שאלה 1

(1) בהינתן פוליגון פשוט עם n קודקודים, ניתן לקבוע בזמן $O(n)$ אם הפוליגון קמור או לא.

<u>נכון!</u> פוליגון פשוט הוא קמור אמ"מ כל הזוויות שלו קטנות מ- 180°, או לחילופין אם ~~...~~ בטיול על הצלעות שלו נגד כיוון השעון כל הפניות הן שמאלה. בדיקת הזווית או כיוון הפניה מתבצעת בזמן קבוע לכן נתחיל ללכת על הפוליגון. אם ניתקל בזווית מעל 180° סימן שהפוליגון קעור. אם כל הזוויות קטנות מ- 180° אז הוא קמור. זה כמובן נעשה בלכל היותר n צעדים - $O(n)$.

~~(3) התחום~~

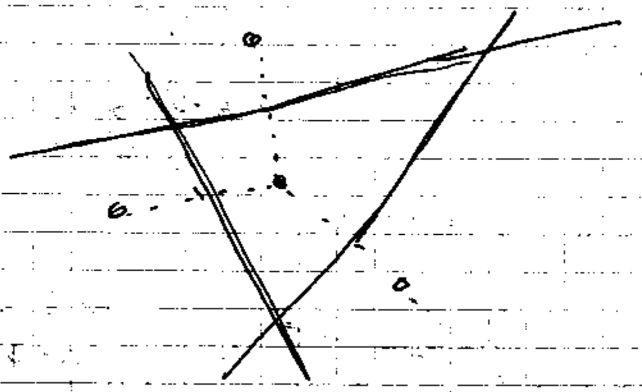
(5) שאילתות טווח מלבניות במישור תמיד נענות מהר יותר בעזרת range-trees מאשר בעזרת kd-trees.

<u>לא נכון!</u> סיבוכיות השאילתא של range tree היא $O(\log^2 n + k)$ ואילו של kd-tree היא $O(\sqrt{n} + k)$.
אפשר לבנות דוגמאות שבהן האחד יעיל מהשני ולהפך.

~~(7) יהי P קבוצה בת n נקודות ויהי T(P)~~
~~שילוש שלה. הטענה אם T(P) היינו שילוש~~
~~Delaunay הפרש יותר מ- O(n).~~
~~לא נכון! למדנו שצלע היא בש~~

(6) אם התאים של דיאגרמת Voronoi של קבוצה P במישור כולם אינם חסומים אז הנקודות של P נמצאות על פוליגון קמור.

נכון! נניח בשלילה שהן לא על פוליגון קמור. אז יש נקודה אחת לפחות שנמצאת בקמור של האחרות. אבל אז התא של הנקודה הזאת בהכרח חסום:
אם $n=3$ זה בכלל מלחי אפשרי. אז נניח ש- $n\geq 4$.
אז יש נקודה אחת שהיא בקמור של 3 אחרות:

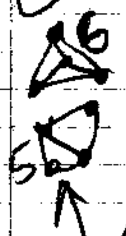


אבל אז כבר בהכרח התא חסום רק מהתבוננות במה שקורה עם 3 נקודות אלה.

(4) אם P,Q פוליגונים עם n נקודות כל אחד רק שבקמור ~~של~~ $CH(P)$ יש $O(n)$ נקודות ומקמור $CH(Q)$ יש $O(1)$ נקודות אז בשילוש של P גשר יש פחות צלעות מאשר בשילוש של Q.

נכון! ראינו שבקבוצת n נקודות עם k נקודות בקמור, בכל שילוש יש $3n-3-k$ צלעות אז אם $K_p \gg K_q$ אז

$$3n-3-k_p < 3n-3-k_q$$



נעיר שזה עובד אם אכן n גדול. מספיק. ב- $n=3$ זה לא נכון כי יש שוויון אבל ב- $n=4$ זה כבר נכון.

