(1) 23.10.06

אלגוריתמים

בקורס נכסה הרבה אלגוריתמים מעניינים ורובם גם יהיו מאוד שימושיים.
נראה גם דברים פחות שימושיים אבל יפים מתמטית.
בתשובות נדון במה שסביר לפתור או לא.

<u>מנהלות</u>: השיעור ביום שני מתקיים אלא אם כן מודיעים אחרת.
קיצצו בתקציב של בודקי תרגילים אז יבדקו אותם אבל לא ממש טוב.
תרגילים - 10% מהציון. ציון נותן לפי התרגילים ובסוף עושים איזה פקטור
כך שבערך 50% מהתלמידים יקבלו את כל 10 הנקודות.
לקורס יש אתר ב- ... cs.
ספרים לקורס - cormen ועוד רשימה תופיע באתר. מדי פעם
נפנה לשם להשלמה.
מי שלא לקח dast שנה שעברה - לא נורא! יהיה בסדר.

אנחנו נלמד מספר טכניקות לבנייה של אלגוריתמים. דוגמה מצוינת היא
מיון של מספרים. נתונים $a_1,\dots,a_n$ ורוצים לסדר אותם בסדר עולה.
שיטה טובה היא הפרד ומשול - מחלקים את הבעיה, פותרים את הבעיה
הקטנה יותר ובסוף מאחדים.
אם יש שתי סדרות ממוינות $b_1 \le \dots \le b_n$
$c_1 \le \dots \le c_n$
אז אפשר לאחד אותן לסדרה ממוינת אחת בזמן ליניארי. - למשל ע"י
השוואת האיברים הקטנים ביותר הנותרים ניתן לרשום בזה אחר זה
את המיון המשולב ולכן מספר הפעולות $\le$ אורך שתי הסדרות $= 2n$.
אם נסמן ב-$T(n)$ את מספר הפעולות למיון רשימה באורך n נקבל ש-

$$T(n) = 2T\left(\frac{n}{2}\right) + n =$$
$$= 2\left(2T\left(\frac{n}{4}\right) + \frac{n}{2}\right) + n =$$
$$= 4T\left(\frac{n}{4}\right) + n + n =$$
$$= 8T\left(\frac{n}{8}\right) + n + n + n = \dots =$$
$$= n + \dots + n = n\log n$$

כאשר n גדול מגיעים ל- ←

אפשר להוכיח ש- nlogn זה הסדר גודל הכי טוב שאפשר להגיע אליו אם ממיינים בעזרת השוואות.

נניח שיש לנו n מעבדים. אז במקרה האידיאלי נצטרך לפחות logn זמן (אם הם כולם יעשו בדיוק את מה שהם צריכים לעשות).
את עניין ההפרדה אפשר להבין שבאמת אפשר לעשות ב- logn כי סה"כ אנחנו מקבלים עץ בגובה logn ובהמשך נראה גם ש- n מעבדים יכולים למזג 2 רשימות באורך n בזמן $O(\log n)$.
ואז נקבל אלגוריתם מיון מקבילי שממיין n מספרים בעזרת n מעבדים בזמן $O(\log^2 n)$. אגב ניתן גם בזמן $O(\log n)$ אבל בזה לא נציג את זה בקורס.

עוד שיטה טובה לפתרון בעיות היא רדוקציה לבעיה שיש לה אלגוריתם יעיל.
לדוגמא - נתונים n מספרים $a_1,\dots,a_n$ ומספר נוסף b וצריך לבדוק אם יש שני אינדקסים $1\le i<j\le n$ כך ש- $a_i+a_j=b$.
טריוויאלי לפתור את הבעיה בזמן $O(n^2)$ ע"י בדיקת כל האפשרויות.
איך אפשר לשפר? קודם כל נמיין את המספרים בזמן $O(n\log n)$.
בה"כ $a_1\le\dots\le a_n$. ועכשיו אפשר לפתור את הבעיה ב-$O(n\log n)$ - לכל a_ℓ ניתן לחפש האם $b-a_\ell$ מופיע ברשימה (חיפוש בינארי לוקח) בזמן $O(\log n)$ וסה"כ $O(n\log n)$.
ומה אם אנחנו מחפשים 3 אינדקסים שסכומים b, או 4 או 5 וכן הלאה? בעצם מה קורה. ראשית רוצים לבדוק האם קיימת קבוצה $S\subseteq\{1,\dots,n\}$ כך ש- $\sum_{\ell\in S} a_\ell=b$? לבעיה הזאת לא ידוע אלגוריתם יעיל. מספר תתי הקבוצות הוא 2^n ולא ידוע משהו שעובד יותר טוב. זו בעיה NP-שלמה... בבעיות כאלה נדון בחישוביות.

הרוסים שאלו את עצמם כמה פעולות אריתמטיות בשביל לחשב מכפלה סקלרית של וקטורים n-ממדיים. לא ניכנס לזה כאן אבל יש הוכחה שיש צורך לפחות ב- n פעולות כפל ו- n-1 פעולות חיבור ולמעשה זה די ברור כי

$$(a_1\dots a_n)\cdot(b_1\dots b_n)=\sum_{i=1}^{n} a_ib_i$$

מה עם מכפלה של וקטור במטריצה?

$$(a_1 \dots a_n)\begin{pmatrix} b_{11} & \dots & b_{1n} \\ \vdots & & \\ b_{n1} & \dots & b_{nn} \end{pmatrix} = (c_1 \dots c_n) \qquad c_k = \sum_{j=1}^{n} a_i b_{ij}$$

כאן באופן טבעי צריך n^2 פעולות כפל ו- n^2-1 פעולות חיבור.
(הוכח מתישהו בשנות ה- 70)

ומה עם כפל מטריצות?

$$\begin{pmatrix} a_{11} & \dots & a_{1n} \\ \vdots & & \\ a_{n1} & \dots & a_{nn} \end{pmatrix}\begin{pmatrix} b_{11} & \dots & b_{1n} \\ \vdots & & \\ b_{n1} & \dots & b_{nn} \end{pmatrix} = (c_{ij}) \qquad c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

כמו שכתוב כאן יש צורך ב- n^3 פעולות כפל ו- n^3-n^2 פעולות חיבור.

אבל ב- 69 איש בשם Strassen הראה שבשביל לכפול 2 מטריצות $A, B \in M_2(F)$ אפשר להסתפק ב-7 כפלים ו- 18 חיבורים במקום ב- 8 כפלים ו- 4 חיבורים.
מה הקטע? במטריצות אפשר גם לכפול בבלוקים וזה בדיוק נותן לנו הפרד ומשול. ואז מקבלים את נוסחת הרקורסיה

$$M(n) = 7M\left(\frac{n}{2}\right) + 18\left(\frac{n}{2}\right)^2$$

↳ הזמן לחיבור מטריצות

הפתרון של הרקורסיה הזאת הוא $O(n^{\log_2 7}) = O(n^{2.81})$ שהחל מ- n מסוים זה יותר טוב מ- $n^3 - n^2$.
בעיית כפל המטריצות היא בעיה פתוחה והאלגוריתם הטוב ביותר הידוע כיום הוא בזמן $O(n^{2.36})$. חסם תחתון שידוע כיום הוא בערך $3.5n^2$

בעיה נוספת היא כפל של פולינומים או של מספרים טבעיים.
יש גם את הבעיה הבאה: המלך ארתור ויועץ הסתרים שלו מרלין שהוא כל יכול אבל מוגבל חישובית. ארתור פיקח אבל לא ממש סומך על מרלין. הוא מוכן לשמוע הוכחות אבל הוא קצר רוח אז הוא

מוכן רק לשמוע נימוקים פולינומיאליים. לקראת מסיבה בארמון ארתור ביקש ממרלין לסדר את האורחים רק שכל אביר ירקוד עם עלמה שהוא מכיר. מרלין הסתכל ברשימה ומיד ראה שזה בלתי אפשרי.
הוא מתחיל לספר לארתור על כל $n!$ האפשרויות שהן לא אפשריות וארתור נמאס לו רוצה לשמוע רק דבר פולינומיאלי ומאיים על מרלין שיתלוש את ראשו אם עד למחרת לא ימצא הסבר. אז מרלין מצליח להוכיח את משפט החתונה של הול (ר' סיכומים בדיסקרטית).
המטלה הבאה שארתור הטיל על מרלין היא להושיב סט אבירים סביב השולחן העגול אבל רק שאלה שלא סובלים אחד את השני לא ישבו יחדיו. שוב אין פתרון ואפשר לעבור על כל $(n-1)!$ האפשרויות.
לזה דווקא אין פתרון טוב אבל אם ארתור מוכן לשמוע הוכחה הסתברותית אז ניתן לספק הוכחה.

(3) 26.10.06
אלגו'

בוקר טוב!

היום נדבר על נושא קצר, פשוט אבל חשוב - אלגוריתמים חמדניים.
(Greedy Alg.) בשני ספרי הקורס יש פרק על
אלג' חמדניים.
נתחיל מדוגמה לדברים כאלה.

בעיה: פרצנו לחנות ואנחנו רוצים לקחת את השלל שערכו רב ביותר,
אבל אנחנו מוגבלים במשקל שאנחנו יכולים לסחוב.
אז יש n פריטים עם ערך ומשקל $(w_1, v_1), \dots, (w_n, v_n)$
w_i - המשקל של הפריט ה-i
v_i - ערך הפריט ה-i
אז צריך לבחור חלק מהפריטים האלה כך שהערך יהיה מקסימלי
והמשקל לכל היותר W.
אנחנו מחפשים סדרה $x_1, \dots, x_n$ כאשר $x_i \in \{0,1\}$ (זה
יקבע לכל פריט אם אנחנו לוקחים אותו או לא) כך ש-
$\sum x_i w_i \le W$ וגם $\sum x_i v_i$ מקסימלי.
אם פרצנו לחנות חשמל אנחנו בבעיה קשה. אבל אם פרצנו
לבית מרקחת עם אבקות אז אנחנו יכולים לקחת גם חלקי
בנזנות.

הבעיה הזאת נקראת בעיית התרמיל השברית
באופן פורמלי:
הקלט: סדרה $(w_1, v_1), \dots, (w_n, v_n)$ ו- W
הפלט: סדרה $x_1, \dots, x_n$ כאשר $x_i \in [0,1]$
ומתקיים $\sum x_i w_i \le W$ ו- $\sum x_i v_i$ מקסימלי.

הנה דוגמה קונקרטית: $(10, 60)$, $(20, 100)$, $(30, 120)$ $W=50$
אם נגדיר $r_i = \frac{v_i}{w_i}$ וזה נותן לנו את הערך הסגולי
של כל איבר ואז כדאי לקחת כמה שיותר מערך סגולי גבוה

ולא כדאי לקחת דברים עם ערך סגולי נמוך.

בדוגמה הקונקרטית שלנו מתקבל:

אוצר (30,120) , (20,100) , (10,60)

ערך סגולי 4 5 6

אז כדאי למלא את התרמיל קודם ב-(10,60) וכן הלאה.

ואז מתקבל 1 1 2/3 כמה לוקחים

קל לבדוק שזה הדבר האופטימלי.

זאת בעיה קלאסית של תכנון לינארי. יש לנו אוסף אילוצים (אי שוויונים) לינאריים, ואנחנו רוצים למקסם משהו לינארי.

לתכנון לינארי יש אלגוריתם ידוע יעיל.

נעיר שהבעיה השברית היא מקרה פרטי (פשוט במיוחד) של תכנון לינארי, וקיים אלגוריתם יעיל פולינומיאלי לפתרון בעיות תכנון לינארי כללי (לא נלמד אותו בקורס הזה, אלא בקורס מתקדם באלגוריתמים).

אלגוריתם חמדני:

נגדיר $r_i = \frac{v_i}{w_i}$ ונמיין את ה-r_i בסדר יורד.

בה"כ $r_1 \ge r_2 \ge \dots \ge r_n$

כל עוד אפשר בוחרים $x_i = 1$ מ-$i=1$

עד שמקבל שאי אפשר יותר ($\sum_{j=1}^{i-1} w_j \le W$ אבל $\sum_{j=1}^{i} w_j > W$)

ואז לוקחים $x_i = \frac{(W - \sum_{j=1}^{i-1} w_j)}{w_i}$ והפסק.

$x_1 = 1, \dots, x_{i-1} = 1, x_i, x_{i+1} = 0, \dots, x_n = 0$

אם הגענו ל-$i=n$ ונגמרו הדברים פשוט לוקחים הכל וזהו.

הוכחת נכונות:

צריך להוכיח כמה דברים.

ראשית נראה ש-$0 \le x_\ell \le 1$. זה ברור לכל $\ell \ne i$.

נוכיח רק עבור x_i אז. אבל זה ברור שהרי $0 \le W - \sum_{j=1}^{i-1} w_j < w_i$

לכן $0 \le x < 1$.

עכשיו נראה שמתקיים $\sum_{\ell=1}^{n} x_\ell w_\ell \le W$. אבל זה די ברור, שהרי

$$\sum_{j=1}^{i-1} w_j + x_i w_i = \sum_{j=1}^{i-1} w_j + \left(W - \sum_{j=1}^{i-1} w_j\right) = W$$

נותר להראות את האופטימליות וזה למעשה גם ברור למדי.
אם נתבונן בפתרון שבו לקח $x_1 = 1$ אז אנחנו יכולים לשחרר משקל מאיברים אחרים ולהוסיף אותו ל-x_1 ובגלל שהערך הסגולי שלו גבוה יותר, יוצא לנו שאפשר להגדיל את הערך ועדיין לעמוד בתנאי המשקל, בסתירה לכך שזה הפתרון.
באופן פורמלי יותר:

<u>למה:</u> נתבונן בפתרון (אופטימלי) $y_1, \dots, y_n$ שעבורו $y_1 w_1 < W$ ונניח $y_1 < 1$. אזי קיים פתרון $x_1, \dots, x_n$ רק ש-$x_1 > y_1$ ו- $\sum_{\ell=1}^{n} x_\ell v_\ell \ge \sum_{\ell=1}^{n} y_\ell v_\ell$ (כלומר: יש עוד פתרון שהוא לפחות טוב באותה מידה)

<u>הוכחה:</u> מכיוון ש-$y_1 w_1 < W$ יש y_i אחר גדול מאפס (זה מעניין אם $n=1$, אז אנחנו מניחים ש-$n>1$). בה"כ $0 < y_2$. נגדיר $x_3 = y_3, \dots, x_n = y_n$.
נבחר $\varepsilon > 0$ רק ש-$x_1 = y_1 + \varepsilon \le 1$ וכן $\varepsilon \cdot \frac{w_1}{w_2} < y_2$ ואז נגדיר $0 \le x_2 = y_2 - \varepsilon \frac{w_1}{w_2} \le 1$ (*). בחירה כזאת של ε היא כמובן אפשרית, ונראה כעת שהרווחנו מהבחירה הזאת:

(*) כאן אנחנו מקטינים את x_2 כדי לפצות את המשקל

קודם נראה שאנחנו עומדים בתנאי הבעיה, כלומר שלא עברנו את המשקל המותר:

$$\sum_{i=1}^{n} x_i w_i = (y_1 + \varepsilon) w_1 + \left(y_2 - \varepsilon \frac{w_1}{w_2}\right) w_2 + \sum_{j=3}^{n} y_i w_i =$$

$$= \sum_{i=1}^{n} y_i w_i + (\varepsilon w_1 - \varepsilon w_1) \overset{\text{הנחה}}{\le} W$$

אז אנחנו עומדים באילוץ. עכשיו נראה שהרווחנו בערך:

$$\sum_{i=1}^{n} x_i v_i - \sum_{i=1}^{n} y_i v_i = (x_1 - y_1) v_1 + (x_2 - y_2) v_2 =$$

$$= \varepsilon \cdot v_1 - \varepsilon \frac{w_1}{w_2} v_2 = \varepsilon \left(v_1 - \frac{w_1}{w_2} v_2\right) =$$

$$= \varepsilon (r_1 w_1 - r_2 w_1) = \varepsilon w_1 (r_1 - r_2)$$

אבל $r_1 \ge r_2 \Leftarrow \Leftarrow \sum_{i=1}^{n} x_i v_i \ge \sum_{i=1}^{n} y_i v_i$ ☺

ועכשיו נחזור להוכחת האופטימליות של האלגוריתם החמדני:
מהלמה ברור שבחירת $x_1=1$ (אם אפשר) לא מקלקלת את האופטימליות.
לכן ניתן לבחור $x_1=1$ ו- $W'=W-W_1$, ואז מתקבלת בעיה דומה
בגודל $n-1$, אבל אם נמשיך רק נקבל בדיוק את האלגוריתם
החמדני שראינו קודם.
מש"ל ☺

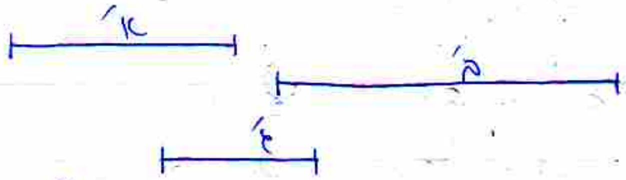
נמשיך בבעיה נוספת שניתן לפתור אותה בעזרת אלגוריתם חמדני.

בעיית שיבוץ המשימות: יש זמני התחלה וזמני סוף וצריך לשבץ
אותן בלו"ז באופן הגיוני, רק שנשלים כמה שיותר, ואסור להיות חפיפה.

באופן פורמלי:
קלט: קטעים $(s_1, f_1), \dots, (s_n, f_n)$
פלט: קבוצה מקסימלית חלקית של קטעים שאינם נחתכים.

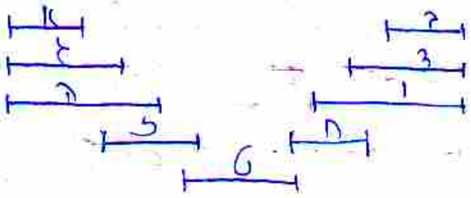
יש כמה דרכים חמדניות שאפשר לנסות בשביל לפתור את זה:
(1) בכל שלב בוחרים את הקטע הקצר ביותר שניתן לבחור.
(2) בכל שלב בוחרים את הקטע שנקודת הקצה שלו f_i קטנה
ביותר מבין הקטעים הנותרים
(3) בכל שלב בוחרים את הקטע שמתנגש עם הכי מעט קטעים אחרים.

טוב, אז הדרכים האלה עובדות:
הנה דוגמא נגדית ל-(1):



לפי (1) היינו בוחרים רק את (ג) אבל ברור שעדיף את (א) ו-(ב);

הנה דוגמה נגדית ל-(3):



לפי האלגוריתם בוחרים ישר את ט' כי הוא מתנגש רק עם שני הקטעים, ואז אפשר לבחור עוד שני קטעים - נניח יוצא (ג', ה', ט') אבל יותר טוב לבחור למשל את (א', ב', ז', ח').

הצעה (2) דווקא עובדת, אבל די בנס.

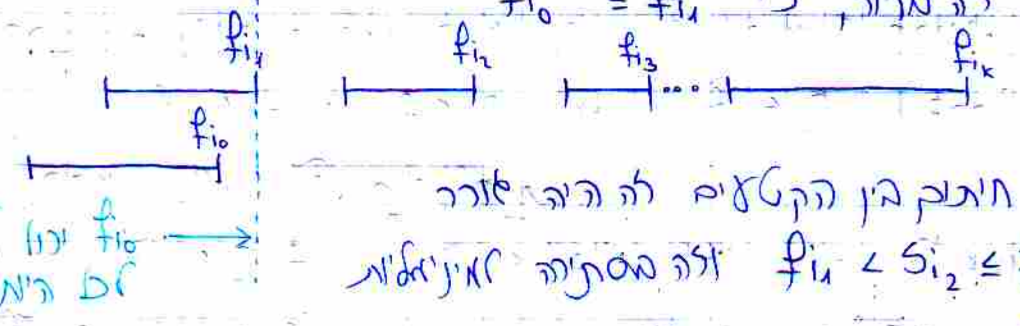
<u>טענה</u>: האלגוריתם החמדני לפי שיטה (2) מוצא פתרון אופטימלי.

<u>הוכחה</u>: יהי $(s_{i_1}, f_{i_1}), \dots, (s_{i_k}, f_{i_k})$ פתרון אופטימלי המכיל k קטעים זרים. בה"כ $f_{i_1} < f_{i_2} < \dots < f_{i_k}$.

יהי i_0 רק ש- $f_{i_0} = \min\{f_i : 1 \le i \le n\}$. לכן $f_{i_0} \le f_{i_\ell}$ עבור $\ell = 1, \dots, k$ (טוב, הוא הקטן ביותר אלה כמובן קטן מהאחרים).

נטען שהקטעים $(s_{i_0}, f_{i_0}), (s_{i_2}, f_{i_2}), \dots, (s_{i_k}, f_{i_k})$ הם קטעים זרים.

טוב, אבל זה ברור, כי $f_{i_0} \le f_{i_1}$



אם היה חיתוך בין הקטעים זה היה אומר ש- $f_{i_1} < s_{i_2} \le f_{i_0}$ וזה בסתירה למינימליות של f_{i_0}.

זהו, וזה בעצם סוף ההוכחה. כי אם יש פתרון אופטימלי אנחנו יכולים להחליף בו את האיבר הראשון באיבר הראשון שהאלג' שלנו בוחר וזה עדיין פתרון אופטימלי. וממשיכים באותו אופן אפשר להחליף את כל האיברים בפתרון באיברים שהאלג' בחר וזה עדיין אופטימלי כי אנחנו נשארים עם אותו מספר קטעים.

הראנו שהצעד הראשון של האלג' החמדני לא מקטין את האופטימום ולכן על הקבוצה הנותרת יש אופטימום בגודל $k-1$, ובאינדוקציה האלג' החמדני מוצא פתרון אופטימלי בגודל k. משל.

דוגמה נוספת שהיא קצת יותר כללית:

מציאת קבוצה בלתי תלויה מקסימלית: נתונה קבוצה של וקטורים $v_1,\dots,v_n \in \mathbb{F}^m$ (שדה כלשהו $\mathbb{F}$) ויש משקלות חיוביים $0<w_1,\dots,w_n$ מחפשים קבוצה $S\subseteq\{1,\dots,n\}$ כך ש-$\{v_i : i\in S\}$ בלתי תלויה לינארית מעל $\mathbb{F}$ וכן $w(S)=\sum_{i\in S} w_i$ מקסימלי.

<u>אלגוריתם חמדני:</u>

בכל צעד בוחרים את הוקטור בעל המשקל הגדול ביותר מבין הוקטורים שאינם תלויים בוקטורים שכבר בחרנו. ממשיכים עד שנגמרו הוקטורים או עד שכל מה שנשאר תלוי במה שכבר בחרנו.

זאת שיטה חמדנית ועובדת בצורה די יעילה: לא צריך לעבור על כל 2^n הקבוצות החוקיות!

להתחתבוננות נתבונן באוסף הבא:

$\mathcal{F}=\{A\subseteq\{1,\dots,n\}$: קבוצת הוקטורים $\{v_i : i\in A\}$ בלתי תלויה לינארית$\}$ (נגדיר שהקבוצה הריקה היא בת"ל)

הטענה הבסיסית שנכונה כאן היא ש-$\mathcal{F}$ מקיימת 2 תכונות חשובות:

1- אם $A\in\mathcal{F}$ ו-$B\subseteq A$ אז $B\in\mathcal{F}$ (דהה?)
זה נקרא ש-$\mathcal{F}$ <u>תורשתית</u>.

2- אם: $A,B\in\mathcal{F}$ ו-$|A|<|B|$ אז קיים $u\in B$ כך ש-$A\cup\{u\}\in\mathcal{F}$. זה נקרא <u>תכונת ההחלפה</u>.
(זה משהו סטנדרטי מאלגברה לינארית ומתבסס על זה שלכל הבסיסים יש אותו מספר איברים)

(6) 30.10.06
אלון

- אלגוריתמים חמדניים -

נתונים n וקטורים $v_1, \dots, v_n \in F^m$ לכל וקטור v_e יש משקל $w_e \geq 0$.
אפשר לייצג אותם במטריצה:

$$m \begin{pmatrix} | & & | \\ v_1 & \dots & v_n \\ | & & | \end{pmatrix}$$

רוצים למצוא קבוצת אינדקסים $A \subseteq \{1, \dots, n\}$ כך ש- $\{v_i : i \in A\}$ בת"ל (כלומר אם $\sum a_i v_i = 0$ אז לכל i $a_i = 0$) וכן המשקל מקסימלי $w(A) = \sum_{i \in A} w_i$

<u>אלגוריתם חמדני</u>:
ממיינים את המשקלות w בסדר יורד.
מה"כ $w_1 \geq w_2 \geq \dots \geq w_n$
בוחרים להוסיף לקבוצת הווקטורים שבחרנו עד כה את הווקטור בעל משקל גבוה ביותר שאינו תלוי לינארית בווקטורים שכבר בחרנו.

נתבונן באוסף $\mathcal{F} = \{A \subseteq \{1, \dots, n\} :$ קבוצת "וקטורים" בלתי תלויים $A\}$
ידוע שמתקיים
(I) אם $B \subseteq A \in \mathcal{F}$ אז $B \in \mathcal{F}$ (תורשתיות)
(II) אם $A, B \in \mathcal{F}$ ו- $|A| < |B|$ אז קיים $u \in B \setminus A$ כך ש- $A \cup \{u\} \in \mathcal{F}$

<u>הגדרה</u>: אוסף קבוצות $\mathcal{F}$ המקיים את התכונות (I) (II) נקרא <u>מטרואיד</u>.

__הוכחת__ נכונות האלגוריתם החמדני:

תהי $T\subseteq\{1,\dots,n\}$ קבוצה בעלת משקל אופטימלי

ותהי S הקבוצה שבוחר האלגוריתם החמדני.

__טענה:__ $|T|=|S|$

__הוכחה:__ אם $|T|<|S|$ קיים $u\in S\setminus T$ כך ש- $T\cup\{u\}$ ב"ת

ואז $w(T\cup\{u\})>w(T)$ מסתירה לאופטימליות של T.

אם $|S|<|T|$ אז $u\in T\setminus S$ כך ש- $S\cup\{u\}$ ב"ת

אבל זה אומר שהאלגוריתם החמדני סיים את העבודה.

נסמן את האיברים ב- $S=\{s_1,\dots,s_k\}$

$T=\{t_1,\dots,t_k\}$

כך שסדר האיברים הנו לפי משקל יורד.

$$w_{s_1}\ge\dots\ge w_{s_k}$$

$$w_{t_1}\ge\dots\ge w_{t_k}$$

אנחנו רוצים להוכיח שמה שמצאנו הוא גם בעל משקל מקסימלי.

מספיק להראות ש- $w(T)\le w(S)$.

נניח בשלילה ש- $w(T)>w(S)$ ונגיע לסתירה. במקרה כזה

לפי ההנחה קיים אינדקס ראשון l כך ש- $w_{s_l}<w_{t_l}$. נגדיר

$B=\{t_1,\dots,t_l\}$ $A=\{s_1,\dots,s_{l-1}\}$ כמובן $|A|<|B|$

$\Leftarrow$ לפי למת ההחלפה קיים $u\in B\setminus A$ כך ש- $A\cup\{u\}$ ב"ת

ואז $w_u\ge w_{t_l}>w_{s_l}$ וזה סותר את העובדה

שהאלגוריתם החמדני בחר ב-s_l ולא ב-s_u.

תהי I קבוצה סופית, ו- $\mathcal{F}$ קבוצה של תת קבוצות של I
המהווה מטרואיד. נתונה פונקצית משקל $\omega: I \to \mathbb{R}_{>0}$
כאשר $\omega(i)$ המשקל של i ב-I.
מחפשים קבוצה ב- $\mathcal{F}$ בעלת משקל מקסימלי.

