

23.10.2006

מנהלות:

לא תמיד יהיה שיעור ביום ב'. כשאין – יודיעו, אחרת יש.

בדיקת תרגילים – יבדקו, אבל לא בדיקה טובה.

תרגילים – 10% מהציון. הציון על כל התרגילים – אבל ינתן פקטור, שיגרום לזה שלפחות 50% מהמגישים יקבלו את כל 10 הנקודות.

נראה מעט אלגוריתמים מקביליים, ובפרט נראה כיצד n מעבדים יכולים למזג שתי רשימות באורך n בזמן $O(\log n)$.הערה: קצת חזרה על מיון מספרים הושמטה עקב עייפות המקלד. עקב חריצותה של דינה <http://notes-heaven.intertent.net>, אפשר להיות סמוכים ובטוחים שיש את החומר הזה אצלה.בקצרה – ניתן למיין n מספרים בעלות של $T(n) = O(n \log_2 n)$. אבל איך עושים את זה אם יש לנו יותר ממעבד 1? נניח, עד n מעבדים?האלגוריתם של איחוד 2 קבוצות המספרים הממוינות הוא לינארית (אחרי החילוק ל-2 ומיון שלהם). זו בעיה. כדי שנגיע לפרקים הבאים – הפתרון יהיה באחד מהשיעורים הבאים. ואז, אם יש לנו n מעבדים,אזי אפשר למצוא אלגוריתם בעלות של $O((\log n)^2)$. אפשר גם בזמן $O(\log n)$.**רדוקציה לבעיה שיש לה אלגוריתם יעיל**

דוגמא: נתונים n מספרים $a_1, \dots, a_n$ ומספר נוסף b . צריך לבדוק אם יש שני מספרים $1 \leq i < j \leq n$ כך ש $a_i + a_j = b$. בדיקת כל האפשרויות – $O(n^2)$ – נמיינ את $a_1, \dots, a_n$ ב $O(n \log n)$ ולכן ברור $a_1 \leq \dots \leq a_n$ ואז בזמן $O(n)$ לפתור את הבעיה (איך? עבור כל a_i מחשבים $a_j = b - a_i$ ואז בודקים האם a_j קיים) (בזמן $O(\log n)$ - חיפוש בינארי).

מה קורה כאשר רוצים לבדוק האם קיימת קבוצה חלקית $S \subseteq \{1, \dots, n\}$ כך ש $b = \sum_{i \in S} a_i$? זה עולה ב- 2^n .

שאלה:

מכפלה סקלרית של $(a_1, \dots, a_n)(b_1, \dots, b_n) = \sum_{i=1}^n a_i b_i$. כמה זה עולה? n פעולות כפל, ו- $n-1$ פעולות חיבור.

אבל, נניח ויש לנו את $(c_1, \dots, c_m) = (a_1, \dots, a_n) \begin{pmatrix} b_{11} & \dots & b_{1n} \\ \vdots & \ddots & \vdots \\ b_{m1} & \dots & b_{mn} \end{pmatrix}$ גם כאן, הפתרון הטריטואלי הוא האידיאלי.

אבל, כאשר רוצים להכפיל שתי מטריצות, מצאו פתרון יותר טוב – ב $O(n^{2.3})$ – והיד עדיין נטויה (הפתרון הטריטואלי הוא $O(n^3)$).

שטראסן (Strassen), 1969, הראה כי אפשר ב $O(n^{\log_2 7})$ – ראה http://en.wikipedia.org/wiki/Strassen_algorithm. בהמשך מצאו אלגוריתם בעלות של

$O(n^{2.36})$ - ראה http://en.wikipedia.org/wiki/Coppersmith-Winograd_algorithm . חסם
תחתון ידוע הוא $O(n^2)$ (ליתר דיוק - $3.5n^2$).

אלגוריתמים חמדניים – Greedy Algorithms

נתחיל עם דוגמא לאלגוריתם. פרצת לחנות - ואתה יכול לגנוב עד משקל w . נתונים n פריטים,

$$(w_1, v_1), \dots, (w_n, v_n), \text{ כאשר מוגדר } w_i \text{ המשקל של הפריט, ו } v_i \text{ ערכו.}$$

צריך להחליט איזה פריטים לוקחים, כלומר – מחפשים סדרה $x_1, \dots, x_n$, כאשר $x_i \in \{0, 1\}$ (מסמן

אם לקחנו אותו או לאו). צריך להתקיים $\sum x_i w_i \leq w$, ו $\sum x_i v_i$ מקסימלי. הבעיה נקראת **בעיית**

התרמי. אנו נפתור את בעיית התרמיל השברית – כאן אנו רוצים $x_1, \dots, x_n$ המקיימים $0 \leq x_i \leq 1$ כך

שמתקיימים הכללים ממקודם.

דוגמא: $(30, 120), (20, 100), (10, 60)$. נגדיר $r_i := \frac{v_i}{w_i}$ (ה'ערך' הסגולי של האיבר i).

$$\left(\frac{2}{3}, 1, 1\right). \text{ בדוגמא זו, עבור } w = 5, \text{ וקטור הבחירה יהיה } \left(\frac{2}{3}, 1, 1\right).$$

הערה: הבעיה השברית היא מקרה פרטי (פשוט במיוחד) של תכנון ליניארי, וקיים אלגוריתם יעיל פולינומיאלי לפתרון בעיות תכנון ליניארי כללי.

אלגוריתם חמדני, במסודר:

נגדיר $r_i = \frac{v_i}{w_i}$. נמיין את r_i בסדר יורד. בה"כ נניח $r_1 \geq r_2 \geq \dots \geq r_n$. כל עוד אפשר בוחרים

$x_i = 1$ החל מ $i = 1$ עד שנקבל שאי אפשר יותר, כלומר $\sum_{j=1}^{i-1} w_j < w$ אבל $\sum_{j=1}^i w_j > w$, ואז ניקח

$$x_i = \frac{\left(w - \sum_{j=1}^{i-1} w_j\right)}{w_i}. \text{ נגדיר את כל ה } x_i \text{ הבאים כ } 0.$$

נשים לב שהאלגוריתם הזה לא אופטימלי במקרה ואסור לנו לחלק לשברים.

הוכחת נכונות: $0 \leq x_i \leq 1$. צריך להראות רק במקרה בו השתמשנו ב $x_i = \frac{\left(w - \sum_{j=1}^{i-1} w_j\right)}{w_i}$.

$$0 \leq x < 1 \text{ בגלל התנאי של בחירתו, וזה גם ולכן } 0 \leq \left(w - \sum_{j=1}^{i-1} w_j\right) < w_i.$$

כעת נראה כי מתקיים $\sum_{l=1}^n x_l w_l \leq w$.

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

(נשים לב כי אנו מניחים ש $\sum_{l=1}^n w_l > w$, אחרת הפתרון האופטימלי הוא הטריביאלי – לקחת הכל).

הוכחת אופטימליות:

למה: נתבונן בפתרון $y_1, \dots, y_n$ שעבורו $y_1 w_1 < w$ ובנוסף $y_1 < 1$, אזי קיים פתרון $x_1, \dots, x_n$ כך

ש $x_1 > y_1$ ו $\sum_{l=1}^n x_l v_l \geq \sum_{l=1}^n y_l v_l$ (בגלל סדר ה r_i), בסתירה לאופטימליות.

כיוון ש $y_1 w_1 < w$ יש y_i אחר גדול מאפס. בה"כ $0 < y_2$. נגדיר $x_3 = y_3, \dots, x_n = y_n$, ונבחר

$$\varepsilon > 0 \text{ כך ש } x_1 := y_1 + \varepsilon \leq 1, \text{ וגם } \varepsilon \frac{w_1}{w_2} < y_2. \text{ אז } 1 \geq x_2 = y_2 - \varepsilon \frac{w_1}{w_2} \geq 0.$$

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

כלומר, מבחינת המשקל – אנחנו סבבה.

בואו נראה את הערך:

$$\sum_{i=1}^n x_i v_i - \sum_{i=1}^n y_i v_i = (x_1 - y_1) v_1 + (x_2 - y_2) v_2 = \varepsilon v_1 - \varepsilon \frac{w_1}{w_2} v_2 =_{v_i = r_i w_i}$$

$$\varepsilon r_1 w_1 - \varepsilon \frac{w_1}{w_2} r_2 w_2 = \varepsilon w_1 (r_1 - r_2) \geq 0$$

הוכחת האופטימליות של האלגוריתם החמדני:

מהלמה, ברור שבחירת $x_1 = 1$ (אם אפשר) לא מקלקלת את האופטימליות. ולכן ניתן לבחור $x_1 = 1$.

נגדיר $w' = w - w_1$ ובעיה דומה בגודל $n - 1$. אם נמשיך רק נקבל את הפתרון החמדני.

בעיית שיבוץ משימות:

נתונים קטעים $(s_1, f_1), \dots, (s_n, f_n)$. ורוצים לבחור קבוצה חלקית של קטעים שאינם נחתכים. רוצים

לבחור מספר מקסימלי של משימות כאלה בכל שלב.

3 אופציות:

1. בוחרים את הקטע הקצר ביותר שניתן לבחור.

2. בוחרים את הקטע שנקודת הקצה שלו f_i קטנה ביותר מבין הקטעים הנותרים.

3. בוחרים את הקטע שמתנגש עם הכי מעט קטעים אחרים.

יש דוגמאות נגדיות לאופטימליות של פתרונות 1 ו-3 (באתר של דינה – <http://notes-heaven.intertent.net>). אבל 2 עובד.

טענה: האלגוריתם החמדני לפי שיטה (2) מוצא פתרון אופטימלי.

הוכחה: יהי $(s_{i_1}, f_{i_1}), (s_{i_2}, f_{i_2}), \dots, (s_{i_k}, f_{i_k})$ פתרון אופטימלי המכיל k קטעים זרים באשר $f_{i_1} \leq f_{i_2} \leq \dots \leq f_{i_k}$. יהי i_0 כך ש $f_{i_0} = \min \{f_i \mid 1 \leq i \leq n\}$. כמובן ש $f_{i_0} \leq f_{i_l}$ עבור $l = 1, \dots, k$.

טענה: גם הקטעים $(s_{i_0}, f_{i_0}), (s_{i_2}, f_{i_2}), \dots, (s_{i_k}, f_{i_k})$ הם קטעים זרים (לשים לב לאינדקסים...).

אם היה חיתוך אזי, $f_{i_0} < s_{i_2} \leq f_{i_2}$ - בסתירה למינימליות של f_{i_0} .

הראנו שהצעד הראשון של האלגוריתם החמדני לא מקטין את האופטימום ולכן על הקבוצה שנותרת יש אופטימום בגודל $k-1$, ובאינדוקציה האלגוריתם החמדני ימצא פתרון אופטימלי בגודל k .

בעיה נוספת: מציאת קבוצה בלתי תלויה מקסימלית. נתונה קבוצה של וקטורים $v_1, \dots, v_n \in F^m$.

שדה כלשהו, ומשקלות חיוביים $0 \leq w_1, \dots, w_n$. נחפש קבוצת וקטורים עם סכימת משקלות מקסימלית.

מחפשים קבוצה $S \subseteq \{1, \dots, n\}$ כך ש $\{v_i \mid i \in S\}$ בלתי תלויים לינארית מעל F

$$w(S) = \sum_{i \in S} w_i$$

אלגוריתם חמדני טריביאלי: מיון ע"פ משקלות, לוקחים את האחד הכי גדול, מסירים את כל התלויים לינארית, וממשיכים.

דוגמא נגדית (בעצם עובדת) – זה כן עובד:

(1,1,1,1) - 4

(0,0,0,1) - 3

(0,0,1,0) - 3

(0,1,0,0) - 3

(1,0,0,0) - 3

נבחר בדוגמא זו את כל הוקטורים פרט לאחרון.

נתבונן באוסף הבא:

$$F = \{A \subseteq \{1, \dots, n\} \mid \{v_i \mid i \in A\} \text{ Are Not Linear Dependent}\}$$

אוסף הקבוצות F מקיים:

1. אם $A \in F$, אז $B \subseteq A$ - $B \in F$ (תורשתית Hereditary).

2. אם $A, B \in F$, וגם $|A| < |B|$, אזי קיים $u \in B$ כך ש $A \cup \{u\} \in F$ (תכונת

ההחלפה).

אלגוריתמים חמדניים

נתונים n וקטורים, $v_1, \dots, v_n \in F^m$ לכל וקטור v_i יש משקל $w_i > 0$. רוצים למצוא קבוצת וקטורים $A \subseteq \{1, \dots, n\}$ כך ש $\{v_i \mid i \in A\}$ קבוצה בלתי תלויה לינארית, והמשקל מקסימלי,

$$w(A) = \sum_{i \in A} w_i$$

אלגוריתם חמדני

ממיינים את המשקולות w בסדר יורד. ב.ה.כ. $w_1 \geq w_2 \geq \dots \geq w_n$. בוחרים להוסיף לקבוצת הוקטורים שבחרנו עד כה את הוקטור בעל משקל גדול ביותר שאיננו תלוי לינארית בוקטורים שכבר בחרנו.

נתבונן באוסף $F = \{A \subseteq \{1, \dots, n\} \mid A \text{ contains Independent vectors}\}$. מתקיים:

1. אם $A \in F$, אז $B \subseteq A$ - $B \in F$ (Hereditary).

2. אם $A, B \in F$, וגם $|A| < |B|$, אזי קיים $u \in B$ כך ש $A \cup \{u\} \in F$ (תכונת ההחלפה).

הגדרה: אוסף הקבוצות F המקיימות את התכונות 1 ו-2 לעיל נקרא מטרואיד.

הוכחת נכונות של האלגוריתם החמדני:

תהי $T \subseteq \{1, \dots, n\}$ קבוצה בעלת משקל אופטימלי (= מקסימום אפשרי), ותהי S הקבוצה שבחר האלגוריתם החמדני.

$$|T| = |S| \quad \text{טענת עזר:}$$

הוכחה: כי אם $|T| < |S|$ אזי קיים $u \in S$ כך ש $T \cup \{u\}$ בלתי תלויה. ואז,

$$w(T \cup \{u\}) > w(T) \quad \text{בסתירה לאופטימליות של } T.$$

אם $|S| < |T|$, אזי קיים $u \in T$ כך ש $S \cup \{u\}$ בלתי תלויים, בסתירה לכך שהאלגוריתם שלנו עבר 'עד הסוף', ולא פספס וקטורים.

נסמן את האיברים ב $S = \{s_1, \dots, s_k\}$, ואת $T = \{t_1, \dots, t_k\}$, כאשר האיברים מסודרים לפי משקל יורד. כלומר

$$w_{t_1} \geq w_{t_2} \geq \dots \geq w_{t_k}$$

$$w_{s_1} \geq w_{s_2} \geq \dots \geq w_{s_k}$$

נראה $w(T) \leq w(S)$. נניח בשלילה כי $w(T) > w(S)$. לפי ההנחה קיים אינקדס ראשון l כך ש $w_{s_l} < w_{t_l}$. נתבונן ב $A = \{s_1, \dots, s_{l-1}\}$, $B = \{t_1, \dots, t_l\}$. כמובן ש $|A| < |B|$. לפי למת ההחלפה, קיים $u \in B - A$ כך ש $A \cup \{u\} \in F$ (כלומר – בלתי תלויה). בכל אופן, $w_u \geq w_{t_l} > w_{s_l}$, וזה סותר את העובדה שהאלגוריתם החמדני בחר ב s_l ולא ב u .

תהי I קבוצה סופית, F קבוצה של תת קבוצות של I , המהווה מטרואיד, ונתונה פונקציית משקל $w: I \rightarrow R_{>0}$, המשקל של $i \in I$, ומחפשים קבוצה ב F בעלת משקל מקסימלי, אזי הראנו שהאלגוריתם החמדני בוחר קבוצה אופטימלית.

הערה: אם F תורשתית אבל תנאי II איננו מתקיים אזי קיימת פונקציית משקל שעבורה האלגוריתם החמדני איננו אופטימלי. בואו נראה אותה:

נניח $A, B \in F$, $|B| > |A|$ ולא קיים $u \in B - A$ כך ש $A \cup \{u\} \in F$.

$$\text{נגדיר } w(x) = \begin{cases} 1, & x \in A \\ 1 - \varepsilon, & x \in B - A \\ \delta, & \text{otherwise} \end{cases} \quad \varepsilon > 0, \delta > 0 \quad \text{נבחר} \quad \text{מספיק קטנים כך}$$

$$\text{ש } w(B) > w(A) + |I|\delta, \text{ כלומר } |B|(1 - \varepsilon) > |A| + |I|\delta$$

אלגוריתם למציאת עץ פורש מקסימלי בגרף

יהי $G = (V, E)$ גרף קשיר (לא מכוון). פונקציית משקל על הצלעות $w: E \rightarrow R_{>0}$. אם

$w(e) < w_0$ לכל e , אזי נגדיר $\tilde{w}(e) = w_0 - w(e)$, אזי מקסימום $\tilde{w}$ יתן מינימום עבור w . האלגוריתם החמדני בוחר אוסף צלעות באופן הבא: עוברים על הצלעות לפי משקל יורד ומוסיפים כל צלע שאיננה סוגרת מעגל בגרף.

$$\text{ב.ה.כ. } V = \{1, \dots, n\} \text{ לצלע } e = (i, j) \text{ נתאים וקטור } \begin{pmatrix} 0 \\ \vdots \\ 1 \\ 0 \\ 1 \\ \vdots \end{pmatrix} \begin{matrix} i \\ \\ j \end{matrix} \in F_2, \text{ כלומר, } V_e =$$

$$v_e(k) = \begin{cases} 1, & k = i, j \\ 0, & \text{otherwise} \end{cases} \text{ נחבר ונכפיל לפי } Z_2.$$

למה: קבוצה חלקית A לקבוצת הוקטורים $v_{e_1}, \dots, v_{e_m}$ $|E| = m$ היא קבוצה בת"ל $\Leftrightarrow A$ מכילה מעגל.

הוכחה: $\Rightarrow$ די להראות שקבוצת וקטורים המתאימה למעגל תלויה לינארית. נניח כי $e_1 = (i_1, i_2), e_2 = (i_2, i_3), \dots, e_k = (i_k, i_1)$ מעגל בגרף. כל קוארדינטה מופיעה בדיוק פעמיים עם ערך 1, ולכן הסכום הוא 0, ע"פ מודולו 2.

$\Leftarrow$ בקבוצה בת"ל יש מעגל. בהנתן תלות לינארית $v_{e_1} + v_{e_2} + \dots + v_{e_k} = 0$ יהי $e_1 = (i_1, i_2)$, קיים וקטור אחר כך שבקוארדינטה i_2 יש 1. בה"כ $e_2 = (i_2, i_3)$. כך נמשיך עד שנגיע ל i_1 , ונסגור מעגל.

מסקנה: קבוצת וקטורים בת"ל מעל F_2 מתאימה לאוסף צלעות שאין בהם מעגל (= איחוד של עצים או יער), ולכן האלגוריתם החמדני מוצא פתרון אופטימלי.

¹ למרות שבהגדרת התלות הלינארית מותר לשים מקדמי 0 בחלק מאיברי הסכימה, ניתן להתעלם מכך, ולהניח שהמעגל פשוט לא כולל אותם ולהמשיך את ההוכחה באופן זהה.

אלגוריתמים דינמיים

לדוגמא, כפל מטריצות – נניח שמכפילים מטריצות של $(n \times m)(m \times k)$. נקבל מטריצה בגודל $n \times k$. חישוב כל איבר במכפלה עולה $O(m)$ ובסה"כ $n \cdot m \cdot k$ פעולות.

2.11.2006

תכנון דינמי (Dynamic Programming)

דוגמא: נתונות n מטריצות, $A_1, \dots, A_n$, כאשר A_i מסדר $P_{i-1} \times P_i$. ברור כי $A_i \times A_{i+1}$ יתן לנו מטריצה M מסדר $P_{i-1} \times P_{i+1}$.

נניח שהכפלת מטריצות A, B מסדרים $p \times q, q \times r$ דורשת $p \cdot q \cdot r$ פעולות $f(p \cdot q \cdot r)$.

