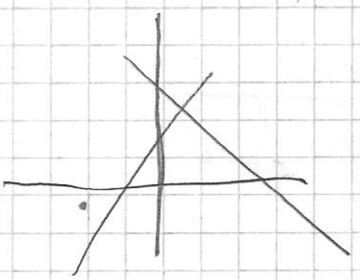
אופטימיזציה גיאומטרית

יש עוד קורס מתקדם, יש אחר, יש מאמרים וספרים. יאדה יאדה. בואו נתחיל.
נתחיל מכמה בע"ת וכמה טכניקות...

חיפוש פרמטרי Parametric Search

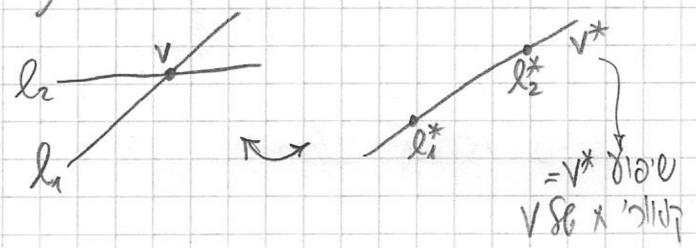
שיטה כבת 30, הומצאה ע"י מגידו (83). נבין אותה דרך דוגמאות.
יש בעיה, שעבורה רוצים למצוא משהו אופטימלי, מינימום או ערך כלשהו אחר.
הנה בעיה שתלווה אותנו:

<u>בעיה</u>: נתונים n ישרים לא אנכיים, במצב כללי במישור, ופרמטר $1 \le k \le \binom{n}{2}$.
<u>המטרה</u>: למצוא את נק' החיתוך ה-k משמאל.



כמובן, בזמן $O(n^2 \log n)$ ועם קצת מאמץ ב-$O(n^2)$ (כמו למצוא חציון) זה די טריוויאלי. רוצים קצת יותר קרוב ללינארי.

הבעיה, שנקראת slope selection, והסיבה היא שהבעיה דואלית לבעיה שבה יש n נק' ורוצים למצוא ישר שהוא בעל השיפוע ה-k ועובר דרך זוג נק' מתוך ה-n.



אז הנה, כאן יש בעיה, ורוצים למצוא את החיתוך ה-k. במקום למצוא את הפתרון האופטימלי x^* [= קואו' x של הקדקוד ה-k-י משמאל] נחפש פרוצדורת הכרעה (Decision Procedure), שמקבלת את כל הקלט ועוד x_0, ועונה על השאלה האם $x_0 \lesseqgtr x^*$?
כלומר במקום לפתור את בע"ת האופטימיזציה, פותרים בע"ת הכרעה.

אפשר לחשוב על פונק' ההכרעה כמנחה חיפוש כמו' חיפוש בינארי, למציאת x^*.
יש לנו $O(n^2)$ מועמדים, ואפשר לעשות עליהם חיפוש. אבל כמובן לא רוצים חיפוש בינארי מפורש על כל המועמדים.

נשכח לרגע מה-slope selection, וניקח בעיית צעצוע שתעזור לנו להבין את החיפוש הפרמטרי.

<u>בעיה</u>: נתונים n ישרים $y=a_ix+b_i$, $i=1,...,n$ וכולם עולים ($a_i>0$). נניח n אי-זוגי.
נסתכל על פונק' החציון M. לכל x, $M(x)=$ ערך ה-y החציוני של הישרים (ב-x).
M פונק' עולה ממש, ולכן חותכת את x פעם אחת בדיוק. יש לה שורש אחד, x^* ורוצים לחשב אותו.

<u>פתרון טריוויאלי</u>: נחתוך את כל הישרים עם ציר x, וניקח את החציון לפי x של נק' החיתוך.
למה? כי כל אלו שחותכים אחרי x^* יהיו שליליים (ויש $\frac{n}{2}$ כאלו) וכנ"ל מימין.