זה סיכום של מה שעשינו כאן. זה בכלל לא חייב להיות קשור לוקטורים וכאלה. זה נכון למטרואידים בכלל. וראינו שהאלגוריתם החמדני שרשמנו מחזיר קבוצה אופטימלית.

<u>הערה</u>: אם $\mathcal{F}$ תורשתית אבל תנאי II איננו מתקיים אז קיימת פונקצית משקל שעבורה האלגוריתם החמדני איננו אופטימלי.

<u>הוכחה</u>: נניח $A,B\in\mathcal{F}$, $|A|<|B|$, אבל לא קיים $u\in B\setminus A$ כך ש- $A\cup\{u\}\in\mathcal{F}$. נגדיר

$$\omega(x)=\begin{cases}1 & x\in A\\ 1-\varepsilon & x\in B\setminus A\\ \delta & \text{אחרת}\end{cases}$$

נבחר ε, δ מספיק קטנים כך ש- $\omega(B) > \omega(A)+|I|\delta$.
כלומר $|B|(1-\varepsilon) > |A| + |I|\delta$ וברור שזה אפשרי (כי $|A|<|B|$).

<u>אלגוריתם למציאת עץ פורש מקסימלי בגרף</u>

יהי $G=(V,E)$ גרף לא מכוון (קשיר) ותהי פונקצית משקל על הצלעות $\omega: E\to\mathbb{R}_{>0}$. אם $\omega(e)<\omega_0$ לכל e.
נגדיר $\bar{\omega}(e)=\omega_0-\omega(e)$ ואז מקסימום עבור $\bar{\omega}$ הוא מינימום עבור ω ואז אפשרי להשתמש באלגוריתם למציאת עץ פורש מינימלי של G עם פונק' משקל $\bar{\omega}$.
האלגוריתם החמדני בוחר אוסף צלעות באופן הבא: עוברים על הצלעות לפי משקל יורד ומוסיפים צלע שאיננה סוגרת מעגל בגרף.

בה"כ $V=\{1,\dots,n\}$. לכל $e=(i,j)$ נתאים

וקטור $\mathbb{F}_2^n \ni v_e = \begin{pmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{pmatrix} \begin{matrix} \\ \\ \leftarrow i \\ \\ \leftarrow j \\ \\ \\ \end{matrix}$ $\qquad \mathbb{Z}_2 = \mathbb{F}_2$

כלומר $v_e(k) = \begin{cases} 1 & k=i,j \\ 0 & \text{אחרת} \end{cases}$

<u>למה:</u> קבוצה חלקית A לקבוצת הווקטורים $v_{e_1},\dots,v_{e_m}$ כאשר $|E|=m$ היא קבוצה בת"ל אמ"מ A מכילה מעגל.

<u>הוכחה:</u>

($\Leftarrow$) די להראות שקבוצת וקטורים המתאימה למעגל תלויה לינארית. נניח שזה מעגל:

$e_1=(i_1,i_2)\ ,\ e_2=(i_2,i_3)\ ,\ \dots\ ,\ e_k=(i_k,i_1)$

כל קואורדינטה מופיעה בדיוק פעמיים. עם ערך 1 ולכן הסכום (מודולו 2) הוא אפס.

($\Rightarrow$) נניח ש- $v_{e_1}+\dots+v_{e_k}=0$. יהי $e_1=(i_1,i_2)$. קיים וקטור אחר רק שבקואורדינטה i_2 יש 1. בה"כ זה $e_2=(i_2,i_3)$. באותו אופן חייב להיות וקטור רק שבקואורדינטה i_3 יש 1. ממשיכים הלאה באותו אופן. אבל יש מספר סופי של צלעות. בסוף נצטרך לחזור לצלע שכבר היינו בה ולכן יוצא מעגל ☺

<u>מסקנה:</u> קבוצת וקטורים בת"ל לינארית מעל $\mathbb{F}_2$ מתאימה לאוסף צלעות שאין בהן מעגל (= איחוד של עצים או יער).
לכן האלגוריתם החמדני מוצא פתרון אופטימלי.

תם (סוף ימי הזיכרון :-) נושא האלגוריתמים החמדניים.
ניתן ורצוי להרחיב את הידיעות שלנו ע"י קריאה בקורמן.

אלגוריתמים דינמיים

כפל מטריצות:

$$A_{n\times m}\cdot B_{m\times k} = C_{n\times k}$$

חישוב כל איבר במכפלה עולה $O(m)$ ובסה"כ זה יוצא $n\cdot m\cdot k$ פעולות.

למשל:

$$C_{1000\times 3} = A_{1000\times 1}\cdot B_{1\times 2000}\cdot D_{2000\times 3}$$

$B\cdot D$ – עולה 6000

$A\cdot B$ – עולה 2000000

אז סדר מכפלת המטריצות משפיע על מספר הפעולות!

נניח שיש מטריצות שרוצים לכפול

$$\underset{n_1\times n_2}{M_1}\cdot \underset{n_2\times n_3}{M_2}\cdots \underset{n_k\times n_{k+1}}{M_k} = C_{n_1\times n_{k+1}}$$

איך מחשבים את זה בצורה אופטימלית?

מספר האפשרויות לשים פה סוגריים הוא מספר קטלן – מספר מעריכי גדול מאוד!

אבל את הבעיה הזאת אפשר לפתור ע"י חלקי בעיות קטנים יותר

(9) 2.11.06
אלון'

תכנון דינאמי

זו טכניקה חשובה - כדאי לקרוא על זה בשני הספרים.

<u>דוגמה</u> נתונות n מטריצות $A_1,\dots,A_n$ כאשר A_i מסדר $P_{i-1}\times P_i$.
לכל i המכפלה $M_{i,i+1}=A_i\cdot A_{i+1}$ היא מסדר $P_{i-1}\times P_{i+1}$.
בשיעור הקודם ראינו שסדר מכפלת המטריצות משפיע על מספר הפעולות:
נניח שהכפלת מטריצות A, B מסדרים $p\times q$, $q\times r$ בהתאמה דורשת
$p\cdot q\cdot r$ פעולות. קל מאוד לראות דוגמאות שהסדר של חישוב המכפלה
$M_{1,n}=A_1\times A_2\times\dots\times A_n$ משפיע על מספר הפעולות.
נסמן ב- $T[i,j]$ את מספר הפעולות הקטן ביותר הדרוש לחישוב
($M_{i,i}=A_i$) $M_{i,j}=A_i\times A_{i+1}\times\dots\times A_j$. אנחנו רוצים לחשב את $T[1,n]$.
נניח שביצענו את הכפל ככה: $M_{1,n}=(A_1\times\dots\times A_k)\times(A_{k+1}\times\dots\times A_n)$ -
כלומר זו המכפלה האחרונה שביצענו. הפעולות שהלכו לנו הן:

$$T[1,n]=T[1,k]+T[k+1,n]+p_0\cdot p_k\cdot p_n$$

הבעיה היא כמובן שאנחנו לא יודעים מה k. אבל מה שכן ידוע הוא ש-

$$T[1,n]=\min_{1\le k\le n}\{T[1,k]+T[k+1,n]+p_0\cdot p_k\cdot p_n\}$$

כמובן $T[i,i]=0$ וזה נותן לנו הגדרה רקורסיבית כשרה לחלוטין ואפשר לרשום
אלגוריתם שפועל ככה. רק שזה יהיה איום ונורא כי מספר האפשרויות הוא
מעריכי. חוץ מזה אנחנו עושים המון חישובים כפולים. כשמחשבים
את $T[1,4]$ מחשבים בדרך גם את $T[1,3]$, אבל נצטרך לחשב אותו
גם כשנחשב את $T[1,7]$. היינו רוצים לשמור את המידע הזה ולא לחשב
אותו כל פעם מחדש.

בסה"כ אוסף תת הבעיות הוא חישוב מחיר מכפלות מהצורה $M_{ij}=A_i\times\dots\times A_j$
כאשר $i<j$ - שזה בעצם $O(n^2)$ בעיות שכל אחת מהן נדרש למצוא
פתרון אופטימלי.

ניתן לחשב את $T[i,j]$ עבור $j=i+1$
ואח"כ $j=i+2$ וכן הלאה.
וכדי לעבור על כל ה-i-ים. וכמובן מספיק לחשב כל אחד מהם דרכים

להשתמש בפתרונות הקודמים, כי הרי:

$$T[i,j] = \min_{i\le k<j} \{ T[i,k] + T[k+1,j] + p_i p_k p_j \}$$

אבל בכל שלב ותמיד שבתוך הסוגריים כבר חישבנו - כי הרי $T[i,k]$ ו- $T[k+1,j]$ מופיעים קודם בסדר החישוב שלנו שהגדרנו בסוף העמוד הקודם. ולכן לכל k החישוב הזה דורש סה"כ $O(n)$ פעולות. לכל הפרש $d=j-i$ צריך לחשב כל חישוב כזה $O(n)$ פעמים ולכן נקבל בסה"כ $O(n^3)$ פעולות.

עוד דרך שבה ניתן לתכנן היא פשוט לשמור בצד כל דבר שחישבנו ואז כשמתחילים לעשות את הרקורסיה קודם בודקים אם כבר חישבנו את זה וזה יכול לחסוך הרבה חישובים מחדש. אך השיטה הזו מגדילה קצת את הזיכרון.

עוד נקודה - באלגוריתם הזה קיבלנו רק את המחיר. אבל זה לא מספיק לדעת מהו את הדרך לבצע. פשוט כשמחשבים את המינימום שומרים גם את ה-k שבעבורו הוא התקבל $k=S[i,j]$ - ה-k שבעבורו התקבל המינימום לחישוב $T[i,j]$. וכו'. אז מכניסים סוגריים במכפלה והנה הדרך.

<u>קוריוז</u>: שנה שעברה באוניברסיטה זכתה לרשת שמאפשרת לחישובים לרוץ במקביל בכל המחשבים שלא משתמשים בהם באותו רגע (מה שאולי אתם זוכרים את המאווררים במעבדות כי הרעש היה בלתי נסבל).

מי שהכי משתמש ברשת הזו הם הביולוגים החישוביים שמחשבים מרחקים בין מקטעי DNA. עבורנו DNA הוא פשוט סדרה של אותיות מהא"ב $\{A,T,G,C\}$, לדוגמא, $X = ACCTGA\ldots$. מילה היא פשוט שרשור של האותיות - $X = x_1 \ldots x_n$ כאשר $x_i \in \{A,T,G,C\}$.

הבעיה היא בעיית חישוב תת סדרה משותפת ארוכה ביותר לשתי סדרות של אותיות מעל א"ב נתון.

בהינתן מילה $X = x_1 \ldots x_n$ תת סדרה של X היא מילה מהצורה $x_{i_1} \ldots x_{i_k}$ כאשר $i_1 < \ldots < i_k$.

עכשיו נתונים שתי מילים ורוצים למצוא סדרה משותפת.

נגדיר פורמלית את הבעיה:

קלט: שתי סדרות $X = x_1 \dots x_n$

$Y = y_1 \dots y_m$

פלט: אורך תת הסדרה המשותפת ל-X ו-Y הארוכה ביותר מבין אחת הסדרות האלה.

אלגוריתם: בה"כ $m<n$ ואז ל-X יש סה"כ 2^n תת סדרות. לבדוק אם תת סדרה כזאת נמצאת ב-Y זו בדיקה מאוד פשוטה ולוקחת $O(n)$. אם יוצא שהאלגוריתם הוא $O(n2^n)$ שזה נורא ואיום. נשפר אותו.

<u>למה:</u> אם $X = x_1 \dots x_n$, $Y = y_1 \dots y_m$ שתי מילים ותהי $Z = z_1 \dots z_k$ תת מילה משותפת ארוכה ביותר. אזי:

(1) אם $x_n = y_m$ אז $y_m = z_k = x_n$ ו- $z_1 \dots z_{k-1}$ תת מילה משותפת ארוכה ביותר של $x_1 \dots x_{n-1}$ ו- $y_1 \dots y_{m-1}$

(2) אם $x_n \neq y_m$ אזי אם $z_k \neq x_n$ אזי Z היא האופטימום גם עבור $x_1 \dots x_{n-1}$ ו- $y_1 \dots y_m$.

(3) אם $x_n \neq y_m$ ו- $z_k \neq y_m$ אזי Z היא גם אופטימום עבור $x_1 \dots x_n$ ו- $y_1 \dots y_{m-1}$.

<u>הוכחה:</u>

(1) אם $z_1 \dots z_{k-1}$ איננה האופטימום עבור $x_1 \dots x_{n-1}$ ו- $y_1 \dots y_{m-1}$ אזי נבחר W יותר ארוך מ- $k-1$ ואז $W z_k$ יהיה תת מילה משותפת יותר ארוכה - באורך לפחות $k+1$ - בסתירה לאופטימליות של Z.

(2) אחרי הורדת x_n, Z עדיין תת סדרה משותפת, ולכן זה הטוב ביותר, כי אחרת היה משהו ארוך יותר ממש.

(3) סימטרי ל-(2).

הלמה הנ״ל בעצם אומרת לנו מה לעשות: מסתכלים על האות האחרונה. אם הן שוות חוזרים ברקורסיה על התהליך כאשר מוחקים את האותיות האחרונות. אם הן שונות יש לבחור את המקסימום מבין האפשרויות (2) ו-(3).

נרשום באופן מסודר:

נסמן $X_k = x_1 \dots x_k \quad 1 \le k \le n$

$Y_\ell = y_1 \dots y_\ell \quad 1 \le \ell \le m$

נסמן $C[k,\ell]$ – אורך תת המילה המשותפת הארוכה ביותר ל-X_k ו-Y_ℓ.
כאן $k=0,\dots,n$, $\ell=0,\dots,m$ והמשמעות של אפס היא מילה ריקה – כי ירצו לציין שאין בכלל תת מילה משותפת.
מהלמה מתקבלת הנוסחה הבאה:

$$C[k,\ell] = \begin{cases} C[k-1,\ell-1]+1 & x_k = y_\ell \\ \max\{C[k-1,\ell],\, C[k,\ell-1]\} & \text{אחרת} \end{cases}$$

תנאי ההתחלה שלנו הם $C[0,\ell] = C[k,0] = 0$

את הדבר הזה קל מאוד לחשב: נניח הסדרות הן ABC ACD

מתקבלת הטבלה הבאה:

		0	1 (A)	2 (B)	3 (C)
	0	0	0	0	0
A	1	0	1	1	1
C	2	0	1	1	2
D	3	0	1	1	2

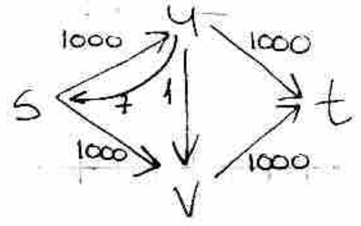
כל מילוי בטבלה דורש להסתכל על 3 מספרים + השוואה בין 2 אותיות – סה״כ $O(n \cdot m)$.
מהטבלה גם ניתן לשחזר בזמן $O(n+m)$ את תת המילה המקסימלית (תחשבו איך)

(11) 6.11.06
אלמו׳

באלגברה פתרנו ∈ מע׳ משוואות. ומה אם יש לנו אוסף של אי-שוויונים?
היינו רוצים למצוא את הנקודות שמקיימות את מע׳ אי השוויונים הללו.

זרימה ברשתות

נתון גרף $G=(V,E)$ גרף מכוון וקיבולות לצלעות.
קיבולת זו פשוט פונקציה
$C: E \to \mathbb{R}^+$



בגרף יש שני קדקודים מיוחדים $s,t \in V$ ואנו רוצים למצוא את קצב הזרימה המקסימלי ב-G מ-s ל-t.
אין טעם להזרים חומר בשני הכיוונים בין שני קדקודים כי הזרמים מבטלים אחד את השני וניתן להסתפק בזרם קטן יותר מכיוון אחד וזרם אפסי בכיוון השני. לכן נחפש רק זרמים כאלה.
לכל צלע $(u,v) \notin E$ נגדיר $C(u,v)=0$ ונגדיר זרימה בגרף G להיות פונקציה $f: V \times V \to \mathbb{R}$ המקיימת:

(I) $f(u,v) \le c(u,v)$ לכל $u,v \in V$ (תנאי קיבולת)

(II) אנטי סימטריות $f(u,v) = -f(v,u)$ לכל $u,v \in V$.

(III) חוק שימור החומר לכל קדקוד $v \in V \setminus \{s,t\}$ מתקיים

$$\sum_{u \in V} f(u,v) = 0$$

כך מסכם את הזרימה שיוצאת מ-v - מספרים חיוביים
ואת הזרימה שנכנסת ל-v - מספרים שליליים
וסה"כ יוצא אפס.

בהינתן זרימה f של G גודל הזרימה $|f| = \sum_{v \in V} f(s,v)$
כי הזרימה נטו היוצאת מ-S.

תרגיל להפסקה: $|f| = \sum_{v \in V} f(v,t)$

בהינתן f של G נרחיב את הפונקציה f לתת קבוצות של V. אם $X,Y \subseteq V$ נגדיר

$$f(X,Y) = \sum_{\substack{x\in X\\ y\in Y}} f(x,y)$$

<u>למה:</u>

(I) לכל $X \subseteq V$: $f(X,X)=0$

$$f(X,X) = \sum_{\substack{x\in X\\ y\in X}} f(x,y) = \sum_{\substack{x\in X\\ y\in X}} -f(y,x) = -f(X,X)$$

(II) לכל $X,Y \subseteq V$ $f(X,Y) = -f(Y,X)$

$$f(X,Y) = \sum_{\substack{x\in X\\ y\in Y}} f(x,y) = \sum_{\substack{x\in X\\ y\in Y}} -f(y,x) = -f(Y,X)$$

(III) אם $X,Y,Z \subseteq V$ ו- $X\cap Y=\emptyset$ אז

$$f(X\cup Y, Z) = f(X,Z) + f(Y,Z)$$

וכן גם בקואורדינטה השנייה.

$$|f| = f(s,V) = f(V,V) - f(V-s,V) =$$
$$= -f(V-s,V) = f(V,V-s)$$
$$= f(V,t) + f(V,V-t-s) = f(V,t)$$

יהי G גרף ו- f זרימה ב- G.

נגדיר לכל $u,v\in V$ את קיבולת השארית לפי f :

$$c_f(u,v) = c(u,v) - f(u,v)$$

ונקבל את גרף השארית G_f שלה אותו גרף רק עם הצלעות עם רשומות השאריות.

<u>למה:</u> תהי f' זרימה בגרף השארית G_f.

נגדיר זרימה חדשה $g = f+f'$ - לכל $u,v\in V$ $g(u,v)=f(u,v)+f'(u,v)$

אז g זרימה בגרף הנתון G וגם $|g|=|f|+|f'|$

הוכחה:

(I) אנטי סימטריה - ברור

(II) $g(u,v) \le c(u,v)$:

$$g(u,v) = f(u,v) + f'(u,v) \le f(u,v) + c(u,v) - f(u,v) = c(u,v)$$

(III) חוק שימור החומר - ברור

האלגוריתם של Ford - Fulkerson

נתבונן בגרף השיורי, עם הקשתות שלגביהן $c_f(u,v) > 0$.
אם יש מסלול פשוט (ללא מעגלים) המחבר בין s ל-t ב-G_f שנקרא לו π אז

$$c(\pi) = \min\{ c_f(u,v) : (u,v) \text{ קשת ב-} \pi \}$$

נגדיר זרימה f' בגרף השיורי באופן הבא:

$$f'(u,v) = \begin{cases} c(\pi) & \text{ קשת ב-}\pi\ (u,v) \\ -c(\pi) & \text{ קשת ב-}\pi\ (v,u) \\ 0 & \text{אחרת} \end{cases}$$

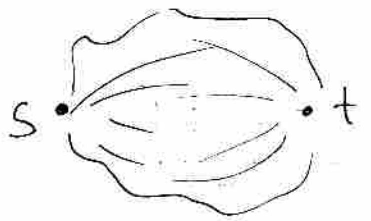
אזי $|f'| = c(\pi)$

האלגוריתם: נתחיל מהזרימה $f(u,v)=0$ לכל (u,v). בכל צעד נתבונן בגרף השיורי G_f. נמצא מסלול פשוט π ב-G_f מ-s ל-t. נחשב את קיבולת המסלול ונגדיר זרימה f' כמו לעיל בגרף G_f ואז נתקן את הזרימה f לזרימה $f+f'$. מסלול כזה π - נקרא מסלול מתקן לזרימה f.

(13) 9/11/06
אלון'

זרימה ברשתות

נתון גרף מכוון $G=(V,E)$ ופונקציה $c: E\to \mathbb{R}^+$ - קיבולת של קשתות הגרף. בגרף שני קודקודים מיוחדים $s,t\in V$.



זרימה הינו פונקציה $f: V\times V \to \mathbb{R}$ כך ש-

(I) $f(u,v) \le c(u,v)$

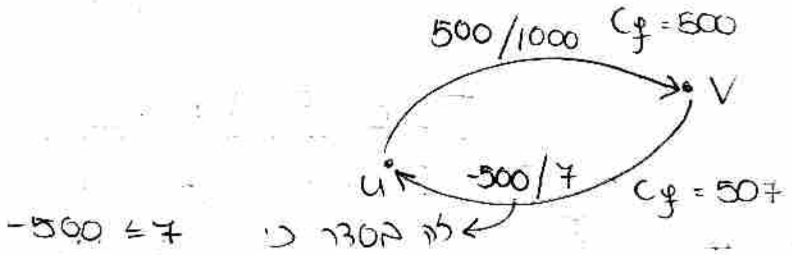
(II) $f(u,v) = -f(v,u)$

(III) לכל $v\notin\{s,t\}$ מתקיים $f(v,V)=0$

(שומר לאן שמאת זרימה לא נוצרת יש מאין).

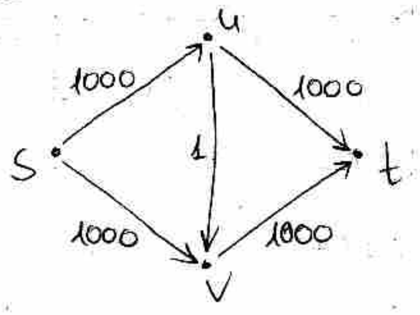
ראינו ש- $|f| = f(s,V) = f(V,t)$

בהינתן זרימה f של G נגדיר את שארית הקיבולת של צלע (u,v) לפי f ע"י $c_f(u,v) = c(u,v) - f(u,v)$



בהינתן זרימה f נגדיר את גרף השארית $G_f=(V,E_f)$

$E_f=\{(u,v): u,v\in V,\ c_f(u,v)>0\}$



נגדיר זרימה:

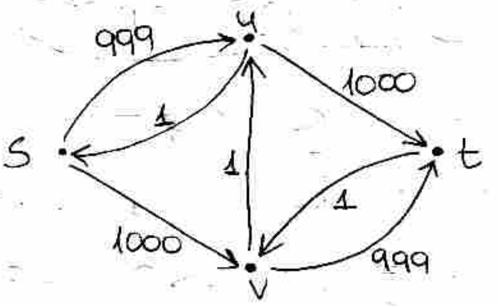
$f(s,u) = f(u,v) = f(v,t) = 1$

$f(u,s) = f(v,u) = f(t,v) = -1$

אחרת $f=0$

$\Leftarrow$ $|f|=1$

גרף השארית נראה כך:



<u>למה</u>: אם f' זרימה בגרף השארית G_f אז הזרימה $g = f + f'$ $\left(g(u,v) = f(u,v) + f'(u,v)\right)$ היא זרימה חוקית בגרף המקורי G וכן $|g| = |f| + |f'|$.

<u>הערה</u>: לכאורה אנחנו מגדירים כאן זרימה גם בצלעות שאולי לא קיימות. אבל זה בסדר כי הנחנו ש- $g(u,v) \le c(u,v)$ ולכן $c(u,v)=0$ אז גם $g(u,v)=0$.

<u>הערה</u>: אם ב-G יש m צלעות ב-G_f יש לכל היותר $2m$ צלעות - כי מקסימום לכל צלע יכולה להתווסף צלע בכיוון ההפוך.

<u>Ford-Fulkerson</u>

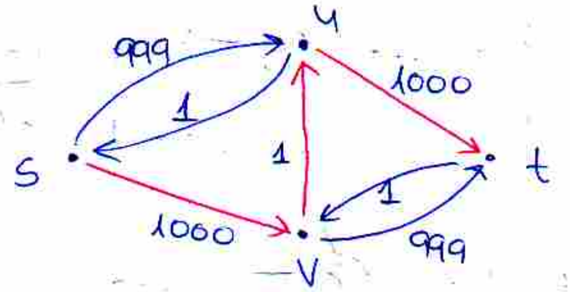
בהינתן G וזרימה f ניצור את גרף השארית.
אם בגרף השארית יש מסלול (פשוט) מכוון π מ-s ל-t (נגדיר את קיבולת המסלול π (בגרף השארית לפי f)

$$c_f(\pi) = \min\{c_f(x,y) : (x,y) \text{ צלע המופיעה ב-} \pi\}.$$

זה בעצם צוואר הבקבוק של המסלול הזה.
(נגדיר זרימה f' בגרף השארית G_f:

$$f'(x,y) = \begin{cases} c_f(\pi) & (x,y) \text{ צלע ב-} \pi \\ -c_f(\pi) & (y,x) \text{ צלע ב-} \pi \\ 0 & \text{אחרת} \end{cases}$$

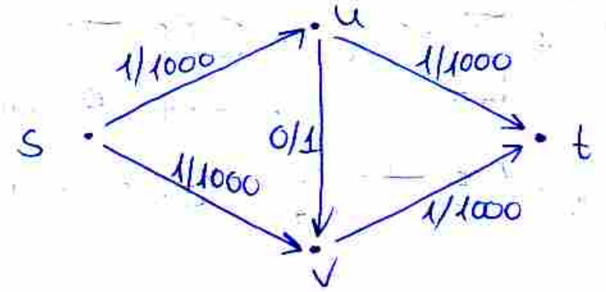
ברור שזו אכן זרימה $|f'| = c_f(\pi)$. לפי הלמה אנחנו יודעים ש- $f+f'$ זרימה גדולה יותר מהזרימה f.
האלגוריתם מוסיף זרימות עד שהצלעות רוויות.



$\pi = s \to v \to u \to t$

$c_f(\pi) = 1$

$\Rightarrow$



$f + f'$

$|f + f'| = 2$

<u>תאור האלגוריתם</u>: מתחילים מזרימה $f \equiv 0$
בוחרים בכל צעד מסלול מתקן בגרף השארית לפי f.
מתקנים את הזרימה ומקבלים זרימה גדולה יותר
ממשיכים עד שאין מסלול בין s ל-t בגרף השאריות.

<u>הגדרה</u>: יהי $G=(V,E)$ גרף מכוון עם פונקציית קיבולת c
וקודקודים מיוחדים $s,t \in V$. חלוקה של קודקודי V לשתי
קבוצות (S,T) כך ש- $V=S\cup T$ ו- $S\cap T=\emptyset$ ו- $s\in S$, $t\in T$
נקראת חתך בגרף G.
יהי (S,T) חתך בגרף G. $f(S,T) =$ הזרימה (נטו)
לרוחב החתך.