רוצים לחשב את המכפלה

$$M_{1,n} = A_1 \times A_2 \times \dots \times A_n =$$

$$(A_1 \times \dots \times A_k) \times (A_{k+1} \times \dots \times A_n)$$

אם נסמן ב- $T[i, j]$ את מספר הפעולות הקטן ביותר הדרוש לחישוב $M_{i,j}$, אזי

$$T[1, n] = T[1, k] + T[k+1, n] + P_0 P_k P_n$$

(כאשר P_0 זה מספר השורות של המטריצה ה-0, P_k מספר השורות של המטריצה ה- k , P_n של ה- n).
למה? כי צריך לחשב את המטריצה $M_{1,k}$, את $M_{k+1,n}$, ואז להכפיל ביניהם!

$$T[1, n] = \min [T[1, k] + T[k+1, n] + P_0 P_k P_n], 1 \leq k < n$$

ועל כן, כמובן שנאלץ להגדיר $T[i, i] = 0$.

בסה"כ, אוסף תת הבעיות שנראה הוא חישוב מחיר מכפלות מהצורה $M_{i,j}$ (בסה"כ

$O(n^2)$ בעיות, ולכל אחת מהן נדרש למצוא פתרון אופטימלי. ניתן לחשב את $T[i, j]$ עבור $j = i+1$, ואחר $j = i+2, \dots$

$$T[i, j] = \min [T[i, k] + T[k+1, j] + P_i P_k P_j] | 1 \leq k < j$$

כל חישוב כזה (כאשר $T[i, k]$, $T[k+1, j]$ כבר חושבו לכל k דורש סה"כ $O(k)$ פעולות = $O(n)$).

לכל הפרש $d = j - i$ צריך לחזור על חישוב כזה $O(n)$ פעמים, ולכן נקבל בסה"כ $O(n^3)$ פעולות.

עבורנו DNA הוא סדרה המורכבת מאותיות $\{A, C, G, T\}$. בהנתן **שתי** סדרות, אנו רוצים למצוא את התת סדרה (AKA 'סדרית') משותפת הארוכה ביותר.

בהנתן מילה $X = x_1, \dots, x_n$ תת סדרה של X היא מילה מהצורה $x_{i_1}, \dots, x_{i_k}$ כאשר $i_1 < i_2 < \dots < i_k$.

הסבר מובן, ופחות פורמלי: לא מחפשים תת סדרה רצופה משותפת! מחפשים סדרה משותפת, אפשר עם 'חורים' לא משותפים.

כלומר, בהנתן

$$X = x_1, \dots, x_n$$

$$Y = y_1, \dots, y_m$$

אנו רוצים לחשב את אורך תת הסדרה הארוכה ביותר המשותפת ל X ול Y (גם רוצים למצוא אחת, לא רק לחשב).

למה: 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 $x_n = y_m$ אזי $z_k = x_n$ 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 של $x_1, \dots, x_{n-1}$ 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 $y_1, \dots, y_{m-1}$.

הוכחה:

1. אם $z_1, \dots, z_{k-1}$ איננו האופטימום עבור $x_1, \dots, x_{n-1}$ ו $y_1, \dots, y_{m-1}$ אזי נבחר W יותר ארוך מ $k-1$, ואז Wz_k יתן תת מילה משותפת באורך $k+1 \leq$ בסתירה לאופטימליות של Z .
2. אחר הורדת x_n עדיין תת סדרה משותפת ולכן זה הטוב ביותר כי אחרת היה משהו ארוך יותר.
3. אנלוגי ל2.

$$1 \leq k \leq n \quad X_k = x_1 \dots x_k$$

$$1 \leq l \leq m, Y_l = y_1, \dots, y_l$$

ונסמן ב $C[k, l]$ את אורך תת הסדרית המשותפת הארוכה ביותר ל X_k, Y_l .

נגדיר:

$$C[k, l] = \begin{cases} C[k-1, l-1] + 1, & x_k = y_l \\ \max \{c[k-1, l], c[k, l-1]\}, & otherwise \end{cases}$$

$$C[0, l] = C[k, 0] = 0$$

6.11.2006

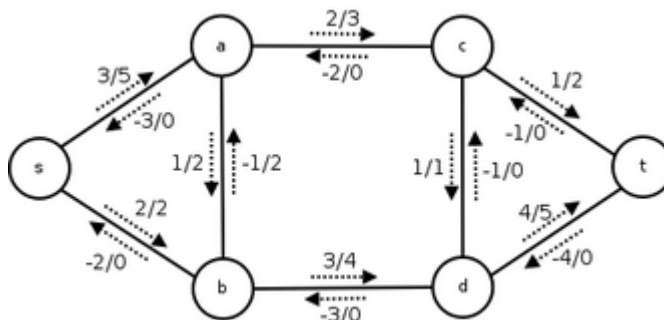
ציטוט השיעור: 'אני משלמת שכר לימוד עבור הפסקות. אני דורשת את ההפסקות שלי' - ד.ד.
 ציטוט השיעור 2: 'כלכם למדתם מספרים שליליים בכיתה ג' או ד', נכון? אני לא רוצה להעליב אותכם, זה באמת חומר מבלבל...' פרופ' אלמוני מסגל הקורס

נתון: n משוואות מהסוג $ax + by \geq c$.

צריך למצוא: $\max_{(x,y)} (ex + fy)$ המקיימים את כל הדרישות.

זה המקרה הכללי. אנחנו נלמד מקרה פרטי.

זרימה ברשתות



נתון גרף מכוון $G = (V, E)$.

קשת (u, v) בגרף מסמנת

חיבור בין u ל- v . פונקציה

$C : E \rightarrow \mathbb{R}_{\geq 0}$ מייצגת את

הקיבולת של החיבור.

קיימים שני קודקודים מיוחדים $s, t \in V$ ($Start, Target$).

אנו דורשים את חוק שימור החומר בגרף – מה שנכנס, גם יוצא. אנו רוצים למצוא את קצב ההזרמה המקסימלי מ- s ל- t . בשביל זה נצטרך להגדיר זרימה.

אין טעם להזרים זרימה בשני הכיוונים בין שני קודקודים כי הזרימות מבטלות אחת את השניה וניתן להסתפק בזרימה קטנה יותר בכיוון אחד ו-0 לכיוון השני. ולכן נחפש רק זרימות כאלה.

לכל צלע $(u, v) \notin E$ נגדיר $C(u, v) = 0$, ונגדיר זרימה בגרף G להיות פונקציה

$f : V \times V \rightarrow \mathbb{R}$ המקיימת:

I. תנאי הקיבולת - $\forall u, v \in V, f(u, v) \leq C(u, v)$ (לא מזרימים מעל הקיבולת)

II. אנטי סימטריות - $\forall u, v \in V, f(u, v) = (-1) \cdot f(v, u)$.

III. חוק שימור החומר – לכל קודקוד $v \in V \setminus \{s, t\}$ מתקיים $\sum_{u \in V} f(v, u) = 0$ (מה שנכנס

הוא מה שיוצא). זה מסביר את הזרימה שיוצאת מ- v (המספרים חיוביים), ואת הזרימה הנכנסת

ל- v (מספרים שליליים). בסוף הכל מתאזן.

בהנתן זרימה f על G מתקיים כמובן $|f| = \sum_{v \in V} f(s, v)$ (מן הסתם זה יהיה שווה ל $\sum_{v \in V} f(v, t)$).

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

בהנתן זרימה f על G נרחיב את הפונקציה f לתת קבוצות של V . אם $X, Y \subset V$ אזי

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

למה:

$$I. \quad \forall X \subset V, f(X, X) = 0.$$

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

$$II. \quad f(X \cup Y, Z) = f(X, Z) + f(Y, Z) \text{ , אזי } X \cap Y = \emptyset, X, Y, Z \subset V$$

$\circ$ (כמו II - עם איחוד זר בקוארדינטה השניה).

$\circ$

$$|f| = f(s, V) = f(V, V) - f(V - s, V) = -f(V - \{s\}, V) = f(V, V - \{s\}) = f(V, t) + \underbrace{f(V, V - t - s)}_{=0(*)}$$

(*) - שכן כל הקודקודים כאן הם פנימיים, וזה מתקיים מחוק שימור החומר).

יהי G גרף ו f זרימה ב G . נגדיר לכל $u, v \in V$ את קיבלות השארית לפי f כך :

$$C_f(u, v) = C(u, v) - f(u, v)$$

וכך נקבל את גרף השארית G_f . (מספר הצלעות בו יכול להיות עד פי 2 מהגרף המקורי).

נתבונן בצלעות עם צלעות שעבורן $C_f(u, v) > 0$.

[יש מספר דוגמאות, שלא נמצאות כאן. בטח כן כאן : [http://www.notes-](http://www.notes-heaven.intertent.net/SemesterA/Algo/index.html)

<http://www.notes-heaven.intertent.net/SemesterA/Algo/index.html>

למה: תהי f' זרימה בגרף השארית G_f . נגדיר זרימה חדשה $g = f + f'$ (כלומר,

$$g(u, v) = f(u, v) + f'(u, v), \text{ אזי } g \text{ זרימה בגרף הנתון } G, \text{ ו } |g| = |f| + |f'|.$$

הוכחה:

I. אנטי סימטריות – ברור.

II. $g(u, v) \leq c(u, v)$

$$g(u, v) = f(u, v) + f'(u, v) \leq f(u, v) + \underbrace{C(u, v) - f(u, v)}_{:=C_f(u, v)} = C(u, v)$$

III. חוק שימור החומר

האלגוריתם של Ford-Fulkerson

http://he.wikipedia.org/wiki/%D7%A8%D7%A9%D7%AA_%D7%96%D7%A8%D7%99%D7%9E%D7%94

נתבונן בגרף השארית עם הצלעות שעבורן $C_f(u, v) > 0$. אם יש מסלול פשוט (ללא מעגלים)

המחבר בין s ל t ב G_f . אם π הוא מסלול כזה, אז $C(\pi) = \min \{C(u, v) | (u, v) \in \pi\}$

נגדיר זרימה f' בגרף השארית באופן הבא:

$$f'(u, v) = \begin{cases} C(\pi), (u, v) \in \pi \\ -C(\pi), (v, u) \in \pi \\ 0, otherwise \end{cases}$$

מתקיים $|f'| = C(\pi)$

האלגוריתם:

נתחיל מהזרימה $f(u, v) = 0$ לכל (u, v) . בכל צעד נתבונן בגרף השארית G_f . נמצא מסלול

פשוט (ללא מעגלים) π ב G_f מ s ל t . נחשב את קיבולת המסלול ונגדיר זרימה f' לאורכו בגרף G_f ,

ואז נתקן את הזרימה f לזרימה $f + f'$.

הערה: מסלול כזה π נקרא מסלול מתקן לזרימה ב f .

9.11.2006

נתון גרף מכוון $G: (V, E)$, $C: E \rightarrow \mathbb{R}_{\geq 0}$ קיבולת על קשתות הגרף. 2 קודקודים מיוחדים $s, t \in V$. מתקיים $c(u, v) = 0$ לכל $(u, v) \notin E$.
הגדרנו פונקציית זרימה $f: V \times V \rightarrow \mathbb{R}$ המקיימת:

I. **תנאי הקיבולת** - $\forall u, v \in V, f(u, v) \leq C(u, v)$ (לא מזרימים מעל הקיבולת)

II. **אנטי סימטריות** - $\forall u, v \in V, f(u, v) = (-1) \cdot f(v, u)$.

III. **חוק שימור החומר** - לכל קודקוד $v \in V \setminus \{s, t\}$ מתקיים $\sum_{u \in V} f(v, u) = 0$ (מה שנכנס

הוא מה שיוצא). זה מסביר את הזרימה שיוצאת מ v (המספרים חיוביים), ואת הזרימה הנכנסת ל v (מספרים שליליים). בסוף הכל מתאזן.

הגדרנו גם שארית קיבולת (ראה שיעור קודם).

בהנתן זרימה f , הגדרנו את גרף השארית $G_f = (V, E_f)$, כאשר
 $E_f = \{(u, v) \mid u, v \in V, C_f(u, v) > 0\}$

Ford-Fulkerson

בהנתן G וזרימה f ניצור את גרף השארית. אם בגרף הארית יש מסלול מכוון פשוט ללא מעגלים π מ s ל t , אזי נגדיר את קיבולת המסלול π (בגרף השארית לפי f):

$$c_f(\pi) = \min \{c_f(x, y) \mid (x, y) \in \pi\}$$

נגדיר זרימה f' בגרף השארית G_f כך:

$$f(x, y) = \begin{cases} c_f(\pi), (x, y) \in \pi \\ -c_f(\pi), (y, x) \in \pi \\ 0, otherwise \end{cases}$$

f' זרימה חוקית. מתקיים $|f'| = c_f(\pi)$, ולכן $f + f'$ זרימה גדולה יותר מהזרימה f .

תאור האלגוריתם: מתחילים מזרימה $f \equiv 0$. בוחרים בכל צעד מסלול מתקן בגרף השארית לפי f , מתקנים את הזרימה, ומקבלים זרימה גדולה יותר. ממשיכים עד שנעצרים כאשר אין מסלול בין s ל t בגרף השארית.

הגדרה: יהי $G = (V, E)$ גרף מכוון עם פונקציית קיבולת c , וקודקודים מיוחדים $s, t \in V$, אזי חתך של קודקודי V לשתי קבוצות S, T ($s \in S, t \in T, V = S \cup T, S \cap T = \emptyset$) נקראת **חתך** של גרף G .
יהי S, T חתך בגרף G . $f(S, T)$ היא הזרימה נטו לרוחב החתך.

טענה: $|f| = f(S, T)$ לכל חתך S, T בגרף.

$$f(S, T) = f(S, V) - f(S, S) = f(S, V) =$$

$$f(s, V) + f(S - \{s\}, V) = |f| + f(S - \{s\}, V) =_* |f|$$

(*) מכיוון $S - \{s\}$ לא מכילה את s, t , כל קודקודים הם פנימיים, ולכן, מחוק שימור החומר,

$$f(S - \{s\}, V) = 0.$$

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

$$f(S, T) = \sum_{\substack{u \in S \\ v \in T}} f(u, v) \leq \sum_{\substack{u \in S \\ v \in T}} c(u, v) = C(S, T) \quad \text{הוכחה:}$$

ולכן, לכל זרימה f מתקיים $\max |f| \leq \min \{c(S, T) \mid (S, T) \text{ חתך בגרף } G\}$.

משפט: תהי f זרימה בגרף G . s, t הקודקודים המיוחדים, אזי התנאים הבאים השקולים:

I. f היא זרימה מקסימלית ב G .

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

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 ונגדיר $S = \{v \mid * \}$ ניתן להגיע מ s ל v בעזרת מסלול בגרף השארית (G_f) . $T = V - S$. נראה כי זה חתך - $s \in S$ (כי הוא שם). $t \notin S$ כי לפי ההנחה אין מסלול מ s ל t ב G_f . ולכן, (S, T) חתך בגרף.

טענה: לכל צלע (u, v) $u \in S, v \in T$ מתקיים $f(u, v) = c(u, v)$

הוכחת: כי אחרת $f(u, v) < c(u, v)$ נמצאת בגרף השארית, ולכן יש

מסלול מ s ל v , ולכן גם מסלול מ s ל u , בסתירה לכך ש $u \notin S$.

הגענו לכך ש $f(S, T) = C(S, T)$.

הערה: ההוכחה נותנת אלגוריתם למציאת חתך כזה בהנתן זרימה מקסימלית.

$III \Rightarrow I$ לכל חתך (S, T) $|f| \leq c(S, T)$ ולכן שיוויון $|f| = c(S, T)$ היא מקסימלית.

מסקנה: אם האלגוריתם של Ford Fulkerson עוצר, אזי הוא מוצא זרימה מקסימלית. אם כל הקיבולות הם מספרים שלמים, אזי בכל צעד האלגוריתם מגדיל את הזרימה לפחות ב1, ולכן לאחר מספר חזרות $|f| \geq$ מקסימלי אזי נסיים.

משפט: אם כל הקיבולות הן מספרים שלמים, אזי קיימת זרימה מקסימלית f כך שהזרימה על כל צלע גם היא מספר שלם.

בעיית הזיווג בגרף דו צדדי:

נתון גרף דו צדדי $G = (U \cup V, E)$, $|U| = |V| = n$, $E \subseteq U \times V$. זיווג שלם בגרף G הוא אוסף של n צלעות $M = \{e_1, \dots, e_n\} \subseteq E$ כך שלכל u_i קיים v_i יחיד כך ש $(u_i, v_i) \in M$. באופן כללי, אנו מחפשים קבוצה מקסימלית $M \subseteq E$ כך שלכל $(u, v) \in M$ אין $v' \neq v$ כך שגם $(u, v') \in M$. נגדיר גרף G' מ G , ע"י הוספת s, t קודקודים מיוחדים, וצלעות (s, U) (קשת מ s לכל $u \in U$), וכן"ל (V, t) . נגדיר את כל הצלעות בגרף עם קיבולת 1.

נריץ על זה FF (Ford Fulkerson). כל צעד דורש $O(|E| + n)$ צעדים, יש לכל היותר n חזרות, ובסה"כ $O(n^3)$. FF מוצא זרימה שלמה מקסימלית. נראה כי $|f| = |m|$ - בהנתן זיווג M קיימת זרימה בגודל $|M|$. נזרים 1 לאורך הצלעות ב M . ולכן $|f| \geq |M|$. בהנתן זרימה שלמה (מספרים שלמים) נבחר צלעות בהן קיימת זרימה 1. זוהי M המתבקשת.

16.11.2006

זרימה ברשתות

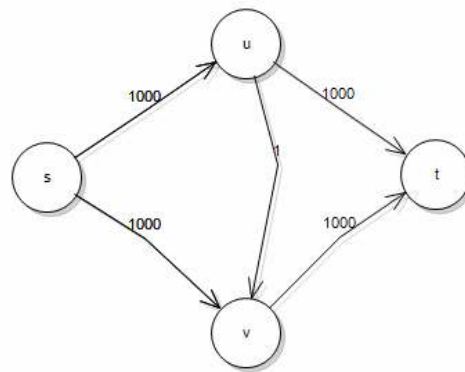
נתון גרף מכוון $G = (V, E)$, ותהי $c: E \rightarrow \mathbb{R}_+$ זרימה f מקיימת $c(u, v) \geq f(u, v)$ שימור חומר בקודקודים פנימיים $(V - \{s, t\})$, $s, t \in V$ קודקודים מיוחדים.

[Ford Fulkerson – תזכורת]

Edmonds Karp

נראה אלגוריתם לחישוב זרימה מקסימלית כאשר מספר האיטרציות של תיקון הזרימה בעזרת מסלול בגרף שארית חסום ע"י פונקציה פולינומיאלית של מספר הקודקודים והצלעות בגרף G .

כאשר בוחרים מסלול מתקן בגרף השארית, נבחר מסלול באורך קצר ביותר¹ מ s ל t .
ניקח דוגמא:

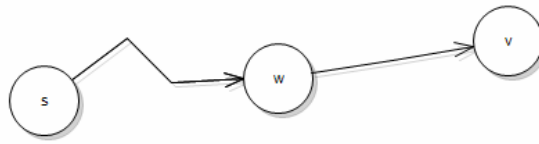


ברור כי אם נבחר את המסלול הקצר יותר קודם, אז נסיים קודם. ברור כי אם FF עובד, גם EK עובד. לצורך הוכחת חסם על מספר האיטרציות ב-EK, בהנתן זרימה f בגרף G נתבונן בגרף השאריות ונגדיר מרחק $\delta_f(v) = [\text{אורך מסלול קצר ביותר מ } s \text{ ל } v \text{ ב } G_f]$ אם יש, אחרת ∞ .

למה: אם f' זרימה המתקבלת בתהליך לאחר זרימה נתונה f , אזי $\delta_f(v) \leq \delta_{f'}(v)$.

הוכחה: די לטפל במקרה ש f' מתקבלת מ f בעזרת מסלול מתקן יחיד (אחרת זה נובע מאינדוקציה).
תהי f' זרימה המתקבלת ב-EK שעבורה קיים קודקוד v שעבורו הטענה איננה נכונה. נבחר זרימה f' **ראשונה** שבה יש דוגמא נגדית, וקודקוד v שעבורו $\delta_{f'}(v)$ מינימלית כך ש v דוגמא נגדית $(\delta_f(v) > \delta_{f'}(v))$.