<u>פתרון בעזרת PS</u>:
פונק' הכרעה: בהינתן x_0, האם $x^* \lesseqgtr x_0$?
דרך מסובכת לעשות זאת, לחשב את $M(x_0)$ ולראות האם חיובי או שלילי. יותר פשוט, לספור כמה ישרים ב-x_0 נמצאים מעל ציר x (m).
אם $m>\frac{n}{2}$ ← $x^*<x_0$
אם $m<\frac{n}{2}$ ← $x^*>x_0$
אם $m=\frac{n}{2}$ ← נהדר. עוצרים. $x^*=x_0$.
פונק' ההכרעה עובדת ב-$O(n)$.

אגב, כאן בכלל יש רק n נק' קריטיות וחיפוש בינארי עליהן יעלה (מיון) $O(n\log n)$.
נתעלם מזה, ונעמיד פנים שיש המון נק' קריטיות.

<u>חיפוש פרמטרי</u>: נריץ את פרוצדורת ההכרעה ב-x^* (הלא ידוע).
נריץ <u>סימולציה</u> של פרוצדורת ההכרעה. הפרוצדורה הרי מבוססת על השוואות מהצורה: האם $a_ix^*+b_i \lesseqgtr 0$?
לא יודעים את התשובה, אבל אפשר לבדוק מתי זה שווה אפס, מתי זה גדול מ-0 וכו'.
אם $x^*>-\frac{b_i}{a_i}$ אז התשובה חיובית. אם < שלילית, אם = אז אפס.
ערכים קריטיים.

אבל הרי אנו יכולים להכריע בדיוק את השאלה הזו ע"י פרוצדורת ההכרעה!
נקרא לפרוצדורת ההכרעה עם הערך הקונקרטי $x_0=-\frac{b_i}{a_i}$ ונחליט לפי התשובה.

וכך נמשיך את סימולצי' פרוצ' ההכרעה, כאשר כל פעם שנתקלים בהשוואה קונקרטית, מכריעים בעזרת פרוצ' ההכרעה.

והרי התשובה של הסימולציה לא מעניינת כי מן הסתם היא תענה שיש $n/2$ ישרים מעל x^*, שכן כי x^* הוא אותו ערך אופטימלי. התשובה לא מעניינת, מעניין מה שמגלים תוך כדי.

כל השוואה שנפתרת ע"י קריאה קונקרטית נותנת לנו אילוץ על x^*:

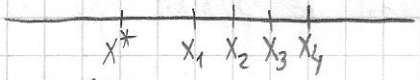
והטווח של x^* הולך ומצטמצם עד שמקבלים תשובה ל-x^*. לא ייתכן שבסוף x^* יהיה בין 3 ל-5 כי רק בנק' אחת יהיו בדיוק התשובות שמתקבלות ב-x^*.

$$\left\{\begin{array}{l} x^* > -\frac{b_1}{a_1} \\ \vdots \\ x^* < -\frac{b_2}{a_2} \\ \vdots \\ x^* > -\frac{b_n}{a_n} \end{array}\right.$$

אוסף כל האילוצים יקבע את x^*

בלי שום התחכמויות, העניין רץ ב-$O(n^2)$ כי לסימולציה n השוואות, ובכ"א קוראים לפרוצ' הקונקרטית שרצה ב-$O(n)$, ולכן, סה"כ: $O(n^2)$.

התחכמויות:



אם ההשוואה הראשונה שנתקלים בה היא עם x_1 ומקבלים $x^*<x_1$ אז כל השוואה אחרת אפשר לענות מיד.
הסדר של הישרים עלול להכתיב את זמן הריצה. אפשר לעשות פרמוטציה אקראית של הישרים ואז אפשר להראות (שבתוחלת) נקרא לפרוצ' ההכרעה (מעט log) פעמים.

איך להתגבר על עניין ה-$O(n^2)$? רעיון נוסף: להשתדל לבצע הרבה השוואות בו זמנית.
כאן, אפשר לעשות את כל ההכרעות בו זמנית, כי הן בלתי תלויות.

נניח שמצאנו m (אצלנו n=m) השוואות שלא תלויות זו בזו ושניתן לבצע את כולן <u>במקביל</u> ונקבל m ערכים קריטיים $x_1, \ldots, x_m$.