(8) תהי P קבוצת n נקודות לא בהכרח במצב כללי. ~~נמצא~~ מציאת הישר l שעובר דרך מספר מקסימלי של נקודות דורש זמן $O(n^2 \log n)$

לא נכון! נסתכל במישור הדואלי. שם מציאת הישר המקסימלי מקבילה למציאת נקודת חיתוך שהיא נקודת החיתוך של המספר המקסימלי של ישרים.

~~ראינו שבזמן line segment intersection~~ לפי מה שלמדנו בתרגול בזמן $O(n\log n)$ אפשר למצוא תיבה חוסמת לכל נקודות החיתוך (מתרגול 6). לכן אפשר עכשיו להשתמש באלגוריתם של line segment intersection שמוצא את נקודות החיתוך בזמן $O(n\log n + k)$. אלא שנבצע קצת לשנות ובמקום רק לדווח את הנקודות גם לספור את כמות הישרים שיצרו את נקודת החיתוך. ברור שהשינוי הזה לא משנה את זמן הריצה. כדי למצוא את הנקודה הרצויה נעבור על כל נקודות החיתוך ונמצא את המקסימלית. סה"כ:

סופר נקודות $k=O(n^2)$

$$\underbrace{O(n)}_{\text{מעבר לדואלי}} + O(n\log n + k) + \underbrace{O(n^2)}_{\text{מציאת המקסימום}} = O(n^2)$$

$O(f(n))$

שאלה 2

יהיו $P_1, \dots, P_k$ פוליגונים קמורים זרים, כל אחד עם k_i קודקודים כך שסך כל הקודקודים $\sum k_i = n$
כל פוליגון מתואר ע"י הקודקודים שלו נגד כיוון השעון.
תהי P_0 תיבה תוחמת של כל הקודקודים.
(נאמר שהפוליגונים מקוננים אם פרט ל-P_0 כל פוליגון מוקף ע"י פוליגון אחר.
בפרט, הפוליגונים הם מקוננים בסדר אם $P_{i+1} \subset P_i$.
הפאה המוגדרת ע"י $P_i - P_{i+1}$ נקראת שכבה i.

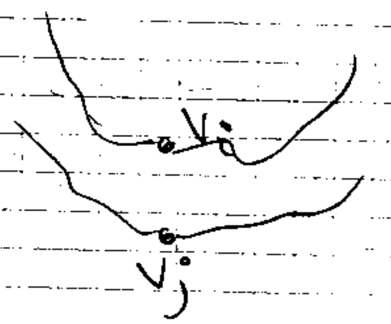
(1) נתאר אלגוריתם שרץ בזמן $O(n)$ שבודק אם $P_0, \dots, P_k$ מקוננים. לפי מה שנאמר צריך לבדוק שזה ולמצוא סדר הכינון שלהם.

אם נראה שאפשר לבדוק זאת עבור שני פוליגונים בגודל n,m בזמן $O(m+n)$, אז אם נבצע את הבדיקה לכל זוג פוליגונים עוקבים בעיגול נקבל זמן $O(\sum k_i + \sum k_i) = O(n)$.
אז נניח שיש לנו שני פוליגונים P, Q בגודל n, m בהתאמה.
נניח בה"כ שהנקודה הכי נמוכה של P היא מתחת לנקודה הכי נמוכה של Q (בדיקה בזמן $O(n+m)$). אז אם הם מקוננים בהכרח $Q \subseteq P$ וזה מה שצריך לבדוק. ~~אם הקודקודים~~ שני הפוליגונים קמורים. ומכאן שאפשר למצוא את החיתוך שלהם בזמן $O(n+m)$. עכשיו יש רק לבדוק שהחיתוך שווה ל-Q וזהו!