<u>טענה</u>: $f(S,T) = |f|$ לכל חתך (S,T) בגרף.
<u>הוכחה</u>: $f(S,T) = f(S,V) - f(S,S)$
שהרי אם $X\cap Y=\emptyset$ אז $f(Z,X)+f(Z,Y) = f(Z,X\cup Y)$
והרי $S\cup T = V$. אבל $f(S,S)=0$ אז $f(S,T)=f(S,V)$
אבל $S=\{s\}\cup(S\setminus\{s\})$. לכן מאותו שיקול
$f(S,V) = f(s,V) + f(S\setminus\{s\}, V) = |f| + f(S\setminus\{s\},V)$
אבל $S\setminus\{s\} \not\ni s,t$ - כלומר אלה כולם קודקודים פנימיים
לכן לפי חוק שימור החומר הסכום כולו מתאפס, כלומר $f(S\setminus\{s\},V)=0$
$\Leftarrow f(S,T) = |f|$

הגדרה: יהי (S,T) חתך בגרף G. הקיבולת של החתך $c(S,T)$ מוגדרת ע"י

$$c(S,T)=\sum_{\substack{u\in S\\ v\in T}} c(u,v)$$

למה: לכל חתך (S,T) ולכל זרימה f ב-G $f(S,T)\le c(S,T)$

הוכחה: $f(S,T)=\sum_{\substack{u\in S\\ v\in T}} f(u,v) \le \sum_{\substack{u\in S\\ v\in T}} c(u,v) = c(S,T)$

ולכן לכל זרימה f מתקיים:

$$\max_{\substack{f \text{ זרימה}\\ \text{ב-}G}} |f| \le \min\{c(S,T) : (S,T) \text{ חתך ב-}G\}$$

☺

משפט: תהי f זרימה בגרף G, s,t הקודקודים המיוחדים. אז התנאים הבאים שקולים:

(I) f היא זרימה מקסימלית ב-G

(II) בגרף השארית G_f אין מסלול מתקן מ-s ל-t

(III) $|f|=c(S,T)$ עבור חתך כלשהו (S,T) ב-G.

הוכחה:

(I $\Rightarrow$ II) כי אחרת ניתן לתקן ולהגדיל את הזרימה f בניגוד להנחת המקסימליות

(II $\Rightarrow$ III) נתבונן בגרף השארית G_f. נגדיר

$S=\{v :$ ניתן להגיע מ-s ל-v בעזרת מסלול ב-$G_f\}$

$T=V\setminus S$

נטען שזה חתך. קודם כל ברור ש-$s\in S$ ו-$t\notin S$ כי לפי ההנחה אין מסלול ב-G_f מ-s ל-t. אז זה באמת חתך.
נטען שלכל (u,v) כך ש-$u\in S, v\in T$ מתקיים $f(u,v)=c(u,v)$
שהרי אחרת $f(u,v)<c(u,v)$ ואז הקשת $(u,v)\in E$
נמצאת בגרף השארית. יש מסלול מ-s ל-v ולכן גם מסלול מ-s ל-u, בסתירה לכך ש-$u\notin S$. לכן $|f|=f(S,T)=c(S,T)$
(נעיר שההוכחה נותנת אלגוריתם למציאת חתך כזה בהינתן זרימה מקסימלית)

(III $\Rightarrow$ I) לכל חתך (S,T) $|f|\le c(S,T)$. לכן שוויון גורר ש-f מקס. ☺

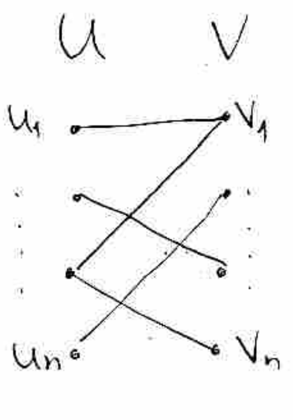
<u>מסקנה</u>: המשפט מוכיח שאם פורד-פולקרסון עוצר, אז זה אומר שהוא מצא זרימה מקסימלית.

אם כל הקיבולות הן מספרים שלמים, אז בכל צעד האלגוריתם מגדיל את הזרימה לפחות ב-1. לכן אחרי מספר חזרות שהוא לכל היותר $|f^*|$ - הזרימה המקסימלית חייבים לסיים.

אם כל הקיבולות הן מספרים שלמים, מובטח לנו כמובן שהזרימה המקסימלית f^* יש לה ערך מספר שלם. הרי באלגוריתם כל פעם חיברנו וחיסרנו מספרים שלמים, לא עשינו חילוק או משהו כזה. אז אין שום סיבה שלא ייצא מספר שלם.

בעיית הזיווג בגרף דו-צדדי

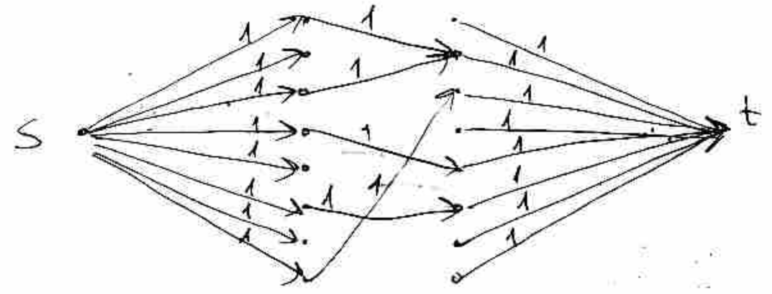
נתון גרף דו צדדי $G=(U \cup V, E)$ (כאשר $|U|=|V|=n$) כאשר $E \subseteq U \times V$



זיווג שלם בגרף G הוא אוסף של n צלעות $M=\{e_1,\dots,e_n\} \subseteq E$ כך שלכל u_i קיים v_j יחיד כך ש $(u_i,v_j) \in M$

<u>באופן כללי</u>, אנו מחפשים קבוצה מקסימלית $M \subseteq E$ כך שלכל $(u,v) \in M$ אין $v \neq v'$ כך שגם $(u,v') \in M$.

נבנה גרף שמכיל את G ונוסיף קודקודים מיוחדים s,t. נחבר את s לכל U ואת V נחבר ל-t. לכל הצלעות נגדיר $c(u,v)=1$.



נחפש זרימה מקסימלית בגרף.
כל צעד ב-FF דורש $O(|E|+n)$ צעדים (חיפוש
מסלול בין קודקודים) ויש לכל היותר n חזרות ובסה"כ $O(n^3)$
FF מוצא זרימה שלמה מקסימלית.
אם $m=|f|$ אז בהינתן זיווג M קיימת זרימה בגודל $|M|$
נזרים 1 לאורך הצלעות ב-M. ולכן $|M| \le |f|$.
מהינתן זרימה שלמה (במספרים שלמים) נבחר צלעות
בהן קיימת זרימה 1. $|f|$ - מספר הצלעות שבחרנו.
זיווג מקסימלי $\le |f|$ $\Leftarrow$ זרימה מקסימלית נותנת זיווג מקסימלי.

(16) 16.11.06
אלגו'

* כדאי להסתכל בספר של Kleinberg & Tardos. יש שם אוסף יפה של דוגמאות של בעיות שאפשר לפתור בעזרת זרימה ברשתות.

זרימה ברשתות

תזכורת (מהסמסטר הקודם - מאלגו'):
נתון גרף $G=(V,E)$ ובו שני קודקודים מיוחדים s,t.
כמו כן נתונה פונקציה $c: E \to \mathbb{R}^+$.
זרימה f היא פונקציה $f: E \to \mathbb{R}$ כך ש-
- לכל $u,v \in V$ $f(u,v) \le c(u,v)$
- מתקיים חוק שימור החומר לקודקודים הפנימיים - $V \setminus \{s,t\}$
- לכל $u,v \in V$ $f(u,v) = -f(v,u)$

גודל הזרימה מוגדר ע"י $|f| = \sum_{v\in V} f(s,v)$

האלגוריתם של Ford-Fulkerson מוצא זרימה מקסימלית
אם גרף נתון $G=(V,E)$ עם פונקציית קיבולת c.
איך הוא פועל?
- מתחילים מבחירת זרימה $f \equiv 0$ שהיא אכן זרימה חוקית.
- בהינתן זרימה, גרף שארית הזרימה G_f הוא גרף עם פונקציית קיבולת שמוגדרת ע"י $c_f(u,v) = c(u,v) - f(u,v)$
- אם f' זרימה ב-G_f אז $f+f'$ זרימה ב-G וגודל הזרימה הוא $|f|+|f'|$.
- אם בוחרים מסלול כלשהו π ב-G_f - מסלול מ-s ל-t - אפשר להזרים לאורכו זרימה f' שהיא בגודל הקיבולת המינימלית לאורך המסלול.

היום נראה אלגוריתם לחישוב זרימה מקסימלית, כאשר מספר החזרות של תיקון הזרימה בעזרת מסלול בגרף השארית חסום ע"י פונקציה פולינומיאלית של מספר הקודקודים והצלעות בגרף.

האלגוריתם של Edmonds-Karp

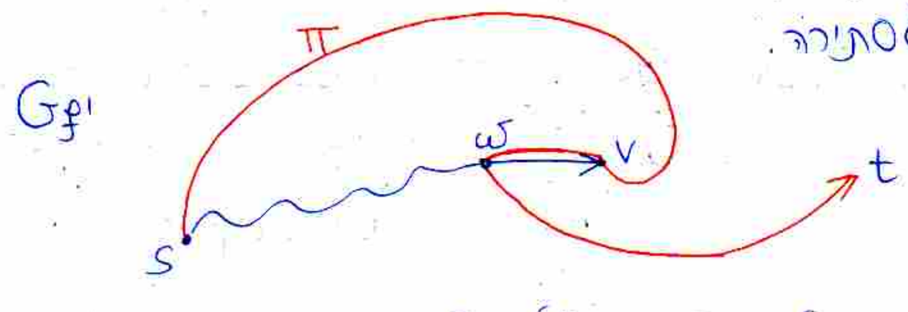
ברמה העקרונית האלגוריתם הזה זהה לאלגוריתם שאפיינו קודם רק שכאן יש דרך מיוחדת לבחור מסלול מתקן - כאשר בוחרים מסלול מתקן בגרף השאריות נבחר מסלול באורך קצר ביותר (מבחינת מספר הקשתות במסלול) מ-s ל-t.

לצורך הוכחת חסם של מספר האיטרציות ב-E-K:
בהינתן זרימה f בגרף G, נתבונן בגרף השאריות G_f ונגדיר מרחק

$$\delta_f(v) = \begin{cases} \text{אורך המסלול הקצר ביותר מ-}s\text{ ל-}v\text{ ב-}G_f & \text{אם יש מסלול קצר ביותר} \\ \infty & \text{אחרת} \end{cases}$$

<u>למה</u>: אם f' זרימה המתקבלת בתהליך לאחר זרימה נתונה f אזי $\delta_f(v) \ge \delta_{f'}(v)$

<u>הוכחה</u>: די לטפל במקרה ש-f' מתקבלת מ-f בעזרת מסלול מתקן יחיד (שהרי אם זה נכון לזרימות עוקבות בוודאי זה נכון לזרימות שבאות אח"כ)
תהי f' זרימה המתקבלת ב-E-K ועבורה קיים קודקוד v שעבורו הטענה איננה נכונה. נבחר זרימה f' ראשונה שבה יש דוגמה נגדית והקודקוד v שעבורו $\delta_{f'}(v)$ מינימלי רק ש-v דוגמה נגדית. כלומר, אנחנו מניחים בשלילה ש-$\delta_f(v) > \delta_{f'}(v)$.
מיד נגיע לסתירה.



המסלול מ-s ל-v הוא מסלול קצר ביותר ב-$G_{f'}$. בפרט, אם $(w,v) \in E_{f'}$ היא הצלע האחרונה במסלול אזי המרחק $\delta_{f'}(w) = \delta_{f'}(v) - 1$ (**)
אבל מהמינימליות של $\delta_{f'}(v)$ נובע ש-$\delta_f(w) \le \delta_{f'}(w)$ (*)

נסמן ש- $(w,v) \notin E_f$ (וזה יראה שהמסלול המתקן
עבר כמו הקו הרתים בעמוד הקודם).
אחרת, כלומר אם $(w,v) \in E_f$ אז $\delta_f(v) \le \delta_f(w)+1$
(זה פשוט נכון לכל גרף כי בשביל להגיע מ-s ל-v אפשר
ללכת מ-s ל-w ואז עוד צלע). אבל לפי (*) נקבל

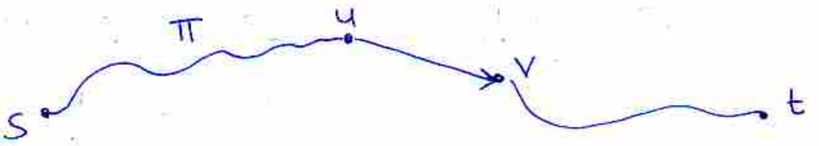
$$\delta_f(v) \le \delta_{f'}(w)+1 = \delta_{f'}(v)$$

וזה בסתירה להנחה שמתקיים אי שוויון חזק הפוך.
⇐ באמת $(w,v) \notin E_f$. אבל $(w,v) \in E_{f'}$ ולכן חייב
להיות שהמסלול המתקן π ב- E_f עבר את הצלע (v,w).
לכן $\delta_f(v) = \delta_f(w)-1 \overset{(*)}{\le} \delta_{f'}(w)-1 \overset{(**)}{=} \delta_{f'}(v)-2$
מה יצא? הנחנו ש- $\delta_{f'}(v) < \delta_f(v)$ וקיבלנו $\delta_f(v) \ge \delta_f(v)+2$
וזו סתירה. ⇐ הוכחנו את הלמה.

<u>למה:</u> מספר האיטרציות ב- E-K הוא $O(|V|\cdot|E|)$

<u>מסקנה:</u> כל צעד באלגוריתם E-K דורש $O(E)$ פעולות
לכן בסה"כ האלגוריתם דורש $O(E^2 \cdot V)$ פעולות.
ידוע ש- $E \le V^2$ ולכן סה"כ נקבל $O(n^5)$ עבור גרף
עם n קודקודים.

<u>הוכחת הלמה:</u> בכל מסלול מתקן π יש צלע (u,v) שעבורה קיבולת
השארית $C_f(\pi) = C_f(u,v)$. כלומר קיבולת השארית של הצלע היא
המינימלית ב- π. אנו מזרימים לאורך π כמות נוספת בגודל $C_f(\pi)$
ולכן בזרימה f' המתקבלת לאחר התיקון $(u,v) \notin E_{f'}$
צלע (u,v) כזאת תיקרא צלע קריטית ב- π.



π מסלול קצר ביותר מ-s ל-t אז $\delta_f(v) = \delta_f(u)+1$.

שאלה: באלגוריתם E-K כמה פעמים צלע מכוונת (u,v) יכולה להיות צלע קריטית?

תשובה: לכל היותר $\frac{|V|}{2}$ פעמים

הוכחה: אם (u,v) קריטית לפי זרימה f אזי לאחר תיקון הזרימה הצלע (u,v) נעלמה וכדי להיות קריטית שוב, היא הייתה חייבת להופיע וזה יכול להיות רק אם בזרימה מאוחרת יותר ומסלול מתקן מחזיר מכיוון ההפוך (v,u).

$$\delta_{f'}(u) = \delta_{f'}(v) + 1 \ge \delta_f(v) + 1 = \delta_f(u) + 2$$

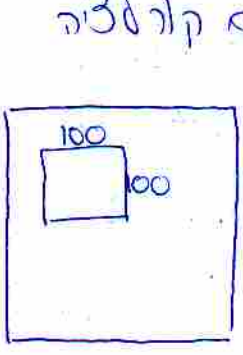
כיוון שגם $\delta_f(v)$ אינו אינסופי אז $|V|-1 \ge \delta_f(v)$.

עכשיו נחזור להוכחת הלמה: בכל איטרציה יש לפחות צלע קריטית אחת ולכן מספר האיטרציות הוא לפחות $|E| \cdot \frac{|V|}{2}$.

הקדמה לטרנספורם פורייה

נניח שיש לנו תמונה ואנחנו רוצים לבחון את אופיעה של תבנית מסויימת. אז יש מטריצה גדולה עם מספרים (נניח שהתמונה ב-grayscale) ותבנית שהיא מטריצה קטנה יותר. אם אנחנו מחפשים מופע מדוייק שלה, אז פשוט אפשר לעבור על כל המטריצה הגדולה ולבדוק. אבל אנחנו מחפשים משהו קרוב - כלומר הופעה של התבנית עם רעש מסוים.

ההנחה היא שבמקום שבו לא מופיעה התבנית אין שום קורלציה בינו לבין התבנית.



אז מסתכלים על התבנית ועל הקטע כווקטורים ועושים ביניהם מכפלה פנימית. אם יש התאמה נקבל מספר גדול. אבל אם אין שום קשר, המכפלה תצא קטנה מאוד.

אבל לעשות מכפלה פנימית המון פעמים (כלומר לעבור על כל התמונה) זה כבד ממש.

אנחנו נדבר על איך אפשר לייעל את העניין...

(18) 20.11.06
אלגו'

דוגמה: יש קבועים $a_0\ a_1 \dots a_n$

וסדרה $b_0\ b_1\ b_2 \dots b_n\ b_{n+1} \dots$

רוצים לחשב את $C_0 = a_0b_0 + a_1b_1 + \dots + a_nb_n$

$C_1 = a_0b_1 + a_1b_2 + \dots + a_nb_{n+1}$

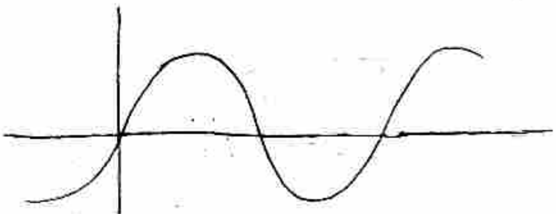
$\vdots$

$C_K = a_0b_K + a_1b_{K+1} + \dots + a_nb_{n+K}$

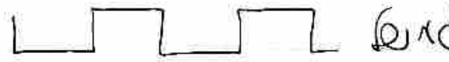
אם מדובר במספרים א-{1,-1} ובאחד מה C_i-ים יש קורלציה בין ה-a-ים ל-b-ים נקבל שהסכום הזה n או לפחות גבוה. ואילו אם אין שום קשר בטח נקבל סכום קטן.

הדבר הזה טוב למשל בשביל להחליק סדרה של נתונים. עושים קונבולוציה עם חלון זה שנראה כך .

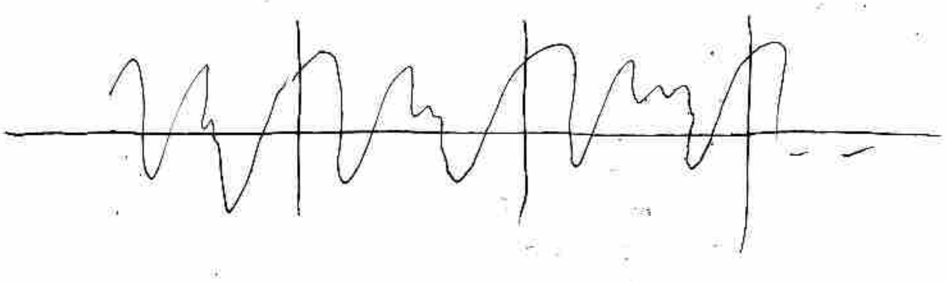
דוגמה: גלי קול אופייניים ע"י תדר. למשל לאות הסיגנלים הם סינוסואידליים נקיים.



ויש גם אינן סיגנלים מחזוריים והם לא בהכרח סינוסואידליים.

למשל  או

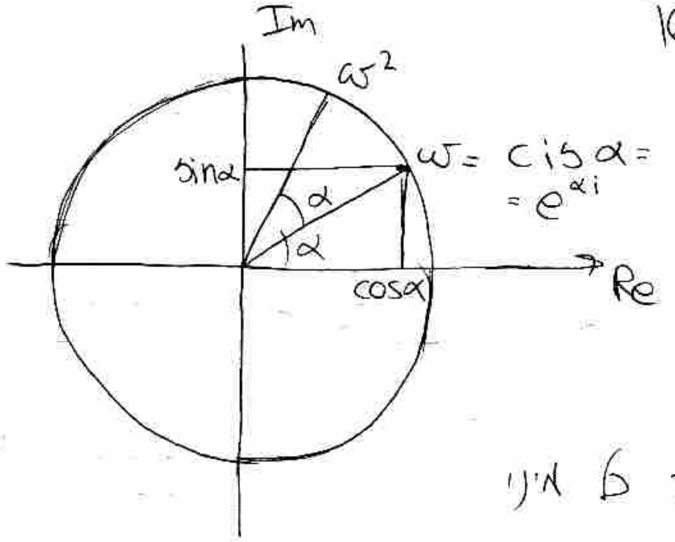
ויכולים להיות דברים מוזרים כמו



קלט: נניח שיש לנו וקטור באורך $N = 1024$. זה וקטור במרחב ה-N מימדי מעל המרוכבים...

את הוקטור הזה אפשר להסתכל כסכום משוקלל של פונקציות δ - זה הבסיס הסטנדרטי.

ומהבסיס הזה כולקת האמפליטודה (עוצמה) של הסיגנל שלנו.
אבל אנחנו רוצים לבחור בסיס שממנו יצעקו התדרים
שמרכיבים את הסיגנל. זה יהיה בסיס שמורכב
מוקטורים מחזוריים ומאוד קל לבחור אותם במרוכבים.



נחלק את מעגל היחידה ל-16
גזרות ונסתכל על
המספרים $\omega^1,\dots,\omega^{16}$
אז אפשר לקחת
וקטור שמורכב מחזרות
של $\omega^1,\dots,\omega^{16}$.
באופן כזה אפשר לבחור 16 מיני
תדרים.

באופן כללי, נבחר בסיס למרחב הווקטורי $\mathbb{C}^N$:
נסמן $\omega = e^{\frac{2\pi i}{N}}$ ואז נבחר בסיס ניקח את הווקטורים

$$\alpha_k = (\omega^{k\cdot 0}, \omega^k, \omega^{2k}, \dots, \omega^{(N-1)k})$$

עבור $k=0,\dots,N-1$.
קל לבדוק שזה באמת בסיס. ואפילו אורתוגונלי.

במקרה ש- $N=1024$ נקבל:

$k=0$ $(1, 1, \dots, 1)$

$\vdots$

$K=64$ $(1, \omega^{64}, \omega^{128}, \dots, 1, \dots)$

מקום 16 (כי $64\cdot 16 = 1024$)

<u>טענה</u>: הבסיס $\{\alpha_k\}_{k=0}^{N-1}$ הוא בסיס אורתוגונלי למרחב $\mathbb{C}^N$

ראשית נזכיר רק מה המכפלה הפנימית שאנחנו משתמשים בה:

$$\alpha = (a_0 \dots a_{N-1})$$
$$\beta = (b_0 \dots b_{N-1})$$
$$\langle \alpha, \beta \rangle = \sum_{l=0}^{N-1} a_l \overline{b_l}$$

בלינארית הוכחנו שזו אכן מ"פ כשרה.

<u>למה</u>: יהי F שדה ו- $\omega \in F$ מקיים $\omega^N = 1$ אבל $\omega \neq 1$.
אזי $\sum_{l=0}^{N-1} \omega^l = 0$

<u>הוכחה</u>: כמו שעשינו בתרגול בלינארית שנה שעברה, נכפיל את הסכום ב- ω ואז נקבל:

$$\omega\left(\sum_{l=0}^{N-1}\omega^l\right) = \sum_{l=0}^{N-1}\omega^{l+1} = \sum_{l=1}^{N}\omega^l = 1+\sum_{l=1}^{N-1}\omega^l = \sum_{l=0}^{N-1}\omega^l$$

אם נסמן $X=\sum_{l=0}^{N-1}\omega^l$ אז בעצם קיבלנו ש- $\omega X = X$
כלומר $(\omega-1)X=0$ ⇐ מכיוון ש- $\omega\neq 0$ נקבל $X=0$.

<u>הגדרה</u>: $\omega\in F$ נקרא שורש יחידה פרימיטיבי מסדר N
אם $\omega^N=1$ אבל לכל $0<j<N$ $\omega^j\neq 1$

<u>למה</u>: יהי F שדה ! $\omega\in F$ שורש יחידה פרימיטיבי מסדר N. אזי:

$$\sum_{l=0}^{N-1}\omega^{kl} = \begin{cases} N & \text{N מחלק את k} \\ 0 & \text{אחרת}\end{cases}$$

<u>הוכחה</u>: ברור שאם N מחלק את K בפשוט מחברים 1 N פעמים וכמובן יוצא N.
אחרת נרשום $k=qN+r$ כאשר $0<r<N$ ואז

$$\omega^k = \omega^{qN+r} = \underbrace{(\omega^N)^q}_{1}\omega^r = \omega^r$$

מאחר ש- $r\neq 0$.. נקבל $\omega^r\neq 1$ ואז לפי הלמה הקודמת $\sum_{l=0}^{N-1}(\omega^r)^l=0$, שהרי $(\omega^r)^N=1$. ☺

טענה: הבסיס $\{\alpha_k\}_{k=0}^{N-1}$ הוא בסיס אורתוגונלי.

הוכחה:

$$\langle \alpha_k , \alpha_r \rangle = \sum_{l=0}^{N-1} \omega^{kl}\cdot\overline{(\omega^{rl})} = \sum_{l=0}^{N-1} \omega^{kl}\omega^{-rl} =$$

$$= \sum_{l=0}^{N-1} \omega^{(k-r)\cdot l}$$

אבל ברור ש- $|k-r| \le N$ מתקיים. ולכן לפי הלמה הקודמת:

$$\sum_{l=0}^{N-1} \omega^{(k-r)l} = \begin{cases} 0 & k\neq r \\ N & k=r \end{cases}$$

מסקנה: עכשיו אנחנו באמת יודעים שהיה מוצדק להגיד ש- $\{\alpha_k\}$ בסיס. כי קיבלנו קבוצה של N וקטורים אורתוגונליים ב- $\mathbb{C}^N$ $\Leftarrow$ הם בלתי תלויים $\Leftarrow$ הם בסיס.

דרך אחרת להראות שהם בלתי תלויים היא להסתכל על המטריצה:

$$V = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & & \omega^{N-1} \\ \vdots & \omega^2 & (\omega^2)^2 & & (\omega^{N-1})^2 \\ & \vdots & \vdots & & \vdots \\ 1 & \omega^{N-1} & (\omega^2)^{N-1} & & (\omega^{N-1})^{N-1} \end{bmatrix}$$

(העמודות: $\alpha_0, \ldots, \alpha_{N-1}$)

זוהי מטריצת ון-דר-מונדה והדטרמיננטה שלה שונה מאפס משום שאם $0 \le j < k < N$ מתקיים $\omega^j \neq \omega^k$. אחרת אז $\omega^{k-j}=1$ ואז $k=j$ כי ω שורש יחידה פרימיטיבי מסדר N. ואז, בגלל שהדטרמיננטה שונה מאפס הוקטורים בלתי תלויים.

בדרך שלנו אבל הוכחנו שהם אורתוגונליים שזה חשוב. אפשר להפוך אותו לאורתונורמלי ע"י $\alpha_k' = \frac{1}{\sqrt{N}}\alpha_k$.

אם $\alpha_0', \dots, \alpha_{N-1}'$ בסיס אורתונורמלי ו- β -
וקטור כך שהוא $\beta = b_0\alpha_0' + \dots + b_{N-1}\alpha_{N-1}'$, כלומר

$$\beta = \sum_{\ell=0}^{N-1} b_\ell \alpha_\ell'$$

היינו רוצים למצוא את המקדמים וזה מתקבל בקלות מהחומר של שנה שעברה:

$$\langle \beta, \alpha_k' \rangle = \sum_{\ell=0}^{N-1} b_\ell \langle \alpha_\ell', \alpha_k' \rangle = b_k \langle \alpha_k', \alpha_k' \rangle = b_k$$

חישוב מקדם בודד $O(N)$ וסה"כ $O(N^2)$. אנחנו רוצים לעשות את זה יותר מהר!

נניח ש- N הוא חזקה של 2 - $N=2^\ell$
בהינתן סדרה $(a_0, \dots, a_{N-1})$ מבסיס הסטנדרטי (כלומר זו סדרת הנתונים שקיבלנו מההתחלה).
נגדיר פולינום $f(x) = a_0 + a_1x + \dots + a_{N-1}x^{N-1}$
ואז

$$A\cdot V = \left(f(1), f(\omega), f(\omega^2), \dots, f(\omega^{N-1})\right)$$

ω הוא שורש יחידה פרימיטיבי מסדר N.

$$\omega^N = 1$$

$(\omega^2)^{\frac{N}{2}} = 1$ ו- ω^2 שורש יחידה פרימיטיבי מסדר $\frac{N}{2}$
וכן הלאה...