¹ מבחינת מספר הצלעות



נתבונן ב- G_f - 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 $\delta_f(w) = \delta_f(v) - 1$. מהמינימליות של $\delta_f(v)$ נקבל $\delta_f(w) \leq \delta_f(v) - 1$.

טענה: $(w, v) \notin E_f$

הוכחה: אחרת, $\delta_f(v) \leq \delta_f(w) + 1 \leq \delta_f(w) + 1 = \delta_f(v)$, שיוון חזק הפוך $(\delta_f(v) > \delta_f(v))$. בסתירה להנחה שמתקיים אי

אם $(w, v) \notin E_f$ אבל $(w, v) \in E_f$ ולכן חייב להיות שהמסלול המתקן π ב- E_f בחר את הצלע (v, w) (הפוך ממה שחשבנו, (w, v)) - כלומר, המסלול התחיל ב- s , עבר ל- w , ומשם איכשהו ל- v . כלומר:

$$\delta_f(v) = \delta_f(w) - 1 \leq \delta_f(w) - 1 = \delta_f(v) - 2$$

כלומר, הנחנו (בשלילה) כי $\delta_f(v) < \delta_f(v)$ וקיבלנו כי $\delta_f(v) \geq \delta_f(v) + 2$. לכן הגענו לסתירה.

למה: מספר האיטרציות ב- EK הוא $O(|E||V|)$.

מסקנה: כל צעד באלגוריתם EK דורש $O(E)$ פעולות ולכן בסה"כ האלגוריתם דורש $O(E^2V)$ פעולות. מאחר ו- $E \leq V^2 = n^2$ אזי נקבל בסה"כ $O(n^5)$ עבור גרפים עם n קודקודים.

הוכחת הלמה: בכל מסלול מתקן π יש צלע (u, v) שעבורה קיבולת השארית $c_f(\pi) = c_f(u, v)$. כלומר, קיבולת השארית על הצלע היא המינימלית ב- π . אנו מזרימים לאורך π זרימה נוספת בגודל $c_f(\pi)$ ולכן בזרימה f' המתקבלת לאחר התיקון $(u, v) \notin E_f$. צלע (u, v) כזו תקרא **צלע קריטית** ב- π . מאחר ו- π מסלול קצר ביותר מ- s ל- v אזי $\delta_f(v) = \delta_f(u) + 1$.

שאלה: במהלך EK כמה פעמים צלע מכוונת (u, v) יכולה להיות צלע קריטית?

תשובה: לכל היותר $\frac{|V|}{2}$.

הוכחה: אם (u, v) קריטית לפי זרימה f , אזי לאחר תיקון הזרימה הצלע (u, v) נעלמה וכדי להיות קריטית שוב, היא חייבת להופיע, וזה יכול להיות רק אם זרימה מאוחרת יותר ומסלול מתקן בחרו בכיוון ההפוך (v, u) (כלומר, היא יכולה לבוא ולהופיע כל פעם בכיוון השני).

$$\delta_f(u) = \delta_f(v) + 1 \geq \delta_f(v) + 1 = \delta_f(u) + 2$$

כיוון ש $\delta_f(v)$ איננו אינסופי, אזי $|\nu| - 1 \geq \delta_f(v)$. בכל איטרציה יש לפחות צלע קריטית אחת ולכן מספר האיטרציות $\geq \frac{|V|}{2} \cdot |E|$.

הקדמה לטרנספורמציות פורייה

תהי מטריצה, עם אוסף מספרים, ואנו מחפשים מטריצה קטנה יותר, 'דומה' למקורית.

20.11.2006

טרנספורמציות פורייה –



http://he.wikipedia.org/w/index.php?title=%D7%96%27%D7%90%D7%9F_%D7%91%D7%98%D7%99%D7%A1%D7%98_%D7%96%27%D7%95%D7%96%D7%A3%D7%A4%D7%95%D7%A8%D7%99%D7%99%D7%94&oldid=2185956

FFT – אתר שמאפשר ל'שמוע' סדרות פורייה

<http://www.jhu.edu/~signals/listen-new/listen-newindex.htm>

נניח ויש לנו סדרה אינסופית $\{b_i\}_{i=0}^{\infty}$ ו $\{a_i\}_{i=0}^n$ אין סופית, a_n סופית כאמור). אנו נרצה לחשב את

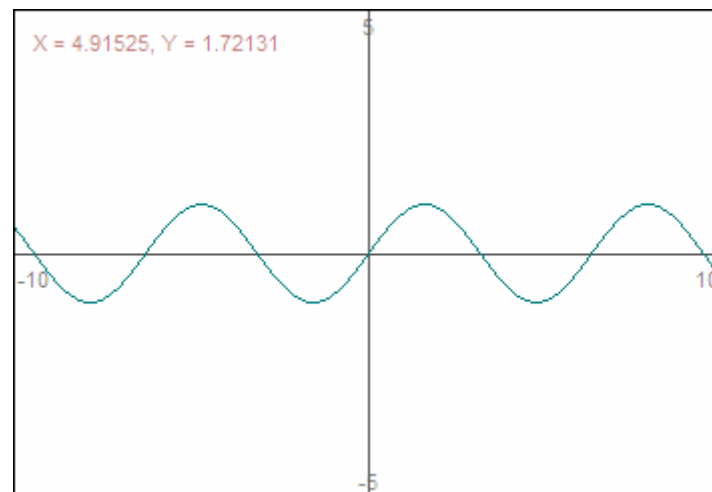
$$C_0 = a_0 b_0 + a_1 b_1 + \dots + a_n b_n$$

$$C_1 = a_0 b_1 + a_1 b_2 + \dots + a_n b_{n+1}$$

...

$$C_k = a_0 b_k + a_1 b_{k+1} + \dots + a_n b_{n+k}$$

יתכן ונרצה לתת משקל שונה לכל a_i (ממוצע משוקלל).

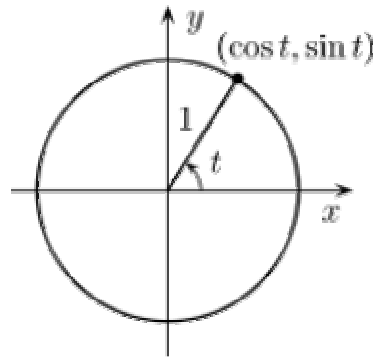


נניח ויש לנו גל :

קלט: וקטור של מספרים ממשיים באורך $N = 1024$. נתבונן בו כוקטור N מימדי מעל המרוכבים $\mathbb{C}$. אפשר לבחור בסיס כרצוננו – לדוגמא, את הבסיס הסטנדרטי. מן הסתם, אם קיבלנו וקטור, אנו צריכים לקבל איתו בסיס, אחרת סתם יש לנו כמה מספרים. אנו ננסה לבחור בסיס טוב יותר מהבסיס הסטנדרטי. אנו נבחר בסיס שמופיעים בו 'איברים מחזוריים'.
לדוגמא:

$$(1,1,1,...,1)$$

$$(1,-1,1,-1,...,1,-1)$$



נתבונן במעגל היחידה:

נגדיר: $w = \cos t + i \sin t$.

(יש כאן עוד שרטוטים – אצל דינה – איך נראים חזקות של w על מעגל היחידה.)

אם ω הוא שורש המתאים ל $t = \frac{\pi}{8}$ במעגל היחידה, אזי אפשר וקטור בסיס של:

$$(\omega^0 = 1, \omega, \omega^2, \dots, \omega^{15}, 1, \omega, \dots)$$

(באופן מחזורי).

נבחר בסיס למרחב: $\mathbb{C}^N$ באופן הבא - $\omega = e^{\frac{2\pi i}{N}}$. כלומר - אם $N = 1024$, אזי

$\omega = \cos \frac{2\pi}{1024} + i \sin \frac{2\pi}{1024}$. (נניח כי $e^{i\alpha} = \cos \alpha + i \sin \alpha$ - אם למדת פונקציות מרוכבות, אתה אמור להבין. אחרת, לא.)
נגדיר וקטורי בסיס:

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

נניח עבור $N = 1024, k = 64$, נשים לב כי $64 \times 16 = 1024$, ועל כן עבור $k = 0$ הוקטור יראה כך:

$$(1, 1, \dots, 1)$$

עבור $k = 1$ זה יראה:

$$\left(1, \omega^{64}, \omega^{128}, \dots, \underset{16\text{'th position}}{1}, \dots \right)$$

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

הוכחה: יהו $\alpha, \beta \in \mathbb{C}^N$. אזי $\langle \alpha, \beta \rangle = \sum_{l=0}^{N-1} \alpha_l \beta_l$ (כמובן בהנחה ש $\alpha = (\alpha_0, \dots, \alpha_{N-1})$,

$$\beta = (b_0, \dots, b_{N-1})$$

למה: יהי F שדה, $\omega \in F$ מקיים $\omega \neq 1, \omega^N = 1$ אזי $X := \sum_{l=0}^{N-1} \omega^l = 0$

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

$$(\omega - 1)X = 0 \Rightarrow X = 0 \text{ ש } \omega X = X$$

הגדרה: $\omega \in F$ נקרא שורש יחידה פרמיטיבי מסדר N אם $\omega^N = 1$, אבל לכל $0 < j < N$, $\omega^j \neq 1$.

למה: יהי F שדה, $\omega \in F$ שורש יחידה פרמיטיבי מסדר N , אזי:

$$\sum_{l=0}^{N-1} \omega^{kl} = \begin{cases} N, k \mid N \\ 0, \text{otherwise} \end{cases}$$

נרשום - $0 \leq r < N, k = qN + r$

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

ועל כן, מתקיים $\sum_{l=0}^{N-1} \omega^{kl} = \sum_{l=0}^{N-1} (\omega^r)^l$. אבל - $(\omega^r)^N = 1$. אם $r \neq 0$ אזי $\omega^r \neq 1$, וסיימנו.

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

$$\langle \alpha_k, \alpha_r \rangle = \sum_{l=0}^{N-1} \omega^{kl} \overline{(\omega^{lr})} = \sum_{l=0}^{N-1} \omega^{kl} \omega^{-rl} = \sum_{l=0}^{N-1} \omega^{(k-r)l} = \begin{cases} 0, k \neq r \\ N, k = r \end{cases}$$

הוכחה: כנדרש.

עוד דרך להסתכל על זה:

$$\begin{pmatrix} 1 & 1 & 1 & & 1 \\ 1 & \omega & \omega^2 & & \omega^{N-1} \\ 1 & \omega^2 & (\omega^2)^2 & \dots & \\ 1 & & & & \\ \dots & & & & \\ 1 & \omega^{N-1} & (\omega^2)^{N-1} & & (\omega^{N-1})^{N-1} \end{pmatrix}$$

וזה מזכיר לנו את... **ונדרמונדה!**

נשים לב כי בחרנו וקטורים אורתוגונליים, לא אורתונורמליים, לכן נגדיר $\alpha'_i = \frac{1}{\sqrt{n}} \alpha_i$, ונעבוד איתו

— זה יהיה אורתונורמלי כמובן.

כעת, אם $\alpha'_0, \dots, \alpha'_{N-1}$ בסיס אורתונורמלי, ו β וקטור כלשהו $\beta = \sum_{i=0}^{N-1} b_i \alpha'_i$, אזי עבור k כלשהו

$$\langle \beta, \alpha'_k \rangle = \sum_{l=0}^{N-1} b_l \langle \alpha'_l, \alpha'_k \rangle = b_k \langle \alpha'_k, \alpha'_k \rangle = b_k$$

חשוב כל מקדם דורש $O(n)$ בסה"כ, $O(n^2)$, ורוצים לעשות זאת מהר יותר.

בקצרה: נניח ש N הוא חזקה של 2, במקרה שלנו $N = 1024 = 2^9$. בהנתן סדרה $a_0, \dots, a_{N-1}$, נגדיר

$$f(x) = a_0 + a_1x + \dots + a_{N-1}x^{N-1} \quad \text{ניקח} \quad \text{את} \quad \text{המטריצה}$$

$$V = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega & \omega^2 & \dots & \omega^{N-1} \\ 1 & \omega^2 & (\omega^2)^2 & \dots & \dots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{N-1} & (\omega^2)^{N-1} & \dots & (\omega^{N-1})^{N-1} \end{pmatrix} \quad \text{נתבונן ב } AV. \text{ למה זה שווה?}$$

$$AV = (f(1), f(\omega), f(\omega^2), \dots, f(\omega^{N-1}))$$

נזכור כי $\omega^N = 1$, שכן ω שורש יחידה פרמיטיבי מסדר N .

$$\frac{N}{2}, (\omega^2)^{N/2} = 1 \text{ שורש יחידה פרמיטיבי מסדר } \frac{N}{2}.$$

נרשום $f(x) = a_0 + a_2x^2 + a_4x^4 + \dots + x(a_1 + a_3x^2 + a_5x^4 + \dots)$ (נפצל ל x ים זוגיים ואי זוגיים, ונוציא גורם משותף x לאי זוגיים).

$$f_{E_{\text{ven}}} = a_0 + a_2x + a_4x^2 + \dots + a_{n-2}x^{\frac{n}{2}-1}$$

$$f_{O_{\text{dd}}} = a_1 + a_3x + a_5x^2 + \dots + a_{n-1}x^{\frac{n}{2}-1}$$

$$f(x) = f_E(x^2) + xf_O(x^2)$$

$$f(\omega^k) = f_E((\omega^2)^k) + \omega^k f_O((\omega^2)^k), \text{ כלומר,}$$

$$T(n) = 2T\left(\frac{n}{2}\right) + O(n) = O(n \log n), \text{ כלומר,}$$

מטרה: רוצים לחשב קונולוציה (Convolution) של סדרת קלט $b_0, \dots, b_n, b_{n+1}, \dots$ בסדרה קבועה $a_0, \dots, a_{n-1}$.

מהי הקונולוציה? הסדרה C_i , המוגדרת:

$$C_0 = a_0 b_0 + \dots + a_{n-1} b_{n-1}$$

...

$$C_k = a_0 b_k + a_1 b_{k+1} + \dots + a_{n-1} b_{n+k-1}$$

הסדרה C_k היא סדרת הקונולוציה של הסדרות a_n ו b_n .

יהי F שדה - ω - שורש יחידה פרימיטיבי מסדר n . $\omega^n = 1, \omega^k \neq 1, 1 \leq k < n$. בהנתן סדרה באורך n , $a = a_0, \dots, a_{n-1}$ אזי טרנספורם פורייה (דיסקרטי) של הסדרה a מוגדר כך:

$$\vec{\alpha} = \sum_{l=0}^{n-1} a_l \omega^{kl}, k = 0, \dots, n-1$$

אפשר לחשב זאת כך:

$$\vec{\alpha} = (a_0, \dots, a_{n-1}) \begin{bmatrix} 1 & 1 & \dots & 1 \\ 1 & \omega & \omega^2 & \omega^{n-1} \\ & \omega^2 & & \\ & & & \\ 1 & (\omega^2)^{n-1} & & (\omega^{n-1})^2 \end{bmatrix} = (\alpha_0, \dots, \alpha_{n-1})$$

(נשים לב – באופן כללי, במטריצה הימנית, האיבר בשורה l בעמודה k הוא ω^{kl}).

אם נסמן את העמודה k ב V_k אזי סדרת הוקטורים $\frac{1}{\sqrt{n}} v_k$ בסיס אורתונורמלי למרחב $\mathbb{C}^n$:

$$\vec{a} = \frac{1}{\sqrt{n}} \sum \alpha_k v_k'$$

חישוב מהיר של טרנספורם פורייה דיסקרטי עבור $n = 2^9$
נשים לב:

1. אם ω שורש יחידה פרימיטיבי מסדר n , אזי ω^2 שורש יחידה פרימיטיבי מסדר $\frac{n}{2}$.

2. $\omega^{n/2} = -1$ - פתרון שונה מ-1 למשוואה $x^2 = 1 \Leftrightarrow$ יכול להיות רק -1 .

נסמן $\omega_1 = \omega^2$, אזי ω^k , עבור $k = 0, \dots, n-1$, נותן פעמיים את כל החזקות של ω_1 -

$$1, \omega_1, \omega_1^2, \dots, \omega_1^{n/2-1}, 1, \omega_1, \dots, \omega_1^{n/2-1}$$

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

שכן $\omega_1^{n/2+k} = (\omega^2)^{n/2+k} = \omega^n \cdot \omega^{2k} = \omega_1^k$

אם נסמן: $A(x) = a_0 + a_1x + \dots + a_{n-1}x^{n-1}$ אזי לכל k $\alpha_k = \sum_{l=0}^n \alpha_l (\omega^k)^l = A(\omega^k)$

נפריד את הפולינום $A(x)$ לחלק זוגי ולחלק אי זוגי.

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

$$A_O(x) = a_1x + a_3x^3 + \dots + a_{n-1}x^{n-1} = x(a_1 + a_3x^2 + \dots + a_{n-1}x^{n-2}) = x\bar{A}_O(x^2)$$

$$\bar{A}_0(t) = a_1 + a_3t + \dots + a_{n-1}t^{\frac{n}{2}-1}$$

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

בסופו של דבר, המסקנה היא $A(x) = \bar{A}_E(x^2) + x\bar{A}_O(x^2)$

$$\omega_1 = \omega^2$$

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

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

מסדר $\frac{n}{2}$.

עבור $k = 0, \dots, \frac{n}{2} - 1, \frac{n}{2}, \frac{n}{2} + 1, \dots, \frac{n}{2} + k$ אנחנו מקבלים

$$A\left(\omega^{\frac{n}{2}+k}\right) = \bar{A}_E(\omega_1^k) - \omega^k \bar{A}_O(\omega_1^k)$$

$$\omega_1^{\frac{n}{2}+k} = \omega_1^k$$

$$\omega^{\frac{n}{2}+k} = -\omega^k$$

$T(n)$ - סה"כ מספר פעולות אריתמטיות במספרים **ממשיים** :

• חיבור מרוכב - 2 פעולות חיבור ממשיות

• כפל מרוכב - 6 פעולות כפל ממשיות

נסמן ב $S(n)$ את מספר החיבורים, $M(n)$ כפלים.

$$S(n) = 2S\left(\frac{n}{2}\right) + n = n \log_2 n$$

$$M(n) = 2M\left(\frac{n}{2}\right) + \frac{n}{2} \log_2 n$$

למה ב $M(n)$ יש רק $\frac{n}{2}$ פעולות? ראינו שעבור $k = \frac{n}{2}, \dots, n$ מדובר באותו ערך רק עם סימן הפוך

, ולכן אין כאן באמת עבודה.

בסה"כ - $5n \log_2 n$ פעולות ממשיות.

טרנספורם הפוך

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

$$\vec{\alpha} = (a_0, \dots, a_{n-1}) \begin{bmatrix} 1 & 1 & \dots & 1 \\ 1 & \omega & \omega^2 & \omega^{n-1} \\ & \omega^2 & & \\ 1 & (\omega^2)^{n-1} & & (\omega^{n-1})^2 \end{bmatrix} = (\alpha_0, \dots, \alpha_{n-1})$$

אזי אם V היא המטריצה האמצעית, אזי $(\alpha_0, \dots, \alpha_{n-1}) V^{-1}$ יתן לנו את $(a_0, \dots, a_{n-1})$. (קיימת הפכית ל- V)

טענה:

$$(T)_{st} = \omega^{-st}, \quad V^{-1} = \frac{1}{n} T$$

בהנתן הטענה, חישוב הכפלת וקטור ב- T הינו חישוב טרנספורם פורייה כאשר בוחרים את ω^{-1} כשורש יחידה פרימיטיבי.

אז, כדי לחשב, נחשב טרנספורם פורייה עם ω^{-1} שורש יחידה פרימיטיבי ואת וקטור התוצאה מחלקים ב- n .