אז משיקולי רציפות
האמת היא שמאחר שהפוליגונים זרים בהכרח ~~אז~~ $P \subseteq Q$ או $Q \subseteq P$, אז ומספיקה רק ההשוואה בין הנקודות הקיצוניות של הקודקודים ... ~~[illegible]~~ $P \subseteq Q$ אמ"מ
$P_{y\,\text{תחתונה}} \ge Q_{y\,\text{תחתונה}}$, $P_{y\,\text{עליונה}} \le Q_{y\,\text{עליונה}}$, $P_{x\,\text{שמאלי קיצוני}} \ge Q_{x\,\text{שמאלי קיצוני}}$, $P_{x\,\text{ימני קיצוני}} \le Q_{x\,\text{ימני קיצוני}}$
וזוהי בדיקה בזמן $O(n+m)$

(2) נתאר אלגוריתם שרץ בזמן $O(n\log n)$
שמחשב את סדר הבינון

~~ראשית נבדוק בזמן $O(n)$ אם הם בכלל מכוונים~~
~~אם לא, אין מה להמשיך. אם הם מכוונים נמצא~~
בזמן $\sum O(k_i) = O(n)$ לכל פוליגון
את הקודקוד הכי נמוך שלו (אם יש שניים לא
משנה, מעניינת אותנו רק הקואורדינטה ה-y של
הקודקוד הכי נמוך). כשהם מכוונים אז הסדר של קואורדינטות
ה-y של קודקודים אלה מתאים לסדר הבינון
מציאת הסדר של קואורדינטות ה-y לוקחת
$O(n\log n)$ ~~וסיום~~.
ברור שהסדר של הקודקודים מתאים לסדר הבינון כי
הנחנו שהם כבר מכוונים. אז אם למשל $v_i < v_j$



ברור שלא יכול להיות ש- $P_j \subseteq P_i$.

לכן, סה"כ כל הבדיקה לקחה

$$\underbrace{O(n)}_{\text{בדיקת הבינון}} + \underbrace{O(n)}_{\text{מציאת קודקודים תחתונים}} + \underbrace{O(n\log n)}_{\text{מיון קודקודים}} = O(n\log n)$$

לבסוף, אם הם מכוונים
אז בהכרח זה הסדר שמצאנו.
תרשים לפי סעיף (ו) נבדוק שהסדר
שמצאנו הוא אכן סדר הכיוון ואז סיימנו.

3) נניח שהפוליגונים $P_0 \dots P_k$ נתונים מסודר:
נתאר מבנה נתונים בגודל $O(n)$ שנבנה בזמן
$O(n\log n)$ (או פחות) כך שבזמן שאילתא
$O(\log^2 n)$ (או פחות) אפשר לענות על שאילתות
מהצורה: בהינתן $q \in P_0$ לקבוע את השכבה i
שהיא שייכת אליה.
אם q על השפה של P_i אז q שייכת לשכבה i.

אלגוריתם נאיבי: למדנו שפוליגון קמור אפשר בזמן $\log n$ לבדוק אם נקודה היא בפנים או לא.
אז נתחיל מ-P_k והפוליגון האחרון שהנקודה עוד בתוכו היא השכבה המבוקשת.

אפשר לשפר: למעשה יש לנו יחס סדר על הפוליגונים (יחס ההכלה) ויש לנו יחס שקובע
אם נקודה היא בפוליגון P_i אז גם ב-P_{i+1} (*)
בזמן $O(\log n)$. (מחלקים את הפוליגון לפלחים, בודקים באיזה פלח הנקודה ואז אם היא בפנים או בחוץ)

אז אנחנו צריכים לעשות חיפוש בינארי על הפוליגונים שמוצא את i כך ש-q ב-p_i ולא ב-p_{i+1}.
מאחר וזה היא ניתנת

אז למעשה אין ממש מבנה נתונים - פשוט נשמור את הפוליגונים. שזה $O(n)$.
ואז נעשה חיפוש בינארי כאשר בודקים את התנאי (*). בחיפוש הבינארי יש לכל היותר $\log n$ שלבים (למעשה $\log k$) ובכל בדיקה משתמשים ב- ~~$O(\log k)$~~ $O(\log n)$ (למעשה $O(\log k_i)$) ולכן סה"כ נקבל בזמן $O(\log^2 n)$.

