8.12.08

גאומטריה חישובית

הערה-

בשעור הקודם ראינו את אלגוריתם sweeping לבדיקת חיתוך בין n קטעים נתונים.

האלגוריתמים שאני נותנים לבעיה זו עובדו גם לבדיקת חיתוך של קבוצת עקומים אחרים (לאו דווקא ישרים) בתנאי ש

(1) העקומים הם x מונוטונים - כל ישר אנכי חותך עקום פעם אחת לכל היותר

(ואם לא, אז ניתן לפרק כל עקום לחלקים x-מונוטוניים, אך דבר זה ישנה את n)

(2) קיימת "קופסאות שחורות" לפעולות הבסיסיות

- לבדוק אם נק' נמצאת מעל / מתחת / על עקום
- למצוא נק' חיתוך של זוג עקומים

בעיה -

נתונים n קטעים זרים ; m נקודות באישור. יש למצוא לכל נקודה את הקטע שנמצא ישירות מעליה (או לדווח שאין כזה).

ניתן לעשות זאת בזמן $O((m+n)\log(m+n))$ ע"י שינוי של האלג' שלנו לחיתוך כך שיוסיף את הנקודות לרשימת ה-x ונבצע חיפוש - לכל נקודה ברשימת ה-y (הנוכחית) למצוא מה מעליה.

למעשה, אם נדייק אז הסיבוכיות היא $O((m+n+k)\log(m+n))$ כאשר k הוא מספר נקודות החיתוך. ליתר דיוק, יש להתחשב ב k גם בתוך ה log, אך מכיוון שהוא חסום ע"י n^2 (וכתוך log זה זהה ל n) אין צורך לציין זאת.

נרצה להבין גם את סיבוכיות המקום...

עבור בדיקת החיתוך בין הקטעים אנו מחזיקים מבני x, y

שגודל רשימת ה-y הוא $O(n)$

שגודל רשימת ה-X הוא $O(n)$ בבדיקת חיתוך

דבר זה נכון גם רוצים את כל החיתוכים, כי הרשימה מכילה גם חיתוכים ואז החסם הוא $\geq 2n+K$.

אבל, ניתן להראות (אך לא נעשה זאת) שגודל רשימת ה X תמיד יהיה קטן או שווה ל $O(n\log^2 n)$

ולכן סיבוכיות המקום היא $O(n\log^2 n)$.

ניתן גם לעשות לבעיה בצורה אחרת - נכריח את רשימת ה-X להיות תמיד בגודל $O(n)$ ובכך נבטיח סיבוכיות $O(n)$. איך נעשה זאת?

נחזיק ברשימת ה-X רק נקודות חיתוך בין צלעות קטעים שכרגע מופיעים ברשימת ה-y ($< n-1$).

אמנם זה יגדיל את סיבוכיות זמן הריצה אך רק בקבוע.

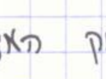
בעיה -

נתונות n נקודות במישור. נרצה למצוא את המרחק המינימלי בין זוג נקודות.

איך נעשה זאת (בעזרת sweep)?

במהלך האלגוריתם נחזיק δ ($\equiv$ המרחק המינימלי בין זוג נקודות משמאל לישר ה sweep). רשימת ה-X תהיה מורכבת מהנקודות (ממוינות).

אם δ מתעדכן כשנתקלים ב p_i אז ה δ החדש הוא מרחק מ p_i לאיזושהי נק' שמאלה

ולכן הוא מרחק בין p_i לנקודה בחצי המעגל  שרדיוסו δ

ובפרט לנקודה במלבן .

ניתן להראות שבגלל הדרישה שאין במלבן נקודות שמרחקן זו מזו קטן מ δ אז יש לכל היותר "6" נקודות כאלו במלבן. לכן צריך

לבדוק את p_i רק עם מס' קבוע של נקודות.

ועדיין, איך נדע מהם הנקודות שאיתן p_i עשויה לעבור את המרחק המינימלי?