אבל צריך להוכיח את הטענה:

$$(W)_{lk} = \omega^{lk} \quad (T)_{st} = \omega^{-st} \quad \text{תהי } W, T \text{ המוגדרות} \quad \text{צ"ל } WT = nI$$

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

שימושים לטרנספורם פורייה

כפל פולינומים –
בהנתן פולינום

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

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

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

$$C_k = \sum_{l=0}^k a_l b_{k-l} \quad \text{לכל } k = 0, \dots, 2n-1 \text{ מתקיים}$$

הערה: פולינום $f(x)$ מעל שדה F ניתן להציג בעזרת המקדמים $f(x) = f_0 + \dots + f_{d-1}x^{d-1}$, ואז $\deg f = d-1$.

אבל את f הנ"ל ניתן לייצג גם באמצעות ערכיו ב d נקודות שונות ב F , $e_0, \dots, e_{d-1}$ נקודות שונות $f(e_0), \dots, f(e_{d-1})$.

טענה: מעל שדה F , בהנתן d נקודות שונות $e_0, \dots, e_{d-1}$ וערכים כלשהם $y_0, \dots, y_{d-1} \in F$ קיים פולינום יחיד $f(x)$ $\deg f \leq d-1$ כך ש $f(e_l) = y_l$ לכל $0 \leq l \leq d-1$.

הוכחה: יחידות: אם $g(x)$ פולינום כנ"ל, אזי לפולינום $h(x) = f(x) - g(x)$ יש d שורשים שונים, אבל דרגת $h(x)$ $\geq d-1 \Rightarrow h = 0$.

קיום: נגדיר $\prod_{\substack{k \neq l \\ 0 \leq k \leq d-1}} (x - e_k)$. הפולינום מתאפס בכל e_k , עבור $k \neq l$, ושונה מ-0 בנקודה e_l . נגדיר

$$f(x) = \sum_{l=0}^{d-1} y_l \cdot \frac{\prod_{\substack{k \neq l \\ 0 \leq k \leq d-1}} (x - e_k)}{\prod_{\substack{k \neq l \\ 0 \leq k \leq d-1}} (e_l - e_k)}.$$

נחזור להכפלת הפולינומים

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

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

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

די לנו לדעת את ערכים על $2n$ נקודות, כדי 'לקבל' את $c(x)$. אבל אנחנו יכולים לבחור את הנקודות כרצוננו...

אלגוריתם לחישוב $C(x)$

הנחה: n חזקה של 2.

ניקח את A , ונעבור באמצעות טרנספורם פורייה בגודל $2n$ ל $\bar{\alpha} = (\alpha_0, \dots, \alpha_{2n-1})$

ניקח את B , ונעבור באמצעות טרנספורם פורייה בגודל $2n$ ל $\vec{\beta} = (\beta_0, \dots, \beta_{n-1})$.
 אפשר פשוט להכפיל את הערכים בנקודות $(2n)$ פעולות כפל) $\vec{C} = (\alpha_0 \beta_0, \dots, \alpha_{2n-1} \beta_{2n-1})$
 ועכשיו נבצע פורייה הפוך מסדר $2n$ ונקבל את C . בסה"כ – זה עולה לנו $O(n \log n)$.

מעבר על סימולטור מהאתר: <http://www.jhu.edu/~signals/fourier2/index.html>

האזנה לטורי פורייה: <http://www.jhu.edu/~signals/listen-new/listen-newindex.htm>

סיכום עד כה :

חישוב טרנספורם פורייה (או טרנספורם הפוך) מסדר n דורש (לכל היותר) $5n \log n$ פעולות אריתמטיות במספרים ממשיים.

בהנתן פולינום $A(x) = a_0 + a_1x + \dots + a_{k-1}x^{k-1}$, אזי ניתן לחשב את מקדמי פולינום המכפלה $B(x) = b_0 + b_1x + \dots + b_{m-1}x^{m-1}$

באופן הבא -

בוחרים n שהוא חזקה של 2, $n \geq k + m$, נשלים את A, B לפולינום מסדר n (עם מקדמים של 0), נחשב $FFT_n(A), FFT_n(B)$

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

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

נחשב את $A \times B$ על ידי מכפלת וקטורים רגילה (קואורדינטה בקואורדינטה), נחשב על התוצאה FFT_n^{-1} , נקבל את הפולינום ההפוך.

מספר הפעולות יוצא $15n \log_2 n$ בממשיים.

קונוולוציה

ניזכר, כי כל הבלאגן של FFT היה בשביל לחשב קונוולוציה. מה זה קונוולוציה:

נניח נתונה $A = a_0, \dots, a_{n-1}$, כאשר n חזקה של 2.

נניח שיש לנו סדרה אינסופית $B = b_0, b_1, \dots, b_{n-1}, b_n, b_{n+1}, \dots, b_{2n-1}, b_{2n}, \dots$

אנו רוצים לחשב את $C_k = \sum_{l=0}^{n-1} a_l b_{k+l}$. חישוב כל C_k דורש $O(n)$.

נזכור כי B - סדרה של מספרים ממשיים, A כלשהו. סדרת C_k נקראת סדרת הקונוולוציה של A, B .

אם אנו רוצים לחשב את $C_0, \dots, C_{n-1}$. נתבונן

$$\bar{A}(x) = a_{n-1} + a_{n-2}x + \dots + a_1x^{n-2} + a_0x^{n-1}$$

$$B(x) = b_0 + b_1x + \dots + b_{n-1}x^{n-1} + b_nx^n + \dots + b_{2n-1}x^{2n-1}$$

$$D(x) = \bar{A}(x)B(x) =$$

נחשב את $0 \cdot x^{3n-1} + \dots + a_0 b_{2n-1} x^{3n-2} + \dots + C_{n-1} x^{2n-2} + \dots + C_1 x^n + \dots + C_0 x^{n-1} + \dots$

(כן, ה C_i כאן יהיה האחד שחיפשונו...) - קיבלנו n C_i ים שונים.

נשים לב אבל, ששילמנו יחסית הרבה – צריך לבחור טרנספורם מסדר $4n$, כלומר – המחיר לכל c_i

$$\frac{15 \times 4n \log(4n)}{n} \text{ כשמחשבים יחד } n \text{ הוא -}$$

שיפורים: אם בוחרים את $B(x)$ כפולינום ממעלה $15n-1$, אזי

$B(x) = b_0 + b_1x + \dots + b_{15n-1}x^{15n-1}$, ואנו בוחרים את $\overline{A}(x) = a_{n-1} + \dots + a_0x^{n-1}$, אזי סכום המעלות $\sim 16n$ - חזקה של 2. $n-1$ האיברים הראשונים אינם מעניינים, וכנ"ל n האיברים האחרונים, אבל באמצע – כן, ולכן קיבלנו בבת אחת $14n$ מקדמים מסדרת הקונוולוציה:

$$\frac{15 \cdot 16n \log(16n)}{14n} \text{ ועל כן, העלות פר מקדם היא}$$

נבדוק מה קורה כאשר בוחרים $B(x)$ מדרגה $16n-1$, ונעזרים בטרנספורם פורייה מסדר $16n$. נשים לב שאין סיבה שנקבל את המכפלה – כי אנו נקבל פולינום מדרגה $16n-1$, למרות שדרגת המכפלה גדולה יותר.

$$\underbrace{A(x)B(x)}_{\deg > 16} = q(x)(x^n - 1) + r(x).$$

$$\overline{A}(x) = a_{n-1} + a_{n-2}x + \dots + a_0x^{n-1} \text{ כלומר, אם}$$

$$B(x) = b_0 + b_1x + \dots + b_{n-1}x^{n-1} + \dots + b_{16n-1}x^{16n-1}$$

$$\overline{A}(x)B(x) =$$

$$d_0 + \dots + d_{n-2}x^{n-2} + \text{ אזי}$$

$$C_0x^{n-1} + C_1x^{n-2} + \dots + C_{15n-1}x^{16n-1} + d_{16n}x^{16n} + \dots + d_{17n-2}x^{17n-2} + 0x^{17n-1}$$

$$\text{אם נציב ב} \underbrace{A(x)B(x)}_{\deg > 16} = q(x)(x^n - 1) + r(x) \text{ את } \omega^k \text{ (} \omega^n = 1 \text{), אזי}$$

$$k = 0, \dots, n-1, \deg r \leq n-1, \overline{A}(\omega^k)B(\omega^k) = q(\omega^k) \cdot 0 + r(\omega^k)$$

אחרי שלב ההכפלה, הטרנספורם ההפוך יתן את מקדמי הפולינום $R(x)$ באשר $R(x)$ הוא השארית של חלוקת הפולינום $D(x)$ ב (x^{16n-1}) , $\deg R(x) \leq 16n$, ואז למעט $n-1$ המקדמים הראשונים, כל המקדמים האחרים הם איברים בסדרת הקונוולוציה, בסה"כ $15n$ איברים מטרנספורם

$$\frac{15 \times 16n \cdot \log(16n)}{15n} \sim 3 \times 5 \log n \text{ של } 16n \text{ קיבלנו מחיר ממוצע -}$$

הסדרה A היא סדרה קבועה, ולכן ניתן לחשב את טרנספורם פורייה שלה פעם אחת ולתמיד ואז המחיר ליחידה יורד ל $2 \times 5 \log n$. אנחנו נרצה להוריד את המכפלה של 2. נזכר מה עשינו – לקחנו את

$$B = b_0, b_1, \dots, b_{16n-1} \Rightarrow C_0, \dots, C_{15n-1}$$

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

סדרת האיברים b היא סדרה ממשית. נתבונן בסדרה הבאה:

$$\overline{B} = b_0 + ib_{15n}, b_1 + ib_{15n+1}, \dots,$$

$$.C_k = \sum_{l=0}^{n-1} a_l \overline{b_{l+k}} = \underbrace{\sum_{l=0}^{n-1} a_l b_{l+k}}_{C_k} + i \underbrace{\sum_{l=0}^{n-1} a_l b_{15n+k+l}}_{C_{15n+k}}$$

ולכן, עבור סדרות ממשיות מחיר יחידה (קוארדינטה) מתקרב ל $5 \log n + c$.

חילוק עם שארית של פולינום ממעלה $O(n)$ דורש $O(n^2)$ פעולות בחילוק ארוך. בהנתן אלגוריתם לכפל פולינום ב $O(n \log n)$ פעולות, ניתן לתאר אלגוריתם לחילוק עם שארית שדורש גם כן $O(n \log n)$ פעולות. נדלג, יש דברים יותר מעניינים.

ראינו כי לחשב את ערכי הפולינום בשורשי היחידה עולה לנו $n \log n$. אבל מה אם בא לנו לחשב אותו ב n נקודות שונות, לאו דווקא שורשי היחידה?

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

טענה: $f(b)$ שווה לשארית בחלוקת $f(x)$ בפולינום $x - b$

$$f(x) = q(x)(x - b) + c$$

$$f(b) = q(b) \cdot 0 + c \Rightarrow f(b) = c$$

אבל חילוק עם שארית עולה לנו די הרבה.

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 $f(x)$. $g(x) = \underbrace{\prod_{i=0}^{n/2} (x - b_i)}_{g_1(x)} \cdot \underbrace{\prod_{i=n/2+1}^n (x - b_i)}_{g_2(x)}$. $T(n)$ לחישוב $g(x)$ הוא

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

אבל אנו לא רוצים את $g(x)$ - אנחנו רוצים לחשב את $f(x)$ ב n נקודות, $b_1, \dots, b_n$.

$$\deg f_1 < \frac{n}{2}$$

$$\deg f_2 < \frac{n}{2}$$

$$f(x) = q_1(x)g_1(x) + f_1(x)$$

$$= q_2(x)g_2(x) + f_2(x)$$

עבור $f(b_k) = f_1(b_k)$, $b_1, \dots, b_{n/2}$

עבור $f(b_k) = f_2(b_k)$, $k = \frac{n}{2} + 1, \dots, n$

$$M(n) = 2M\left(\frac{n}{2}\right) + O(n \log n) =$$

$$O(n \log^2 n)$$

ראינו כי אפשר לחשב פולינום ב n נקודות כלשהן ב $O(n \log^2 n)$.

4.12.2006

השלמה FFT

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

תוצאה נוספת: בהנתן פולינום ממעלה n , $f(x) = a_0 + a_1x + \dots + a_nx^n$ ניתן לחשב את $f(b_1), \dots, f(b_n)$ ב $O(n \log^2 n)$.

בעית חישוב מקדמי פולינום אינטרפולציה

נתונות n נקודות שונות. $a_1, \dots, a_n$, וערכים $b_1, \dots, b_n$, אזי קיים פולינום יחיד $f(x)$, $\deg f \leq n-1$ כך ש $f(a_l) = b_l, \forall 1 \leq l \leq n$.

איך נעשה זאת? נחשב את $g(x)$ בזמן $O(n \log^2 n)$ (נשים לב כי $g(x)$ זה לא $f(x)$...).
ניזכר שניה איך מוגדרת האינטרפולציה של לגרנג'

$$f(x) = \sum_{l=1}^n b_l \frac{\prod_{\substack{i \neq l \\ 1 \leq i \leq n}} (x - a_i)}{\prod_{\substack{i \neq l \\ 1 \leq i \leq n}} (a_l - a_i)} \quad \text{נגדיר } c_l := \frac{b_l}{\prod_{\substack{i \neq l \\ 1 \leq i \leq n}} (a_l - a_i)} \text{ מתקיים:}$$

$$f(x) = \sum_{l=1}^n c_l \prod_{\substack{i \neq l \\ 1 \leq i \leq n}} (x - a_i) =$$

$$\prod_{i=\frac{n}{2}+1}^n (x - a_i) \sum_{l=1}^{\frac{n}{2}} c_l \prod_{\substack{i \neq l \\ 1 \leq i \leq \frac{n}{2}}} (x - a_i) + \prod_{i=1}^{\frac{n}{2}} (x - a_i) \sum_{l=\frac{n}{2}+1}^n c_l \prod_{\substack{i \neq l \\ \frac{n}{2}+1 \leq i \leq n}} (x - a_i)$$

בהנתן c_l לכל l , וחישובי הביניים בחישוב $g(x)$ (ראה שיעור שעבר) נקבל שחישוב $f(x)$ דורש

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

חישוב c_l :

נשים לב כי:

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

$$g'(x) = g_1 + 2g_2x + \dots + (n-1)g_{n-1}x^{n-2} + n \cdot g_n \cdot x^{n-1}$$

$$g'(x) = \sum_{l=1}^n \prod_{i \neq l} (x - a_i)$$

ולכן $g'(a_l) = \prod_{i \neq l} (a_l - a_i)$. נחשב את הערך של $g'(x)$ בנקודות $a_1, \dots, a_n$ ב $O(n \log^2 n)$

צעדים ואז $c_l = \frac{b_l}{g'(a_l)}$ וזה רק עוד n צעדים.

חיפוש טקסט

$$T = T_1 \dots T_m \dots T_n$$

$$P = P_1 \dots P_2 \dots P_m$$

השיטה הפשוטה לזיהוי המיקום של תבנית נתונה באורך m בתוך טקסט באורך n דורשת $O(n \cdot m)$.

נתון א"ב סופי Σ , $\Sigma = \{0,1\}$ או $\Sigma = \{a,b,\dots,z\}$. מילה x בא"ב Σ היא סדרה $x = x_1, \dots, x_n$ כאשר $x_i \in \Sigma$.

נעזר בסימון מיוחד $\varepsilon (\notin \Sigma)$ שמסמן את המילה הריקה שאורכה 0.

$$\begin{aligned} x &= ababca \\ y &= ca \end{aligned} \quad \Sigma = \{a,b,c\}$$

בהנתן שתי מילים x, y , ניתן לשרשר אותן $x.y = ababcaca$.

נסמן Σ^* את המילים בא"ב Σ .

בהנתן מילים $z, x, y \in \Sigma^*$ כך ש $z = x.y$ אזי נאמר ש x היא רישא של z , ו y היא סיפא של z .

בהנתן מילה t ומילה אחרת p נאמר ש p מופיעה ב t אם קיימות מילים x, y כך ש $t = x.p.y$.

$$p = \{ababaca\}, \Sigma = \{a,b,c\}$$

אחרי שקוראים רישא כלשהי של t (נניח x), אנו רוצים לדעת מהי הרישא הארוכה ביותר של p שהיא סיפא של x .

(שרטוטים – אצל דינה)

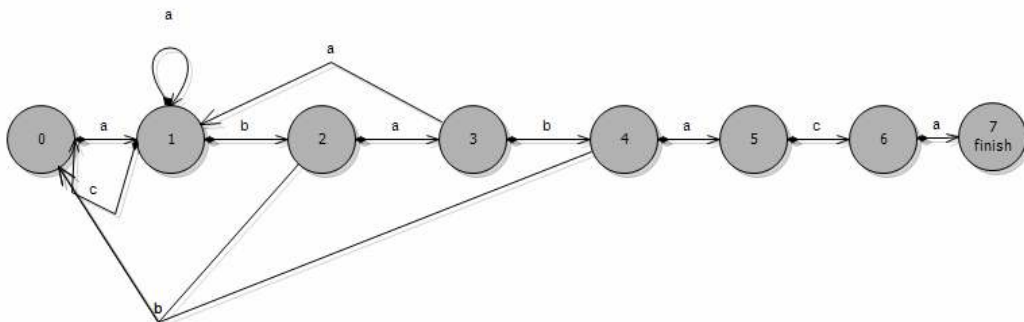
בהנתן תבנית $p = p_1, \dots, p_m$, $p_i \in \Sigma$. נסמן את הרישא של p באורך k .

$$p_k = p_1 p_2, \dots, p_k$$

נגדיר פונקציה $\sigma : \Sigma^* \rightarrow \mathbb{N}$ - אורך הרישא הארוכה ביותר של p שהיא סיפא של המילה x .

בונים אוטומט סופי:

(לא כל הצלעות שורטטו – אפשר להשלים לבד, או להסתכל אצל דינה):



נשים לב שמ7 אנו צריכים לחזור למצב המתאים (לאו דווקא ל0) בהתאם לאות הנקראת.

כאשר נמצאים במצב k וקוראים אות v , $\delta(k, v) = \sigma(p_k.v)$, δ מסמלת מה לעשות בשלב הבא).

כלומר, עלינו לחשב טבלה:

$$\delta : \{0,1,\dots,m\} \times \Sigma \rightarrow \{0,1,\dots,m\}$$

בהנתן הטבלא δ כל שנותר הוא לעבור אות אות על הטקסט ובכל פעם שמגיעים למצב m מודיעים שגילינו מופע של התבנית p בטקסט. נשים לב שבניית הטבלא לוקחת לנו $O(m^2 \cdot |\Sigma|)$

האלגוריתם של Knuth Moris Pratt (עובד ב $O(m+n)$)

<http://www2.toki.or.id/book/AlgDesignManual/BOOK/BOOK5/NODE203.HTM>

פרק 32 Cormen

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

$$\pi(k) = \max \{l \mid * \wedge l < k\}$$

(*) - p_l סיפא של p_k

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

נתון טקסט $t = t[1], \dots, t[n]$. תבנית $p = p[1], \dots, p[m]$ והטבלא π .

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

*

נשווה את $t[i] : p[q+i]$. אם הם שווים, $q \leftarrow q+1, i \leftarrow i+1$. בדוק אם $q = m$ - אז מצאנו

התאמה, ו $q \leftarrow \pi[n]$

אחרת אם $q = 0$ אזי $i \leftarrow i+1$

אם $q > 0$ אזי $q \leftarrow \pi(q)$ וחזור ל *.

String Matching – זיהוי תבניות – Cormen 32

Cormen 32

ציטוט השיעור: 'אני מקווה שאתם מבינים שאני לא טוען שהתכנות שלי נכון'.

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

$$p = P[1]P[2] \dots P[m], t = T[1] \dots T[n]$$

אנו רוצים לזהות את כל המופעים של p ב t .

אם נחלק את $t = x.y$, אנו רוצים למצוא את $\sigma(x)$ - אורך הרישא הארוכה ביותר של p שהיא סיפא של x .

$$0 \leq \sigma(x) \leq m$$

אם $\sigma(x) = k$, אזי $t = \omega \cdot p_k \cdot y$ ורק החלק p_k והאות הראשונה של y קובעים את

$\sigma(p_k v) = \sigma(xv)$, אז לכל k ולכל $v \in \Sigma$ נחשב את $\delta(k, v) = \sigma(p_k, v)$, אזי מעבר אות t על t מאפשר זיהוי של כל המופעים של p .