אז נרשום מחדש את המקדמים הזוגיים והאי-זוגיים:

$$f(x) = a_0 + a_2x^2 + a_4x^4 + \dots + x(a_1 + a_3x^2 + a_5x^4 + \dots)$$

נגדיר:

$$f_E = a_0 + a_2x + a_4x^2 + \dots + a_{N-2}x^{\frac{N}{2}-1}$$
$$f_O = a_1 + a_3x + a_5x^2 + \dots + a_{N-1}x^{\frac{N}{2}-1}$$

ואז

$$f(x) = f_E(x^2) + x f_O(x^2)$$
$$\Rightarrow f(\omega^k) = f_E((\omega^2)^k) + \omega^k f_O((\omega^2)^k)$$

אבל ω^2 שורש יחידה פרימיטיבי מסדר $\frac{N}{2}$ אז הכל עד $k=\frac{N}{2}$
זה יחזור על עצמו. כלומר פירקנו את הבעיה עם

נוסחת רקורסיה $T(n)=2T(\frac{n}{2})+O(n)$

הפתרון של זה הוא $O(n\log n)$

בפעם הבאה (תזכור על זה בפרוטרוט) ונבדק עוד
כל מיני חסכונות.

נדבר גם על הטרנספורם ההפוך ואיך עושים
עם זה עיבוד אות וכו'...

(21) 22.11.06
אלגו'

טרנספורם פורייה (Fourier)

רוצים לחשב קונבולוציה של סדרת קלט $b_0, b_1, \dots, b_n, b_{n+1}, \dots$
בסדרה הקבועה $a_0, a_1, \dots, a_{n-1}$:

$$c_0 = a_0 b_0 + \dots + a_{n-1} b_{n-1}$$
$$\vdots$$
$$c_k = a_0 b_k + \dots + a_{n-1} b_{n+k-1}$$

הסדרה (c_k) היא סדרת הקונבולוציה של הסדרות (a_n) ו-(b_n)!

יהי F שדה ויהי ω שורש יחידה פרימיטיבי מסדר n, כלומר
$\omega^n = 1$ אבל $\omega^k \neq 1$ $1 \le k < n$.

בהינתן סדרה באורך n $\bar{a} = (a_0, \dots, a_n)$
טרנספורם פורייה (דיסקרטי) של הסדרה $\bar{a}$ מוגדר כך:
$$\alpha_k = \sum_{l=0}^{n-1} a_l \omega^{kl} \qquad k = 0, \dots, n-1$$

בכתיב מטריציוני זה בעצם

$$(a_0 \dots a_{n-1}) \begin{pmatrix} 1 & 1 & \cdots & 1 & \cdots & 1 \\ 1 & \omega & & \vdots & & \omega^{n-1} \\ \vdots & & & \omega^{kl} & & \vdots \\ 1 & \omega^{2(n-1)} & & & & \omega^{(n-1)^2} \end{pmatrix} = (\alpha_0, \alpha_1, \dots, \alpha_{n-1})$$

(החץ k מצביע על העמודה, החץ l על השורה.)

אם נסמן את העמודה ה-k ב-V_k אזי סדרת הווקטורים $\{\frac{1}{\sqrt{n}} V_k\}_{k=0}^{n-1}$ ($= V_k'$)
היא בסיס אורתונורמלי למרחב F^n.
המספרים האלה הם מקדמי פורייה ומתקבל $\bar{a} = \frac{1}{\sqrt{n}} \sum \alpha_k V_k'$
בעצם אנו מתעניינים בעוצמה של תדר מסוים בגלים, ואז
α_k היא העוצמה של התדר של V_k.

בד"כ מדברים על שדה המרוכבים $\mathbb{C}$ ומשתמשים בשורש
היחידה $\omega = e^{\frac{2\pi}{n} i}$ ...
יתכן ויהיו מספר שורשי יחידה פרימיטיביים מסדר n, אז בד"כ בוחרים
שורש יחידה מסוים כדי להיות קונסיסטנטיים ולדבוק בו.

חישוב מהיר של טרנספורם פוריה דיסקרטי עבור $n=2^9$:

נשים ❤: אם ω שורש יחידה פרימיטיבי מסדר n

אזי: (I) ω^2 שורש יחידה פרימיטיבי מסדר $\frac{n}{2}$

(II) $\omega^{\frac{n}{2}}=-1$ שהרי פתרון שונה מ-1 למשוואה $X^2=1$ יכול להיות רק -1

(בגלל הפרימיטיביות לא יכול להיות $\omega^{\frac{n}{2}}=1$)

(III) נסמן $\omega_1=\omega^2$. אזי $\{\omega_1^k\}_{k=0}^{n-1}$ נותן פעמיים את כל התלכות של ω_1 - $1,\omega_1,\omega_1^2,\dots,\omega_1^{\frac{n}{2}-1},1,\omega_1,\dots,\omega^{\frac{n}{2}-1}$

שהרי $\omega_1^{\frac{n}{2}+k}=(\omega^2)^{\frac{n}{2}+k}=\omega^n\cdot\omega^{2k}=\omega_1^k$

(IV) אם נסמן $A(X)=a_0+a_1X+\dots+a_{n-1}X^{n-1}$

אזי לכל k $\alpha_k=\sum_{\ell=0}^{n}a_\ell(\omega^k)^\ell=A(\omega^k)$

ומשתמש בכל התובנות האלה אמשיך לעשות לבעיה הפרד ומשול.

נפריד את הפולינום $A(x)$ לחלק זוגי וחלק אי־זוגי:

$A_E(x)=a_0+a_2x^2+a_4x^4+\dots+a_{n-2}x^{n-2}$ (כי n זוגי)

$A_O(x)=a_1x+a_3x^3+\dots+a_{n-1}x^{n-1}=$

$=x(a_1+a_3x^2+\dots+a_{n-1}x^{n-2})=x\bar{A}_O(x^2)$

כאשר $\bar{A}_O(t)=a_1+a_3t+\dots+a_{n-1}t^{\frac{n}{2}-1}$

ובאותו אופן $A_E=\bar{A}_E(x^2)$ כאשר $\bar{A}_E(t)=a_0+a_2t+\dots+a_{n-2}t^{\frac{n}{2}-1}$

עכשיו קיבלנו ש- $A(x)=\bar{A}_E(x^2)+x\bar{A}_O(x^2)$

$\Leftarrow$. $k=0,\dots,n-1$ $\alpha_k=A(\omega^k)=\bar{A}_E(\omega_1^k)+\omega^k\bar{A}_O(\omega_1^k)$.

עבור $k=0,\dots,\frac{n}{2}-1$ נקבל פעמיים בעיית חישוב פולינומים ממעלה $\frac{n}{2}-1$

בכל שורשי היחידה מסדר $\frac{n}{2}$:

עבור $\frac{n}{2},\frac{n}{2}+1,\dots,\frac{n}{2}+k$ מתקבל $A(\omega^{\frac{n}{2}+k})=\bar{A}_E(\omega_1^k)-\omega^k\bar{A}_O(\omega_1^k)$

שהרי $\omega_1^{\frac{n}{2}+k}=\omega_1^k$ ו- $\omega^{\frac{n}{2}+k}=-\omega^k$.

עכשיו נספור כמה פעולות נצטרך בשביל לחשב את זה.

נסמן $T(n)$ = סה"כ מספר פעולות אריתמטיות במספרים ממשיים,

ואז חיבור מרוכב זה 2 פעולות וכפל מרוכב זה 6 פעולות

נסמן ב- $S(n)$ חיבורים וב- $M(n)$ כפלים. ואז

$$S(n) = 2S(\tfrac{n}{2}) + n \Rightarrow S(n) = n\log n$$

בשלב חיבור ב הפולינומים.

בשלב הכפל אנחנו (צריכים) חזקות של שורש היחידה הפרימיטיבי, אבל לא כל הזמן חוזר על עצמו ולכן פשוט נכין אותם מראש ולא נצטרך לחשב אותם כל פעם מחדש.

$$M(n) = 2M(\tfrac{n}{2}) + \tfrac{n}{2} \Rightarrow M(n) = \tfrac{n}{2}\log n$$

יש את החלק שמחברים והחלק שמחסרים. מספיק לחשב רק פעם אחת ולדעת מתי לכפול ומתי לחבר.

וסה"כ ב פעולות חיבור ו-6 פעולות כפל נותן לנו חסם $5n\log n$ פעולות

ומה זה הבעיה ההפוכה? - מישהו נותן לנו טרנספורם פוריה של אות ואנחנו רוצים לחשב ממנו את האות עצמו.

בהינתן $\alpha_0,\dots,\alpha_{n-1}$ רוצים לחשב את $a_0,\dots,a_{n-1}$

כמו קודם, מתקיים הקשר

$$(a_0,\dots,a_{n-1})\underbrace{\begin{pmatrix} 1 & 1 & \cdots & 1 & \cdots & 1 \\ 1 & \omega & & & & \\ \vdots & & & \omega^{k\ell} & & \\ 1 & \omega^{(n-1)} & & & & \end{pmatrix}}_{V} = (\alpha_0,\dots,\alpha_{n-1})$$

כלומר $\bar{a}V = \bar{\alpha}$. $\Leftarrow$ $\bar{\alpha}V^{-1} = \bar{a}$

אבל V היא מטריצה אורתוגונלית ולכן $V^{-1} = \frac{1}{n}\bar{V}^t = \frac{1}{n}[\omega^{-st}]$

ולכן חישוב הכפלת וקטור ב- $[\omega^{-st}]$ הוא טרנספורם פוריה כאשר בוחרים את ω^{-1} השורש יחידה פרימיטיבי.

חישוב טרנספורם הפוך: מחשבים טרנספורם פוריה עם ω^{-1} השורש יחידה פרימיטיבי ואת הוקטור התוצאה מחלקים ב- n.

נראה שבאמת: $V^{-1} = \frac{1}{n}\left[\omega^{-st}\right]$

$$\ell(\omega^{\ell k})(\omega^{-st}) = \ell(-^{t}) = \begin{pmatrix} n & 0 \\ 0 & n \end{pmatrix} = nI$$

$$\sum_{k=0}^{n-1}\omega^{\ell k}\omega^{-kt} = \sum_{k=0}^{n-1}\omega^{(\ell-t)k} = \begin{cases} 0 & \ell \neq t \\ n & \ell = t \end{cases}$$

שימושים של טרנספורם פורייה

כפל פולינומים: בהינתן פולינומים $A(x) = a_0 + a_1x + \dots + a_{n-1}x^{n-1}$

$B(x) = b_0 + b_1x + \dots + b_{n-1}x^{n-1}$

נרצה לחשב את המכפלה:

$$C(x) = A(x)B(x) = c_0 + c_1x + \dots + c_{2n-2}x^{2n-2} + \overset{=0}{c_{2n-1}}x^{2n-1}$$

$$c_k = \sum_{\ell=0}^{k} a_\ell b_{k-\ell} \qquad \text{לכל } k = 0, \dots, 2n-1$$

באופן ישיר יש פה $O(n^2)$ פעולות.

אנחנו נראה איך אפשר לעשות את זה ב- $O(n\log n)$

נזכיר שפולינום $f(x) \in F[X]$ נוכל להציג בעזרת המקדמים $f(x) = f_0 + f_1x + \dots + f_{d-1}x^{d-1}$ כאשר $\deg f = d-1$.

אבל נוכל גם להציג את f ע"י d נקודות שונות ב- F - בהינתן $e_0, \dots, e_{d-1} \in F$ יש פולינום יחיד ממעלה לכל היותר $d-1$ שעובר בנקודות $f(e_0), \dots, f(e_{d-1})$ (הוכחנו את זה בלינארית, אבל זה גם ברור כי אם היו שניים כאלה אז ההפרש ביניהם היה פולינום ממעלה גדולה מאפס והיו לו יותר מ- $d-1$ שורשים.)

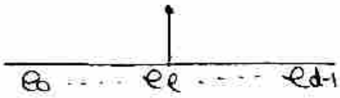
<u>טענה</u>: יהא F שדה. בהינתן d נקודות שונות $e_0, \dots, e_{d-1}$ וערכים $y_0, \dots, y_{d-1} \in F$ קיים פולינום יחיד $f(x)$ עם $\deg f \le d-1$ כך ש- $f(e_\ell) = y_\ell$ לכל $0 \le \ell \le d-1$.

<u>הוכחה</u>: יחידות - אם $g(x)$ פולינום נוסף אזי לפולינום $h(x) = f(x) - g(x)$ יש d שורשים שונים אבל $\deg h \le d-1 \Leftarrow h \equiv 0$.

קיום - כאן אינטרפולציה. הפולינום $\prod_{\substack{k\neq\ell \\ 0\le k\le d-1}}(x-e_k)$ מתאפס בכל e_k ל- $k \neq \ell$ ושונה מאפס בנקודה e_ℓ, אבל אנחנו רוצים ש"יצא" לנו הערך הנכון,

(23)

אז הפולינום $f_\ell(x)=\frac{\prod_{k\neq \ell}(x-e_k)}{\prod_{k\neq \ell}(e_\ell-e_k)}$ מקבל 0 בכל $e_\ell \neq x$ ו-1 ב- $x=e_\ell$



לכל אחד מהפולינומים $f_\ell(x)$ (ראה רב.)

ואז $f(x)=\sum_{\ell=0}^{d-1} f_\ell(x)$ מקיים את הדרוש.

עכשיו נחזור למטרה המקורית שלנו - להכפיל פולינומים.

$$A(x) = a_0 + a_1 x + \dots + a_{n-1} x^{n-1}$$

$$B(x) = b_0 + b_1 x + \dots + b_{n-1} x^{n-1}$$

$$C(x) = A(x)B(x) = c_0 + c_1 x + \dots + c_{2n-1} x^{2n-1}$$

כדי לדעת את $C(x)$ כולו מספיק לחשב את $C(x)$ ב- $2n$ נקודות שונות. אם נניח ש-n חזקה של 2 פשוט נחשב את הטרנספורם פורייה של $\bar{a}$ ו-$\bar{b}$ עם $2n$ שורשי יחידה (אנחנו מניחים שיש $2n$ שורשי יחידה)

$$A(x) \xrightarrow{FFT} \bar{\alpha} = (\alpha_0, \dots, \alpha_{2n-1})$$

$$B(x) \xrightarrow{FFT} \bar{\beta} = (\beta_0, \dots, \beta_{2n-1})$$

הטענה היא ש $FFT(A\cdot B)=(\alpha_0\beta_0, \dots, \alpha_{2n-1}\beta_{2n-1})$. אז אפשר לעשות טרנספורם פורייה הפוך מסדר $2n$ ולקבל את $C(x)=A(x)B(x)$

סה"כ היה לנו $O(n\log n)+O(n\log n)+O(n\log n) = O(n\log n)$ פעולות.

אתר מגניב עם הדגמות של גלים:

www.jhu.edu/~signals/listen-new/listen-newindex.htm

חישוב התמרת פורייה (או ההתמרה ההפוכה) אורך n דורש לכל היותר $5n\log n$ פעולות אריתמטיות במספרים ממשיים.

בהינתן פולינומים
$$A(x)=a_0+a_1x+\dots+a_{k-1}x^{k-1}$$
$$B(x)=b_0+b_1x+\dots+b_{m-1}x^{m-1}$$

אזי ניתן לחשב את מקדמי פולינום המכפלה באופן הבא:

בוחרים n שהוא חזקה של 2 ו- $k+m\le n-1$.

נסמן $A=(\underbrace{a_0\, a_1 \dots a_{n-1}\, 0\dots 0}_{n\ \text{איברים}})$ ובאותו אופן B ו- AB

$$FFT_n(A)=(A(\omega^0),\dots,A(\omega^{n-1}))$$
$$FFT_n(B)=(B(\omega^0),\dots,B(\omega^{n-1}))$$

$$FFT_n(A\cdot B)=FFT_n(A)\otimes FFT_n(B)$$

($\otimes$ – מכפלה איבר-איבר)

$$\Rightarrow\quad AB=FFT_n^{-1}(FFT_n(A)\otimes FFT_n(B))$$

$\Leftarrow$ אפשר להכפיל פולינומים ב- $15n\log n$ פעולות אריתמטיות בממשיים

(הערה: לא הוספנו כאן את מכפלת התמרות האיברים וגם בסוף יש לחלק ב-n, אבל כשאמרנו שהתמרת פורייה לוקחת $5n\log n$ זו הייתה הגזמה, כך שבסוף הכל מתקזז)

קונבולוציה - נניח ש-n חזקה של 2 ונתונה n-יה
$$\mathbb{C}\supseteq A=(a_0,a_1,\dots,a_{n-1})$$
וסדרה אינסופית $\mathbb{R}\supseteq B=(b_0,b_1,\dots,b_{n-1},b_n,\dots)$

ורוצים לחשב את הקונבולוציה שלהם $(C_k)_{k=0}^{\infty}$ $\quad C_k=\sum_{\ell=0}^{n-1}a_\ell b_{k+\ell}$

חישוב C_k דורש $O(n)$ פעולות.

מיד קופץ לנו לעין כפל פולינומים. למה? - כי זה ממש דומה! ☺

אם רוצים לחשב את $c_0, \dots, c_{n-1}$ (נתכל ש)

$$\bar{A}(x) = a_{n-1} + a_{n-2}x + \dots + a_1 x^{n-2} + a_0 x^{n-1}$$
$$B(x) = b_0 + b_1 x + \dots + b_{n-1}x^{n-1} + b_n x^n + \dots + b_{2n-1}x^{2n-1}$$

$$\Rightarrow D(x) = \bar{A}(x)B(x) = _ + _\,x + \dots + \overset{?}{_}\,x^{n-1} + \dots + _\,x^{2n-1} + \dots$$

קל לראות שזה c_0 (המקדם של x^{n-1})

קל לראות שזה c_{n-1} (המקדם של x^{2n-1})

באותו אופן, אם $D(x) = \sum_{i=0}^{3n-2} d_i x^i$ אז לכל $n-1 \le i \le 2n-1$ מתקיים $d_i = c_{i-n+1}$

$\Leftarrow$ אם נעשה התמרה באורך $4n$ (כלומר בחזקה של 2) ובאמצעותה נכפול את $\bar{A}(x)$ ב- $B(x)$ אז נקבל בבת אחת n איברים של הקונבולוציה במחיר $15 \times 4n \cdot \log 4n$.
בממוצע זה אומר של c_k עלה לנו $15 \cdot 4 \log 4n = 60 \log 4n$.

<u>שיפורים</u>: אם בוחרים את $B(x)$ כפולינום ממעלה $15n-1$

$$B(x) = b_0 + b_1 x + \dots + b_{15n-1}x^{15n-1}$$
$$\bar{A}(x) = a_{n-1} + a_{n-2}x + \dots + a_0 x^{n-1}$$

אזי סכום המעלות הוא כ- $16n$ - חזקה של 2.
n האיברים הראשונים ! n- האחרונים אינם מעניינים אותנו
אז בבת אחת אנחנו מקבלים $14n$ איברים מהקונבולוציה
ואז המחיר הממוצע יוצא $\frac{15 \cdot 16n \log 16n}{14n}$.

מה קורה כאשר בוחרים את $B(x)$ מדרגה $16n-1$ ~~עדיין נעבוד~~ בהתמרה מסדר $16n$. לא עובד כי $A(x)B(x)$ הוא לא מדרגה מוגבלת ב- $16n$.

$$\bar{A}(x) = a_{n-1} + a_{n-2}x + \dots + a_0 x^{n-1}$$
$$B(x) = b_0 + b_1 x + \dots + b_{n-1}x^{n+1} + \dots + b_{16n-1}x^{16n-1}$$
$$\bar{A}(x)B(x) = d_0 + \dots + d_{n-2}x^{n-2} + c_0 x^{n-1} + \dots + c_{15n-1}x^{16n-1} + d_{16n}x^{16n} + \dots + d_{17n-2}x^{17n-2} + 0 \cdot x^{17n-1}$$

אך אם עושים התמרה בגודל $16n$ זה לא מספיק.

אבל אפשר לרשום $\bar{A}(x)B(x) = q(x)(x^n-1) + r(x)$

כאשר $\deg r(x) \le n-1$.

אחרי שלב ההכפלה התמרה ההפוכה תיתן את מקדמי הפולינום $r(x)$ כאשר $r(x)$ הוא השארית של חלוקת הפולינום $\bar{A}(x)B(x)$ ב- $(x^{16n}-1)$.

$\deg r(x) \le 16n-1$ ואז לפחות $n-1$ המקדמים הראשונים כי המקדמים האחרים הם איברים בסדרת הקונבולוציה - סה"כ $15n$ איברים.

קיבלנו מחיר ממוצע $\frac{15\times 16n \log 16n}{15n} \sim 15\log n$

הסדרה A היא קבועה ולכן ניתן לחשב את התמרת פורייה שלה פעם אחת ולתמיד ואז המחיר ליחידה יורד ל- $2\times 5\log n$.

נראה איך אפשר לצמצם את זה עוד.

$$B = b_0\, b_1 \,\dots\, b_{16n-1} \Rightarrow c_0, \dots, c_{15n-1}$$

$$b_{15n}\, b_{15n+1} \,\dots\, b_{31n-1} \Rightarrow c_{15n}, \dots, c_{30n-1}$$

סדרת האיברים b הוא סדרה ממשית. נסתכל על

$$\bar{B} = b_0 + i\, b_{15n},\ b_1 + i\, b_{15n+1},\ \dots$$

כאשר הסדרה A גם כן ממשית

$$c_k = \sum_{\ell=0}^{n-1} a_\ell \bar{b}_{\ell+k} = \underbrace{\sum_{\ell=0}^{n-1} a_\ell b_{\ell+k}}_{c_k} + i \underbrace{\sum_{\ell=0}^{n-1} a_\ell b_{15n+k+\ell}}_{c_{15n+k}}$$

$\Leftarrow$ עבור סדרות ממשיות מחיר היחידה מתקרב ל- $5\log n + c$.

חילוק עם שארית של פולינומים ממעלה $O(n)$ דורש $O(n^2)$ פעולות בחילוק ארוך. בהינתן אלגוריתם לכפל פולינומים ב- $O(n\log n)$ פעולות ניתן לתאר אלגוריתם לחילוק עם שארית שדורש גם כן $O(n\log n)$ פעולות. אבל על זה נדלג.

חישוב פולינום $f(x)$ מעלה n ב-n נקודות שונות:

$$f(x) = a_0 + a_1 x + \dots + a_n x^n$$

<u>טענה:</u> $f(b)$ שווה לשארית מחלוקת $f(x)$ בפולינום $x-b$.

<u>הוכחה:</u> $f(x) = q(x)(x-b) + c \Leftarrow f(b) = q(b)\cdot 0 + c$

$\Leftarrow f(b) = c$. (:

צעד הכנה: נחשב את מקדמי הפולינום $g(x) = \prod_{i=1}^{n}(x-b_i)$ כאשר $b_1, \dots, b_n$ הן הנקודות בהן אנו רוצים לחשב את $f(x)$. באופן ישיר זה לוקח $O(n^2)$. אבל

$$g(x) = \left[\prod_{i=1}^{n/2}(x-b_i)\right]\left[\prod_{i=n/2+1}^{n}(x-b_i)\right] := g_1(x)\, g_2(x)$$

$$\Rightarrow T(n) = 2T\left(\tfrac{n}{2}\right) + O(n \log n)$$
$$\Rightarrow T(n) = O(n(\log n)^2)$$

עכשיו לחשב את $f(x)$ ב-n נקודות $b_1, \dots, b_n$.
אפשר לחלק פולינומים ב-$O(n \log n)$ ולרשום

$$f(x) = q_1(x) g_1(x) + f_1(x) = q_2(x) g_2(x) + f_2(x)$$

כאשר $\deg f_1 < \frac{n}{2}$, $\deg f_2 < \frac{n}{2}$

עבור $b_1, \dots, b_{\frac{n}{2}}$: $f(b_k) = f_1(b_k)$
ועבור $b_{\frac{n}{2}+1}, \dots, b_n$: $f(b_k) = f_2(b_k)$

$$\Rightarrow M(n) = 2M\left(\tfrac{n}{2}\right) + O(n \log n)$$
$$\Rightarrow M(n) = O(n(\log n)^2)$$

רק שבחישוב צריך תמיד לזכור את ה-g-ים שמחשבים כי הנחנו כאן שהם ידועים.

(26) 4/12/06

יום קודם שמה לסמלול התדש

בהינתן נקודות $a_1,\dots,a_n$ אפשר לחשב את מקדמי הפולינום $g(x)=\prod_{i=1}^{n}(x-a_i)$ ב- $O(n\log^2 n)$ פעולות.

תוצאה נוספת. בהינתן פולינום ממעלה n $f(x)=\sum_{i=1}^{n}a_i x^i$ ניתן לחשב את $f(b_1),\dots,f(b_n)$ ב- $O(n\log^2 n)$ פעולות.

בעית חישוב מקדמי פולינום אינטרפולציה

נתונות n נקודות שונות $a_1,\dots,a_n$ וערכים $b_1,\dots,b_n$

קיים פולינום יחיד $f(x)$ כך ש $\deg f\le n-1$!

$f(a_i)=b_i$ לכל $1\le i\le n$

נחשב את מקדמי $f(x)$:

- נחשב את מקדמי $g(x)$ כמו שעשינו בזמן $O(n\log^2 n)$

$$f(x)=\sum_{\ell=1}^{n} b_\ell\cdot\frac{\prod_{i\ne\ell}(x-a_i)}{\prod_{i\ne\ell}(a_\ell-a_i)}$$

נסמן $c_\ell=\frac{b_\ell}{\prod_{i\ne\ell}(a_\ell-a_i)}$ ואז נקבל ש-

$$f(x)=\sum_{\ell=1}^{n}c_\ell\prod_{i\ne\ell}(x-a_i)=\sum_{\ell=1}^{n/2}c_\ell\prod_{i\ne\ell}(x-a_i)+\sum_{\ell=\frac n2+1}^{n}c_\ell\prod_{i\ne\ell}(x-a_i)=$$

$$=\prod_{i=\frac n2+1}^{n}(x-a_i)\sum_{\ell=1}^{n/2}c_\ell\prod_{\substack{i\ne\ell\\1\le i\le\frac n2}}(x-a_i)+\prod_{i=1}^{n/2}(x-a_i)\sum_{\ell=\frac n2+1}^{n}c_\ell\prod_{\substack{i\ne\ell\\\frac n2+1\le i\le n}}(x-a_i)$$

חישובי המכפלות של $g(x)$ הם בעצם $\prod_{i=1}^{n/2}(x-a_i)$, $\prod_{i=\frac n2+1}^{n}(x-a_i)$ ואותם חישבנו קודם. ואז מהינתן c_ℓ לכל ℓ נקבל חישוב $f(x)$ רקורסיבי.

$$T(n)=2T(\tfrac n2)+\underbrace{O(n\log n)}_{\text{בשביל להכפיל את הפולינומים}}=O(n\log^2 n)$$

נותר לנו לחשב את c_ℓ:

$$g(x)=\prod_{i=1}^{n}(x-a_i)=g_0+g_1x+\dots+g_nx^n$$

$$g'(x)=g_1+2g_2x+3g_3x^2+\dots+ng_nx^{n-1}=\sum_{\ell=1}^{n}\prod_{i\ne\ell}(x-a_i)$$

ולכן $g'(a_\ell)=\prod_{i\ne\ell}(a_\ell-a_i)$.

⇐ נחשב את הערך של $g'(x)$ בנקודות $a_1,\dots,a_n$
(ב $O(n\log^2 n)$ צעדים) ואז $c_e = b_e / g'(a_e)$
וזה רק עוד n צעדים.