בכל זמן, נחזיק רשימת y שתכלול רק חלק מהנק' שמשמאל לישר ה sweep - ושם יהיו ממויינות לפי סדר y עולה.

נאתר את p_i ברשימה ונעבור ממיקום זה למעלה ולמטה לפי הסדר של למרחק $\pm\delta$ מ p_i (בציר ה y).

עבור כל נקודה שנעבור עליה נבדוק אם היא במלבן של p_i. אם היא לא $k\delta$ (כלומר מרחקה בציר ה x מהישר $\geq \delta$) נוציא אותה מהרשימה, אם היא כן נבדוק את מרחקה מ p_i ונעדכן את δ אם צריך.

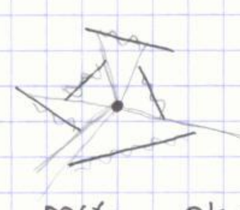
בכל צעד אני מבצעים מס' צעדים. מס' הצעדים בהם נק' מוצאות מרשימת ה y בסה"כ $\leq n$ ובנוסף בכל p_i מתבצעים $\leq c$ "צעדים אמיתיים" לכן מס' הצעדים בסה"כ $\leq cn$

לזה יש להוסיף את עלות המיון ($O(n\log n)$) ולכן בסה"כ מקבלים [illegible] ב $O(n\log n)$

בעיה-

נתונים n קטעים זרים במישור ונקודה o. נרצה לחשב את מפת הנראות עבור o

דוגמה-



איך נעשה זאת בעזרת sweep?

נעבור א sweep שמתקדם על ציר ה x ל sweep סיבובי ע"י קרן שיוצאת מ-o.

רשימת x ← רשימת θ

רשימת Y ← רשימת Γ

ברשימת ה-θ "מצאו" נקודות הקצה של הקטעים ממוינות לפי θ

ברשימת ה-Γ נמצאים הקטעים שהקרן חותכת לפי מרחק שלה מ-O

האתחול יהיה $\theta = 0$ ועבורו נמצא מהם הקטעים שהקרן ההתחלתית חותכת ונמיין לפי מרחק שלה.

בכל אירוע קריטי (נקודת קצה) נוסיף את הקטע לרשימת ה-Γ (או נמחוק אותו)

הנקודה O רואה בכל רגע את הקטע הראשון ברשימת ה-Γ.

הערה -

בצורה זו ניתן גם בהינתן n קטעים לחשב את G כושים נק' הקצה הרואות זו את זו -

ביצוע sweep סביב כל נקודת קצה

עלות אלגוריתם זה היא $O(n^2 \log n)$

בעיה -

נתונים n קטעים נחתכים. נרצה לבנות את המפה המישורית שהם מגדירים (במבנה DCEL למשל) בסיבוכיות $O(n+k)$ (k - מס' נקודות החיתוך)

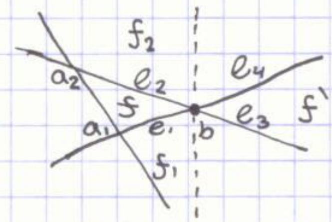
הקלט מורכב מקווים ישרים - $k = \Theta(n^2)$

באתחול נמיין את הישרים לפי הסדר ב $x = -\infty$ (סדר שיפועים יורד).

נריץ sweep למציאת נק' חיתוך

ברשימת ה Y נחזיק את רשימת הישרים בסדר הנוכחי וגם את רשימת הפאות ביניהם (ניתן לכל פאה "שם" כלשהו). למעשה, המשמעות היא שאנחנו מחזיקים את רשימת הקשתות של המפה שהישר חותך ועבור G קשת אנו יודעים מה נקודת הקצה השמאלית (אך לא את הימנית).