(קצת שרטוטים – אצל דינה)

אם $v = P[k+1]$ (האות שאנו קראנו עכשיו ב t) אזי $k \leftarrow k+1$

אחרת נגדיר $\pi : \{1, \dots, m\} \rightarrow \{0, \dots, m-1\}$

$$\pi(k) = \max \{l \mid p_l \mid l < k, *\}$$

p_k (*) סיפא של p_l

ונבצע $k \leftarrow \pi(k)$.

(נרשום את p , ועבור כל איבר, את π המתאים לאינדקס שלו:

	0	0	1	2	3	4	5	6	0	1
P	a	b	a	b	a	b	a	b	c	a

כדי להבין את את (π)

תאור האלגוריתם בהנתן π

התחלה: $k = 0$.

עבור $i = 1 \dots n$

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

אם $P[k+1] = T[i]$ בצע $k \leftarrow k+1$

אם $k = m$

הכריזו על התאמה במקום הזה

$$k \leftarrow \pi(k)$$

הערכת זמן ריצה

בהסתכלות ראשונית, נדמה כי זה $O(mn)$, לא יותר טוב מהאלגוריתם הטריטיואלי.

נבחן זאת באופן מדויק יותר. בכל מחזור של הלולאה על המשתנה i המשתנה k יכול לגדול רק ב 1 (כי $k \leftarrow \pi(k)$ רק יכול להקטין אותו...).

אם נשערך את פעולת ההגדלה פי שניים, אזי פעולת הקטנת k תקבל ערך 0, ואז הלולאה הפנימית במחירים החדשים היא בחינם. ולכן כל חזרה על הלולאה על i דורשת $O(1)$, בסה"כ $O(n)$.

כעת, נותר לנו רק...

חישוב מהיר של π בהנתן תבנית P

$\pi(1) = 0$. נניח שחישבנו את $\pi(k)$, ונחשב את $\pi(k+1)$. נגדיר $\pi^i(x)$ כהפעלה i פעמים של π .

$$\pi^*(k) = \{\pi^l(k) \mid l \geq 1\} = \{l \mid *\}$$

$*$ = הרישא p_l היא סיפא ממש ל p_k .

אם הרישא p_{l+1} היא סיפא של p_{k+1} אזי p_l סיפא של p_k , ולכן $l \in \pi^*(k)$.

אם נסמן $E_k = \{l \in \pi^*(k) \mid P[k+1] = P[l+1]\}$, אזי ברור ש $E_k = \{l < k \mid *\}$ $p_{l+1} - *$ סיפא של p_{k+1} .

טוב, התבלבלנו קצת. בואו ננסה להסביר את זה מילולית

אנו עוברים בסדר אורכים יורד על הקבוצה $\pi^*(k)$, ובודקים לכל l בקבוצה האם $P[k+1] = P[l+1]$. בפעם הראשונה שזה קורה, אפשר להגדיר $\pi(k+1) = l+1$. אם לא נעצרנו, נקבל $\pi(k+1) = 0$.

אלגוריתם - חישוב π

$$1. \quad l \leftarrow 0, \pi(1) \leftarrow 0$$

$$2. \quad \text{עבור } k = 2 \dots m$$

a. כל עוד $0 < l$ ו $P[l+1] \neq P[k]$ בצע $l \leftarrow \pi(l)$ (מותר, מאחר וחישבנו

עבור k ימים קטנים יותר)

b. אם $P[k+1] = P[l+1]$ אזי $l \leftarrow l+1$

c. נגדיר $\pi(k) := l$

מאותם שיקולים בדיוק בחישוב סיבוכיות זמן הריצה של האלגוריתם הראשי, כאן העלות היא $O(m)$.

הקדמה לאלגוריתמים הסתברותיים

מפליא, אבל יש בעיות שאנו יודעים לפתור יותר טוב באמצעות הטלות מטבע. לדוגמא – בעיית הפילוסופים הסועדים.

14.12.2006

אלגוריתמים הסתברותיים

הנחה: אפשר להטיל מטבע 'לא מוטה' כלומר נתונים מספר גדול כרצוננו של משתנים מקריים $r_1, \dots, r_k$

כאשר r_i יכול לקבל 0 או 1. $\Pr(r_i = 0) = \Pr(r_i = 1) = \frac{1}{2}$ והמשתנים בלתי תלויים (Pr - Probability).

נשים לב שבפועל אין אופציה להטלת מטבע אמיתית במחשב, זה רק פסאודורנדומי.

אנו רוצים לבחור מספר מקרי בתחום $\{0, 1\}$ (מקרי מתוך תחום $\equiv$ בהתפלגות אחידה).

$S = \{0, 1\}^n$ או לחילופין מספר טביע x בתחום $0 \leq x < 2^n$. מה נעשה? נטיל n מטבעות,

$$P(x) = \left(\frac{1}{2}\right)^n \sum_{i=1}^n r_i \cdot 2^{i-1} \quad x \text{ ההסתברות לכל } x \text{ אזי } r_1, \dots, r_n \in \{0, 1\}$$

מה קורה כאשר רוצים לבחור x טבעי $0 \leq x < 3$? נשים לב שאנו דורשים התפלגות אחידה. אופציות: 1. להטיל 'הרבה' מטבעות, ואז לעשות מודולו. זה יתקרב להתפלגות אחידה. זה מקרה פרטי של

חלוקת התוצאות האפשריות ל-3 קבוצות. נשים לב שזה לעולם לא יגיע באמת ל $\frac{1}{3}$.

2. דרך מדויקת יותר: בוחרים מספר טבעי x בתחום $0 \leq x < 2^2 = 4$. נבחר x בעזרת שתי

הטלות מטבע. אם x קיבלנו קטן או שווה ל-2 נעצור, אחרת – נטיל שוב. בקצרה – מטילים

עד שמקבלים מספר בטעות.

a . באופן כללי: אם רוצים לבחור מספר מקרי בין $\{0, \dots, a\}$, עבור a טבעי כלשהו.

נבחר חזקה של 2 כך ש $a < 2^n$ (נשים לב שתמיד אפשר לבחור n כך

ש $2^n \leq 2a$). נבחר x בתחום $0 \leq x < 2^n$ ואם $x \leq a$ נעצור, אחרת נזרוק את התוצאה ונחזור על הניסוי.

נניח שאנו יכולים לדגום באופן יעיל בהתפלגות אחידה בעזרת הטלות מטבע בתוך תחום Ω , ואנו

רוצים לדגום בקבוצה חלקית $S \subseteq \Omega$ כך ש $\frac{|S|}{|\Omega|} = p$. בנוסף נניח שניתן לבדוק ביעילות האם

$x \in S$ או לא.

טענה: האלגוריתם הבא דוגם באופן אחיד מתוך S – בוחרים x מקרי מתוך Ω ובודקים אם נפלנו לתוך S . אם כן, עוצרים עם x , אחרת, חוזרים על הניסוי באופן בלתי תלוי.

הוכחה פורמלית: שאכן מקבלים התפלגות אחידה כאשר עוצרים. יהי $a \in S$, ונראה

$$\Pr(x = a) = \frac{1}{|S|} \text{ שההסתברות}$$

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

תוחלת הזמן עד לעצירה – ההסתברות ל k חזרות עד להצלחה הוא $p \cdot (1-p)^{k-1}$ (כאשר p הוא

$$\frac{|S|}{|\Omega|}.$$

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

במקרה שלנו, בחירה אחידה מתוך $\{0, \dots, a\}$ כאשר $a < 2^n \leq 2a$ אזי $\frac{1}{2} \leq p$ ולכן תוחלת מספר הטלות המטבע $2n \geq$.

בעיית מציאת חתך מינימלי בגרף קשיר $G = (V, E)$

חתך בגרף זו חלוקה של הקודקודים $V = A \cup B$, כך ש $A, B \neq \emptyset$, $A \cap B = \emptyset$. המחיר של החתך הוא מספר הצלעות בגרף בין הקודקודים ב A ו B .

קיים אלגוריתם פולינומיאלי במספר הקודקודים - נבחר קודקוד $s \in V$. בה"כ $s \in A$. נבחר $s \neq t \in V$ כלשהו. נגדיר רשת זרימה עם קיבולת 1 בשני הכיוונים על כל צלע. חישוב זרימה מקסימלית מ s ל t יתן חתך מינימלי, בהנחה ש $t \in B$. נעבור על $n-1$ ה t ים האפשריים, ונבחר את החתך הקטן ביותר ($n = |V|$).

נוסח אלגוריתם הסתברותי

נגדיר פעולת כיווץ של שני קודקודים בגרף או 'מולטי גרף'. מרשים צלעות מקבילות (או – יותר מצלע אחת בין שני קודקודים). פעולת כיווץ על הקודקודים 2,5 תהפוך את הקודקוד המאוחד ל 2.5.

האלגוריתם של Karger

בוחרים באופן מקרי צלע ומכווצים לאורכה (את שני הקודקודים שהיא מחוברת אליהם). ממשיכים בפעולות הכיווץ, כל פעם בוחרים צלע באופן מקרי, ומכווצים. אחרי $n-2$ חזרות, ישארו 2 קודקודים. שני הקודקודים מייצגים קבוצת קודקודים בגרף המקורי. נבחר חתך לפי הקבוצות האלה.

נחזור על התהליך $n^2 \log n$ פעמים ונבחר את התוצאה הטובה ביותר בתור תשובה.

טענה: בהנתן חתך מינימלי בגרף, ההסתברות שהאלגוריתם יחזיר את החתך הזה בתור תשובה (עבור

$$\text{חזרה בודדת}) \leq \frac{1}{\binom{n}{2}} < \frac{1}{n^2}.$$

נניח שמספר הצלעות בחתך המינימלי הזה הוא k . כדי לקבל את החתך הזה אסור לבחור באחת מ k הצלעות שבחתך.

בצלע הראשונה ההסתברות להכשל היא $\frac{k}{|E|}$. אם החתך המינימלי הוא $k \Leftarrow$ דרגת כל קודקוד

$$k \leq |E| \leq \frac{nk}{2} \text{ ועל כן ההסתברות להכשל היא } \frac{2}{n} = \frac{k}{\frac{1}{2}kn} \text{ ועל כן ההסתברות להצליח}$$

בצעד הראשון $1 - \frac{2}{n} \leq$ אחרי j צעדים של הצלחה, מספר הקודקודים ירד ל $n - j$. כיוון שחתך במולטי גרף הנוכחי הוא חתך בגרף המקורי, גם כאן דרגת כל קודקוד $k \leq$. ולכן, ההסתברות להצליח, כלומר לא לבחור אחת מ k הצלעות המיוחדות (שכולן נותרו) הוא $1 - \frac{2}{n - j}$. ההסתברות להצליח עד

$$\left(1 - \frac{2}{n}\right) \left(1 - \frac{2}{n-1}\right) \dots \left(1 - \frac{2}{n-j}\right) \dots \left(1 - \frac{2}{3}\right) =$$

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

הסוף היא

מסקנה: ההסתברות למצוא חתך בגודל מינימלי $\frac{1}{n^2} \leq$. ההסתברות לכשלון

$$\left(1 - \frac{1}{n^2}\right)^{n^2 \log n} = \left[\left(1 - \frac{1}{n^2}\right)^{n^2}\right]^{\log n} = e^{-\log n} < \frac{1}{n}$$

(כמובן בהנחה שחוזרים $n^2 \log n$).

18.12.2006

בדיקת ראשוניות

נתון מספר טבעי $2 < a < 2^n$. רוצים לבדוק האם a ראשוני.
 a ראשוני $\Leftrightarrow$ לכל b , $1 < b < a$ מתקיים b איננו מחלק את a .

בדיקת חלוקה דורשת $O(n^2)$ פעולות ביטיות. אבל יש מספר גדול של b (כאלה).

הגדרה שקולה: a ראשוני אם לכל $1 \leq b < a$

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

למה? אם a ראשוני זה נכון. אחרת – אם $b \mid a$ (כלומר $b > 1$ מחלק את a אזי

$$b^{a-1} \not\equiv 1 \pmod{a}$$

אחרת $b \mid 1$ $\Rightarrow b^{a-1} = a + ma \Rightarrow b \mid 1$ $b > 1$.

למעשה, אם a ו b יש מחלק משותף $d > 1$ אזי a איננו ראשוני.

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

נוסח פתרונות אחרים:

יהי a מספר פריק.

1. אם b איננו זר ל a $\Leftrightarrow b^{a-1} \not\equiv 1 \pmod{a}$ ולכן b מעיד ש a מספר פריק. $a = p \cdot q$,

אזי $\mathbb{Z}_a^* = \{b \mid \gcd(b, a) = 1\}$. $\psi(a) = (p-1)(q-1) = p \cdot q - p - q + 1$.

כלומר, אם $p, q \sim 2^{n/2}$, $a \sim 2^n$, אזי $\frac{2^{n/2}}{2^n} = 2^{-n/2}$ הסיכוי לבחור $b \notin \mathbb{Z}_a^*$ כאשר

בוחרים b מקרי $1 \leq b < a$. בקיצר – אם בוחרים מספר מקרי בטווח, הסיכוי לבחור אחד שיש לו גורם משותף, הוא נורא קטן.

שאלה: האם זה נכון שלכל a פריק, מרבית ה b ים בתחום $1 \leq b < a$ מקיימי $b^{a-1} \not\equiv 1 \pmod{a}$ או לחילופין, האם יתכן שלכל $b \in \mathbb{Z}_a^*$ מתקיים $b^{a-1} \equiv 1 \pmod{a}$?

התשובה היא לא! עבור 561 - איננו ראשוני, אבל הבדיקה הנ"ל מראה שהוא עונה על התנאים. זה נקרא מספר Carmichael. בשנת 1994 הצליחו להוכיח שלפחות $n^{2/7}$ מספרים עד n הם מספרי Carmichael עבור n ים גדולים.

נעזר בעובדה נוספת הידועה עבור ראשוניים: אם a ראשוני, אזי למשוואה הברורה מאליה של $x^2 \equiv 1 \pmod{a}$ יש בדיוק 2 פתרונות ב $\mathbb{Z}_a^*$. למה? $x = 1, -1$ פותרים את זה.

מסקנה: אם נמצא $d \in \mathbb{Z}_a^*$ כך ש $d \not\equiv \pm 1 \pmod{a}$ אבל $d^2 \equiv 1 \pmod{a}$ אז a פריק.

בדיקת ראשוניות הסתברותית (1977, Rabin)

יהי a , בה"כ a איננו זוגי.

נציג $a-1 = 2^u \cdot t$ כאשר t בלתי זוגי. $u \geq 1$. בוחרים b מקרי בתחום $1 \leq b < a$ ונחשב את $b^{a-1} \pmod{a}$.

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

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

מתבוננים בסדרה $b_0, \dots, b_u$. אם $b_u \neq 1$ עוצרים ומכריזים ש a פריק.

אם יש b_l כך ש $b_l \neq \pm 1 \pmod{a}$ אבל $b_{l+1} = 1$ נכריז ש a פריק. אחרת, נכריז a אולי ראשוני (בסבירות גבוהה).

הגדרה: יהי a מספר פריק. $1 \leq b < a$ כך שבבדיקת b מכריזים על פריקות a , אזי נאמר ש b הוא עד לפריקות a .

נגדיר $w(a) = \{b \mid b \text{ is witness for the decomposability of } a\}$.

משפט: אם a מספר פריק בלתי זוגי אזי $|w(a)| \geq \frac{a-1}{2}$.

משפט מתורת המספרים:

נתבונן בחבורה הכפלית מודולו a $\mathbb{Z}_a^*$. אזי החבורה הזאת היא ציקלית, כלומר קיים $g \in \mathbb{Z}_a^*$ כך ש $\{1, g, g^2, \dots, g^{\varphi-1}\}$ הם כל האברים בחבורה $\mathbb{Z}_a^*$, $|\mathbb{Z}_a^*| = \varphi = \varphi(a)$. במקרים הבאים $a = 2, 4$, $a = p^e$ או $a = 2 \cdot p^e$. כאשר p ראשוני $2 < p$ ו e מספר טבעי כלשהו.

הוכחת המשפט על מספר העדים

1. קיים $b \in \mathbb{Z}_a^*$ כך ש $b^{a-1} \not\equiv 1 \pmod{a}$ כלומר a איננו מספר Carmichael. נגדיר

$B = \{b \in \mathbb{Z}_a^* \mid b^{a-1} \equiv 1 \pmod{a}\}$. $B \neq \emptyset$ ולכן $1 \in B$. סגורה תחת כפל מודולו

a . $B \Leftarrow (b_1 \cdot b_2)^{a-1} \equiv b_1^{a-1} b_2^{a-1} \equiv 1 \cdot 1 \equiv 1 \pmod{a}$ תת חבורה של החבורה הכפלית

$\mathbb{Z}_a^*$. $B \subset \mathbb{Z}_a^*, B \neq \mathbb{Z}_a^*$, ועל כן ממשפט לגרנז' $|B| \mid |\mathbb{Z}_a^*|$. ועל כן $|B| \leq \frac{|\mathbb{Z}_a^*|}{2}$. אם

$$|w(b)| \geq \frac{a-1}{2} \quad b \notin \mathbb{Z}_a^*$$

2. מקרה 2: a מספר Carmichael ולכן לכל $b \in \mathbb{Z}_a^*$, $b^{a-1} \equiv 1 \pmod{a}$.

a. הערה: נשים לב שבמקרה הזה לא יתכן ש $a = p^e$ עבור p ראשוני. כלשהו.

b. הוכחה להערה: יהי g איבר שהחזקות שלו נותנות את כל איברי $\mathbb{Z}_{p^e}^*$ (ל g קוראים

שורש פרימיטיבי מודולו p^e). אזי הסדר של g בחבורה $\mathbb{Z}_{p^e}^*$ שווה למספר

האיברים בחבורה $\varphi(p^e) = p^e - p^{e-1} = p^{e-1}(p-1)$. ועל כן

$g^{p^{e-1}} \equiv 1 \pmod{p^e}$. אם $g^{p^{e-1}} \equiv 1 \pmod{p^e}$ הסדר של g מחלק את

$p^e - 1$, כלומר $p^{e-1}(p-1) \mid p^e - 1$, וזה כמובן לא יתכן – סתירה – שכן

$p^{e-1}(p-1) \mid p^e - 1$ מתחלק ב p בעוד $p^e - 1$ זר ל p .

מסקנה: אם a מספר Carmichael אזי בפרוק לגורמים ראשוניים של a מופיע יותר מאשר גורם ראשוני אחד. $a = p_1^{e_1} p_2^{e_2} \dots p_l^{e_l} = n_1 \cdot n_2$. $n_1, n_2 > 1$. מספרים זרים זה לזה - $n_1 = p_1^{e_1}, n_2 = p_2^{e_2} \dots p_l^{e_l}$. לכל $b \in \mathbb{Z}_a^*$ במהלך הבדיקה נקבל תמיד $b_u \equiv 1 \pmod{a}$. נתבונן בסדרות $(b_0, b_1, \dots, b_n)$ המתקבלת עבור b , ונאמר שהזוג (b, j) הוא **מתאים** אם $b_j = b^{t \cdot 2^j} \equiv (-1 \pmod{a})$.

נשים לב ש $(-1, 0)$ הוא זוג מתאים, כי t הוא בלתי זוגי - $a - 1 = 2^u \cdot t$

$$(-1)^t \equiv (-1) \pmod{a}$$

יהי j המספר המקסימלי שעבורו יש b כך ש (b, j) זוג מתאים. נתבונן ב $B = \{b \in \mathbb{Z}_a^* \mid b^{t \cdot 2^j} \equiv \pm 1 \pmod{a}\}$. אם $b \in \mathbb{Z}_a^*$ אבל $b \notin B$ אזי b הוא עד. בנוסף B סגורה תחת כפל. נותר להראות ש $B \subset \mathbb{Z}_a^*, B \neq \mathbb{Z}_a^*$ ועל כן B תת חבורה.