סה"כ כל התהליך כולו $= O(n\log^2 n) + O(n\log^2 n) + O(n) =$
$= O(n\log^2 n)$

זיהוי תבניות

נתון טקסט ארוך $T = T_1 \dots T_m \dots T_n$ ותבנית $P_1 \dots P_m$
השיטה הפשוטה לזיהוי המיקום של תבנית נתונה באורך m בתוך טקסט באורך n דורשת $O(n\cdot m)$ צעדים

נתון א"ב סופי Σ (פשוט קבוצה של סימנים והם יהיו האותיות)
מילה x בא"ב היא סדרה $X = X_1 \dots X_m$ כאשר $X_i \in \Sigma$
ו- $|X|$ האורך של המילה הוא n.
נעזר בסימן מיוחד ε (שאינו אות בא"ב) והוא מסמן את המילה הריקה שאורכה 0.

דוגמה $\Sigma = \{a,b,c\}$

$X = abcabb \quad |X| = 6$
$y = ca \quad |y| = 2$
$\varepsilon \quad |\varepsilon| = 0$

בהינתן שתי מילים x,y ניתן לשרשר אותן.
אם $X = X_1 \dots X_n$ ו- $Y = Y_1 \dots Y_m$ אז השרשור הוא

$x \cdot y = X_1 \dots X_n Y_1 \dots Y_m$

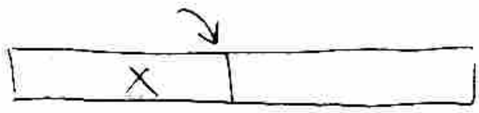
נסמן ב- Σ^* את כל המילים (הסופיות) בא"ב Σ. (זוהי קבוצה בת מניה)

בהינתן מילים $x,y,z \in \Sigma^*$ כך ש- $z = x \cdot y$ אזי נאמר ש-x הוא רישא של z ו- y הוא סיפא של z.

בהינתן מילה t ומילה אחרת p נאמר ש-p מופיעה ב-t אם קיימות מילים $x,y \in \Sigma^*$ כך ש-

$$t = x.p.y$$

דוגמא $\Sigma = \{a,b,c\}$ $P = ababaca$

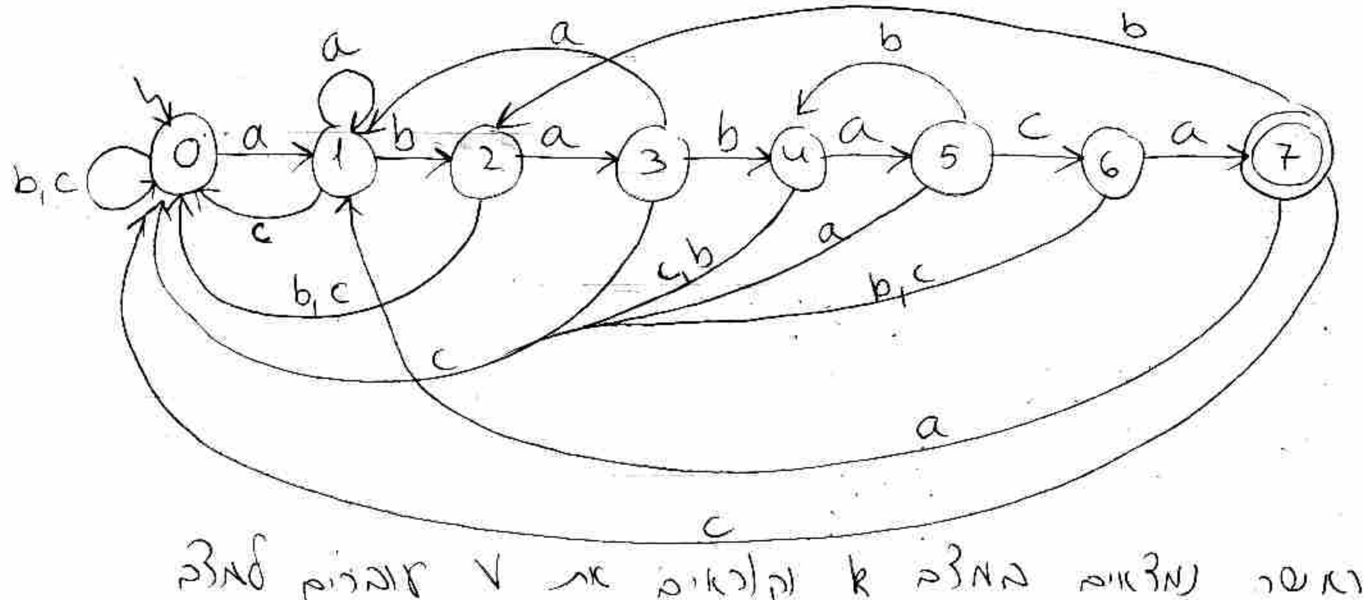
$t =$ 

אחרי שקראנו רישא כלשהו של t (נקרא לה x), אנו רוצים לדעת מהי הרישא הארוכה ביותר של P שהיא סיפא של x.

בהינתן תבנית $p = p_1 \cdots p_m$, $p_i \in \Sigma$ נסמן את הרישא של p באורך k - $p_k = p_1 \cdots p_k$

נגדיר פונקציה $\sigma : \Sigma^* \to \mathbb{N}$

σ - האורך של הרישא הארוכה ביותר של p שהיא סיפא של המילה x.



כאשר נמצאים במצב k וקוראים את v עוברים למצב

$$\delta(k,v) = \sigma(p_k . v)$$

$\delta : \{0,\dots,m\} \times \Sigma \to \{0,\dots,m\}$ - הפונקציה שאומרת לאן עוברים. מחשבים אותה מראש ואז נשתמש לחפש זה לוקח $O(n)$.

בהינתן הטבלה δ מה שנותר הוא לעבור אות אות על הטקסט ובכל פעם שמגיעים למצב m מודיעים שגילינו מופע של התבנית P בטקסט.

הטבלה הנ"ל יכולה לקחת הרבה זיכרון. אז אנחנו נרצה אלגוריתם שעושה את אותו הדבר אבל לא שומר את הטבלה.

האלגוריתם של Knuth-Moriss-Pratt

נגדיר פונקציה π $\pi : \{1,\dots,m\} \to \{0,\dots,m-1\}$

$\pi(k) = \max\{\ell : P_\ell \text{ סיפא של } P_k : \ell < k\}$

כלומר הרישא הארוכה ביותר שהיא סיפא ממש של הרישא באורך k.

נתון טקסט $t = t[1]\dots t[n]$ ותבנית $p = p[1]\dots p[m]$ והטבלה π.

המצב ההתחלתי $q=0$, המקום בטקסט i, $i=1$.

(*) נשווה את $p[q+1] : T[i]$. אם הם שווים אזי $q \leftarrow q+1$, $i \leftarrow i+1$

אם $q=m$ אזי מצאנו התאמה. $q \leftarrow \pi(m)$.

אחרת אם $q=0$ אזי $i \leftarrow i+1$

אם $q>0$ $q \leftarrow \pi(q)$ ותחזור ל-(*)

(28) 7.12.06
אלגו

זיהוי תבניות   String Matching

(cormen, ch.32)

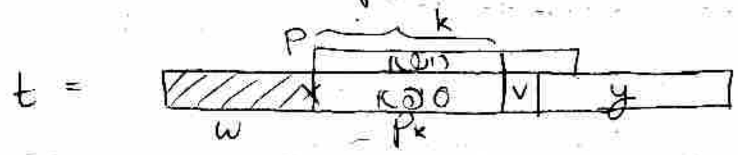
יש א"ב סופי Σ ושתי מילים $p,t \in \Sigma^*$.

$t = T[1] \ldots T[n]$ $p = P[1] \ldots P[m]$

רוצים לזהות את המופעים של p ב-t

בהינתן $x \in \Sigma^*$ נגדיר $0 \le \sigma(x) \le m$ ע"י

אורך הרישא הארוכה ביותר של p שהיא סיפא של x = $\sigma(x)$



נסמן $P_k = P[1] \ldots P[k]$.

אם $\sigma(x) = k$ אז $t = w \cdot P_k \cdot y$ ורק החלק P_k

והאות הראשונה של y ((נניח היא v)) קובעים את $\sigma(x.v) = \sigma(P_k.v)$

אם לכל k ולכל $v \in \Sigma$ נחשב את $\delta(k,v) = \sigma(P_k.v)$.

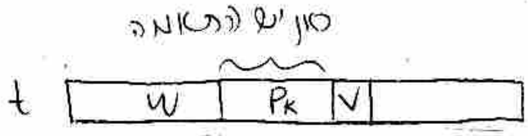
ואז מעבר אות אות על t מאפשר זיהוי של כל המופעים של p.

גודל הטבלה δ תלוי גם באורכי המילים וגם בגודל הא"ב שלה יכול להיות ממש גדול. אז אנחנו רוצים למצוא דרך לדעת לאן לעבור במצב הבא, אבל בלי שנצטרך לשמור את הטבלה בזיכרון.

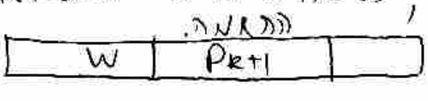
אנחנו נצליח להיפטר ממנה, אבל זה יעלה לנו בעוד פעולות. אנחנו לא נשמור אלא כל פעם נחשב את התו הבא מחדש.

אולי אפשר לתאר את זה במילים.

נניח שאנחנו במצב הבא:

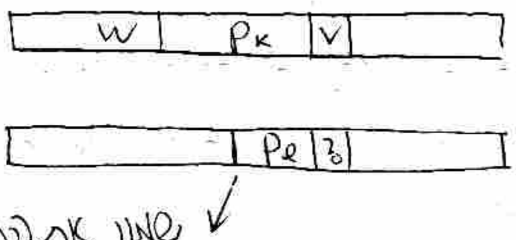


בודקים אם $P[k+1] = v$. אם כן, זה אומר שיש התאמה ב $k+1$



ועדכן אז $k \leftarrow k+1$

אם לא $V = p[k+1]$ אז נראה שאין התאמה בכלל
וצריך לקדם את התבנית שלנו. התבנית מתקדמת כמו מין
חלון רצה על T ובודקת אם בחלון יש התאמה.
אז נזיז את התבנית למקום הראשון שבו שוב יש התאמה
עד התו V, רק שנשארת בעיה שם V לא מה שנזכור שאין התאמה.



שמנו את התבנית במקום
שבו ההתאמה הבאה נמצאת.

הרישא המקסימלית באורך קטן מ-k שהיא סיפא של p_k = l.

ועכשיו שוב נבדוק מה יש ב-? וחוזר חלילה.
כי סדרה של הרחבות p_k והבטחנו שמצאנו התאמה
(אם יש כזאת)

אז כרגע: בכל שלב עושים את הדבר הבא:
אם $V = P[k+1]$ אזי $k \leftarrow k+1$
אחרת
נגדיר $\pi : \{1,\dots,m\} \rightarrow \{0,1,\dots,m-1\}$
$\pi(k) = \max\{l : P_l ; l<k, P_l \text{ סיפא של } P_k\}$
$k \leftarrow \pi(k)$.

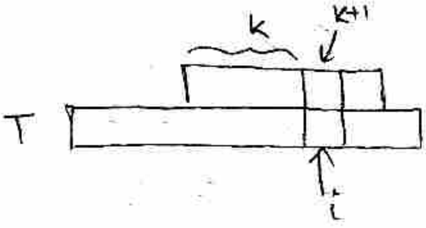
<u>דוגמה:</u> $P = a\ b\ a\ b\ a\ b\ a\ b\ c\ a$

$\pi(k) = 0\ 0\ 1\ 2\ 3\ 4\ 5\ 6\ 0\ 1$

אם נחשב את π באופן נאיבי זה לוקח $O(m^2)$
כי כל פעם מחפשים התאמה שלה $O(m)$ ואת זה צריך
לעשות עבור m מקומות.
היינו רוצים לחשב את π בזמן לינארי.

אבל בינתיים, נניח שנתונה לנו הטבלה π...

אז הנה אלגוריתם שמחפש תבנית:



1) התחלה: $k=0$

2) עבור $i = 1,\dots,n$

2.1) כל עוד $0<k$:

2.1.1) אם $P[k+1] \neq T[i]$ בצע $k \leftarrow \pi(k)$

2.3) אם $P[k+1] = T[i]$ בצע $k \leftarrow k+1$ (*)

2.4) אם $k=m$ הכרז על התאמה במקום i ובצע $k \leftarrow \pi(m)$

בנקודה (*) הערך של k הוא הערך שהיה מתקבל בחישוב של פונקצית המעבר δ.

<u>הערכת זמן ריצה:</u>

בכל מעבר של הלולאה על המשתנה i המשתנה k יכול לגדול רק ב-1 (כי ב-(2.1.1) הוא יכול רק לקטון)

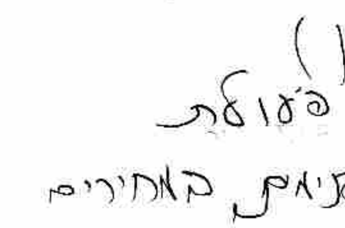
אם נשערך את פעולת ההגדלה פי שניים אזי פעולת הקטנת k תקבל ערך 0 ואז הלולאה הפנימית במחירים החדשים הינה חינם. לכן כל חזרה של הלולאה של i דורש $O(1)$ ובסה"כ $O(n)$.

<u>חישוב מהיר של π בהינתן תבנית P:</u>

נניח שידועים את $\pi(i)$ לכל $1 \le i \le k$ ונחשב את $\pi(k+1)$.

נגדיר $\pi^*(k) = \{\pi^l(k) : l \ge 1\} =$

$= \{ l :$ הרישא P_l היא סיפא ממש של $P_k \}$



אם הרישא P_{l+1} היא סיפא של P_{k+1} אזי P_l סיפא של P_k. לכן $l \in \pi^*(k)$.

עוברים בסדר אורכים "יורד" על הקבוצה $\pi^*(k)$ ובודקים לכל l בקבוצה האם $P[l+1] = P[k+1]$. בפעם הראשונה שזה קורה אפשר להגדיר $l+1 := \pi(k+1)$. אם לא נעצרנו נקבל $\pi(k+1)$

חישוב Π:

1) $\Pi(1) \leftarrow 0$

2) $\ell \leftarrow 0$

3) עבור $k = 2,\dots,m$ // חישוב $\pi(k)$ בהנחה שחישבנו קטנים יותר

3.1) כל עוד $0 < \ell$ וגם $P[\ell+1] \neq P[k]$ בצע $\ell \leftarrow \pi(\ell)$

3.2) אם $P[\ell+1] = P[k]$ אזי $\ell \leftarrow \ell+1$

3.3) $\pi(k) \leftarrow \ell$

ניתוח זמן ריצה:

כמו במקרה הקודם, אנחנו לא יודעים כמה לוקח שלב (3.1) אבל הוא לא קורה יותר פעמים מאשר (3.2)
אז אם ניתן ל(3.2) מחיר 2, אז (3.1) הוא בחינם. ⇐ סה"כ יש m איטרציות - $O(m)$
(השיטה הזאת נקראת amortization)

* יום שני הקרוב אין שיעור!

אלגוריתמים הסתברותיים

- בעיית הפילוסופים הסועדים
- סנכרון של מעבדים

יש בעיות שניתן להוכיח שלא קיים אלגוריתם דטרמיניסטי עבורן!

(30) 14.12.06
א'/ג'

<u>הודעה חשובה:</u> בשבוע הבא יש שיעור ביום שני
אין שיעור ביום חמישי

<u>אלגוריתמים הסתברותיים Randomized Alg.</u>

<u>הנחה:</u> אפשר להטיל מטבע הוגן.
כלומר נתונים מספר גדול כרצוננו של משתנים מקריים
$r_1, r_2, \ldots, r_k, \ldots$ כאשר כל r_i יכול לקבל ערך 0 או 1
רק ש- $Pr(r_i=0) = \frac{1}{2} = Pr(r_i=1)$ והמשתנים הם בלתי תלויים.

אם רוצים לבחור מספר מקרי (תמיד כשנאמר מקרי מתוך תחום הכוונה היא מהתפלגות אחידה, אלא אם נאמר אחרת).
מהתחום $\{0,1\}$. באופן טבעי פשוט נטיל מטבע.

אם רוצים לבחור מספר מקרי בתחום $S = \{0,1\}^n$ (או לחילופין מספר טבעי x מהתחום $0 \le x < 2^n$) -
מטילים n מטבעות $r_1, \ldots, r_n \in \{0,1\}$ ואז
$0 \le x = \sum_{i=1}^{n} r_i 2^{i-1} < 2^n$ - ! $P(x) = \left(\frac{1}{2}\right)^n$.

מה קורה כאשר רוצים לבחור x טבעי $0 \le x < 3$,
כלומר מהתחום $\{0,1,2\}$?
הנה רעיון: נטיל 20 מטבעות, זה נותן לנו כ- 10^6 אפשרויות ונסתכל שלהן מודולו 3. זה לא ייתן לנו התפלגות אחידה מושלמת (כי זה לא מתחלק ב-3) אבל זה מאוד קרוב לזה.
באופן כללי אם מחליטים מראש להטיל n מטבעות, ניתן לחלק את התוצאות האפשריות ל-3 קבוצות ולהחליט מהי התוצאה בהתאם לקבוצה שאליה נפלנו.

אבל אז לא $N_0 \in \mathbb{N}$ $\quad Pr(X=0) = \frac{N_0}{2^n} \neq \frac{1}{3}$

<u>שיטה לקבל הסתברות אחידה מדוייקת</u>:

בוחרים מספר טבעי X בתחום $0 \le X < 2^2 = 4$ $\quad \{0,1,2,3\}$

(נבחר X בעזרת שתי הטלות מטבע.

אם קיבלנו מספר קטן או שווה ל-2 נעצור ואחרת, אם קיבלנו 3 נחזור ונבחר X מחדש (בעזרת הטלות מטבע חדשות).

באופן כללי: אם רוצים לבחור מספר אקראי בתחום $\{0,1,\dots,a\}$ עבור a טבעי כלשהו. (נבחר חזקה של 2 רק ש $a<2^n$, נבחר X בתחום $0 \le X < 2^n$ ואם $X \le a$ נעצור, אחרת נזרוק את התוצאה ונחזור על הניסוי.

נניח שאנו רוצים לדגום באופן יעיל בהתפלגות אחידה בעזרת הטלות מטבע בתוך תחום Ω ואנו רוצים לדגום בקבוצה חלקית $S \subseteq \Omega$ ונתון $p = \frac{|S|}{|\Omega|}$.

בנוסף נניח שניתן לבדוק ביעילות האם $x \in S$ או לא.

<u>טענה</u>: האלגוריתם הבא דוגם באופן אחיד מתוך S:

בוחרים x אקראי מתוך Ω ובודקים אם נפלנו לתוך S. אם כן, עוצרים עם התשובה x. אחרת, חוזרים על הניסוי באופן בלתי תלוי.

<u>הוכחה של פיסיקאים</u>: סימטריה. אין ב-S שום נקודה מיוחדת, הבחירה של Ω היא אחידה. אין שום סיבה שיגרום לנו משהו לא אחיד.

<u>הוכחה אמיתית</u>: נבחר $a \in S$ ונראה ש- $Pr(X=a \mid X \in S) = \frac{1}{|S|}$

($X \in S$ זה המאורע שעצרנו).

$$Pr(X=a \mid X \in S) = \frac{Pr(X=a \wedge X \in S)}{Pr(X \in S)} = \frac{Pr(X=a)}{|S|/|\Omega|} = \frac{1/|\Omega|}{|S|/|\Omega|} = \frac{1}{|S|}$$

(השוויון הראשון: הגדרה של הסתברות מותנית.)

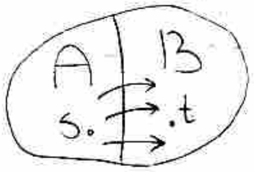
(השוויון השני: אם $X=a$ אז הדרישה ש-$X \in S$ היא בעצם מיותרת.)

<u>תוחלת הזמן עד לעצירה</u> -

$$\sum_{k=1}^{\infty} k(1-p)^{k-1}p = \frac{1}{p}$$

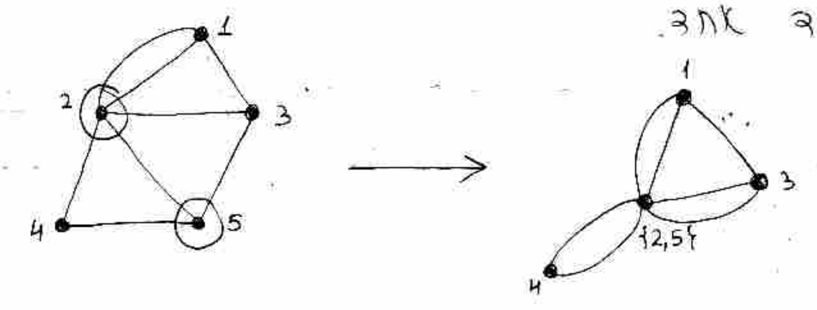
במקרה שלנו, בחירה אחידה מתוך $\{0,\dots,a\}$
$a<2^n\le 2a$ אזי $p\ge\frac{1}{2}$ ולכן תוחלת מספר הטלות
המטבע הוא לכל היותר $2n$ אז זה יעיל מאוד!

<u>בעיה קומבינטורית נחמדה</u> (זה לא אוקסימורון?)
בעיית מציאת חתך מינימלי בגרף קשיר לא מכוון $G=(V,E)$
חתך בגרף כזו חלוקה של הקודקודים $V=A\cup B$ רק ש-
$A,B\neq\emptyset$ ו- $A\cap B=\emptyset$.
המחיר של החתך הוא מספר הצלעות בגרף בין קודקודים ב-A
ו- B. מחפשים חתך עם מחיר מינימלי.
קיים אלגוריתם פולינומיאלי במספר הקודקודים -
נבחר קודקוד $s\in V$. בה"כ $s\in A$. נבחר $s\neq t$ כלשהו
ונגדיר רשת זרימה עם קיבולת 1 בשני הכיוונים על כל צלע.
חישוב זרימה מקסימלית מ-s ל-t יתן חתך



מינימלי בהנחה ש- $t\in B$. זה לוקח זמן פולינומיאלי.
נעבור על $n-1$ ה-t-ים האפשריים ונבחר את
החתך הקטן ביותר. סה"כ האלגוריתם פולינומיאלי.

ניתן אלגוריתם הסתברותי לפתרון הבעיה.
נגדיר פעולת כיווץ של שני קודקודים בגרף (או מולטיגרף - יכולות
להיות יותר מצלע אחת בין שני קודקודים). הופכים את שני
הקודקודים לקודקוד אחד.



אלגוריתם הסתברותי למציאת חתך מינימלי (Karger)

בוחרים באופן אקראי צלע ומכווצים לאורכה.
ממשיכים בפעולות הכיווץ, כל פעם בוחרים צלע באופן אקראי.
אחרי $n-2$ חזרות "ישארו 2 קודקודים.
שני הקודקודים מייצגים קבוצת קודקודים מהגרף המקורי.
נחזיר חתך לפי הקבוצות האלה.
נחזור על התהליך $n^2 \log n$ פעמים ונבחר את התוצאה הטובה ביותר בתור התשובה.

<u>טענה:</u> בהינתן חתך מינימלי בגרף ההסתברות שהאלגוריתם יחזיר את החתך הזה אחרי איטרציה בודדת היא לפחות $\frac{1}{\binom{n}{2}} > \frac{1}{n^2}$.

<u>הוכחה:</u> נניח שמספר הצלעות בחתך המינימלי הוא k. מה הסיכוי שאף פעם לא בחרנו את k הצלעות הללו? כדי לקבל את החתך הזה אסור לאלגוריתם לבחור באחת מ-k הצלעות שבחתך.
בצעד הראשון ההסתברות להיכשל היא $\frac{k}{|E|}$. אם החתך המינימלי הוא בגודל k אז דרגת כל קודקוד היא לפחות k.
$\Leftarrow |E| \geq \frac{nk}{2} \Leftarrow$ ההסתברות להיכשל $\leq \frac{k}{\frac{1}{2}kn} = \frac{2}{n}$ ...
$\Leftarrow$ ההסתברות להצליח בצעד הראשון $\geq 1-\frac{2}{n}$.
אחרי j צעדים (של הצלחה), מספר הקודקודים יורד ל- $n-j$. כיוון שחתך במולטיגרף הנוכחי הוא חתך בגרף המקורי גם כאן דרגת כל קודקוד $\geq k$.
לכן ההסתברות להצליח, כלומר לא לבחור אחת מ-k הצלעות המיוחדות (שכולן נותרו) היא לפחות $1-\frac{2}{n-j}$.
ההסתברות להצליח עד הסוף היא $(1-\frac{2}{n})(1-\frac{2}{n-1})\dots(1-\frac{2}{n-j})\dots(1-\frac{2}{3}) =$

$$= \frac{n-2}{n}\cdot\frac{n-3}{n-1}\cdot\frac{n-4}{n-2}\cdots\frac{2}{4}\cdot\frac{1}{3} = \frac{2}{n(n-1)} = \frac{1}{\binom{n}{2}}$$

<u>מסקנה:</u> ההסתברות למצוא חתך בגודל מינימלי $\geq \frac{1}{n^2}$.
ההסתברות לכישלון היא לכל היותר $1-\frac{1}{n^2}$. ואם חוזרים על זה $n^2 \log n$ מקבלים $(1-\frac{1}{n^2})^{n^2 \log n} \sim e^{-\log n} < \frac{1}{n}$

(32) 18.12.06
אליאן

בדיקת ראשוניות

נתון מספר a כך ש- $2<a<2^n$ ורוצים לבדוק האם a ראשוני.

a ראשוני $\Leftrightarrow$ לכל b כאשר $1<b<a$ b אינו מחלק את a.

בדיקת חלוקה דורשת $O(n^2)$ פעולות ביטים אבל יש מספר גדול של b-ים. לפעמים מספיק לבדוק רק עד $\sqrt{a}$ אבל זה עדיין ממש מלא.

למה זה טוב לדעת אם מספר הוא ראשוני? זה יעזור לנו בהצפנה. בעוד שהכפלה של מספרים זו משימה פשוטה, חלוקה של מספר לגורמים היא בעיה לא קלה בכלל. ועל זה מתבססים בצופן RSA - צריך לבחור 2 מספרים ראשוניים אבל בשביל לבחור כאלה, צריך לבחור 2 מספרים אקראיים ולבדוק אם הם ראשוניים. עוד על כך בהרצאות הבאות.

<u>הגדרה שקולה:</u> a ראשוני אם לכל $1\le b<a$

$$a^{b-1}\equiv 1 \pmod{a}$$

<u>הוכחה:</u>

($\Leftarrow$) משפט פרמה הקטן

($\Rightarrow$) אם $b>1$ $b|a-1$ אז $b^{a-1}\not\equiv 1 \pmod{a}$. אחרת

$b^{a-1}=1+m\cdot a \Leftarrow 1=b^{a-1}-ma$

$\Leftarrow$ $b|1$ סתירה לכך ש- $b>1$.

בדיקה כזאת דורשת $O(n^3)$ פעולות ביטים.

יהי a מספר פריק. אזי

(I) אם b איננו זר ל-a $\Leftarrow b^{a-1} \not\equiv 1 \pmod a$

לכן b מעיד ש-a מספר פריק.

אם $a = p \cdot q$ כאשר p, q ראשוניים שונים

$$\mathbb{Z}_a^* = \{ b : (b,a) = 1 \}$$

$$|\mathbb{Z}_a^*| = \varphi(a) = (p-1)(q-1) =$$
$$= pq - p - q + 1 =$$
$$= a - p - q + 1$$

$\Leftarrow$ אם $p, q \sim 2^{n/2}$, $a \sim 2^n$

הסיכוי לבחור $b \notin \mathbb{Z}_a^*$ הוא

$$\frac{2^{n/2}}{2^n} = 2^{-\frac{n}{2}}$$

כאשר בוחרים b אקראי $1 \leq b < a$.