אפשר להריץ עליהן חיפוש בינארי ואז יש רק $\log m$ קריאות לפרו' ההכרעה הקונקרטית. ברגע שנעשה זאת, נדע את התשובה לכל הערכים הקריטיים.

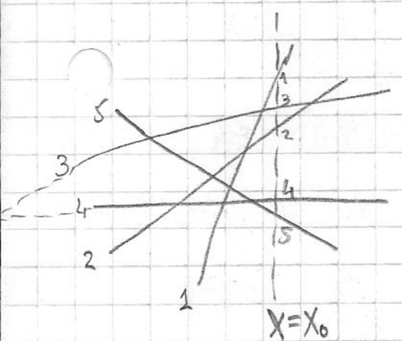
כלומר, יודעים מראש את כל השאלות. מריצים במקביל, מקבלים ערכים קריטיים ומקבלים זמן ריצה של $O(n\log n)$.

מה שמבקש זה לבצע את סימולציית פרוצדורת ההכרעה ע"י אלגוריתם <u>מקבילי</u>, עם מספר קטן של צעדי זמן מקבילים. כלומר, רוצים לארגן את ההשוואות במספר קטן של שכבות שתלויות אחת בשניה. אבל האלג' שלנו הוא <u>סדרתי</u>. פשוט לפי המקבילי יוצרים רשימת השוואות שאותם רוצים לעשות, ואותם מבצעים.

שוב, רוצים לארגן את השלב העיקרי של סימולציית פרוצ' ההכרעה בצורה כמה שיותר מקבילית, וככה נוכל בצורה סדרתית לעבוד מעט שלבים.

<u>דוגמה לבעיית ה Slope Selection</u>

קודם כל, פונק' הכרעה יעילה. יש לנו ישרים $l_1, \ldots, l_m$, ופרמטר $1 \le k \le \binom{m}{2}$, ורוצים למצוא את נק' החיתוך x^* ה-k משמאל.



פרוצ' ההכרעה: בהינתן x_0, האם $x^* \lesseqgtr x_0$

אפשר לספור כמה נק' חיתוך נמצאות משמאל ל-x_0.

אם המספר יהיה קטן מ-k, אז x^* מימין וכו'.

$$\text{אם } < k \Leftarrow x^* < x_0$$
$$> k \Leftarrow x^* > x_0$$
$$= k \Leftarrow x^* = x_0$$

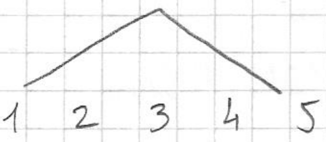
איך עושים זאת? זה יחסית קל. נמיין את הישרים לפי סדר השיפועים יורד של שלהם. נניח שזה הסדר $l_1, \ldots, l_m$. זהו סדר הישרים ב-$x = -\infty$.

נחשב גם את סדר הישרים על $x = x_0$, ונקבל פרמוטציה של הסדר שלהם ב-$-\infty$.

<u>אבחנה</u>: 2 ישרים l_i, l_j נחתכים משמאל ל-x_0 אם סדרם ב-x_0 הפוך לסדרם ב-$-\infty$.

⇐ זהו מס' ה-inversions, מס' נק' החיתוך משמאל ל-x_0 = מספר ההיפוכים (inversions) בפרמוטציה π (הפרמוטציה ב-x_0).

כיצד סופרים את מס' ה-inversions - זהו תרגיל במבני נתונים. שמים את הישרים בעץ חיפוש, אבל מכניסים אותם לפי סדרם ב-π. בכל הכנסה, נספור כמה גדולים ממנו יש בעץ, וזה כמה היפוכים נוצרו.



5	3	4	2	1	סדר הכנסה:
1	2	3	4	5	העץ:
4	2	2	1		⇐ 9 חיתוכים

ועכשיו לחיפוש הפרמטרי:

מריצים סימולציה ב-x^* הלא ידוע. יש כמה שלבים לסימולציה:

(1) מיון הישרים לפי שיפועם לא תלוי ב-x^*. זה לא חלק בסימולציה, וזה לא בעיה.

(2) מיון הישרים על x^*, כמובן תלוי מאוד ב-x^*.