B מכילה את כל ה'לא עדים' וגודלה $\frac{|\mathbb{Z}_a^*|}{2} \geq$ ולכן סיימנו. די 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

השאריט הסיני קיים w , $0 \leq w < a$, $w \equiv b \pmod{n_1}$ ו $w \equiv 1 \pmod{n_2}$. זר ל n_1 .

$\gcd(w, n_1) = 1$ כי $\gcd(b, a) = 1$ ועל כן $\gcd(b, n_1) = 1$. ועל כן -

$\gcd\left(w, \underbrace{n_1 n_2}_{=1}\right) = 1$ ועל כן $w \in \mathbb{Z}_z^*$. אם $w^{t \cdot 2^j} \equiv \pm 1 \pmod{a}$ אזי

$w^{t \cdot 2^j} = \pm 1 + k(n_1 n_2)$ ולכן או $w^{t \cdot 2^j} \equiv 1 \pmod{n_1}$ ו $w^{t \cdot 2^j} \equiv 1 \pmod{n_2}$ או שניהם

-1. אבל $w^{t \cdot 2^j} \equiv b^{t \cdot 2^j} \equiv -1 \pmod{n_1}$ - סתירה - $w \notin B$. מ.ש.ל.

25.12.2006

ציטוט השיעור¹: 'השערת רימן היא אחת הבעיות הפתוחות הגדולות במתמטיקה. אתם תוכיחו אותה בתרגיל'.

בדיקת ראשוניות

נתון מספר טבעי $a < 2^m$ ורוצים לבדוק האם a ראשוני. לראשוניים יש שתי תכונות:

1. מודולו a ל 1 יש בדיוק 2 שורשים ריבועיים $\{1, -1\}$

2. $b^{a-1} \equiv 1 \pmod{a}$, $1 \leq b < a$

הנחה: $2 < a$, אזי $a-1 = 2^u \cdot t$, בלתי זוגי. בוחרים b מקרי ומחשבים מודולו a את $b^t, b^{2^t}, b^{4^t}, \dots, b^{2^{u^t}}$. זוהי סדרה של העלאות בריבוע.

אם $b_u \neq 1$ לא ראשוני. ואם עוברים ממספר 1 ± 1 גם אז a לא ראשוני.

אם a איננו ראשוני אזי b כזה הוא עד לפריקות a .

$$W(a) = \{b \mid 1 \leq b < a, b \text{ witnesses decomposability of } a\}$$

הוכחנו (בשעתיים בפעם שעברה, הפרטים לא חשובים) כי אם a פריק אזי $|W(a)| \geq \frac{a-1}{2}$

נקבל אלגוריתם הסתברותי:

בהנתן a בוחרים b מקרי $1 \leq b < a$ ובודקים האם $b \in W(a)$. זה דורש $O(n^3)$ פעולות

ביטיות. (כאשר $a < 2^n$ - זה ה n המדובר...)

אחרי בדיקה אחת כזאת: 1. אם מצאנו עד לפריקות אזי a פריק. 2. אחרת, ההסתברות ש a פריק

$$\geq \frac{1}{2}$$

אחרי 100 חזרות נקבל שאו שגילינו ש a פריק בוודאות או ש a ראשוני (בהסתברות שגיאה $>$

$$\frac{1}{2^{100}})$$

Aks 2004 נתנו אלגוריתם **דטרמיניסטי** פולינומיאלי לבדיקת ראשוניות, אבל ב $O(n^{12})$.

בחירת מספר ראשוני $2^n > p$ בהסתברות אחידה על פני המספרים הראשוניים

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

משפט המספרים הראשוניים:

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

$$2. \zeta(s) = \sum_{n=1}^{\infty} \frac{1}{n^s} = \prod_{p \text{ prime}} (1 - p^{-s})^{-1}, s \in \mathbb{C}$$

וקצת יותר שימושי (מאחר ו 1 לא עוזר, ו 2 לא תורם), אזי **לכל x מתקיים:**

¹ טוב, נו, לא באמת.

$$\frac{7}{8} < \frac{\pi(x)}{x/\ln x} < \frac{9}{8}$$

מסקנה: מספר המספרים הראשוניים בני n ביטים גדול מ $\frac{2^n}{2n}$. עבור $n = 1024$ בערך $\frac{1}{2000}$ מהמספרים הם ראשוניים (עד 2^{1024}).

אם רוצים לבחור ראשוני מקרי, נבחר מספר מקרי בתחום עד 2^n ונבדוק האם הוא ראשוני. נחזור עד להצלחה. תוחלת מספר החזרות הוא n .
אם בבדיקת הראשוניות חוזרים על הבדיקה k פעמים, אזי הסתברות השגיאה (כלומר, הכרזה על מספר פריק כראשוני) $\frac{1}{2^k} >$. נשים לב שהחזרה על הבדיקה (n פעמים) לא מגדילה את הסיכוי לשגיאה – כי אנו חוזרים רק אם מצאנו מספר פריק, ואז אין סיכוי לשגיאה.

בהנתן שני מספרים ראשוניים, p, q , $2^{n/2} > p, q$, ניתן להכפיל אותם באופן יעיל, ולמכפלה $N = pq$ יש פחות מ n ביטים.

בעיית פירוק מספרים טבעיים לגורמים ראשוניים

נתון מספר a ($2^n > a$). מהו האלגוריתם הטוב ביותר לפירוק a לגורמים? אפשר בזמן $2^{n/2}$ ($\sim \sqrt{a}$) ניתן למצוא גורם של a , לחלק ולהמשיך.

בשנות ה-60 מצאו פתרון ב $O(2^{n/4})$.

בשנות ה-70 מצאו פתרון ב $2^{O(\sqrt{n \log n})}$. עבור 1024 זה יוצא בערך 100.

ב-1992 מצאו פתרון ב $2^{O(n^{1/3} (\log n)^{2/3})}$. כלומר – אם מגדילים את n ל $8n$, אזי זמן הריצה עולה בריבוע.

בקיצור, לפרק מספרים זה קשה.

ב-1995 הראו שאם ניתן לבנות מחשבים קוונטים אזי יהיה קל לפרק מספרים. בינתיים ההישג הכי גדול של מחשב קוונטי הוא לפרק את 15 ל $3 \cdot 5$.

הצפנה

מפתח חד פעמי

Alice ו Bob רוצים לדבר ביניהם. בהצפנה קלאסית ל A ול B יש מפתח הצפנה ופענוח משותף שנבחר באופן מקרי מתחום כלשהו $k \in \{0,1\}^n$. אם A ו B בחרו k מקרי כזה מראש, ואז A רוצה לשלוח הודעה $m \in \{0,1\}^n$ ל B אזי אם $k = k_1 \dots k_n$ ו $m = m_1 \dots m_n$, נגדיר

$$x \oplus y = x + y \pmod{2} \text{ (xor).}$$

A יכולה לחשב את הוקטור $C = k \oplus m = (k_1 \oplus m_1, \dots, k_n \oplus m_n) = (c_1, \dots, c_n)$ ולשלוח את C ל B.

נשים לב, שלכל m כאשר k נבחר באופן מקרי, אזי C יהיה מפולג באופן אחיד. B יכול לחשב את $C \oplus k = m + k \oplus k = m$, ולקרוא את ההודעה.

אם אורך המפתח שווה לאורך האינפורמציה, לא ניתן לפענח זאת. אבל זו שיטה לא כל כך נוחה לשימוש ביום יום (אלא אם אתה רוצה להעביר קוד לפצצה גרעינית, ואפשר להעביר אותו קודם, או דברים כאלה).

D-H (1977) - שיטת המפתח הפומבי

נניח שכל אחד מן השחקנים B (או A) יכולים לבחור מפתח הצפנה e_B (או e_A , עבור Alice) ומפתח פענוח d_B . יהי M תחום ההודעה $\{0,1\}^n$. נגדיר $E_{e_B}: M \rightarrow M$ הצפנה, $D_{d_B}: M \rightarrow M$ פונקציית הפענוח, כך ש $\forall m \in M, D_{d_B}(E_{e_B}(m)) = m$.

נשים לב שאם D ו E היא פונקציית הזהות, אזי זה יעבוד ואפילו יהיה פשוט. זה רק לא יהיה מוצפן. לכן אנו נרצה שיתקיימו התכונות הבאות:

- כל שחקן שיוודע את e_B יכול לחשב ביעילות את $C = E_{e_B}(m)$ לכל m .
- אם שחקן יודע את e_B אזי גם אם הוא רואה הודעה מוצפנת $C = E_{e_B}(m)$ אבל איננו יודע את m לא יכול לחשב בזמן סביר מהו m (למעט מספר זניח של m).

נשים לב, כי Alice לא צריכה לפגוש את Bob (למרות שזה עשוי לפגוע בקשר הרומנטי שביניהם), כי אין בעיה ש Bob יפרסם את e_B - זה לא עוזר לפתוח דברים שמוצפנים על פיו. אם A רוצה לשלוח הודעה ל B, אזי $C = E_{e_B}(m)$, ושולחת את B. B יודע את d_B , ולכן יכול לחשב את $D_{d_B}(C) = m$ ולקרוא את המסר.

כלומר, אנו מחפשים שיטה כזאת כך שחשוב E, D יעילים, אבל בפרט לא ניתן לחשב את מפתח הפענוח ממפתח ההצפנה.

מערכת כזו של מפתח הצפנה פומבי מאפשרת גם חתימות דיגיטליות. אם $D_{d_B}(E_{e_B}(m)) = m \Leftrightarrow D_{d_B}(E_{e_B}(m)) = m$ כי $D = E^{-1}$.

B פרסם את e_B ושמר בסוד את d_B . על כן, B הוא היחיד שיכול לחשב את הפונקציה D_{d_B} . ולכן, אם B רוצה לחתום על הודעה כלשהי m , אזי הוא יכול לחשב את $s = D_{d_B}(m)$, ושולח ל A את (m, s) .

כדי לבדוק שזו חתימה של B, פשוט נבדוק כי $E_{e_B}(s) = m$.

$$E_e : M \rightarrow M$$

$$D_d : M \rightarrow M$$

$$D_d = E_e^{-1}$$

הן מפתחות הצפנה, e - הצפנה, d - פענוח. כך שאין אלגוריתם יעיל לפענוח (כלומר חישוב D_d למרות שידועים את e).

ב-1977 – RSA – פיתחו אלגוריתם המבוסס על הקושי של פרוק מספרים לגורמים.

יש שיטות נוספות הנעזרות בבעיות קשות אחרות.

שיטת RSA – תהליך יצירת המפתחות

B בוחר שני מספרים ראשוניים מקריים (p, q) . בני $\frac{n}{2}$ ביטים כ"א. נחשב את המכפלה

$$N = p \cdot q \quad (\text{ל } N \text{ יש עד } n \text{ ביטים}).$$

$$\mathbb{Z}_N^* = \{a \mid 1 \leq a < N, \gcd(a, N) = 1\}$$

נחשב את $|\mathbb{Z}_N^*|$

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

$$a \in \mathbb{Z}_N^* \text{ מתקיים } a^{(p-1)(q-1)} \equiv 1 \pmod{N}.$$

$$\text{בוחרים } 1 \leq e < \varphi \text{ כך ש } \gcd(e, \varphi) = 1.$$

$$\text{נחפש } d, \text{ כאשר } 1 \leq d < \varphi \text{ כך ש } e \cdot d \equiv 1 \pmod{\varphi}.$$

איך נמצא d כזה? כאשר מחשבים את המחלק המשותף המכסימלי לפי האלגוריתם של אוקלידס, מקבלים d, f המקיימים $de + f\varphi = 1$. נתבונן על תוצאה זו מודולו φ ונקבל

$$de + 0 = de \equiv 1 \pmod{\varphi}.$$

המפתח הפומבי של B יהיה הזוג (N, e) . תחום ההודעות $M = \{m \mid 0 \leq m < N\}$. פונקציית ההצפנה היא:

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

כדי לפענח:

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

$$\text{טענה: } D_d(E_e(m)) = m, m \in M \text{ לכל } m \in M.$$

$$\text{נזכור כי } e \cdot d = 1 + k \cdot \varphi, \text{ וכי } m^\varphi \equiv 1 \pmod{N}$$

$$\text{אם } \gcd(m, N) = 1, \text{ אז } m^{\varphi} \equiv 1 \pmod{N}, \text{ אז } m^{ed} \equiv m^{1+k\varphi} \equiv m \pmod{N}.$$

$$\text{אם } p \mid m \text{ אזי } \neg(q \mid m) \text{ (אחרת } N \leq m).$$

$$e \cdot d \equiv 1 \left(\text{mod} \underbrace{(p-1)(q-1)}_{=\varphi} \right) \Rightarrow e \cdot d \equiv 1 \pmod{(q-1)}$$

ולכן, בדומה למקרה הקודם, אבל מודולו q , אזי

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

ולכן, אם עושים את אותו תהליך כמו קודם, יוצא:

$$m^{e \cdot d} \equiv \bar{m} \pmod{q}$$

($\bar{m}$ - השארית של m בחלוקה ב m). וכן $m^{e \cdot d} \equiv 0 \pmod{p}$ (שכן m מתחלק ב p). כלומר:

$$m^{\varphi} \equiv m \pmod{N}, N \text{ שמודולו } N, \text{ ממשפט השאריות הסיני, או יודעים, שמודולו } N, \begin{cases} m \equiv \bar{m} \pmod{q} \\ m \equiv 0 \pmod{p} \end{cases}$$

הערה: כדי באמת להשתמש בשיטה, אזי כדי מומלץ לבחור עוד סדרה מקרית r בגודל 512, ולשלוח רק 512 ביטים שלי, ואת החלק המקרי, כדי שמידע כמו 0,1 אכן יוצפן, וגם כדי שחזרה על שתי הודעות לא תראה בקו כהודעה זהה.

הערכת הסיבוכיות:

1. יצירת זוג מפתחות:

a. $O(n^4)$ (גודל המפתח - 1024 לדוגמא)

2. הצפנה ופענוח:

a. פעולות כפל מודולו N דורשת $O(n^2)$ פעולות.

b. להצפנה צריך $O(\log e)$ פעולות. אם בחרנו e קטן (לדוגמא - 7,17) אזי הצפנה

תהיה $O(n^2)$ בסה"כ.

c. לפענוח צריך לבחור d כלשהו כללי, ולכן הוא בן n ביטים. לכן פענוח דורש $O(n^3)$ פעולות. (זה גם מחיר חישוב חתימה בשיטת RSA).

הערה: RSA לא יעילה באמת לקצבים גדולים. לכן משתמשים בה בד"כ בשביל להחליף מפתחות זמניים, ואז עובדים איתם.

הערה: אם אכן לכל אלגוריתם שמצליח לפענח הודעות RSA בהסתברות הצלחה 1% (עבור הודעות מקריות), אז דרוש לו זמן שהוא לפחות $2^{n^{1/10}}$ (הוא מספר הביטים של מודולו N).

אם 2^{ϵ} ($\epsilon > 0$) קבוע כלשהוא) הוא אכן הקושי של חישוב ההופכי להצפנת RSA (לא יודעים להוכיח את זה), אזי אם לוקחים $1 \leq r < N$ מקרי, ומפעיל

$$r \rightarrow E_e(r) = r_1 \rightarrow E_e(r_1) \rightarrow \dots \rightarrow r_k = E_e(r_k)$$

אזי נקבל סדרה $r_0 = r, r_1, \dots, r_k$.

לכל אחד מהם נגדיר ביט $b_0, \dots, b_k$ בהתאם לזוגיות של r_k .

נתבונן ב

$$b_0, \dots, b_{n^{1/10}}$$

$$c_0, \dots, c_{n^{10}}$$

כאשר c_i הטלות מטבע. אם אפשר להבדיל בין הסדרות הללו, אזי אפשר לשבור את RSA. לכן, בהנחה ש RSA לא שביר כרגע, אפשר להשתמש בטריק לעיל בשביל לייצר מספרים רנדומיים. מאחר ואפשר להמשיך את סדרת ה b_i ים עד $b_{2^{n^{1/10}}}$, וזה עדיין יראה מקרי, אזי אפשר ליצור ככה מספרים אקראיים בקלות.

אם רוצים להריץ אלגוריתם שדורש m אקראיים, אזי $m \sim 2^{n^{1/100}}$.

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

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

על כן – די במספר קטן מאוד של ביטים מקריים (אמיתיים) כדי לקבל הרבה מספרים אקראיים.

שאלה: יש כספת, שאנו רוצים ש 7 אנשים יוכלו לפתוח, אבל רק כשיש שני מפתחות כלשהן מתוך ה-7. מה עושים?

(בניסוח אחר):

נתון סוד S , ב.ה.כ. S הוא מספר טבעי $0 \leq s < p$ (כאשר p ראשוני). ורוצים לחלק את הסוד S בין n שחקנים ($n < p$) כך שלכל קבוצה בגודל $k \geq$ לא יהיה שום מידע על S , אבל קבוצות בגודל $k + 1$ תוכל לשחזר את S .

בוחרים פולינום פולינום מקרי $f(x)$ שנראה $f(x) = S + a_1x + \dots + a_kx^k$ כאשר $a_1, \dots, a_k$ מקריים בתחום $[0, \dots, p - 1]$.

לכל שחקן p_r , $r = 1 \dots n$ נותנים את הסוד הבא, $s_r = f(r) \pmod{p}$.

ברור $k + 1$ שחקנים יכולים לשחזר את הפולינום (שכן הוא ממעלה $k + 1$).

למה קבוצה בגודל k לא יכולה לדעת כלום? ב.ה.כ. זו הקבוצה $\{1, 2, \dots, k\}$. הם יודעים את $s_1, \dots, s_k$.

אם אני לוקח את $s_1, \dots, s_k$ ועוד נקודה $s = f(0) \pmod{p}$. בעזרתה - אפשר לחשב את

הפולינום. בנוסף, אם יודעים את S אז זה קובע $s_1, \dots, s_k$.

על כן, יש p^k אפשרויות לבחירת $a_1, \dots, a_k$, ויש p^k אפשרויות ל $s_1, \dots, s_k$. לכן, חייבים $k + 1$ שחקנים כדי לגלות את S .

1.1.2007

ציטוט השיעור: 'זה לא אלגוריתם יותר יעיל, אבל זה אלגוריתם הרבה יותר מסובך' תלמיד אלמוני.

יהי p ראשוני $n < p$. נתון סוד $s \in F_p$, $0 \leq s < p$. רוצים לתת את הסוד S בידי n שחקנים כך שייתכן:

- I. לכל קבוצת שחקנים אין כל מידע מהן S .
- II. כל קבוצה בגודל $k + 1$ יכולה לחשב ביחד את S .

בוחרים פולינום מקרי מעל השדה F_p $f(x) = S + a_1x + \dots + a_kx^k$. איברים מקריים $a_1, \dots, a_k$ מהשדה. ניתן לשחקן p_k את $s_k = f(a_k)$, $k = 1, \dots, n$. מתקיים בבירור כי כל קבוצה בגודל $k + 1$ מחזיקה $k + 1$ נקודות אינטרפולציה לפולינום ממעלה $k \geq$, ויש פולינום יחיד כזה, ולכן הוא $f(x)$, ולכן הם יכולים לחשב גם את S .

הוכחת I – נראה שלכל קבוצה של k שחקנים, התהליך שבו יצרנו את המספרים שהם קיבלו זהה לתהליך שבו נבחרים k מספרים בלתי תלויים בהסתברות אחידה על פני כל ה k יות האפשריות של מספרים קטנים מ p . בה"כ קבוצת השחקנים היא $p_1, \dots, p_k$. הם קיבלו את $s_1, \dots, s_k$. נראה שלכל סדרה של k מספרים יש הסתברות $\frac{1}{p^k}$.

אנו בחרנו את $a_1, \dots, a_k$ בהסתברות אחידה. אם נבחר s , זה יקבע את $s_1, \dots, s_k$. נשים לב כי $s = f(0)$. אבל בהנתן $f(0)$, נקבע s . $f(0)$ קובע את s , ולהפך. לכן ההעתיקה שמעתיקה שמשמשת ב s כדי לעבור מ $a_1, \dots, a_k$ ל f היא חז"ע ועל. מאחר ו $a_1, \dots, a_k$ נדגם באופן אחיד, אזי גם f נדגם באופן אחיד. הבחירה האחידה של a ים משרה התפלגות אחידה על $s_1, \dots, s_k$.