יתכן שעבור a פריק, אחוז ה-b-ים בתחום $1 \leq b < a$ מקיימים $b^{a-1} \not\equiv 1 \pmod a$,

או לחילופין, יתכן שלכל $b \in \mathbb{Z}_a^*$

$$b^{a-1} \equiv 1 \pmod a$$

a-ים כאלה נקראים מספרי Carmichael והמספר 561 הוא מספר כזה לדוגמא.

יודעים להוכיח שלפחות $n^{2/7}$ מהמספרים עד n הם מספרי Carmichael עבור n-ים גדולים.

נעבר לעובדה נוספת הידועה עבור ראשוניים:

אם $a > 2$ ראשוני אז למשוואה $x^2 \equiv 1 \pmod a$ יש בדיוק שני פתרונות ב-$\mathbb{Z}_a^*$: הפתרונות הם כמובן $x = 1, a-1$.

⇐ אם נמצא $d\in \mathbb{Z}_a^*$ כך ש-
$d \not\equiv \pm 1 \pmod a$
אבל $d^2\equiv 1 \pmod a$ זה אומר שמצאנו
עוד שורש ⇐ a לא יכול להיות ראשוני,
כלומר a פריק.

בדיקת ראשוניות הסתברותית (1977 Rabin)

ניקח a שהוא בה"כ אי-זוגי.
נציג $a-1 = 2^u\cdot t$ כאשר t אי-זוגי, ו-$1\le u$.
בוחרים b מקרי בתחום $1\le b<a$ ונחשב
את $b^{a-1} \pmod a$.

$$b_0 = b^t \pmod a,\ b_1 = b_0^2 \pmod a,\ \dots,\ b_u = b_{u-1}^2 \pmod a$$

$$b_1 = b^{t\cdot 2}$$

$$\vdots$$

$$b_u = b^{t\cdot 2^u} = b^{a-1} \pmod a$$

מתבוננים בסדרה $(b_0,\dots,b_u)$.
אם $b_u\neq 1$ עוצרים ומכריזים ש-a פריק.
אם יש b_ℓ כך ש- $b_\ell\neq\pm1$ אבל
$b_{\ell+1}=1$ נכריז ש-a פריק
אחרת a אולי ראשוני.

<u>הגדרה:</u> יהי a מספר פריק, $1\le b<a$ כך
שהבדיקה b מכריזים על פריקות a. אזי (אומר
ש-b הוא עד לפריקות a.
נסמן $W(a)=\{ b :$ b עד לפריקות a

משפט: אם a מספר פריק אי זוגי אז

$$|W(a)| \ge \frac{a-1}{2}$$

- **משפט מחזוריות המספרים:** נתבונן בחבורה הכפלית מודולו a $\mathbb{Z}_a^*$. אזי החבורה הזאת היא ציקלית.

כלומר, קיים $g \in \mathbb{Z}_a^*$ כך ש

$$\mathbb{Z}_a^* = \{1, g, g^2, \dots, g^{\varphi-1}\}$$

כאשר $|\mathbb{Z}_a^*| = \varphi = \varphi(a)$

הוכחת המשפט על מספר העדים:

מקרה I קיים $b \in \mathbb{Z}_a^*$ כך ש- $b^{a-1} \equiv 1 \pmod a$

כלומר a אינו מספר Carmichael (נגדיר

$$B = \{b \in \mathbb{Z}_a^* : b^{a-1} \equiv 1 \pmod a\}$$

$1 \in B \Leftarrow B \neq \emptyset$. B סגורה תחת כפל מודולו a:

$$(b_1 b_2)^{a-1} = b_1^{a-1} b_2^{a-1} \equiv 1 \cdot 1 = 1 \pmod a$$

⇐ B תת חבורה של החבורה הכפלית $\mathbb{Z}_a^*$

⇐ הכלה ש- $B \subsetneq \mathbb{Z}_a^*$ אז ממשפט לגרנז'

$$|B| \mid |\mathbb{Z}_a^*|$$

$$\Rightarrow |B| \le \frac{|\mathbb{Z}_a^*|}{2} = \frac{a-1}{2}$$

אם $b \notin \mathbb{Z}_a^*$ אזי הוא [illegible] עד לפריקות a.

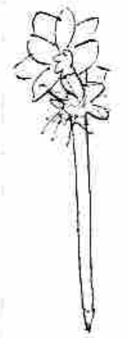
⇐ $|W(b)| \ge \frac{a-1}{2}$.

מקרה II a מספר Carmichael לכן לכל $b \in \mathbb{Z}_a^*$ $b^{a-1} \equiv 1 \pmod a$. נשים ♡ שבמקרה זה לא יכול להיות ש- $a = p^e$ עבור p ראשוני כלשהו. כי אחרת, יהי g איבר שהחזקות שלו נותנות את כל איברי $\mathbb{Z}_{p^e}^*$ (ל- g קוראים שורש פרימיטיבי מודולו p^e)

הסדר של g בחבורה $\mathbb{Z}_{p^e}^*$ שווה למספר
האיברים בחבורה $\varphi(p^e)=p^e-p^{e-1}=p^{e-1}(p-1)$

$\Leftarrow$ $g^{p^{e}(p-1)}\equiv 1 \pmod{p^e}$.

אם $g^{p^e-1}\equiv 1 \pmod{p^e}$ אז הסדר של g מחלק
את p^e-1 , כלומר $p^{e-1}(p-1) \mid p^e-1$ אבל
זו סתירה כי $p^{e-1}(p-1)$ מתחלק ב-p ואילו
p^e-1 אינו מתחלק ב-p.



$\Leftarrow$ אם a מספר Carmichael אזי בפירוק
לגורמים ראשוניים של a מופיע יותר מאשר ראשוני אחד.
נניח $a=p_1^{e_1}p_2^{e_2}\cdots p_l^{e_l}=n_1\cdot n_2$ כאשר $n_1,n_2>1$
מספרים זרים זה לזה. למשל $n_1=p_1^{e_1}$, $n_2=p_2^{e_2}\cdots p_l^{e_l}$
לכל $b\in\mathbb{Z}_a^*$ באלגוריתם הבדיקה נקבל תמיד $b_u\equiv 1 \pmod a$.
נתבונן בסדרות $(b_0,b_1,\dots,b_u)$ המתקבלות עבור כל b.
נאמר שהזוג (b,j) הוא <u>מתאים</u> אם $b_j=b^{t\cdot 2^j}=-1 \pmod a$
נשים לב כי $(-1,0)$ הוא זוג מתאים, כי
t הוא אי-זוגי, ואז $a-1=2^u\cdot t$ $(-1)^t=-1 \pmod a$
יהי j המספר המקסימלי שעבורו יש b כך ש-(b,j)
זוג מתאים. נגדיר

$$B=\{b\in\mathbb{Z}_a^* : b^{t\cdot 2^j}=\pm 1 \pmod a\}$$

אם $b\in\mathbb{Z}_a^*$ אבל $b\notin B$ אזי b הוא עד.
קבוצה B סגורה תחת כפל, ולכן היא תת חבורה.
נותר להראות ש-$B\subsetneq\mathbb{Z}_a^*$.
B מכילה את כל ה"לא עדים" ואז $\geq \frac{|\mathbb{Z}_a^*|}{2}$
ולכן נסיים.
כדי להציג איבר $b\in\mathbb{Z}_a^*$, $b\notin B$. יהי (b,j)
זוג מתאים $b^{t\cdot 2^j}\equiv -1 \pmod a$. $a=n_1\cdot n_2$. לפי משפט
השאריות הסיני קיים $0\le w<a$ כך ש-

$$w\equiv b \pmod{n_1}$$
$$w\equiv 1 \pmod{n_2}$$

$(b,n_1)=1 \Leftarrow (b,a)=1$ ו- $(\omega,n_1)=1$

$\omega\in\mathbb{Z}_a^* \Leftarrow (\omega,n_1\cdot n_2)=1 \Leftarrow (\omega,n_2)=1$

אזי $B=\{b\in\mathbb{Z}_a^* \,;\, b^{t\cdot 2^j}\equiv\pm1 \pmod a\}$

אם $n_1,n_2|a$ $\quad \omega^{t2^j}\equiv\pm1 \pmod a$

$\omega^{t2^j}\equiv\pm1+kn_1\cdot n_2$

לכן $\omega^{t2^j}\equiv1 \pmod{n_1}$ או $\omega^{t2^j}\equiv1 \pmod{n_2}$!

או שניהם -1

אבל [illegible] n_1

$\omega^{t2^j}\equiv1^{t2^j}\equiv1 \pmod{n_2}$

$\omega^{t2^j}\equiv b^{t2^j}\equiv-1 \pmod{n_1}$ סתירה.

$\Leftarrow \omega\notin B$

(35) 25.12.06
אלגו׳

יום הולדת שמח אגדו!!

בדיקת ראשוניות

נתון מספר טבעי $a<2^n$ ורוצים לבדוק האם a ראשוני.
לראשוניים יש שתי תכונות
(I) מודולו a ל-1 יש בדיוק שני שורשים ריבועיים $\{1,-1\}$
(II) עבור $1\le b<a$ $b^{a-1}\equiv 1 (\mathrm{mod}\ a)$

ניתן להניח ש $2<a$ ואז $a-1=2^u\cdot t$ כאשר t אי-זוגי.
בוחרים b אקראי מתחום $[1,a)$ ומחשבים מודולו a את $b^t, (b^t)^2, b^{4t}, \dots, b^{2^u t}=b_u$ - סדרה של העלאות בריבוע.
אם $b_u\neq 1$ אז a לא ראשוני ואם עוברים ממספר שונה מ ± 1 ל-1 גם אז a לא ראשוני.
b כזה נקרא עד לפריקות a. נסמן
$W(a)=\{b : 1\le b<a$ b עד לפריקות $a\}$
הוכחנו שאם a פריק אזי $|W(a)|\ge\frac{a-1}{2}$
נקבל אלגוריתם הסתברותי:
בהינתן a בוחרים b אקראי $1\le b<a$ ובודקים האם $b\in W(a)$. זה דורש $O(n^3)$ פעולות ביטיות.
אחרי בדיקה אחת רואים: (1) אם מצאנו עד לפריקות a פריק
(2) אחרת ההסתברות ש-a פריק $\ge\frac{1}{2}$

אחרי 100 חזרות נקבל שאו שמצאנו ש-a פריק או ש-a ראשונה בהסתברות שגיאה $\ge\frac{1}{2^{100}}$.

* ב- 2004 כמה חברה הודים מצאו אלגוריתם דטרמיניסטי פולינומיאלי לבדיקת ראשוניות שעובד ב $O(n^{12})$.

<u>בחירת מספר ראשוני p $2^n \geq p \geq 2$</u>

<u>בהסתברות אחידה על פני המספרים הראשוניים בתחום</u>

משפט המספרים הראשוניים: נסמן ב- $\pi(x)$ את מספר המספרים הראשוניים הקטנים מ- x. אזי

$$\lim_{x\to\infty} \frac{\pi(x)}{x/\ln x} = 1$$

חוץ מזה $\frac{7}{8} < \frac{\pi(x)}{x/\ln x} < \frac{9}{8}$ לכל x.

המסקנה מכאן היא שמספר המספרים הראשוניים בני n ביטים הוא לפחות $\frac{2^n}{2n}$.

למשל עבור $n=1024$ אז בערך $\frac{1}{2000}$ מהמספרים הם ראשוניים עד 2^{1024}.

אם רוצים לבחור ראשוני מקרי נבחר מספר מקרי מתחום 2^n ונבדוק האם הוא ראשוני. נחזור עד להצלחה.

תוחלת מספר החזרות היא n.

אם בבדיקת הראשוניות חוזרים על הבדיקה k פעמים אזי הסתברות השגיאה (כלומר הכרזה על מספר פריק כראשוני) קטנה מ- $\frac{1}{2^k}$, וזו גם השגיאה בהכרזת "מספר ראשוני" באלגוריתם הבחירה של ראשוני מקרי כאן, כי כל פעם שאומרים שמספר הוא פריק זה בטוח נכון. השגיאה יכולה לקרות רק במספר שנבחר בסוף.

בהינתן שני מספרים ראשוניים p,q ניתן להכפיל אותם בעזרת אלגוריתם יעיל. אם $p,q \le 2^{n/2}$ אז למכפלה $N=p\cdot q$ יש בערך n ביטים.

בעיית פירוק מספרים טבעיים לגורמים ראשוניים

נתון מספר $a < 2^n$. מהו האלגוריתם הטוב ביותר לפרק a לגורמים?

- בזמן $2^{n/2}$ ($\sim\sqrt{a}$) ניתן למצוא גורם של a לחלק וככה להמשיך.
- ב-1960 שיפרו את זה ל- $2^{n/4}$
- מ-1970 זה השתפר ל- $2^{O(\sqrt{n\log n})}$
- ב-1992 הגיעו ל- $2^{O(n^{1/3}(\log n)^{2/3})}$ ומאז לא השתפר.
- ב-1995 גילו שאם ניתן לבנות מחשבים קוונטיים אז קל לפרק מספרים לגורמים.

בחוסר סימטריה הזה בין מכפלה לפירוק משתמשים בהצפנה. בהצפנה קלאסית ל-A ול-B יש מפתח הצפנה ופענוח משותף שנבחר באופן מקרי מתחום כלשהו $K\in\{0,1\}^n$.

אם A (עליס) ו-B (בוב) בחרו K רצה מראש ואליס רוצה לשלוח הודעה $m\in\{0,1\}^n$ ל-B אז אם $K=k_1\ldots k_n$ ו- $m=m_1\ldots m_2$ אז אליס יכולה לחשב את הווקטור $C=K+m$

$$(k_1\oplus m_1,\ldots,k_n\oplus m_n)=(c_1,\ldots,c_n)$$

ולשלוח את C ל-B

לכל m כאשר בוחרים k באופן מקרי ואחיד אז C מפולג באופן אחיד. אז E (איב) שמסתכלת מבחוץ לא יכולה להבין מה קורה.

אבל בוב יודע את k ונשים לב ש- $C \oplus k = m$
אז הוא דווקא יכול לפענח את ההודעה.
הבעיה העיקרית בשיטה הזאת היא שאליס ובוב צריכים להיפגש כדי להחליף מפתח.

D-H - שיטת מפתח פומבי.
נניח שכל אחד מהמשתתפים B יכולים לבחור מפתח הצפנה e_B ומפתח פענוח d_B וקיימת שיטת הצפנה
$E_{e_B}: M \to M$ ושיטת פענוח $D_{d_B}: M \to M$
(תחום ההודעות $\{0,1\}^n$)
ומתקיים $D_{d_B}(E_{e_B}(m)) = m$ לכל $m \in M$.
אנו רוצים שיתקיימו התכונות הבאות:
- כל שחקן שיודע את e_B יכול לחשב את $E_{e_B}(m)=c$ לכל m
- אם שחקן יודע את e_B אזי גם אם הוא רואה הודעה מוצפנת $C=E_{e_B}(m)$ ~~את~~ אינו יודע ~~את m~~ לחשב בזמן סביר מהו m (לא מספר זניח של m-ים).

במקרה כזה B יכול לפרסם בפומבי את מפתח ההצפנה e_B ולשמור בסוד את מפתח הפענוח d_B. אז אם A רוצה לשלוח הודעה ל-B היא מחשבת את $c=E_{e_B}(m)$ ושולחת אותו ל-B. E רואה את c אבל לא יודע לפענח אותו (לפי הנחה 2), אבל B יודע את d_B אז הוא מקבל $m=D_{d_B}(c)$.
מחפשים שיטה כזאת שחישוב E ו-D יעילים אבל לא ניתן לחשב את מפתח הפענוח ממפתח ההצפנה.
מערכת כזו של מפתח הצפנה פומבי מאפשרת גם חתימה דיגיטלית.
$m = D_{d_B}(E_{e_B}(m)) \leftrightarrow m=E_{e_B}(D_{d_B}(m))$. B הוא היחיד שיודע לחשב את D_{d_B}.
למשל, אם B רוצה לחתום על הודעה m הוא יכול לחשב $s=D_{d_B}(m)$ ושולח ל-A את "(m,s)"
וכדי לבדוק שזו חתימה של B בודקים $E_{e_B}(s)=m$.

(37) 28.12.06
אלגו'

הצפנה בשיטת המפתח הפומבי

1997 - Diffie-Hellman - ישנן פונקציות הצפנה ופענוח

$E_e : M \to M$

$D_d : M \to M$

כך ש- $D_d = E_e^{-1}$
כאשר e מפתח הצפנה ו- d מפתח פענוח, אבל אין אלגוריתם יעיל לפענוח למרות שיודעים את e.

1997-8 - RSA, Rabin - שיטות שמבוססות על הקושי של פירוק מספרים לגורמים.

יש שיטות נוספות שנעזרות בבעיות קשות אחרות אבל אנחנו לא נדון בהן כאן.

RSA

B בוחר שני מספרים ראשוניים אקראיים p,q (בני $n/2$ ביטים) ומחשב את המכפלה $N = p \cdot q$. (ל- N יש n ביטים).

נתבונן בחבורה הכפלית מודולו N:

$$\mathbb{Z}_N^* = \{ a : 1 \le a < N \wedge gcd(a,N) = 1 \}$$

$$|\mathbb{Z}_N^*| = (p-1)(q-1) = \varphi$$

⇐ לפי משפט אוילר לכל $a \in \mathbb{Z}_N^*$ מתקיים

$$a^{(p-1)(q-1)} \equiv 1 \pmod{N}$$

נבחר e כך ש- $gcd(e,\varphi) = 1$ כאשר $1 \le e < \varphi$

נחשב את d כך ש- $1 \le d < \varphi$ ו- $e \cdot d \equiv 1 \pmod{\varphi}$
בשביל למצוא d כזה מריצים את האלגוריתם של
אוקלידס. כאשר מחשבים את המחלק המשותף המקסימלי
מקבלים מספרים d, f כך ש- $de + f\varphi = 1$
אם d יצא שלילי במקרה זה מוסיפים לו כפולות
של φ עד שמקבלים $1 \le d < \varphi$.

המפתח הפומבי של B יהיה הזוג (N, e).
תחום ההצפנות הוא $M = \{m : 0 \le m < N\}$
ועל N מגדירים

$$E_{(e,N)}(m) = m^e \pmod{N}$$

$$D_{(d,N)}(c) = c^d \pmod{N}$$

<u>טענה</u>: לכל $m \in M$ $D_d(E_e(m)) = m$
<u>הוכחה</u>: - אם $\gcd(m,N) = 1$ אז

$$(m^e)^d \equiv m^{ed} \cdot 1^{-k} \equiv m^{ed}(m^{\varphi})^{-k} \equiv m^{ed-k\varphi} \equiv$$

($m^{\varphi} \equiv 1 \pmod{N}$)

$$\equiv m \pmod{N}$$

($ed - k\varphi = 1$)

- אם $\gcd(m,N) \neq 1$ (זה מקרה לא מעניין, כי אם
זה כך אז שברנו את ההצפנה. אבל הטענה נכונה
גם אז).
אם $p \mid m \Rightarrow q \nmid m$ (אחרת $m \ge N$)

$$ed \equiv 1 \pmod{\underbrace{(p-1)(q-1)}_{\varphi}}$$

$$ed \equiv 1 \pmod{(q-1)}$$

לכן בדומה לחישוב הקודם אבל בחישוב מודולו q:

$$m^{q-1} \equiv 1 \pmod{q} \Rightarrow m^{(p-1)(q-1)} \equiv 1 \pmod{q}$$

$\Rightarrow m^{ed} \equiv m \pmod{q}$. ברור ש-
$m^{ed} \equiv 0 \pmod{p}$ (כי m מתחלק ב- p)

נסתכל על אלו המשוואות

$$m \equiv m \pmod q$$

$$m \equiv 0 \pmod p$$

ממשפט השאריות הסיני אנחנו יודעים שמודולו N

$$m^{\varphi} \equiv m \pmod N$$

הערכת סיבוכיות

יצירת זוג מפתחות -

מציאת ראשוני אקראי - $O(n^4)$

הצפנה ופענוח -

פעולת כפל מודולו N דורשת - $O(n^2)$

להצפנה צריך $O(\log e)$ ואם בחרנו e קטן ההצפנה דורשת $O(n^2)$.

- d הוא בן n ביטים $\Leftarrow$ פענוח זה $O(n^3)$

זה גם מחיר חישוב חתימה בשיטת RSA.

אם אכן לכל אלגוריתם שמצליח לפענח הודעות RSA בהסתברות הצלחה של 1% (עבור הודעות אקראיות) דרוש זמן שהוא לפחות $2^{n^{\varepsilon}}$ ($\varepsilon > 0$ קבוע כלשהו), כלומר אם זה הקושי של חישוב ההופכי להצפנת RSA, אז אם בוחרים מספר אקראי $1 \le r < N$ ומחשבים

$$r_1 = E_e(r) \quad r_2 = E_e(r_1) \quad \ldots \quad r_k = E_e(r_{k-1})$$

אז מקבלים סדרה $r_0 = r, r_1, r_2, \ldots, r_k, \ldots$

לכל אחד מהם נגדיר ביט $b_0, b_1, \ldots, b_k, \ldots$ מתאים לזוגיות של r_k.

אם יש אלגוריתם פולינומיאלי שמבחין בין $b_0, b_1, \ldots, b_{n^{10}}$, לבין $c_0, c_1, c_2, \ldots, c_{n^{10}}$ כאשר c_i סדרה של הטלות מטבע אקראיות אז אפשר גם לשבור את ה-RSA.

למעשה את הסדרה $b_0, b_1, \ldots, b_{n^{10}}, \ldots, b_{2^{n^{1/100}}}$

נראית מקרית למאמרו לכל אלגוריתם יעיל.

בגלל שזה מספר כ"כ גדול אפשר לבחור n יחסית קטן.

אם רוצים להחליף אלגוריתם שדורש m ביטים מקריים

$$m \sim 2^{n^{\frac{1}{100}}}$$

$$\log m = n^{\frac{1}{100}}$$

$$\Rightarrow (\log m)^{100} = n$$

נתון סוד S. בה"כ נניח שהסוד S הוא מספר שלם $0 \le S < p$ קטן מ-p ראשוני

ורוצים לחלק את הסוד S בין n שחקנים $(n<p)$

כך שכל קבוצה בגודל $\le k$ לא יהיה שום מידע על S אבל קבוצות בגודל $k+1$ תוכלנה לשחזר את S.

בוחרים פולינום מקרי

$$f(x) = S + a_1 x + \ldots + a_k x^k$$

כאשר $a_1, \ldots, a_k$ מספרים מקריים בתחום $[0, p-1]$.

לשחקן P_r $(r=1,\ldots,n)$ נותנים את הסוד

$$S_r = f(r) \pmod p$$

כשיש $k+1$ שחקנים הם ביחד יכולים לשחזר את הפולינום ע"י אינטרפולציה.

למה קבוצה בגודל k לא יכולה לדעת שום דבר?

הקבוצה $\{1, \ldots, k\}$ יודעת את $\{S_1, \ldots, S_k\}$

$$S = f(0)$$

בעזרת S אפשר לחשב את $a_1, \ldots, a_k$. כמובן בהינתן $a_1, \ldots, a_k$ ו-S אפשר לחשב את $S_1, \ldots, S_k$.

זה מגדיר העתקה חח"ע ועל בין $\{a_1, \ldots, a_k\}$ לבין $\{S_1, \ldots, S_k\}$. אבל $\{a_i\}$ מקריים ואחידים. $\Leftarrow$ ל-k שחקנים פשוט יש k מספרים אקראיים וזה כלום!

(39) 1.1.07
אלגו'

שנה אזרחית שמחה!

חלוקת סוד בין n שחקנים

יהי p ראשוני כך ש $p>n$.

נתון סוד $s \in F_p$ $0 \le s < p$.

רוצים לתת את הסוד s בידי n שחקנים כך שיתקיים:

(I) לכל קבוצה בת k שחקנים אין כל מידע על s

(II) כל קבוצה בגודל k+1 יכולה לחשב מיד את s.

בוחרים פולינום מקרי מעל השדה F_p:

$$f(x) = s + a_1x + \dots + a_kx^k$$

כאשר $a_1,\dots,a_k$ איברים מקריים מהשדה.

לשחקן ה- e נותנים את $s_e = f(e)$.

ברור ש-(II) מתקיים כי כל קבוצה בגודל k+1 מתאימה k+1 נקודות אינטרפולציה לפולינום ממעלה לכל היותר k, ויש פולינום יחיד כזה לכן (וגם $f(x)$) ואז $f(0)=s$, כלומר הם יכולים לחשב את s.

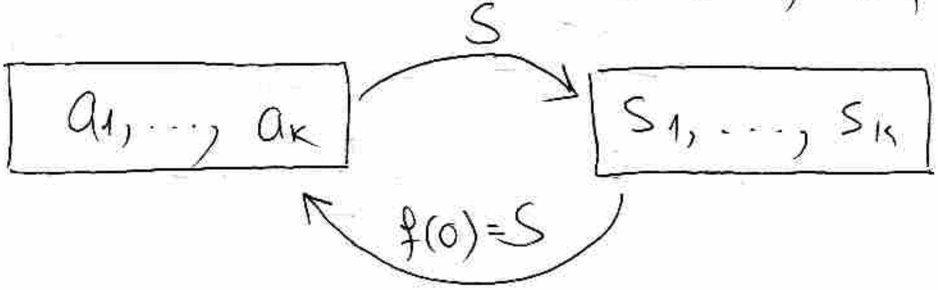
(I) הוא קצת יותר tricky. נראה שכל קבוצה של k שחקנים התהליך שבו ידעו את המספרים שהם קיבלו זהה לתהליך שבו נבחרים k מספרים בהסתברות אחידה על כל ה-k-יות האפשריות של מספרים ב- F_p.

בה"כ השחקנים הם $1,\dots,k$ (לכל קבוצה אחרת ההוכחה תהיה זהה). השחקנים האלה קיבלו את $s_1,\dots,s_k$. נראה שלכל סדרה של k מספרים יש הסתברות $\frac{1}{p^k}$.

נניח שבחרנו $a_1,\dots,a_k$ מהתפלגות אחידה.

יש התאמה חח"ע ועל בין $\{s_1,\dots,s_k\}$

לבנין $\{a_1, \ldots, a_k\}$



בסה"כ קיבלנו שההעתקות המוגדרות ע"י החוצים הן חח"ע ועל. לכן ההתאמה של a-ים משרה התפלגות אחידה על $s_1, \ldots, s_k$.

<u>הערה</u>: הדבר הזה לא מתקיים בכלל על שדות אינסופיים. למשל בשדה בן מניה זה בלתי אפשרי, כי אי אפשר בכלל להגדיר הסתברות אחידה.

<u>שימוש אחר בטכניקה זו.</u>

נניח שיש לנו k איברים בשדה מודולו q. - $a_0, \ldots, a_{k-1}$

ניתן לבנות את הפולינום

$$g(x) = a_0 + a_1 x + \ldots + a_{k-1} x^{k-1}$$

ונחשב את $n \geq k$ הערכים $g(1), \ldots, g(n)$

נסמן אותם בהתאמה $b_1, \ldots, b_n$.

מכל k מה-b-ים ניתן לשחזר את k ה-a-ים.

לכן אם מאבדים מסיבה כלשהי $n-k$ מה-b-ים

לא איבדנו אינפורמציה.

למשל, אם יש $n=10$ דיסקים ורוצים להגן בפני נפילה של 3 מהם, אזי בוחרים $k=7$.

יהי F שדה ויהי $p(x_1, \ldots, x_n)$ פולינום ממעלה d מעל F.

אפשר להציג אותו בצורה מלאה או בצורה של נוסחה, או בעזרת תוכנית שמחשבת בדרך כלשהי את

ערכי הפולינום בנקודה נתונה.