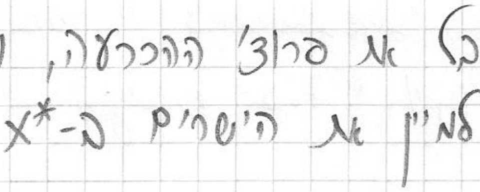
(3) סופרים כמה היפוכים. לא תלוי ב-x^*, תלוי רק בסדר (משלב (2)). זה קומבינטורי לגמרי (על העץ). ברגע שמקבלים את π אפשר לעצור.

וזה חשוב, כי הרי אנו רוצים לקבל את פרוצ' ההכרעה, והדבר היחיד שבאמת צריכים לעשות עם סימולציה זה למיין את הישרים ב-x^*.

אם כן, מחפשים אלג' מקבילי שיבצע מיון של n מספרים, ורוצים כמה שיותר חישובים בכל שלב.

מסתבר שניתן למיין בזמן מקבילי $O(\log n)$ ע"י $O(n)$ מעבדים, ואפילו אפשר לעשות זאת ע"י <u>רשת מיון</u> (Sorting Network).

דרך פשוטה (לא הכי יעילה) למיין זה merge-sort. זהו מיון שעובד יפה בצורה מקבילית. ה-merge קצת יותר קשה, כי אפשר לעשות אותו בזמן לינארי אבל סדרתי. אז:



n/2 מעבדים עושים חיפוש בינארי לאן להכניס n/2 איברים

אפשר להאיץ להוריד ל-$O(1)$

$\log n$ רמות בעץ × $\log n$ זמן מקבילי ברמה אחת

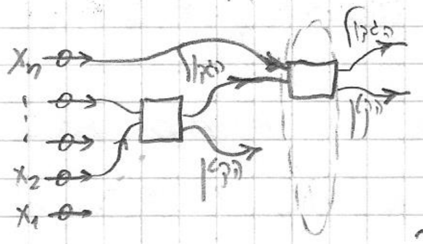
זה n/2 מעבדים שעובדים $\log n$ זמן (לא יעיל אבל סבבה)

והנה ניתן לבצע את המיון ב-$O(\log^2 n)$.

אפשר לחשוב גם על מימוש מקבילי של quicksort.

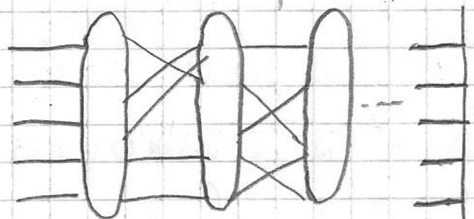
בקיצור מיון זה משהו מאוד מקבילי אם עושים אותו כמו שצריך.

רשת מיון:



מכל צומת יש חוט שהולך ימינה, יש שערים שאליהם נכנסים 2 חוטים ויוצאים גדול וקטן, עד שבסוף, לכל סדרי ההתחלה, סדר החוטים יהיה מהגדול לקטן.

המקביליות בעצם נמדד בכמה שכבות צריך כדי לסדר בהם את כל השערים.

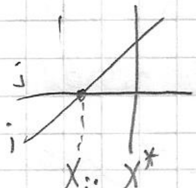


ויש הוכחה, כאמור, שניתן לעשות זאת עם logn שלבים. כלומר, ניתן לממש את זה מראש.

המציאו את זה 3 הונגרים וקראו לזה AKS network. (semeredi)

מוכרים ל-slope selection, וכמו יודעים שניתן לבצע את המיון על $x=x^*$ ב-$O(\log n)$ צעדים מקביליים שבכ"א מבצעים $O(n)$ השוואות.

מהן ההשוואות האלו? $a_ix^*+b_i : a_jx^*+b_j$ (מעל או מתחת?). הערך הקריטי של ההשוואה הוא בעצם נק' החיתוך בין 2 הישרים, שבה סדרם מתחלף



$$a_ix_{ij}+b_i=a_jx_{ij}+b_j$$

$\Leftarrow$ מחשבים את הערכים הקריטיים ומבצעים חיפוש בינארי ביניהם למיקום x^* ביניהם $\longrightarrow$ $O(\log n)$ קריאות לפרוצדורה הקונקרטית שעולה $O(n\log n)$.