שימוש אחר בטכניקה הזו - RAID

נניח שיש לנו k איברים $(a_0, \dots, a_{k-1})$ בשדה מודולו p . ניתן לבנות את הפולינום $g(x) = a_0 + a_1x + \dots + a_{k-1}x^{k-1}$ נחשב את n הערכים $(g(1), \dots, g(n))$ ($n \geq k$). מכל k מה b ים ניתן לשחזר את k ה a ים ולכן אם מאבדים מסיבה כלשהי $n - k$ מה b ים, לא איבדנו אינפורמציה. אם יש לנו $n = 10$ דיסקים ורוצים להגן בפני נפילה של 3 מהם – אזי בוחרים $k = 7$.

ננסה לראות האם פולינום כלשהו הוא זהותית 0. יהי $\mathbb{F}$ שדה. יהי $p(x_1, \dots, x_n)$ פולינום ממעלה d מעל השדה.

$p(x, y) = 5 + 7x^2y^3 - 17y^4$
בהנתן פולינום אפשר להציג פולינום בצורה מלאה או בצורה של נוסחה לחישוב הפולינום או בעזרת תוכנית שמחשבת בדרך כלשהו את ערכי הפולינום בנקודה נתונה.
אם $d = \deg p(x)$ ויש n משתנים צריך $(d + 1)^n$ נקודות אינטרפולציה כדי לחשב את מקדמי $p(x)$.

$$(x_1 + x_2) \times (x_3 + x_4) \times (x_5 + x_6) \times \dots \times (x_{n-1} + x_n)$$

המעלה של הפולינום המוגדר על ידי הנוסחה הזו היא $\frac{n}{2}$. מספר המונומים השונים בהצגה המלאה $2^{n/2}$

$$p(x_1, x_2) = ((x_1 - 3x_2 + 7) \times (\dots), \dots)$$

הלמה של Zippel-Schwartz

יהי $\mathbb{F}$ שדה, $0 \neq p(x_1, \dots, x_n)$ פולינום ממעלה $d \geq 1$ מעל $\mathbb{F}$. ותהי $S \subseteq \mathbb{F}$ קבוצה סופית כלשהי. אזי, אם בוחרים $a_1, \dots, a_n \in S$ בהתפלגות אחידה בלתי תלויה, אזי ההסתברות

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

הוכחה: באינדוקציה על n . עבור $n = 1$, לפולינום $p(x)$ יש לכל היותר d שורשים בשדה $\mathbb{F}$ ולכן

$$\frac{d}{|S|} \geq \text{ההסתברות שהוא יתאפס}$$

כעת, נניח ל $n - 1$, ונוכיח עבור n נרשום:

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

כך ש $0 \neq p_k(x_1, \dots, x_{n-1})$, $k \leq d$

נשים לב שכך $\deg p_k(x_1, \dots, x_{n-1}) \leq d - k$

$$\Pr(p(a_1, \dots, a_n) = 0) \leq \Pr(p_k(a_1, \dots, a_{n-1}) = 0) + \Pr(*) \leq \text{Induction Hypothesis, for } n-1 \text{ and for } n=1$$

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

$$(*) - a_n \text{ מאפס את } p(a_1, \dots, a_{n-1}, x_n) \neq 0$$

נימוק - אם זה קורה, נניח שההצגה של $a_1, \dots, a_{n-1}$ אפסה את כל ה p_l , $l = 0, \dots, k$ ואז כל בחירה של a_n נותנת 0.

דוגמא לשימוש

בהנתן מטריצה מסדר $n \times n$ $A = (a_j)$, $\det(A) = \sum_{\pi \in S_n} \text{sgn}(\pi) \prod_{i=1}^n a_{i\pi(i)}$. הפרמנטה של A

$$\text{מוגדרת } Per(A) = \sum_{\pi \in S_n} \prod_{i=1}^n a_{i\pi(i)} \text{ (כמו דטרמיננטה, בלי הסימן).}$$

ידועים אלגוריתמים יעילים לחישוב $\det(O(n^3))$ טריוויאלי, אפשרי ב $O(n^{2.36})$, אבל האלגוריתמים הטוב ביותר לחישוב $Per(A)$ הוא אקספוננציאלי.

בהנתן גרף דו צדדי $G = (L \times R, E)$, אפשר לשאול האם קיים זיווג מלא בגרף וכן לשאול מהו מספר הזיווגים המלאים בגרף.

הערה: אם נגדיר מטריצה A

$$a_{ij} = \begin{cases} 1, (i, j) \in E \\ 0, otherwise \end{cases}$$

אזי מספר הזיווגים המלאים בגרף $G = Per(A)$. אבל אנו לא יודעים לחשב Per בזמן טוב, רק $\det$.

אם $\det(A) \neq 0$ אבל כמובן ייתכן שיש זיווגים מלאים ובכל זאת התוצאה היא 0 (יש אותו מספר זיווגים עם סימן + ו-).

נגדיר A חדשה. יהיו $x_{11}, x_{12}, \dots, x_{nn}$ משתנים. $a_{ij} = \begin{cases} x_{ij}, (i, j) \in E \\ 0, otherwise \end{cases}$. נחשב את

הדטרמיננטה, $\det A$. מתקיים $\det A \equiv 0 \Leftrightarrow$ אין זיווגים מלאים.

מתקיים $\deg \det A \leq n$. אם נציב ערכים מתוך תחום S שגודלו $100n$ ונחשב את הערך של

$\det A$ בנקודה הזאת. ההסתברות ש $\det(A) \neq 0$ ובכל זאת נקבל 0, קטנה מ $\frac{1}{100}$.

4.1.2007

בעיות אופטימיזציה

בעיות המוגדרות ע"י

- צורת פתרון מותר (משפחה F)
- איכות הפתרון – מוגדרת ע"י $q(s)$

דוגמאות:

- עץ פורש מינימום
- מסלול קצר ביותר
- צביעת גרף במינימום צבעים

צביעת גרף במינימום צבעים

- קלט: גרף
- פלט: צבע לכל קודקוד בגרף, כך שאסור שקודקודים סמוכים יהיו צבועים באותו צבע. לדוגמא – גרף לא קשיר אפשר לצבוע בצבע אחד לדוגמא.

נשאל האם יש לנו דרך לקרב את הפתרון – להגיע לפתרון 'טוב' בזמן זול יחסית.

אלגוריתם C – קרוב

הרעיון: מותר לתת פתרון בעל איכות קרובה ל Opt (האופטימלי). מדוע: כי נצליח לתת פתרון מקורב ביעילות.

נאמר לבעיית מינימוזציה הוא C-קירוב אם לכל קלט לאלגוריתם איכות הפלט $q(s)$ מקיימת $q(s) \leq c \cdot q(opt)$.

לדוגמא, מתי נגיד שאנו ב-C קירוב לבעיית מקסימיזציה – כאשר $q(s) \geq \frac{q(opt)}{c}$. (כלומר – לא הפסדתי יותר מדי...)

"מרשים": אלגוריתם פולינומיאלי שנותן $1 + \epsilon$ קרוב לבעיה אקספוננציאלית, לכל $\epsilon > 0$.

בעיית התזמון על מכונות זהות

קלט: n משימות הדורשות זמנים $t_1, \dots, t_n$. מספר k של מכונות.

פתרון: חלוקה של המשימות ל k המכונות

איכות הפתרון: זמן הסיום של המכונה האחרונה.

דוגמא: 3 מכונות, משימות שדורשות - (13,12,5,6,10,2,4).

פתרון לדוגמא: $\left(\begin{array}{c|c|c} 4 & & \\ \hline 10 & 13 & 5 \\ 2 & 12 & 6 \\ \hline M_1 & M_2 & M_3 \end{array} \right)$. איכותו: 25.

אלגוריתם 2 קירוב לבעיה (השיא העולמי - 1.8 קירוב).

נסמן $f_1 = 0, \dots, f_k = 0$ מסמן את הזמן שבו מכונה j מסיימת בהקצאה הנוכחית.

for $i=1$ to n

$f_j := \min \{f_i \mid 0 \leq i \leq k\}$ המכונה שמסיימת ראשונה נכון לעכשיו

allocate t_i to M_j

$f_j := f_j + t_i$

הרצה עבור קלט הדוגמא:

$$q(s) = 21 \quad q(opt) = 18$$

4	2	10
13	12	5
M_1	M_2	M_3

טענה: $q(s) \leq 2q(opt)$

סימונים: $q(opt[x])$: הביצוע של opt על x ($0 \leq x \leq n$) המשימות הראשונות.

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

$f_j[x]$ - זמן הסיום של המכונה f_j בהקצאה של x המכונות הראשונות.

קושי: איך נוכיח משהו על $q(opt)$ בלי לדעת מהו opt ?

הרעיון: נחפש דברים בגנות האופטימלי - $q(opt[x]) = \frac{\sum_{i=1}^x t_i}{k} = \frac{\sum_{j=1}^k f_j[x]}{k}$ (חלוקה מאוזנת היא

אידיאלית, אם היא אפשרית).

בנוסף - $\{f_j[x] \mid 1 \leq j \leq k\} \leq q(opt)[x]$ וכי

$$\forall x, t_x \leq q(opt)[x]$$

טענה: $q(s) \leq 2q(opt)$

נוכיח באינדוקציה על x : לכל x $q(s)[x] \leq 2q(opt)[x]$

הוכחה באינדוקציה על x :

בסיס האינדוקציה - עבור $x=1$ - $q(s)[1] = q(opt)[1] = t_1$

צעד האינדוקציה - מהנחת האינדוקציה, -

$$q(s)[x-1] \leq 2q(opt)[x-1]$$

$$q(s)[x] = \max \left\{ \underbrace{\min \{f_j[x-1]\}}_{\text{the machine we've decided to allocate } t_x \text{ to}} + t_x, \underbrace{q(s)[x-1]}_{\text{previous quality}} \right\}$$

מתקיים $\min \{f_j[x-1]\} + t_x \leq 2q(opt)[x]$, ומכאן נובעת הטענה.

סיבוכיות: מבצעים n פעמים מציאת מינימום ($O(\log k)$ וחיבור - $O(1)$) – בסה"כ - $n \log k$. בנוסף, מדובר באלגוריתם מקוון (on-line): מקבל את המשימות בזמן אמת (אפשר למיין את המשימות מראש).

יש קלטים רעים – עבורים $q(s) = 2q(opt) - \varepsilon$. לדוגמא – 2 מכונות, מקבלים 6 משימות של 1, ואז משימה אחת של 6. הפתרון שלנו יתן פתרון של 9, לעומת אופטימלי של 6.

מצב אחר – 100 מכונות - 99×100 משימות של 1, ומשימה אחת של 100. לאחר ה- 99×100 משימות הראשונות, כל המכונות יוקצו 99 משימות של 1, ואז מכונה אחת תשלם עוד 100 – נקבל עלות של 199, כאשר אלגוריתם אופטימלי יתן 100.

בעיית הכיסוי ע"י קבוצות (set cover)

קלט: קבוצה $X = \{1, \dots, n\}$ (תכונות) ו m תתי קבוצות של X $S_1, \dots, S_m$ (לכל איש – כמה תכונות – כל תכונה – מספר מ $1 \dots n$).
(יש לי אנשים עם תכונות – קירח, תינוק, מתולתל, ... – ואני רוצה לקחת מינימום אנשים כך שיהיו לי אנשים מכל תכונה).

פתרון (קרוב $\ln n$) – אלגוריתם חמדני

רעיון: בכל איטרציה נבחר את הקבוצה שמכסה הכי הרבה איברים (מאלה שעדיין לא כוסו).

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

Algorithm:

$C = \{ \}$

while $X \neq \emptyset$

i:=תוספת הגדולה ביותר

$C := C \cup \{i\}$

$X := X \setminus S_i$ $(O(|S_i|))$

for all $j, S_j := S_j \setminus S_i$ $(O\left(\sum_j |S_j|\right))$ – פולינומיאלי

נראה כי $|C| \leq \ln n |opt|$

סימונים: $x[j]$ – האיברים שנותרו ב x אחרי ש j קבוצות נלקחו

$S_i[j]$ – האיברים שנותרו ב S_i אחרי ש j קבוצות נלקחו

$n[j]$ – $|X[j]|$ – כמה איברים נשארו אחרי ש j קבוצות נלקחו.

מתקיים:

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

$$|S_i[j]| \geq \frac{n[j]}{|opt|} \Leftrightarrow |opt| \geq \frac{n[j]}{|S_i[j]|} - j \quad \text{לכל } j \quad 2.$$

a. הוכחה:

- i. Opt צריך לקחת את כל $x[j]$.
- ii. כל הקבוצות שנלקחו עד האיטרציה ה- j ית לא מכילות איברים מ- $x[j]$.
- iii. $S_i[j]$ היא הקבוצה הכי גדולה שנותרה.

$$|Opt| \geq \frac{n}{\left| \underbrace{S_1}_{\text{biggest group}} \right|}, \text{ בהתחלה,}$$

$$n[j+1] \leq n[j] \left(1 - \frac{1}{|opt|} \right) \quad \text{מסקנה:}$$

נפעיל פעמיים את אי השוויון, ונקבל:

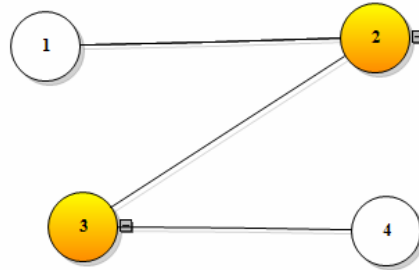
$$\begin{aligned} n[j+1] &\leq n[j] \left(1 - \frac{1}{|opt|} \right) \leq n[j-1] \left(1 - \frac{1}{|opt|} \right) \left(1 - \frac{1}{|opt|} \right) \leq \dots \\ &\leq n \left(1 - \frac{1}{|opt|} \right)^j < ne^{-\frac{j}{|opt|}} \end{aligned}$$

קיבלנו ש $n[j+1] < ne^{-\frac{j}{|opt|}}$. אנו רוצים להוכיח ש $|C| \leq \ln n \cdot |opt|$ (איכות הפתרון שלנו).
 כלומר: אחרי $\ln n |opt|$ איטרציות, X התרוקן, ובכל איטרציה מגדילים את $|C|$ באחד. נציב
 $j = \ln n |opt|$ ב $n[j+1] < ne^{-\frac{j}{|opt|}}$ נקבל את הנדרש - $n[j+1] = 1$.

11.1.2007

בעיית כיסוי קודקודים לצלעות גרף (Vertex Cover)

נתון גרף (לא מכוון) $G = (V, E)$, $|V| = n$. אנו נשמה למצוא קבוצה בגודל מינימלי של קודקודים $C \subseteq V$, כך שלכל צלע $e = (u, v) \in E$ מתקיים $\{u, v\} \cap C \neq \emptyset$.



לדוגמא - הקבוצה C היא $2,3$.

כאשר בוחרים באופן איטרטיבי קודקודים בעלי דרגה גבוהה ביותר נקבל קירוב עד כדי $O(\log n)$. ניתן לבנות דוגמאות של גרפים שבהן האלגוריתם נותן יחס כזה.

אלגוריתם קירוב עד יחס קירוב 2

רעיון - במקום לבחור קודקודים, נבחר צלעות.

מתחילים עם $C = \emptyset$. בכל צעד בוחרים צלע $e = (u, v)$ בקבוצה E . מוסיפים את u ואת v ל- C ,

כלומר $C \leftarrow C \cup \{u, v\}$ ואז מורידים מ- E את כל הצלעות שכבר מכוסות ע"י C .

ממשיכים עד ש $E = \emptyset$.

טענה: האלגוריתם נותן יחס קירוב של 2 .

הוכחה: נשים לב שלאוסף הצלעות שבחרנו במהלך האלגוריתם אין קודקוד משותף. ולכן הקבוצה C היא

אוסף צלעות זרות. יהי C^* אוסף כיסוי קודקודים אופטימלי כלשהו, אזי ב- C^* חייב להיות לפחות נציג

אחד לפחות מכל צלע שהכנסנו ב- C . ועלכן $|C| \leq 2|C^*|$. מספר הצלעות ב- C .

כלומר - $|C^*| \leq \text{מספר הצלעות שאספנו במהלך האלגוריתם} = \frac{|C|}{2} \leq 2|C^*|$.

כדאי להעיר, שכמה שהאלגוריתם הזה פשוט, האלגוריתם הטוב ביותר שידוע נותן יחס קירוב של $2 - \varepsilon$, כש ε שואף ל- 0 כש n גדל. ידוע שאם קיים אלגוריתם קירוב יעיל המבטיח יחס קירוב

$\frac{7}{6} = 1 + \frac{1}{6} \geq$ אזי קיים אלגוריתם יעיל המוצא את האופטימום. ההשערה שיחס טוב מ- 2 יגרור תוצאה

דומה.

בעית הסוכן הנוסע

נתון אוסף של $v_1, \dots, v_n$ ומחירים עבור הנסיעה בין כל שתי ערים - לכל $i \neq j$, $w_{ii} = 0$,

ו $w(v_i, v_j) = w_{ij} \geq 0$. המחיר סימטרי - $w_{ij} = w_{ji}$.

מסלול עבור הסוכן הוא פרמוטציה של $\{1, \dots, n\}$ - $\pi(1), \dots, \pi(n)$, סדר הערים בהן הסוכן יבקר

מ $\pi(n)$ הוא יחזור ל $\pi(1)$. מחיר המסלול π הוא

$$w_{\pi(1)\pi(2)} + w_{\pi(2)\pi(3)} + \dots + w_{\pi(n-1)\pi(n)} + w_{\pi(n)\pi(1)}$$

אנו נניח הנחה נוספת – אי שיוויון המשולש – לכל i, j, k - $w_{ij} + w_{jk} \geq w_{ik}$, וניתן אלגוריתם קירוב בפקטור 2.

אלגוריתם קירוב

בהנתן הגרף על n קודקודים, עם משקל w_{ij} לצלע בין i, j , נמצא עץ פורש מינימלי. נבנה עץ פורש מינימלי (ממושקל). נבחר שורש לעץ, ונבצע הילוך 'אצבעות' על פני קודקודי העץ – כאשר עוברים על כל הקודקודים (אפילו יותר מפעם אחת), ובדרך עוברים פעמיים על כל צלע.

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

(יש כאן שרטוט – אצל דינה)

מהמסלול עם חזרות על קודקודים, נוריד את הקודקודים המופיעים יותר מפעם אחת – חוץ מחזרה לשורש. מאי שיוויון המשולש, לא יתכן שהעלנו את המחיר. כך נקבל מסלול π העובר דרך כל קודקוד פעם אחת וחוזר לשורש. כאמור, מאי שיוויון המשולש, $w(\pi) \leq 2w(T)$.

יהי H מסלול העובר דרך כל קודקוד פעם אחת וחוזר לשורש, אזי $w(H) \geq w(T)$, כי כאשר מורידים H מ H צלע כלשהי נקבל עץ פורש שמשקלו קטן ממשקל $w(H)$. ועל כן

$$w(\pi) \leq 2w(T) \leq 2w(H)$$

עבור בעיית הסוכן הנוסע בין נקודות במישור (מחיר = מרחק) קיים אלגוריתם קירוב $1 + \varepsilon$, לכל $\varepsilon > 0$ - אבל ככל ש ε שואף ל 0 , דרגת הפולינום עולה מהותית. אין לנו יכולת אמיתית להריץ אותו על

$$\varepsilon = \frac{1}{10} \text{ ו } 10 \text{ נקודות.}$$

בעיית השמה המספקת מספר מקסימלי של משוואות לינאריות מעל השדה בן שני איברים

ניקח את $\mathbb{F}_2$. יש משתנים $x_1, \dots, x_n$, ו m משוואות, לדוגמא –

$$m \text{ equations } \begin{cases} x_1 + x_7 + x_{15} = 0 \\ x_1 + x_3 + x_{19} = 1 \\ \dots \end{cases}$$

הנחה: כל המשוואות שונות זו מזו. אנו רוצים למצוא השמה $x_1, \dots, x_n \in \{0, 1\}$ כך שיתקיימו

מקסימלי של משוואות. אם אפשר לקיים את כולם, זו לא בעיה – גאוס ב $O(n^3)$. אבל מה אם לא?