<u>דוגמה:</u> $(x_1+x_2)(x_3+x_4)\cdots(x_{n-1}+x_n)$

המעלה של הפולינום המוגדר ע"י הנוסחה הנ"ל היא $\frac{n}{2}$ אבל מספר המונומים השונים בהצגה המלאה הוא $2^{n/2}$.

בהינתן פולינום כלשהו כזה היינו רוצים לדעת האם הוא פולינום האפס. הדרך היא שבשביל לקבוע את זה באופן דטרמיניסטי הדבר היחיד שאנחנו יכולים לעשות הוא לפתוח סוגריים וזה לא משהו בכלל כי יש מספר אקספוננציאלי של מונומים שצריך לחשב.

אבל יש אלגוריתם הסתברותי יעיל לבעיה.

* אם $d=\deg p$ ויש n משתנים אז צריך $(d+1)^n$ נקודות אינטרפולציה כדי לחשב את מקדמי $p(\vec{x})$. אז גם זה מעריכי

<u>הלמה של Zippel-Schwartz</u>

יהי F שדה $0\not\equiv p(x_1,\dots,x_n)$ פולינום ממעלה לכל היותר d מעל F. תהי $S\subseteq F$ קבוצה סופית כלשהי. אם בוחרים $a_1,\dots,a_n\in S$ בהתפלגות אחידה וב"ת אז

$$\Pr(p(a_1,\dots,a_n)=0)\le \frac{d}{|S|}$$

<u>הוכחה:</u> באינדוקציה על n.

עבור $n=1$ לפולינום $p(x_1)$ יש לכל היותר d שורשים בשדה ולכן ההסתברות שהוא יתאפס היא לכל היותר $\frac{d}{|S|}$.

נניח ל-$(n-1)$ ונוכיח ל-n. נניח ש-

$$p(x_1,\dots,x_n)=p_0(x_1,\dots,x_{n-1})+p_1(x_1,\dots,x_{n-1})x_n+\dots+p_k(x_1,\dots,x_{n-1})x_n^k$$

ואשר $0\not\equiv p_k(x_1,\dots,x_{n-1})$.

נשים ♡ ש- $d-k\ge \deg p_k(x_1,\dots,x_{n-1})$.

ואז

$$\Pr(p(a_1,\dots,a_n)=0) \le \Pr(\underbrace{p_k(a_1,\dots,a_{n-1})=0}) +$$

אם לא קורה (נניח שההצבה של $a_1,\dots,a_{n-1}$ איפסו את ה- p_k זה פולי. אז בחירה של a_n תיתן אפס

*

$$+\Pr\left(\text{מאפס } a_n \text{ את } p(a_1,\dots,a_{n-1},x_n) \;\middle|\; p(a_1,\dots,a_{n-1},x_n)\not\equiv 0\right) \le$$

$$\le \frac{d-k}{|S|} + \frac{k}{|S|} = \frac{d}{|S|}$$

ה"א של $n-1$

זה עבור משתנה אחד

☺

בהינתן מטריצה $(a_{ij}) = A \in M_n(F)$

הדטרמיננטה של A $\quad \det A = \sum_{\pi\in S_n} \mathrm{sign}(\pi)\prod_{i=1}^{n} a_{i,\pi(i)}$

הפרמננטה של A מוגדרת $\quad per A = \sum_{\pi\in S_n}\prod_{i=1}^{n} a_{i,\pi(i)}$

לא ידוע אלגוריתם יעיל לחשב את הפרמננטה, בעוד שידועים אלגוריתמים יעילים לחישוב $\det A$, ב-$O(n^3)$ למשל. אבל האלגוריתם הטוב ביותר לחישוב $Per(A)$ הוא אקספוננציאלי.

למה הפרמננטה טובה בכלל?

בהינתן גרף דו-צדדי $G=(L\ R,\ E)$

אפשר לשאול האם קיים זיווג מלא בגרף וכן לשאול מהו מספר הזיווגים המלאים השונים בגרף.

יש אלגוריתם הסתברותי שאומר אם קיים זיווג מלא בגרף.

<u>הערה</u>: אם נגדיר מטריצה A ע"י $a_{ij} = \begin{cases} 1 & (i,j)\in E \\ 0 & \text{אחרת} \end{cases}$
אז $per(A)$ = מספר הזיווגים המלאים בגרף G.
למה? כי זה עובר על כל הפרמוטציות האפשריות ובודק באילו מהן מכולם יש צלע (אחרת המכפלה מתאפסת).

אם $\det A \neq 0$ אז ברור שיש זיווג מלא אבל ייתכן שהדברים שם יתאפסו, כלומר שיש זיווג מושלם אבל בכל זאת $\det A = 0$. אז זאת לא שיטה טובה.

נגדיר A חדשה. יהיו $x_{11}, x_{12}, \dots, x_{nn}$ n^2 משתנים.

$$a_{ij} = \begin{cases} x_{ij} & (i,j)\in E \\ 0 & \text{אחרת} \end{cases}$$

$\Leftarrow$ $\det A \equiv 0$ אמ"מ אין זיווגים מלאים בגרף.

ברור ש- $\deg \det A \leq n$. אם נציב ערכים מקריים מתוך תחום S שגודלו $100 \cdot n$ ונחשב את הערך של $\det A$ בנקודה הזאת אז ההסתברות ש- $\det A \neq 0$ ובכל זאת התוצאה שנקבל היא 0 קטנה מ- $\frac{1}{100}$.

יום חמישי - יש מרצה מחליף

יום שני הבא - אין שיעור

(42) 4/1/06
אלגו

שיעור עם אורנה קופרמן

בעיות אופטימיזציה

כבר ראינו דוגמה לבעיית אופטימיזציה בקורס הזה - בעיית זרימה.

בעיית אופטימיזציה היא בעיה המוגדרת ע"י שני דברים
- צורת פתרון מותר (משפחה F של פתרונות חוקיים)
- איכות הפתרון (פונקציה q(s) שרוצים למזער או למקסם).

דוגמאות:
- עץ פורש מינימלי
  (עץ פורש – צורת פתרון; מינימלי – איכות פתרון)
- מסלול קצר ביותר
  (מסלול – צורת פתרון; קצר ביותר – איכות פתרון)

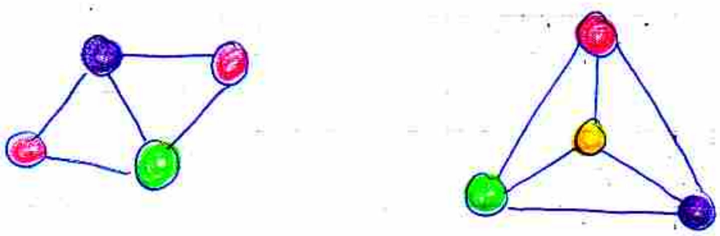
אלה בעיות קלות - יש להן פתרון איכותי ביותר שאפשר למצוא בזמן פולינומיאלי.

- צביעת גרף במינימום צבעים
  (צביעת גרף – צורת פתרון; במינימום צבעים – איכות פתרון)

קלט: גרף

פלט: צביעה כך שכל שני קודקודים סמוכים צבועים בצבעים שונים.

בחישוביות נראה שאת הבעיה הזאת אי אפשר לפתור בזמן פולינומיאלי.



נשאלת השאלה האם את הבעיות האלה אפשר לפתור בקירוב (כלומר להתפשר על האיכות) אבל בזמן קצר יותר מהנדרש לפתרון האופטימלי.

אלגוריתם C - קירוב:

הרעיון הוא שמותר לתת פתרון בעל איכות קרובה ל-Opt (Opt הוא האלגוריתם האופטימלי). הסיבה שאנחנו מתפשרים על האיכות היא שהאלגוריתם יהיה יעיל יותר בזמן.

פורמלית –

אלגוריתם לבעיית מינימיזציה הוא C-קירוב אם לכל קלט לאלגוריתם איכות הפלט $q(s)$ מקיימת

$$q(s) \leq c \cdot q(Opt)$$

באופן דומה, אלגוריתם לבעיית מקסימיזציה הוא C-קירוב אם לכל קלט לאלגוריתם איכות הפלט $q(s)$ מקיימת

$$q(s) \geq \frac{q(Opt)}{c}$$

מה שמעניין בכל העסק הזה הוא שאלגוריתם פולינומיאלי נותן $1+\varepsilon$ קירוב לבעיה אקספוננציאלית לכל ε.

בעיית התזמון של מכונות זהות

<u>קלט:</u> n משימות הדורשות זמנים $t_1, \dots, t_n$

k מכונות זהות

<u>פתרון:</u> חלוקה של המשימות למכונות

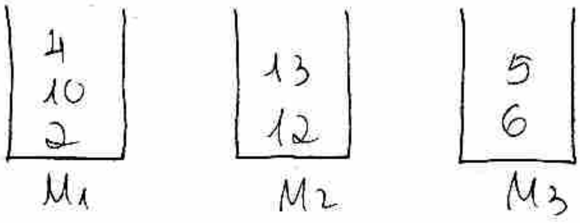
<u>איכות הפתרון:</u> זמן הסיום של המכונה האחרונה.

(כל המכונות מתחילות באותו זמן ובמקביל וכל אחת מהירות, אבל כל רגע נתון כל מכונה יכולה לעבוד אך ורק על משימה אחת).

דוגמה: 3 מכונות

משימות: 13, 12, 5, 6, 10, 2, 4

פתרון אפשרי:



אורך פתרון זה הוא 25.

אין אלגוריתם פולינומיאלי שמוצא את הפתרון הכי טוב. כל מה שאפשר לעשות הוא פחות או יותר לנסות את כל האפשרויות שלה אקספוננציאלי k^n

* השיא העולמי הוא $1+\varepsilon$-קירוב

אלגוריתם 2 - קירוב לבעיה

נסמן f_j - הזמן שבו מכונה j מסיימת מהקצאה הנוכחית.

$f_1 = f_2 = \dots = f_k = 0$

for $i = 1$ to n

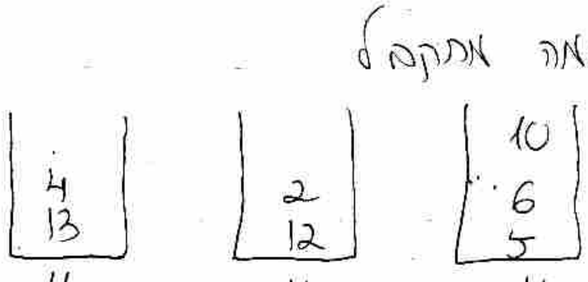
  $j =$ המכונה שמסיימת ראשונה מהקצאה עד כה

  allocate t_i to M_j

  $f_j = f_j + t_i$

הרעיון הוא שכל שלב מגיעה משימה ואנחנו שמים אותה במכונה שכרגע סכום המשימות שלה הוא הכי קטן.

עבור קלט הדוגמה מתקבל



$q(S) = 21$

אז השתפרנו!

* בפתרון האופטימלי יש $q(Opt) = 18$

טענה: $q(S) \le 2q(Opt)$ לכל S.

סימונים: $0 \le x \le n$

$q(Opt[x])$ - הביצוע של Opt על x המשימות הראשונות

$q(S[x])$ - הביצוע של האלגוריתם שלנו על x המשימות הראשונות.

$f_j[x]$ - זמן הסיום של המכונה f_j מהקצאה של x המשימות הראשונות.

קושי: יש כאן קושי קונספטואלי. איך נוכיח משהו על $q(Opt)$ כאשר אנחנו לא יודעים בכלל מהו Opt?

רעיון: נחפש דברים בגנות האופטימלי.

למשל, ברור ש-

$$\frac{\sum_{j=1}^{k} f_j[x]}{k} = \frac{\sum_{i=1}^{x} t_i}{k} \le q(Opt)$$

כי הרי המצב האופטימלי הוא שהם יהיו מחולקים באופן מאוזן אבל זה לא תמיד יכול להסתדר.

עוד דבר הוא ש-

$$\min_j f_j[x-1] \le \min_j f_j[x] \le q(Opt)[x] \quad (*)$$

כי המינימום חייב להיות קטן מהממוצע

וברור גם ש- $t_x \le q(Opt)[x] \quad (**)$

כי האופטימלי חייב לשבץ את t_x איפשהו.

הוכחת הטענה: באינדוקציה על x. נראה שלכל x

$$q(S)[x] \le 2q(Opt)[x]$$

עבור $x=1$ כמובן.. מתקיים

$$q(S)[1] = q(Opt)[1] = t_1 \quad \checkmark$$

(44)

נניח כעת ש- $q(S)[X-1] \le 2q(Opt)[X-1]$

ונוכיח עבור X :

$$q(S)[X] = \max\{\min_j f_j[X-1] + t_X \,,\, q(S)[X-1]\}$$

כי או שהמכונה שהוספנו
אליה עכשיו את x הפכה
לאיטית ביותר או שכבר
קודם היתה מכונה עם משימה
ממש גדולה ואפילו שהוספנו
לכולן משימה זה עדיין לא
יותר ארוך מהמשימה הגדולה.

נשים לב ש- $\min_j f_j[X-1] + t_X \le 2q(Opt)[X]$

(כי נחבר את (*) ואת (**))

לפי הנחת האינדוקציה

$$q(S)[X-1] \le 2q(Opt[X-1]) \le 2q(Opt[X])$$

$$\Rightarrow q(S)[X] \le 2q(Opt)[X]$$

כמה דברים על האלגוריתם שראינו:

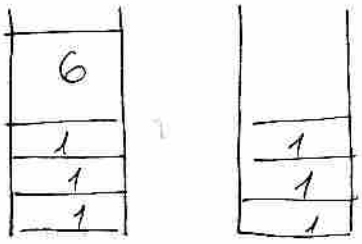
- סיבוכיות: for $i=1 \dots n$

n פעמים [
מציאת מינימום $O(\log k)$ ← למשל ע"י ערמה
$f_j = f_j + t_i$ $O(1)$

סה"כ $O(n \log k)$ שזה בוודאי יותר
טוב מהפתרון האקספוננציאלי.

- זהו אלגוריתם מקוון (on-line) - מקבל את המשימות בזמן אמת (אי אפשר למשל, למיין את המשימות מראש)
- יש קלטים רעים שעבורם $q(S) = 2q(Opt) - \varepsilon$
למשל, אם יש 2 מכונות והקלטים הם
1, 1, 1, 1, 1, 1, 6

אז ייצא הפתרון



והנה דוגמא שעוד יותר מתקרבת לשניים:

יש 100 מכונות והמשימות הן 99x100 משימות של 1 ועוד משימה של 100.

בסוף האלגוריתם שלנו תהיה מכונה אחת שיש בה 199 והשאר יהיה 99.

ברור שבפתרון אופטימלי בכל המכונות יהיה 100. אז היחס הוא $\frac{199}{100} \approx 2$.

בעית הכיסוי ע"י קבוצות

קלט: קבוצה $X=\{1,\dots,n\}$

m תתי קבוצות של X $S_1,\dots,S_m$

פתרון: חלק מהקבוצות $S_{i_1},\dots,S_{i_k}$ כך ש- $X=\bigcup_{j=1}^{k} S_{i_j}$

איכות הפתרון: מספר הקבוצות שנבחרו.

דוגמא: רוצים לראות איך השתייה על הירח משפיעה על אנשים עם תכונות שונות (זקן, שער שחור, הריון, תלתלים, תינוק וכו') אבל רוצים להטיס לחלל כמה שפחות אנשים.

אז כשמגיעים מועמדים לניסוי רוצים לבחור מתם את אלה כך שכמה שפחות מהם יהיה להם את כל התכונות הדרושות.

אז עכשיו אנחנו תקועי מוטיבציה.

ניתן אלגוריתם חמדני שהוא $\ln n$ - קירוב.

<u>הרעיון</u>: בכל איטרציה נבחר את הקבוצה שמכסה הכי הרבה איברים (מאלה שעדיין לא כוסו).

input : $X = \{1, \dots, n\}$ $\quad S_1, \dots, S_m$

$C = \{\}$

while $X \neq \{\}$

לכל היותר n פעמים כי בכל איטרציה מורידים מ-X לפחות איבר 1

- $i =$ אינדקס הקבוצה הקטנה ביותר $\quad O(\log m)$
- $C = C \cup \{i\}$ $\quad O(1)$
- $X = X \setminus S_i$
- for all j $\quad S_j = S_j \setminus S_i$

} [illegible]

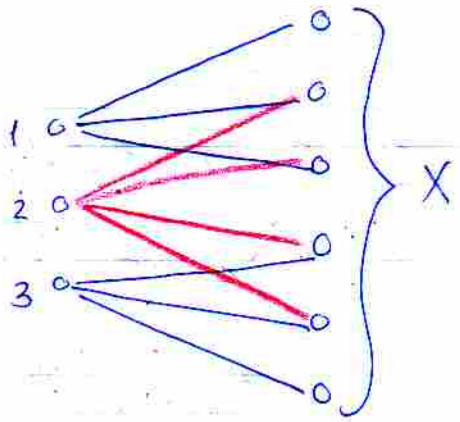
אפשר לממש את זה בכל מיני מבני נתונים - מטריצה עם תכונות רשימה מקושרת או משהו כזה.

קודם כל, נראה שהאלגוריתם הזה לא אופטימלי.

למשל:

הפתרון האופטימלי יהיה לקחת את 3,1

אבל בגלל שב-2 יש הכי הרבה איברים הוא "יבחר ראשון" ואז ניאלץ לקחת באלגוריתם שלנו את 1 ו-3.



<u>טענה</u>: $|C| \leq \ln n \cdot |Opt|$

<u>סימונים</u>:

$X[j]$ - האיברים שנותרו ב-X אחרי ש-j קבוצות נלקחו

$S_i[j]$ ______ " ______ S_i ______ " ______

$n[j] = |X[j]|$ מספר האיברים שנותרו ב-X אחרי ש-j נלקחו.

דברים רעים על Opt :

① לכל j $\quad n[j+1] = n[j] - |S_i[j]|$

עבור S_i שנבחרה באיטרציה ה-j-ית

② לכל j $\quad |Opt| \geq \frac{n[j]}{|S_i[j]|}$

מה זה אומר ולמה זה נכון?

- Opt צריך לקחת את כל $X[j]$ - הוא חייב לכסות את כל האיברים שנשארו ב-X אחרי האיטרציה ה-j-ית כי הם עוד לא כוסו.
- כל הקבוצות שנלקחו עד האיטרציה ה-j-ית לא מכילות איברים מ-$X[j]$ כי כל מה שנלקח ירד מ-X.
- $S_i[j]$ היא הקבוצה הכי גדולה שנותרה.

ועכשיו זה צריך להיות ברור. זה הרעיון של חלוקה מאוזנת.

את (2) אפשר גם לרשום כך: $|S_i[j]| \geq \frac{n[j]}{|Opt|}$.

הוכחת הטענה:

מ-(1) ו-(2) מתקבל

$$(*)\quad n[j+1] \leq n[j]\left(1-\frac{1}{|Opt|}\right) \leq$$

$$\leq n[j-1]\left(1-\frac{1}{|Opt|}\right)\left(1-\frac{1}{|Opt|}\right) \leq$$

$$\leq \dots \leq$$

$$\leq n[0]\left(1-\frac{1}{|Opt|}\right)^j = n\left(1-\frac{1}{|Opt|}\right)^j <$$

$$< n e^{-\frac{j}{|Opt|}}$$

כי לכל x, j מתקיים $\left(1-\frac{1}{x}\right)^j < e^{-\frac{j}{x}}$

אנחנו רוצים להוכיח $|C| \leq \ln n \, |Opt|$

נוכיח: אחרי $\ln n \cdot |Opt|$ איטרציות, X התרוקן. עכשיו, אם $j = \ln n \, |Opt|$ אז $n[j+1] < 1$ (הצבה ב-(*))

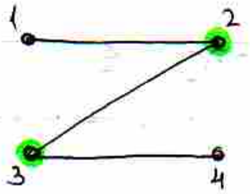
(11)

אלגוריתמי קירוב

בעית כיסוי קודקודי צלעות גרף (vertex cover)

נתון גרף לא מכוון $G=(V,E)$, $n=|V|$ ומחפשים קבוצה של קודקודים בגודל מינימלי $C\subseteq V$ כך שלכל צלע $e=(u,v)\in E$ מתקיים $C\cap\{u,v\}\neq\emptyset$, כלומר לפחות צד אחד של הצלע נמצא ב-C.

עד כמה שזה נשמע פשוט, לא ידוע אלגוריתם יעיל לפתור את הבעיה.



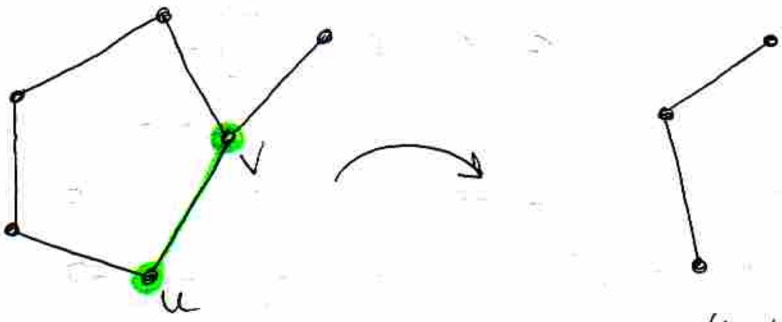
$C=\{2,3\}$

אלגו' מאוד הגיוני הוא לבחור בכל שלב את הקודקוד הבא שמכסה הכי הרבה צלעות.
כאשר בוחרים באופן איטרטיבי קודקודים בעלי דרגה גבוהה ביותר מקבלים קירוב עד כדי $O(\log n)$ ואפשר לבנות דוגמאות שבהן זה באמת התוצאה שמתקבלת ע' האלגוריתם. (ומה $\log n$? - בעצם מה שנעשה כאן הוא שימוש ב Set-Cover וראינו בשיעור הקודם שהוא $\log n$ - מקרב :)

אבל יש אלגוריתם פשוט עם יחס קירוב 2:
מתחילים עם $C=\emptyset$
בכל צעד בוחרים צלע $e=(u,v)$ כלשהי ב-E
ומוסיפים את u ואת v ל-C. $C\leftarrow C\cup\{u,v\}$.
לאחר מכן מורידים מ-E את כל הצלעות שכבר מכוסות.
ממשיכים עד ש-$E=\emptyset$.

טענה: האלגוריתם נותן יחס קרוב 2.

הוכחה: נשים ♥ שלאוסף הצלעות שנבחרו במהלך האלגוריתם אין קדקוד משותף. ולכן C אוסף צלעות זרות.



יהי C^* כיסוי קודקודים אופטימלי כלשהו. אזי ב-C^* חייב להיות נציג אחד לפחות מכל צלע שהכנסנו ל-C

$\Leftarrow$ $|C^*| \geq$ מספר הצלעות ב-C $= \frac{|C|}{2}$.

כלומר $|C| \leq 2|C^*|$.

זה האלגוריתם הטוב ביותר שידוע מזה 30 שנה.
ידוע שאם קיים אלגוריתם פולינומיאלי עם יחס קרוב $\frac{7}{6}$ אז כבר ניתן לפתור את הבעיה בצורה אופטימלית בזמן פולינומיאלי.

בעיית הסוכן הנוסע (traveling salesman)

יש סוכן שצריך לבקר בערים מסוימות ויש לו טבלה של מרחקים בין הערים. הוא רוצה לבקר בכל הערים ובסוף לחזור הביתה רק שהמרחק שהוא ינסע הוא מינימלי.

נתון אוסף של n ערים $v_1, \dots, v_n$ ומחירים עבור הנסיעה בין כל שתי ערים לכל $i \neq j$ $W(v_j, v_i) = W(v_i, v_j) = W_{ij} \geq 0$.

מסלול עבור הסוכן הוא פרמוטציה $\pi : \{1, \dots, n\} \circlearrowleft$.

$\pi(1), \dots, \pi(n)$ הוא סדר הערים שבהן הסוכן מבקר (כאשר הוא מתחיל ב-$\pi(1)$ ומ-$\pi(n)$ הוא חוזר ל-$\pi(1)$.

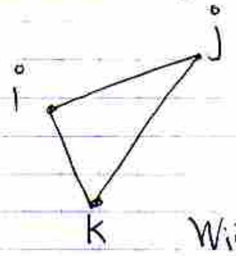
המחיר של π הוא $W(\pi) = W_{\pi(n), \pi(1)} + W_{\pi(n-1), \pi(n)} + \dots + W_{\pi(1), \pi(2)}$

(47)

אם אין שום הנחה על פונקציית המחיר ניתן לפתור את הבעיה באופן אופטימלי בזמן יעיל.

אבל רשים הנחות על המחיר הדברים מסתבכים.

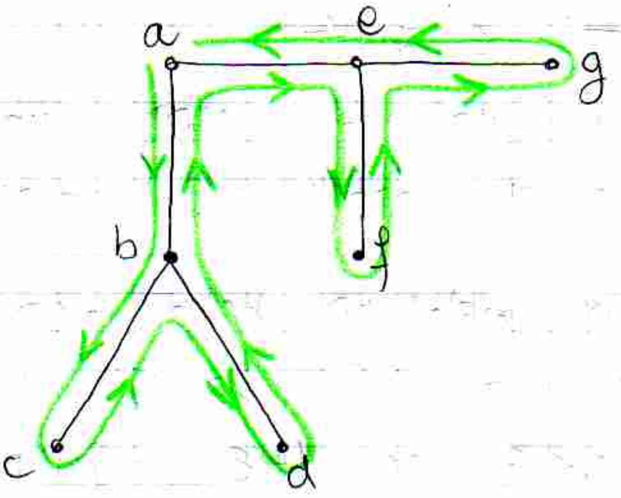
אנחנו נניח שפונקציית המחיר מקיימת אי שוויון המשולש לכל i,j,k $\quad W_{ij} + W_{jk} \ge W_{ik}$



אלגוריתם קירוב לבעיית הסוכן הנוסע

בהינתן הגרף על n קודקודים עם משקל W_{ij} לצלע בין i ו-j. נבנה עץ פורש מינימלי.

נבחר שורש לעץ ונבצע הילוך אצבעות על פני קודקודי העץ כאשר עוברים פעמיים על כל הקודקודים (אפילו יותר מפעם אחת) ובדיוק פעמיים על כל צלע.

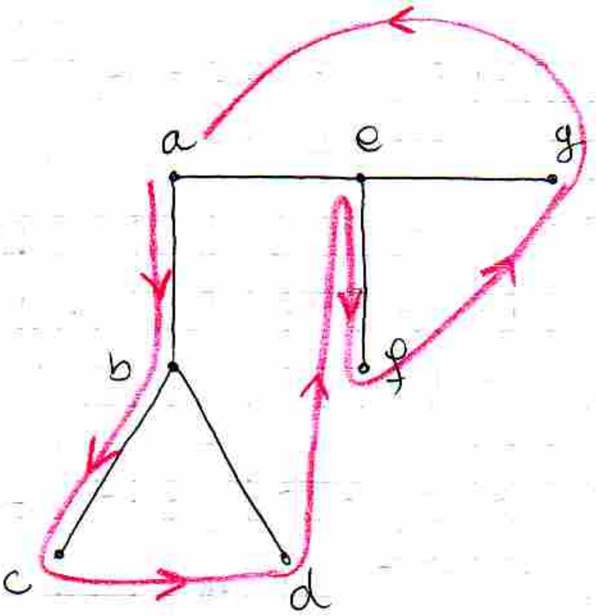


נסמן ב $w(T)$ את משקל העץ האופטימלי. אזי מחיר המסלול שלנו הוא בדיוק $2w(T)$.

אבל המסלול שלנו לא עומד בתנאים (הוא לא תמורה).

אז מהמסלול עם החזרות נוריד את הקודקודים המופיעים יותר מפעם אחת (חוץ מהחזרה לשורש).

בגלל אי שוויון המשולש נקבל מסלול T' שעובר דרך כל קודקוד פעם אחת וחוזר לשורש ומחירו לכל היותר $2w(T)$.