שאלה 3

תהי $P=\{p_1 ... p_n\}$ קבוצה n נקודות קבועות
במישור במקום כללי. R רובוט שצורתו דיסק ברדיוס r.
תהי q נקודה ונסמן ב- $R(q)$ את השטח שתופס
הרובוט כשמרכזו נמצא ב- q.
q ~~חופשית~~ חופשית מהתנגשויות אם אין נקודות מ- P בפנים של
$R(q)$ (אפשר על השפה)
המשימה היא לתכנן מסלול T מנקודת התחלה s לנקודת
סיום g כך שכל הנקודות בדרך חסרות התנגשויות.
מסלול הוא שרשרת סגמנטים שמוגדרת ע"י סדרת נקודות.
אורך המסלול T הוא סכום אורכי הסגמנטים (מרחק אוקלידי).

(1) נתאר מבנה נתונים שמאפשר לענות באופן יעיל על
שאילתות מהצורה: בהינתן q, האם היא חסרת התנגשויות.

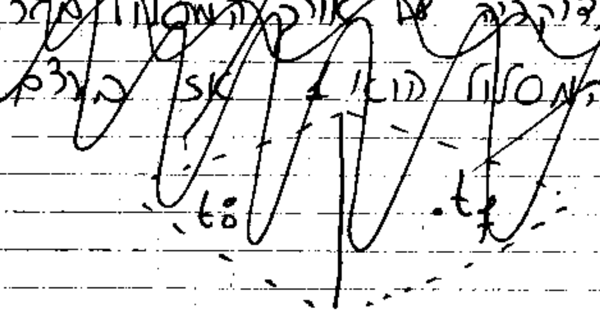
~~נשנה את הנקודות של~~ אפשר להסתכל על זה ככה:
יש לנו נקודות ו- q מסמלת שאילתת טווח מעגלית.
ראינו בתרגול שאפשר להשתמש ב- kd-tree
עבור שאילתת טווח מעגלית. האלגוריתם הוא
אותו אלגוריתם אלא שבמקום
שבו בודקים אילו תאים נחתכים עם התיבה צריך לשנות
לבדיקה של חיתוך עם מעגל. זה קצת מסבך יותר בפועל
הנוסחאות אבל זה לא מגדיל את הסיבוכיות של
זמן השאילתא.
סיבוכיות המקום של kd-tree היא $O(n)$
וזמן הבנייה הוא $O(n \log n)$.
גם זמן השאילתא של מעגל אינו שונה מזה של מלבן
ולכן זמן השאילתא הוא $O(\sqrt{n} + k)$.
ולמעשה לא צריך את החלק של ה- k כי אנחנו לא צריכים

לבדוק את הנקודות שבמעגל אלא רק לענות על שאלה
מינימלית. אם במקרה כלשהו מצאנו שיש אזור
שנמצא כולו בתוך המעגל סימן שהנקודה אינה
חופשית מהתנגשויות.
נעיר רק שבהתאם לתנאי השאלה - הבדיקה אם
נקודה בתוך המעגל או לא צריך לא לכלול את
המקרה שהנקודה על השפה.

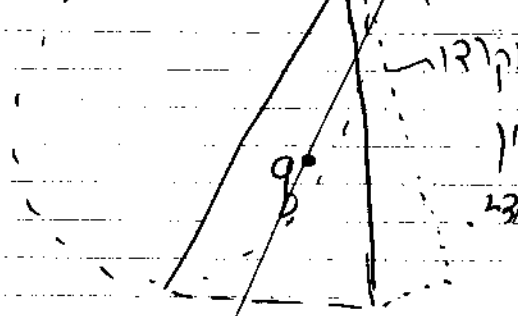
(2) נניח ש- t_0 ו- t_f חסרות התנגשויות. נתאר אלגוריתם
שרץ בזמן $O(n\log n)$ ומבנה נתונים בגודל
$O(n)$ שמחשב מסלול בין t_0 ל- t_f (אם יש כזה).