$\Leftarrow$ $O(n\log n)$ לביצוע צעד מקבילי אחד של הסימולציה.

נכפיל ב-$O(\log n)$ צעדים מקביליים $\Leftarrow$ סה"כ: $O(n\log^3 n)$.

כל השוואה קונקרטית מצמצמת את הטווח של x^*. בסיום, או קודם לכן, ניתקל ב-x^* כערך קריטי. לא "יתכן" שלא ניתקל ב-x^*, כי

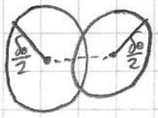
ואלו נתקל בהשוואה עם x^* האמיתי בסוף, יהיה לנו אינטרוול $a<x^*<b$
כי שאלנו את כל השאלות וענינו הכל ול-a ו-b צריכה להיות אותה ריצה, ובקיצור לא יתכן זאת כי יש רק x^* אחד שבו הפרוצדורה נפסקת כפי שהסימולציה אמורה להסתיים.

אגב, בשביל לעשות חיפוש בינארי לא צריכים למיין! מספיק למצוא חציון בזמן לינארי, ואז לזרוק חצי וכך הלאה. האמת, אפילו יש לא מעט מן המיותר בלמיין את כולם, כי לא באמת צריכים את הסדר בין כולם, וצריך רק את החציונים...

<u>עוד דוגמה:</u>

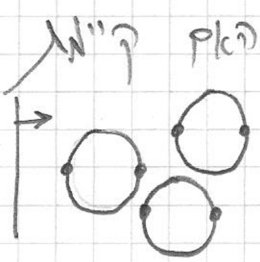
נתונות n נק' במישור ורוצים למצוא מרחק מינימלי בין זוג מהן.
האמת באותה דרך אפשר לפתור את הבעיה לקלטים או יצורים יותר מסובכים אבל לעת עתה נשאר עם נקודות.
אם כן, רוצים למצוא את δ^*, ולשם כך רוצים פרוצ' הכרעה, בהינתן δ_0, האם $\delta_0 \gtreqless \delta^*$. כלומר בהינתן δ_0, האם יש זוג נק' שמרחקן אחת מהשניה קטן מ-δ_0?
אם לא, אז $\delta_0<\delta^*$, ואם כן, יש זוג נק' במרחק $<\delta_0$ ואז $\delta_0>\delta^*$, ואם לא שניהם אז $\delta^*=\delta$.

נצייר סביב כל נק' עיגול ברדיוס $\frac{\delta_0}{2}$ ונשאל: האם קיים זוג מעגלים נחתכים?



$\iff$ המרחק בין המרכזים $<\delta$

בהינתן אוסף של מעגלים, אפשר בעזרת sweep לבדוק האם קיימת נק' חיתוך, בזמן (מעגלים) $O(n\log n)$: לאחר מיון והכנסה ל-x-str,
נקודות במעגל, מכניסים את החלק העליון והתחתון ל-y-str בנפרד,
לאחר ש-2 החלקים יהיו צמודים עד שמישהו יחתוך אותם, ואז בעצם נעצור... כל פעם שמכניסים או מוציאים מעגל, זה משפיע רק על שכניו ב-y-str.



העניין הוא שמיון והכנסה ל-x-str זה קל, אבל איך לנצל מקבילים sweep? הכל שם תלוי בהכל!

אבל יש פתרון. בעצם לא צריך למקבל את כל המימוש של ה-sweep.

<u>מקבול:</u>

~~אפשרות: קודם כל, בגלל שכל המעגלים הם באותו רדיוס, אם יש חיתוך בין שני מעגלים, אז הוא יהיה בין אלה עליון של המעגל להוא התחתון של השני.~~
(מורה שמה לב שלא נכון)

<u>אפשרות</u>: כדי ~~[illegible]~~ שזוג מעגלים יחתכו, המרכזים צריכים להיות לא רחוקים אחד מהשני על ציר x.

$|p_x-q_x|<\delta$

עושים לוקליזציה לאיפה עלולים להיות נק' חיתוך, ויוצרים את בע״ת ונמשיך עם הפרטים בשבוע הבא.