פתרון – אם בוחרים השמה מקרית – הטלת מטבע ב"ת לכל משתנה, אזי ההסתברות שמשוואה כלשהי

תתקיים היא בדיוק $\frac{1}{2}$. נגדיר משתנה מקרי $X_l = \begin{cases} 1, l' \text{th equation is satisfied} \\ 0, \text{otherwise} \end{cases}$. התוחלת

$$E(X_l) = \frac{1}{2} x_l \text{ של } y = X_1 + \dots + X_m \text{ מקרי משתנה} \quad E(y) = \sum E(X) = \frac{m}{2}$$

כל שתי משוואות שונות הן בלתי תלויות לינארית.

18.1.2007

אלגוריתמים מקבילים

לרשותנו זכרון בדומה למחשב עם מעבד יחיד. לרשותנו הרבה מעבדים. אבל לרשותנו הרבה מעבדים $P_1, \dots, P_n$.

במחזור פעולה של המכונה המקבילית:

- כל מעבד יכול לקרוא תא מהזכרון המשותף
- כל מעבד מבצע צעד חישוב
- כל מעבד כותב לתוך הזכרון המשותף

לדוגמא: חיבור של n מספרים בעזרת $\frac{n}{2}$ מעבדים (ב.ה.כ. $n = 2^k, k \in \mathbb{N}$)

נתונים n מספרים $x_1, \dots, x_n$, כאשר x_k נמצא בתא k בזכרון.

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

ומקבל תוצאה y_k . בשלב הכתיבה הוא כותב את y_k בתא k בזכרון, $k = 1, \dots, \frac{n}{2}$.

בזמן $O(1)$ ירדנו מבעיה בגודל n לבעיה בגודל $\frac{n}{2}$. לאחר $\log n$ פעולות נקבל את הסכום בתא 1.

סה"כ מספר הפעולות – בזמן t יש $\frac{n}{2^t}$ פעולות, וסה"כ $n - 1$ פעולות.

למה: אם קיים אלגוריתם מקבילי לביצוע משימה בזמן t , ומספר לא מוגבל של מעבדים, וסה"כ מספר הפעולות הוא W אזי ניתן לבצע את החישוב בעזרת k מעבדים במגבלת זמן של $\frac{W-t}{k} + t$.

הוכחה: בשלב l אנו מבצעים במקביל w_l פעולות, $w = \sum_{i=1}^l w_i$. ניתן להתחיל לבצע את w_{l+1} אחרי

$$\sum_{i=1}^l \left\lceil \frac{w_i}{k} \right\rceil \leq \sum_{i=1}^l \left(\frac{w_i - 1}{k} + 1 \right) \leq \frac{w - l}{k} + l.$$

למודל החישוב המקבילי שהצגנו קוראים PRAM (Parallel Random Access Machine).

מיון במקביל בעזרת רשת מיון

אנו מקבלים כקלט n מספרים $x_1, \dots, x_n$. (שרטוט – אצל דינה).

בכל צעד ניתן לבצע עד $\frac{n}{2}$ השוואות. הראנו כי קיים חסם תחתון של $\Omega(n \log n)$ (בDAST), ולכן

חסם תחתון מייד למספר השלבים למיון בעזרת רשת מיון הוא $O(\log n)$. קיימות רשתות כאלה –

בטובות שבהן זה קבוע של 5000 בערך. זה לא קל להגיע ל- $O(\log n)$ – אנחנו נגיע ל- $O((\log n)^2)$

רשת מיון בעומק $O(\log^2 n)$ - Odd-Even Merge

למה: בהנתן תאור של רשת מיון הרשת פועלת נכון וממיינת כל פרמוטציה אם היא ממיינת נכון את כל הקלטים $x_1, \dots, x_n$ כאשר $x_i \in \{0, 1\}$ לכל i .

הוכחה: נשים לב שאם רשת מיון ממיינת נכון קלט $a_1, \dots, a_n$ אזי אם $f: \mathbb{R} \rightarrow \mathbb{R}$ פונקציה

מונוטונית עולה, אז $x \leq y \Rightarrow f(x) \leq f(y)$.

אזי הרשת פועלת באופן זהה על הסדרה $f(a_1), \dots, f(a_n)$. (שכן רשת מיון עובדת עם פעולות $\max$, והן עובדות).

כעת נוכיח בדרך השלילה – נניח שעבור קלט $a_1, \dots, a_n$ הפלט a_i עובר למקום נמוך יותר מ a_j למרות

ש $a_j < a_i$. נבחר פונקציה f , $f(x) = \begin{cases} 0, & x \leq a_j \\ 1, & \text{otherwise} \end{cases}$. מתקיים $f(a_i) = 1, f(a_j) = 0$.

אזי עבור הקלט $f(a_1), \dots, f(a_n) \in \{0, 1\}^n$ נקבל שהרשת אינה ממיינת נכון בסתירה להנחה.

למיון בעזרת מיזוגים יש $\log n$ שלבים, ואם נראה כיצד ניתן למזג רשימות ממוינות בעזרת רשת מיון יעילה, נקבל בסה"כ רשת מיון טובה.

מיזוג במקביל בעזרת רשת מיון

נתונים שתי רשימות $a_0, \dots, a_{M-1}$ ו $b_0, \dots, b_{M-1}$ שתיהן ממוינות בסדר עולה. נפריד את הרשימות למקומות הזוגיים והבלתי זוגיים.

$$\begin{array}{cc} \overbrace{(a_0, a_2, \dots, a_{M-2})}^{a \text{ even}} & \overbrace{(a_1, a_3, a_5, \dots, a_{M-1})}^{a \text{ odd}} \\ \underbrace{(b_1, b_3, \dots, b_{M-1})}_{b \text{ odd}} & \underbrace{(b_0, b_2, b_4, \dots, b_{M-2})}_{b \text{ even}} \end{array}$$

נאחד את $a \text{ even}$ ו $b \text{ odd}$ ל $c_0, \dots, c_{M-1}$, ואת $a \text{ odd}$ ו $b \text{ even}$ ל $d_0, \dots, d_{M-1}$.

כעת נתבונן ב $\underbrace{c_0, d_0}_{}, \dots, \underbrace{c_k, d_k}_{}, \dots, \underbrace{c_{M-1}, d_{M-1}}_{}$, ונבצע במקביל השוואה בין זוגות (c_k, d_k) וזהו – סיימנו.

מספר השלבים למיון רשימות באורך M הוא $\log M$.

$$\text{even}(A) = 2, 4, \text{Odd}(A) = 3, 8$$

$$\text{odd}(B) = 5, 7, \text{even}(B) = 1, 6$$

$$\left. \begin{array}{l} C = 2, 4, 5, 7 \\ D = 1, 3, 6, 8 \end{array} \right\} \xrightarrow{\text{via sorting network}} (1, 2, 3, 4, 5, 6, 7, 8)$$

$$\begin{array}{l} A = 2, 3, 4, 8 \\ B = 1, 5, 6, 7 \end{array} \quad \text{דוגמא:}$$

מהוכחת הלמה, די להוכיח את הטענה עבור רשימות ממוינות A, B המכילות רק אברים מ $\{0,1\}$.

אם מספר האפסים ב A הוא a

אם מספר האפסים ב B הוא b .

מספר האפסים ב $even(A)$ הוא $\left\lfloor \frac{a}{2} \right\rfloor$, וב $odd(A)$ $\left\lfloor \frac{a}{2} \right\rfloor$. באופן דומה עבור B . על כן, מספר

האפסים ברשימה C יהיה $\left\lfloor \frac{a}{2} \right\rfloor + \left\lfloor \frac{b}{2} \right\rfloor$.

מספר האפסים ברשימה D יהיה $\left\lfloor \frac{a}{2} \right\rfloor + \left\lfloor \frac{b}{2} \right\rfloor$.

נשים לב כי ההפרש בין מספר האפסים ב C וב D הוא לכל היותר 1. (c הוא מספר האפסים ב C),

אם $c = d$ - אזי רשת המיון לא צריכה לעשות כלום.

אם $c = d + 1$, אזי גם רשת המיון לא צריכה לעשות כלום (זה כבר מסודר).

במקרה $c = d - 1$, אזי תהיה השוואה אחת בלבד ברשת המיון, שתתקן -

$\underbrace{00\dots0}_{2c} | \underbrace{10}_{\text{fix here!}} | 11\dots1$

ולכן המיזוג עובד לכל שתי סדרות ממוינות.

מסקנה: ניתן לבנות רשת מיון למיון n מספרים ($n = 2^k, k \in \mathbb{N}$), בעומק $O(\log^2 n)$.

הוכחה: מיזוג רשימות באורך M דרש עומק $\log M$. ולכן כאשר 'נריץ' או נבנה רשת מיון כאשר

עוקבים אחרי האלגוריתם למיון בעזרת מיזוגים, נקבל

$$T(n) = T\left(\frac{n}{2}\right) + \log\left(\frac{n}{2}\right) = \log \frac{n}{2} + \log \frac{n}{4} + \dots \log 2 = O\left(\frac{\log n \cdot \log(n-1)}{2}\right) \leq$$

$$O(\log^2 n)$$

($\log$ הוא בגלל עומק רשת המיון)

22.1.2007

הודעה:

– בחינה

אלגוריתמים מקביליים במודל PRAM

חישוב FFT על n איברים בעזרת n מעבדים ($n = 2^k$) האלגוריתם הסדרתי שקיבלנו ניתן למקבל בצורה ברורה, שכן פירקנו בעיה מסדר n לשתי בעיות בלתי תלויות מסדר $\frac{n}{2}$ וכל אחת מ- n התוצאות הסופיות התקבלה בעזרת מספר קבוע של פעולות.

$$\text{אם } T(n) \text{ הזמן המקבילי, } T(n) = T\left(\frac{n}{2}\right) + O(1).$$

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 $O(\log^2 n)$ בעזרת $O(n^4)$ מעבדים. (וכך ניתן לבדוק בעזרת האלגוריתם ההסתברותי שתארנו האם קיים זיווג מלא בגרף דו צדדי).

בעיה

נניח ויש לנו ביטוי כדוגמת:

$$((a \times b + c) \times d + e) \times f + g$$

אפשר לבנות מזה עץ¹.
נרצה להראות שאפשר כל ביטוי כזה לחשב בזמן לוגרימתי. אפילו אם העץ לא מאוזן בעליל (ובעומק של $O(n)$).

נתונים משתנים $x_1, \dots, x_n$. ביטוי אלגברי מעל שדה $\mathbb{F}$ במשתנים הללו הוא

1. איבר מהשדה

2. x_i כלשהו

3. אם φ, ψ ביטויים אלגבריים אז גם $(\varphi \times \psi)$ הוא ביטוי, וגם $(\varphi + \psi)$ הוא ביטוי.

(כאן גודל הביטויים המתקבלים הוא סכום גודלי הביטויים (φ, ψ)).

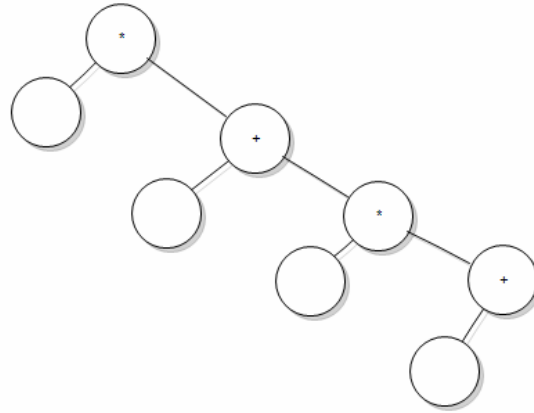
נשים לב שביטוי בגודל n ניתן לחשב באופן סדרתי בעזרת חישוב של $n - 1$ פעולות חיבור / כפל.

לכל ביטוי ניתן להתאים באופן טבעי עץ בינארי.

¹ מי שמתקשה – <http://notes-heaven.intertent.net> זה המקום בשבילו.

בהינתן ביטוי שמתאים לו עץ בגובה h אזי ברור שניתן לחשב אותו בזמן h (בעזרת לכל היותר n מעבדים).

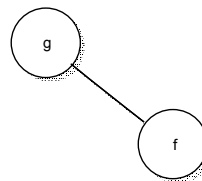
חישוב מקבילי של ביטויים בצורת מסרק



לעיתים אפשר לאזן, אבל לא תמיד.

נתבונן בפעולה לדוגמא $y = f(x) = x + 7$. (יש כאן כמה שרטוטים, אצל דינה).

נתבונן בכל קודקוד כפעולה לינארית על קלט אחד עם מקדמים ידועים – באופן כללי, אם יש לנו עץ בצורת מסרק, לדוגמא



כאשר $y = f(x) = ax + b$ (נכנס ל' f , x יוצא y , ו y נכנס ל' g).
 $z = g(y) = cy + d$

הרכבה של שתי פונקציות לינאריות נותנת פונקציה לינארית וחישוב מקדמי ההרכבה דורש 3 פעולות יחידות זמן. לכן, נחשוב על המסרק הזה כאוסף של פונקציות לינאריות.

ואז, בעזרת הרכבת זוגות עוקבים מהשורש ועד לפעולה האחרונה ניתן ב $O(1)$ לעבור למסרק חדש

באורך $\left\lceil \frac{n}{2} \right\rceil$. לאחר $O(\log n)$ זמן נקבל פו' לינארית אחת, וקלט לפונקציה.

הערה: ניתן להרחיב את האלגוריתם גם לחילוק, ואז הפונקציות כולן יהיו מהצורה $f(x) = \frac{ax+b}{cx+d}$

הערה 2: אפשר לבצע זאת גם עם פעולות לוגיות (and, or, ...).

מה נעשה כאשר לא מדובר בביטוי בצורת מסרק?

נחשב ביטוי כללי באופן הבא:

1. גיזום עלים (Rake) - עלה תמיד יהיה איבר בשדה, אז אפשר לחבר אותו לצומת שמעליו, ולהפוך אותו לפי' לינארית. אם לקודקוד יש שני בנים שהם עלים ניתן למחוק את העלים, והקודקוד עצמו הופך לעלה.
אם לקודקוד יש בן אחד שהוא עלה ובן שני שהוא שורש של תת עץ כלשהו, ניתן לגזום את העלה והקודקוד הופך לפונקציה לינארית, ויש לו רק בן אחד.
2. כיווץ מסלולים (Compress) - הרכבת שתי פונקציות לינאריות עוקבות לפונקציה אחת.

נגדיר דרך לייצר זוגות באופן הבא - יורדים לאורך המסלולים מהשורש לעלים. בוחרים זוגות בצורה חמדנית בהתאם למי שנתקלים בו קודם.
בצעד הכיווץ עצמו הופכים כל זוג קודקודים לקודקוד בודד, בעזרת חישוב הרכבת הפונקציות הלינאריות.

לאחר צעד הגיזום הראשון יש לנו עצים בינאריים עם קודקודים פנימיים עם בן אחד ושני בנים ועלים שהם קודקודים ללא בנים.

יהי T עץ בינארי בן n קודקודים שיש בו l עלים ו- p קודקודים פנימיים עם שני בנים.

טענה: $p + 1 = l$

הוכחה: באינדוקציה על גובה העץ. בעץ בגובה 0 יש קודקוד 1 שהוא השורש והוא גם עלה $p = 0$, $l = 1$, והתנאי מתקיים.

צעד האינדוקציה: נתבונן בשורש. אם יש לו בן בודד - אז לא שינינו את מספר העלים ולא את מספר הקודקודים הפנימיים, והדבר מתקיים ע"פ הנחת האינדוקציה.

אם יש לו שני בנים - אזי אם לבן הימני יש p_1 קודקודים פנימיים, ו- l_1 עלים, ולשמאלי p_2 ו- l_2 בהתאמה, אז

$$l = l_1 + l_2 = p_1 + 1 + p_2 + 1 = \overbrace{p_1 + p_2 + 1}^p + 1$$

$$p = p_1 + p_2 + 1$$

מ.ש.ל.

כעת, נניח שבעץ שלנו יש n קודקודים, l עלים, p קודקודים פנימיים עם שני בנים, וצורך צעד הכיווץ יצרנו r זוגות. יהי h מס' הקודקודים עם בן יחיד (שלא נמצא להם בן זוג) ולכן מתאים לו באופן יחיד קודקוד שהוא עלה או קודקוד עם שני בנים.
מסקנה $h \leq l + p$.

מספר הקודקודים שאנו מצליחים להוריד מן העץ בפעולה של Rake + Compress - $l + r$ - כי מורידים l עלים ומ- $2r$ קודקודים נקבל רק r קודקודים בעץ החדש.

$$n = l + p + 2r + h \leq 2(l + p) + 2r < 4l + 2r \Rightarrow l + r > \frac{n}{4}$$

ולכן לאחר לכל היותר $O(\log n)$ חזרות נוכל לחשב את ערכו של הביטוי.

מחיקת כל קודקוד דורשת לכל היותר 3 פעולות, ולכן בסה"כ ברור שמספר הפעולות $3n \geq$. למעשה

ניתן להראות חסם של $\frac{3}{2}n$. ולכן בעזרת $\frac{n}{\log n}$ מעבדים ניתן לחשב את הביטוי בזמן $O(\log n)$.

25.1.2007

אלגוריתמים מקוונים (On-Line)

בעיית הסקן

בבעיה זו קל לתת פתרון מקוון הדורש לכל היותר מחיר כפול מן הפתרון האופטימלי של מישור שיודע את העתיד – שכירה, עד שעלות השכירה שווה לעלות קניה. ואז קונים.

Caching – ניהול זכרון מטמון

http://en.wikipedia.org/wiki/CPU_cache

לתוכנית יש סדרה של בקשות לקריאה (/כתיבה) מהזכרון ויש לנו זיכרון מהיר (מטמון) בגודל k . אנו רוצים לנהל את הזכרון המהיר בצורה הטובה ביותר.

אחרי מילוי הזכרון המהיר כל פניה לתא זכרון שאיננו שם מחייבת את הבאת התא הזה לזכרון המהיר ופינוי של איבר כלשהו מהזכרון המהיר (Cache miss).

הערה: אם יודעים מראש את כל הסדרה אז פתרון אופטימלי הוא לזרוק מהזכרון המהיר את האיבר שזקוקים לו הכי רחוק בעתיד.

אם מניחים 'מה שהיה הוא שיהיה', אפשר להסתכל על העבר, להניח שהעתיד יהיה זהה, ולהעיף את הדברים האחרונים שהשתמשו בהם. (LRU – least recently used). הבעיה היא שאין שום דרך לנהל את זה בפועל... (בסיס נתונים מסובך וגדול מדי).

בפועל משתמשים בקירוב ל-LRU – **אלגוריתם סימון**

מסמנים כל איבר בזכרון המהיר אליו אנו פונים. אם כל האיברים מסומנים, מוחקים את כל הסימונים. **כלל ההחלפה:** לבחור את אחד מאלה שלא מסומנים, ולהחליף אותו.

טענה: המחיר לכל אלגוריתם סימון ולכל סדרת פניות לזכרון $k \geq$ המחיר האופטימלי.

הוכחה: על כל $k + 1$ פעמים שאני משלם פה, ראינו $k + 1$ איברים בסה"כ, ולכן האלגוריתם הטוב ביותר, שילם לפחות פעם אחת.

טענה: לכל אלגוריתם החלפה דטרמיניסטי קיימת סדרה שעבורה המחיר יהיה פי k מהמחיר האופטימלי.

הוכחה: נתבונן בסדרה של גישות ל $k + 1$ איברים. תמיד ניגש לאיבר שירד מהCache לפני כן. אנו נשלם עבור כל גישה שהיא, כאשר אלגוריתם אופטימלי ישלם על אחת ל $k + 1$.

כעת נתבונן באלגוריתם סימון הסתברותי – זורקים איבר מקרי לחלוטין. תוחלת היחס לסדרה הגרועה ביותר היא $\log k$.

הערה היסטורית – זה התברר ככישלון מוחלט. עבור תוכניות כלליות זה עבד מצוין. עבור אלגברה לינארית, המתכנתים שתכנתו ע"פ התנהגות הCache, לא יכלו יותר לעשות אופטימיזציות.