~~ראשית נשלש את הנקודות. יש $O(n)$ צלעות
בשילוש. נעבור על כל הצלעות ו"נמחק" את אלו
שבאורך קטן מ- $2r$ - שם הרובוט בטוח לא יוכל
לעבור.~~ ~~הנקודות שיהיו אמצעי המשולשים.~~
~~עכשיו ניצור את הגרף הדואלי אלא שלא נחבר
בין משולשים ששכנים בצלע שנמחקה (כלומר
הצלע קצרה מ- $2r$).~~
~~בגרף הדואלי הזה נחפש מסלול בין המשולש
של t_0 למשולש של t_f. המסלול הזה (אם
קיים) עובר רק דרך צלעות של משולשים שאורכן
גדול מ- $2r$ ולכן הרובוט יוכל לעבור שם.~~

~~הערה: באינדוקציה על אורך המסלול בגרף
הדואלי: אם אורך המסלול הוא 1 אז ברור~~



האינטואיציה היא שהרובוט עובר בין משולשים שמוגדרים
ע"י נקודות P וכדי לעבור דרך משולש בהצלחה
הוא צריך להיכנס דרך צלע אחת ומבלי להיתקל בכלום
ולצאת דרך צלע אחרת. אם למשולש אין שתי צלעות
גדולות מ- $2r$ אז המשולש
הזה הוא dead-end - אפשר להיכנס אליו אבל לא לצאת
את האלגוריתם למציאת מסלול בגרף בכלל לא יעבור דרך
המשולש הזה. ואם יש שתי צלעות באורך גדול מ-$2r$
אז בטוח מעגל ברדיוס r סביב
מרכז המשולש לא נוגע בנקודות
המשולש. בעצם זה לא נכון



כי יכול להיות שהמשולש צר קטן מדי.
אז השיטה תגביל את המסלול
רציף ונראה לי שאחרי שמוצאים מסלול רציף אפשר לבחור בו נקודות
שישמשו כעוגנים.

③ המסלול לא בהכרח הקצר ביותר. האלגוריתם למציאת
מסלול בגרף לא מוצא אותו לפי שום קריטריון, אלא
רק אחד מהמסלולים ואין מניעה שייבחר דווקא
מסלול ארוך...

האינטואיציה: לעבור כמה שיותר רחוק מכל המכשולים.
~~רעיון אחר שאין לי זמן לפתח ואולי הוא טוב יותר:~~
נסתכל על דיאגרמת voronoi של הנקודות. ברור
שאם יש מסלול אז יש מסלול שעובר על קודקודים וצלעות הדיאגרמה
כי הם מרוחקים במידה שווה מכל הנקודות שמהוות מכשול.
אז מה שצריך לעשות זה למצוא מסלול מין
קודקוד ששייך לתא של t_0 לקודקוד ששייך לתא של

t_f. בניית הדיאגרמה לוקחת $O(n\log n)$
ומציאת קודקוד של התא t_0 זה בצורה הפשוטה
ביותר $O(n\log n)$ (עוברים על כל התאים
שהם קמורים ובודקים את t_0 בפנים) וכנ"ל
ל-t_f (נזכיר שגודל הדיאגרמה הוא $O(n)$).

כעת, ~~ומשיבים~~ נריץ באמצעות אלגוריתם כמו
BFS או DFS ונמצא מסלול בין התאים
ומאחר שנתון ש t_0 ו-t_f חופשיים מהתנגשויות
אז נצטרך למסלול מהתחלה את t_0 ומסוף את
t_f ונקבל מסלול כמו שצריך.

כן שצריך שהמסלול יהיה מרוחק r מכל המכשולים.
אז בריצה של BFS לא ניתן לו לצלול קודקודים
זוגיים שהצלע ביניהם מרוחקת פחות מ-r מהקודקוד
של התא שלה (כמו בעצם כאילו עברנו על כל
הצלעות מהתחלה ומחקנו את אלו שאי אפשר לעבור
דרכן כי

$$\frac{r > \lambda}{r > \lambda}$$

מאחר שגודל הדיאגרמה $O(n)$ אז גם הכי הרבה
הזה ייקח זמן לינארי.

אם יש מסלול כמו שצריך בטוח נמצא אחד כזה
ואם אין אז לא נמצא שום מסלול ונדווח
שלא קיים מסלול כנדרש