דוגמה -



בנקודה b

- e_1, e_2 מסתיימים (ידועים $e_1=(a_1,b)$ $e_2=(a_2,b)$)
- אחרי e_1 בפאה f_1 מופיעה הקשת החדשה e_3 (ידועים $e_3=(b,?)$)
- אחרי e_2 בפאה f_2 מופיעה הקשת החדשה e_4 (ידועים $e_4=(b,?)$)
- נוצרת פאה חדשה f'
- ו f_1, f_2 נשארות ללא שינוי

המפה המתקבלת נקראת המערך (Arrangement) של הישרים הנתונים

סיבוכיות הפתרון היא $O(n^2\log n)$

אם הקלט מורכב מקטעים חסומים אז האלגוריתם יותר בעייתי (כי, לדוגמה, ב $\times$ יש רק פאה אחת)

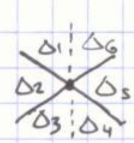
במקרה זה, במקום לחלק את הפאות נחלק פרוק שלהן לטרפזים אנכיים כאשר כל טרפז חסום ע"י ≤ 4 קשתות (2 מהן מוצאות בקטעים המקוריים ושתיים אנכיות)

מכל נק' קצה וכל נק' חיתוך נעלה קטע אנכי מעלה ומטה עד שיפגש במשהו נוסף (או עד $\pm\infty$)

נחשב פרוק טרפזי נ"ל ע"י sweep:

כמו קודם, רשימת ה y מתחזקת קשתות של המפה, אבל הפסים הרווחים ביניהם הם טרפזים

דוגמה -

עבור 

, בנקודת החיתוך אנו נסגור את הטרפזים $\Delta_{1,2,3}$

וניצור טרפזים חדשים $\Delta_{4,5,6}$

הפלט יהיה טרפזים כאשר כל טרפז יודע מיהן 4 הצלעות שלו

וכאן עם אחד הטרפזים הסמוכים לו מימין ומשמאל (דרך קירות אנכיים)
וכמה טרפזים כאלו יש?

ב $x=-\infty$ יש טרפז אחד

נק' קצה שמאלית יוצרת 2 טרפזים חדשים

נק' " ימנית " 1 " "

" חיתוך " 3 " "

כלומר בסה"כ יש עד (ואפילו בדיוק) $1+3(n+K)$ טרפזים

הערה-

חישוב כל החיתוכים בין n הקטעים כפי שעשינו לוקח ($O((n+K)\log n)$
אם $K=\Theta(n^2)$ אז אלגוריתם זה יותר איטי מהטריוויאלי ($O(n^2)$)
ואכן, קיימים אלגוריתמים אחרים שמחשבים את החיתוכים בזמן
$O(K+n\log n)$

דוגמא-

קיים אלגוריתם רנדומי אינקרמנטלי שמוסיף את הקטעים אחד אחרי השני
בסדר אקראי ומתחזק את הפירוק הטרפזי. אלגוריתם זה לוקח,
בתוחלת, $O(K+n\log n)$.

בעיה-

נתונות שתי מפות מישוריות M_1, M_2 מיוצגות ע"י DCEL. נרצה לחשב
את ה overlay - המפה המתקבלת מציור שתי המפות זו על גבי
זו.

ניתן לפתור ע"י sweep כשכאן קודם רשימת ה y מתחזקת צלעות של
ה overlay ופאות בניהן או טרפזים בניהם. בצורה זו נקבל את
הפירוק של ה overlay ומשם ניתן לשחזר את הפאות של ה overlay.
חישוב הפירוק הטרפזי יעלה $O((n_1+n_2+K)\log(n_1+n_2))$ (כאשר n_1
סיבוכיות M_1 ו- n_2 סיבוכיות M_2)

וכמה יעלה השחזור?

אם ניקח את הטרפזים ואת יחסי השכנות ביניהם דרך צלעות אנכיים, ואם נחשוב על מבנה זה כעל גרף בו

צמתים = טרפזים

קשתות = שכנויות

אז הבעיה שקולה לחישוב רכיבי קשירות בגרף כל רכיב קשירות הוא פאה של ה overlay (ולכן לינארית)