יהי H מסלול המטייל דרך כל קדקוד פעם אחת וחוזר לשורש. אזי $W(H) \geq W(T)$

כי כאשר מורידים מ-H צלע כלשהי נקבל עץ פורש שמשקלו קטן ממשקל $W(H)$.

$$\Rightarrow \quad W(\Pi) \leq 2W(T) \leq 2W(H)$$

בפרט זה נכון עבור H המסלול האופטימלי.

⇐ האלגוריתם שלנו מקרב פי יחס 2.

עבור בעיית הסוכן הנוסע בין נקודות במישור (מחיר = מרחק ולכן מתקיים אי שוויון המשולש) קיים אלגוריתם קירוב פולינומיאלי המביא $1+\varepsilon$ לכל $\varepsilon>0$ אבל הוא מתנהג רע מאוד כפונקציה של ε ומעניין רק מבחינה תיאורטית.

<u>בעיית השמה המספקת מספר מקסימלי של משוואות ליניאריות מעל $\mathbb{Z}_2$</u>

המקרה המעניין הוא כאשר מס' המשוואות אינו הומוגני וכן כל המשוואות שונות

$$\begin{cases} a_{11}x_1+\dots+a_{1n}x_n=b_1 \\ \vdots \\ a_{m1}x_1+\dots+a_{mn}x_n=b_n \end{cases}$$

אם יש פתרון אז בעזרת האלגוריתם של גאוס אפשר תוך $O(n^3)$ למצוא אותו.

אבל מה קורה אם אין פתרון? היינו רוצים למצוא השמה למשתנים $x_1,\dots,x_n \in \mathbb{Z}_2$ כך שנספק מספר מקסימלי של משוואות.

אם בוחרים השמה מקרית באופן אחיד אז ההסתברות שמשוואה כלשהי תתקיים היא בדיוק $\frac{1}{2}$ (קיום המשוואה תלוי במשתנה האחרון, ויש לו סיכוי $\frac{1}{2}$ להיות 0 וסיכוי $\frac{1}{2}$ להיות 1).

נגדיר משתנה מקרי

$$X_l = \begin{cases} 1 & \text{המשוואה ה-}l\text{ מתקיימת} \\ 0 & \text{אחרת} \end{cases}$$

(לא קשור - משתנים שונים.)

לכל l $\quad E[X_l] = \frac{1}{2} \Rightarrow$

נגדיר משתנה מקרי $Y = \sum_{l=1}^{m} X_l \Rightarrow E[Y] = \frac{m}{2}$

אבל ב-$\mathbb{Z}_2$ שתי משוואות שונות הן בלתי תלויות לינארית. התלות הלינארית הזאת מתבטאת בכך שהמשתנים המקריים X_l הם בלתי תלויים בזוגות וזה מאפשר לנו להשתמש באי שוויון צ'בישב... אז נקבל תוצאה שלא רק שהתוחלת היא $\frac{m}{2}$ אלא שגם סטיית התקן היא קטנה. וזה נותן לנו יחס קרוב $\frac{1}{2} - \varepsilon$.

(49) 18.01.07
אלעד

אלגוריתמים מקביליים

יש בעיה בגודל n ויש כמה מעבדים שבא לנו (יכול להיות תלוי ב-n).
יש לנו גם זיכרון יחיד בדומה למחשב עם מעבד אחד וכל המעבדים
יכולים לקרוא ולכתוב לזיכרון הזה. נסמן את המעבדים ב-$\{P_i\}$.
במחזור פעולה של המכונה המקבילית מתקיים הדבר הבא:

(I) כל מעבד יכול לקרוא תא מהזיכרון המשותף

(II) כל מעבד מבצע צעד חישוב

(III) כל מעבד כותב לתא בזיכרון המשותף.

המודל הזה נקרא PRAM (Parallel RAM)

יש כאן בעיה: מה קורה כשהרבה מעבדים רוצים לכתוב לאותו תא?
יש כמה אופציות לטפל בזה-

- על המתכנת לדאוג שלא תהיה כתיבה לאותו מקום אחרת המחשב מתפוצץ.
- אפשר שתהיה כתיבה משותפת בתנאי שהערך שרוצים לכתוב זהה לכל המעבדים
- אם יש למעבדים מספרים סידוריים אז אפשר רק למעבד עם המספר הנמוך ביותר לכתוב (זה גם מעשי כי כשבונים ממשים מק' מרובות מעבדים אז לא באמת יש זיכרון אחד, אלא לכל מעבד יש זכרון משלו והם מחוברים ברשת. ואז כשמידע עובר ברשת אפשר לקבוע את זה שמגיע מהמחשב עם המספר הנמוך ביותר ולעצור את השאר).

דוגמה: חיבור של n מספרים בעזרת $n/2$ מעבדים (בה"כ $n=2^k$).
נתונים n מספרים $x_1,\dots,x_n$ כאשר x_k נמצא בתא k בזיכרון.

x_1	x_2	...	x_k	...	x_n

נחלק את המספרים לזוגות x_{2k-1}, x_{2k}. המעבד P_k קורא את
המספרים x_{2k-1}, x_{2k}, מבצע את פעולת החיבור ומקבל תוצאה y_k.

ואז כותב בשלב הראשון את התוצאה a_k לתא k במערך, עבור $k=1,\dots,\frac{n}{2}$.
בזמן $O(1)$ וקבלנו מבעיה שגודלה n לבעיה בגודל $\frac{n}{2}$ ולכן אחרי $\log n$
צעדים כאלה נקבל את הסכום בתא המערך 1.
סה"כ מספר הפעולות - בזמן t יש $\frac{n}{2^t}$ פעולות וסה"כ $n-1$ פעולות.

טוב, אז זה מודל טוב ונחמד אבל הוא לא מציאותי בכלל כי אף אחד לא נותן לנו מספר לא מוגבל של מעבדים. בחיים האמיתיים יש מספר סופי של מעבדים וזהו!

<u>למה:</u> אם קיים אלגוריתם מקבילי לבעיה שרץ בזמן T ומספר לא מוגבל של מעבדים וסה"כ מספר הפעולות הוא W, אזי ניתן לבצע את החישוב במערכת k מעבדים במגבלת זמן $\frac{W-T}{k}+T$

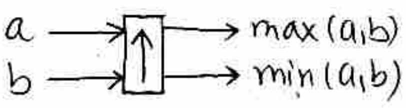
<u>הוכחה:</u> בשלב l אנו מבצעים במקביל W_l פעולות ומתקיים $W=W_1+W_2+\dots+W_T$. ניתן להתחיל לבצע את W_{l+1} אחרי השלמת W_l. W_l פעולות אפשר לבצע ע"י k מעבדים בזמן $\left\lceil\frac{W_l}{k}\right\rceil$ (שהרי קודם לכן הן התבצעו במקביל אז בוודאי גם עם k מעבדים אפשר במקביל). ואז

$$\sum_{l=1}^{T}\left\lceil\frac{W_l}{k}\right\rceil \le \sum_{l=1}^{T}\left(\frac{W_l}{k}+1\right) \le T+\frac{W-T}{k}$$

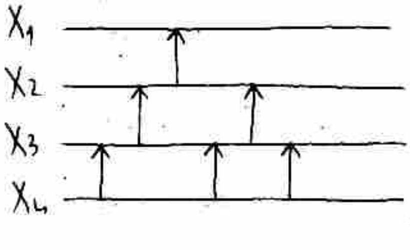
הבעיה עם אלגוריתמים מקביליים היא שיש מלא אלגוריתמים שהם אינם סדרתיים ובכלל לא ברור אם אפשר למקבל אותם.

<u>~~##~~ מקבול בעזרת רשת מיון</u>

אנו מקבלים כקלט n מספרים $x_1,\dots,x_n$
הפעולה הבסיסית שנבצע היא זו



לדוגמא, הנה מיון מאוד לא יעיל:
זה פשוט מיון בועות ובכלל
לא ניצלנו פה את המקביליות.



(50)

בכל צעד ניתן לבצע עד $\frac{n}{2}$ השוואות וידוע שבשלבו מיון שמבוסס על השוואות צריך לבצע לפחות $O(n\log n)$ השוואות לכן חסם תחתון מיידי למספר השלבים למיון בעזרת רשת מיון הוא $O(\log n)$ עם קיומה רשתות כאלה רק שהקבועים שלהן די גדולים והן מסובכות אז אנחנו לא נדון בהן.

נראה רשת מיון בעומק $O(\log^2 n)$ Odd-Even-Merge.

<u>למה:</u> בהינתן תאור של רשת מיון, הרשת פועלת נכון וממיינת כל פרמוטציה אמ"מ היא ממיינת נכון את הקלטים $x_1,\dots,x_n$ כאשר $x_i\in\{0,1\}$ לכל i.

<u>הוכחה:</u> נשים ♥ שאם רשת מיון פועלת על קלט $a_1,\dots,a_n$ אזי אם $f:\mathbb{R}\to\mathbb{R}$ פונקציה מונוטונית עולה ($x\le y \Rightarrow f(x)\le f(y)$) אזי הרשת פועלת באופן זהה על הסדרה $f(a_1),\dots,f(a_n)$, פשוט כי זה נכון לכל צעד בסיסי של השוואה ואז באינדוקציה זה נכון לכל הרשת.

נניח בשלילה שעבור קלט $a_1,\dots,a_n$ הפלט a_i עובר למקום נמוך יותר מ- a_j למרות ש- $a_i > a_j$. (נגדיר פונקציה $f:\mathbb{R}\to\mathbb{R}$ ע"י

$$f(x)=\begin{cases}0 & x\le a_j\\ 1 & \text{אחרת}\end{cases}$$

כי בוודאי פונקציה מונוטונית ומתקיים $f(a_i)=1$, $f(a_j)=0$. אם נפעיל את הרשת על $f(a_1),\dots,f(a_n)$ נקבל ש- $f(a_i)=1$ עובר למקום נמוך יותר מאשר $0=f(a_j)$ וזה בסתירה להנחה שהרשת ממיינת נכון כל סדרה ב- $\{0,1\}^n$. (11)

למיון בעזרת מיזוגים יש $\log n$ שלבים ואם נראה כיצד ניתן למזג רשימות ממוינות בעזרת רשת מיון יעילה נקבל בסה"כ רשת מיון טובה.

<u>מיזוג באמצעות בעזרת רשת מיון</u> (M חזקה של 2)

נתונות שתי רשימות ממוינות בסדר עולה $a_0,a_1,\dots,a_{M-1}$

$b_0,b_1,\dots,b_{M-1}$

נפריד את הרשימות למקומות הזוגיים והאי-זוגיים

$a_0,a_2,\dots,a_{M-2}$ $\quad$ $a_1,a_3,\dots,a_{M-1}$

$b_1,b_3,\dots,b_{M-1}$ $\quad$ $b_0,b_2,\dots,b_{M-2}$

נמזג את הרשימות הזוגיות עם הרשימות האי-זוגיות ונקבל

$c_0, c_1, \ldots, c_{M-1}$ $\qquad$ $d_0, d_1, \ldots, d_{M-1}$

נמזג את שתי אלה באופן הבא:

$c_0, d_0, c_1, d_1, \ldots, c_{M-1}, d_{M-1}$

נותר לבצע במקביל השוואה בין כל ה- c_k, d_k וסיימנו!

מספר השלבים למיזוג רשימות באורך M הוא $\log M$.

דוגמה: $A = 2,3,4,8$ $\quad$ $odd(A) = 3,8$ $\quad$ $even(A) = 2,4$

$B = 1,5,6,7$ $\quad$ $even(B) = 1,6$ $\quad$ $odd(B) = 5,7$

$\Downarrow$ $\qquad$ $\Downarrow$

$C = 2,4,5,7$ $\qquad$ $D = 1,3,6,8$

$\Downarrow$

2 1 4 3 5 6 7 8

12 34 56 78

OMG! It works!

נראה שהשיטה הזו עובדת. באופן דומה ללמה הקודמת ניתן להוכיח שדי להוכיח את הטענה עבור רשימות ממוינות A,B שמכילות רק איברים מ- $\{0,1\}$

$$A = \overbrace{0 \ldots 0}^{a} 1 \ldots 1$$

$$B = \overbrace{0 \ldots 0}^{b} 1 \ldots 1$$

אם מספר האפסים ב-A הוא a ומספר האפסים ב-B הוא b אזי מספר האפסים ב- $even(a)$ הוא $\lceil \frac{a}{2} \rceil$ ומספר האפסים ב- $odd(A)$ הוא $\lfloor \frac{a}{2} \rfloor$ וכן עבור B. לכן מספר האפסים ב-C הוא $c = \lceil \frac{a}{2} \rceil + \lfloor \frac{b}{2} \rfloor$ ומספר האפסים ב-D הוא $d = \lfloor \frac{a}{2} \rfloor + \lceil \frac{b}{2} \rceil$. נשים ♥ שההפרש בין c ו-d הוא לכל היותר 1. אם $c=d$ או $c=d+1$

$c_0 d_0 c_1 d_1 \ldots c_k d_k \ldots c_{M-1} d_{M-1}$

$c=d$ $\quad \underbrace{00 \ldots 0}_{2c} 11 \ldots 1$

$c=d+1$ $\quad \underbrace{0 \ldots 0}_{2d} 01 \ldots 1$

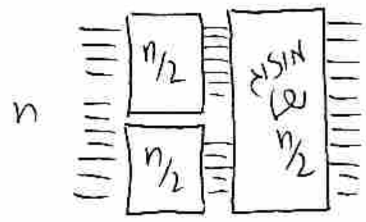
} בשני המקרים הללו ברור שהכל בסדר.

במקרה $c=d-1$ נקבל $\underbrace{00 \ldots 0}_{2c}|10|1 \ldots 1$ וההשוואה הסופית תתקן את המצב.

לכן המיזוג עובד על שתי סדרות ממוינות.

<u>מסקנה</u>: ניתן לבנות רשת מיון למיון $n=2^k$ מספרים בעומק שהוא $O(\log^2 n)$, ומספר הפעולות יהיה סה"כ $O(\frac{n}{2}\log^2 n)$.

<u>הוכחה</u>: מיזוג רשימות באורך M דורש עומק $\log M$ לכן נעשה "נכון": נבנה רשת מיון רקורסיבית אחרי האלגוריתם למיון בעזרת מיזוגים ונקבל



$$T(n) = T(\tfrac{n}{2}) + \log\tfrac{n}{2} =$$
$$= \log\tfrac{n}{2} + \log\tfrac{n}{4} + \dots \log 2 =$$
$$= \frac{(\log n)(\log n - 1)}{2} \le \frac{\log^2 n}{2}$$

בכל שלב יש $n/2$ פעולות ⇐ מספר הפעולות הוא $O(\frac{n}{2}\log^2 n)$. ☺

זה מודל מאוד מוגבל של חישוב מקבילי. בשיעור הבא נדון במודל חזק יותר שבו אפשר לעשות דברים.

(52) 22.1.07
אוגו'

דוגמאות פשוטות:

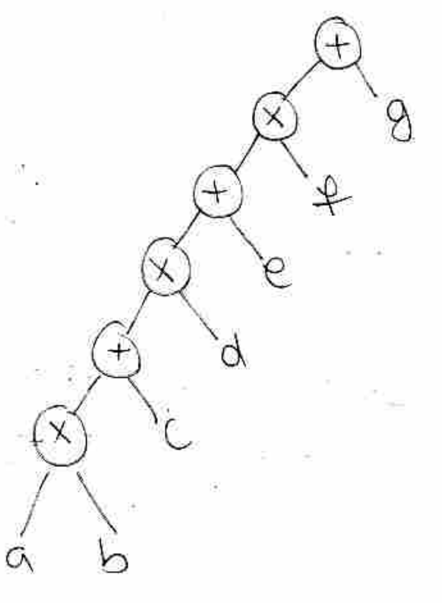
- חישוב FFT עם $n=2^k$ איברים בעזרת n מעבדים.
את האלגוריתם הסדרתי שתיארנו ניתן למקבל בצורה ברורה.
פרקנו בעיה מסדר n לשתי בעיות בלתי תלויות מסדר $\frac{n}{2}$
וכל אחת מ-n התוצאות הסופיות התקבלה בעזרת מספר
קבוע של פעולות.
אם $T(n)$ הזמן המקבילי אז $T(n) = T(\frac{n}{2}) + O(1)$
שזה נותן לנו $O(\log n)$.

- כפל מטריצות מסדר $n \times n$ - כל איבר במכפלה ניתן לחשב בזמן
$O(\log n)$ בעזרת n מעבדים, ולכן בעזרת n^3 מעבדים ניתן
להכפיל מטריצות בזמן $O(\log n)$.

- ניתן לחשב דטרמיננטה מסדר $n \times n$ בזמן $O(\log^2 n)$
ובעזרת $O(n^4)$ מעבדים. וכן ניתן לבדוק בעזרת
האלגוריתם ההסתברותי שתיארנו האם קיים זיווג מלא בגרף
דו-צדדי.

- נניח שרוצים לחשב את $(((a\times b+c)\times d+e)\times f+g)\times \dots$
אפשר לייצג את זה בצורת עץ:

צורת מסרק



לא ברור בכלל איך אפשר לקדד את החישוב הזה

(בניגוד לכל פעולה אסוציאטיבית שניתן לחשב בקלות ב-$O(\log n)$ למשל $a_1 + \dots + a_n$).

באופן כללי, נתונים משתנים $x_1, \dots, x_n, \dots$.

ביטוי אלגברי מעל שדה F במשתנים הללו הוא:

(I) איבר מהשדה } ביטויים אלגבריים מגודל 1

(II) x_i כלשהו

(III) אם Ψ, φ ביטויים אזי $(\varphi + \Psi)$ ו-$(\varphi \times \Psi)$

ביטויים אלגבריים שגודלם סכום הגדלים של φ ו-Ψ.

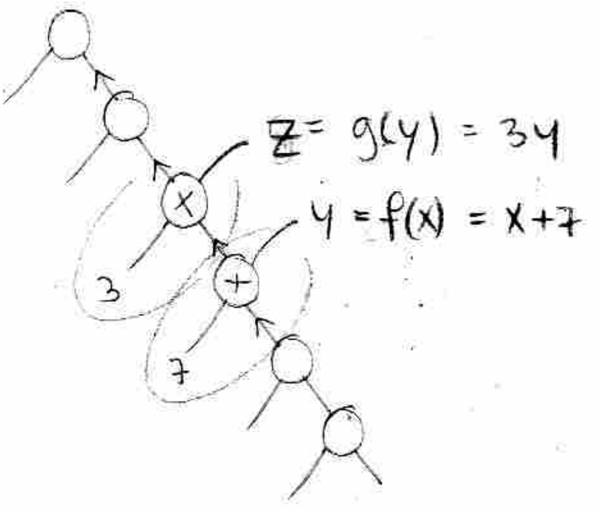
נשים ♡ שביטוי בגודל n ניתן לחשב באופן סדרתי בעזרת חישוב של $n-1$ פעולות חיבור/כפל.

לכל ביטוי ניתן להתאים באופן טבעי עץ בינארי.

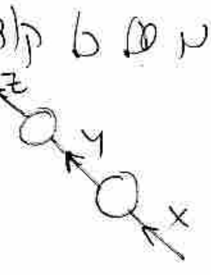
למשל לביטוי $\varphi + \Psi$ מתאים

מהירות ביטוי שמתאים לו עץ בגובה h ברור שניתן לחשב אותו בזמן h בעזרת לכל היותר n מעבדים.

חישוב מקבילי של ביטויים בצורת מסרק:



נתבונן על כל קדקוד כפעולה לינארית. כל קלט אחד עם מקדמים ידועים.

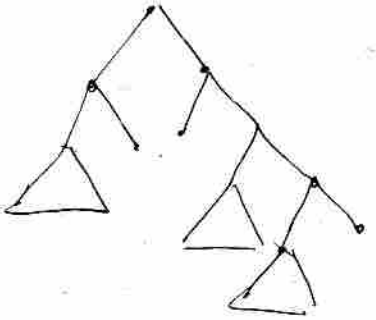


$$y = ax + b = f(x), \quad z = cy + d = g(y)$$

$$\Rightarrow z = g(f(x)) = c(ax+b) + d = (ca)x + (cb+d)$$

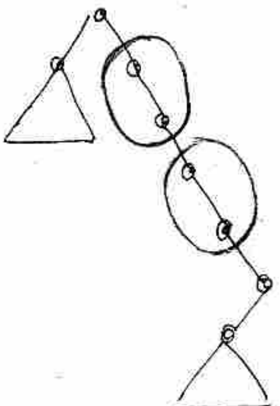
הרכבה של שתי פונקציות לינאריות נותנת פונקציה לינארית וחישוב מקדמי ההרכבה דורש 3 פעולות ושתי יחידות זמן.
אז בעזרת הרכבת זוגות עוקבים מהשורש ועד לפעולה האחרונה ניתן ב- $O(1)$ זמן ; $O(n)$ מעבדים לעבור למסרק חדש באורך $\lceil \frac{n}{2} \rceil$
אחרי $\log n$ חזרות נקבל פונקציה לינארית אחת וקלט לפונקציה הזאת ונסיים את החישוב ב- $O(1)$ זמן.
נעיר רק שאם יש גם פעולת חילוק הפונקציות הן מהצורה $f(x) = \frac{ax+b}{cx+d}$ ואפשר לחשב בקלות ע"י כפל מטריצות 2x2 את המקדמים של הרכבה של פונקציות מהצורה הזאת.
כמו כן את התהליך שעשינו ניתן לעשות לעץ עם פעולות בוליאניות כלשהן - למשל פעולות לוגיות.

עכשיו נניח שהביטוי שלנו הוא לא מסרק אלא משהו כללי.
באופן כללי נוכל לחשב רק:



(I) קיצוץ עלים (Rake)
אם לקודקוד יש שני בנים שהם עלים, ניתן למחוק את העלים והקודקוד עצמו הופך לעלה שהוא תוצאת החישוב.
אם לקודקוד יש בן אחד שהוא עלה ובן שני שהוא שורש של תת עץ כלשהו, ניתן לקצץ את העלה והקודקוד הופך לפונקציה לינארית ויש לו רק בן יחיד.

(II) כיווץ מסלולים (compress)



הרכבת שתי פונקציות לינאריות עוקבות לפונקציה אחת.
נגדיר דרך לעבור זוגות: נרוץ לאורך המסלולים מהשורש לעלים, נחבר זוגות בצורה חמדנית בהתאם לקודקוד שנתקלים בו קודם.

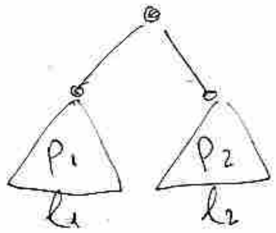
בכל שלב הריצה הופכים כל זוג קודקודים לקודקוד בודד בעזרת חישוב הרכבת הפונקציות הלינאריות.

לאחר שלב הגיזום הראשון יש לנו עצים בינאריים עם קודקודים פנימיים עם בן אחד ושני בנים ועלים שהם קודקודים ללא בנים.

<u>טענה:</u> יהי T עץ בינארי בן n קודקודים שיש בו l עלים ו- p קודקודים פנימיים עם שני בנים. אזי $l = p+1$.

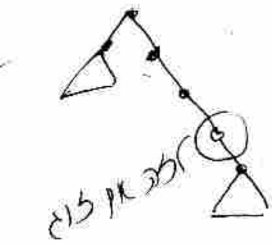
<u>הוכחה:</u> באינדוקציה על גובה העץ. בעץ בגובה 0 יש קודקוד אחד שהוא השורש והוא גם עלה. אז $p=0$, $l=1$.

צעד האינדוקציה:



$$l = l_1 + l_2 = p_1+1+p_2+1 = (p_1+p_2+1)+1 = p+1$$

נניח שבעץ שלנו יש n קודקודים, l עלים ו- p קודקודים פנימיים עם שני בנים ובשלב זה של הריצה ישנן r זוגות.
יהי h מספר הקודקודים עם בן יחיד שלא נמצא להם בן זוג. לכן מתאים לו באופן יחיד קודקוד שהוא עלה או קודקוד עם שני בנים. $\Leftarrow h \le l+p$.



מספר הקודקודים שאנו מצליחים להוריד מהעץ בפעולה של Rake + Compress הוא $l+r$ כי מורידים l עלים ומ-2r קודקודים נקבל רק r קודקודים בעץ החדש.

$$n = l+p+2r+h \le 2(l+p)+2r < 4l+2r \le 4(l+r)$$

לכן לאחר לכל היותר $O(\log n)$ חזרות נוכל לחשב את ערכו של הביטוי. מחיקת כל קודקוד דורשת לכל היותר 3 פעולות לכן בסה"כ מכיוון שמספר הפעולות הכללי $\le 3n$ ולמעשה ניתן להראות חסם של $\frac{3}{2}n$ לכן בעזרת $\frac{n}{\log n}$ מעבדים ניתן לחשב את הביטוי בזמן $O(\log n)$.

(54) 25.1.07
אלגו'

אלגוריתמים מקוונים   on-line algorithms

בעיית הסקי: אנחנו הולכים לגלוש ואין לנו ציוד. אנחנו לא יודעים אם נאהב את זה או לא ומתי נפסיק. כל יום אנחנו מחליטים אם לשכור ציוד או לקנות ציוד.
קל לתת פתרון מקוון הדורש לכל היותר מחיר כפול מן הפתרון האופטימלי של מישהו שיודע את העתיד - שוכרים עד שעלות השכירה מסתכמת לעלות קנייה ואז קונים.

ניהול זיכרון מטמון - cache - לתכניות יש סדרה של בקשות לקריאה מהזיכרון ויש לנו זיכרון מהיר (מטמון) בגודל k.
אנחנו רוצים לנהל את הזיכרון המהיר בדרך הטובה ביותר.
לאחר מילוי הזיכרון המהיר כל פנייה לתא זיכרון שאינו שם מחייבת את הבאת התא הזה לזיכרון המהיר ופינוי של איבר כלשהו מהזיכרון המהיר לזיכרון האיטי (cache miss).
אם יודעים מראש את כל הסדרה אז הפתרון האופטימלי הוא לזרוק מהזיכרון המהיר את האיבר שצריכים לו הכי רחוק בעתיד.

מהניסיון המעשי את כל ה - LRU - (last recently used)
זורקים את האיבר שפנינו אליו לאחרונה הכי רחוק בעבר (אבל קשה לחשב את זה בפועל).
בפועל משתמשים בקירוב ל- LRU - <u>אלגוריתם סימון</u>.
מסמנים כל איבר בזיכרון המהיר אליו אנו פונים. אם כל האיברים מסומנים מורידים את כל הסימנים.
<u>בכל החלפה</u>: בוחרים איבר כלשהו שאינו מסומן ומחליפים אותו.

<u>טענה</u>: המחיר לכל אלגוריתם סימון על כל סדרת פניות לזיכרון הוא לכל היותר $k \times$ המחיר האופטימלי.

~~הוכחה: אלגוריתמי סימון פועלים במחזורים – כל פעם שמוחקים את כל הסימונים.~~

נתבונן במהלך האלגוריתם מאז מחיקת כל הסימונים למחיקה הבאה.

<u>טענה</u>: לכל אלגוריתם החלפה דטרמיניסטי קיימת סדרה שעבורה המחיר יהיה פי k מהמחיר האופטימלי.

באלגוריתם סימון הסתברותי צריכים לבחור מקרי לא מסומן. תוחלת היחס לסדרה הגרועה ביותר הינו $\log k$.

בנו כזו מכונה כזאת (ע"י אינטל) אבל לא התברר כנסיון מוצלח. הסיבה לכך היא שהרבה אלגוריתמים מסתמכים על כך שיודעים מה מצב הזיכרון. ושולפים משם באופן מקרי לא פוגע בהנחות של האלגוריתמים!

<u>מבחן</u>

כמו שנים קודמות.