28.10.2019

פתרון בעיות באלגוריתמים שיעור 1

מה נלמד בקורס?

1. פתרונות לא נאיביים שייעלו בעיות מוכרות, שגרות שימושיות
   ← בעיית מנוע החיפוש
2. מבני נתונים (שנלמד גם לממש) שמרחיבים יכולות שלמדנו
   ← Skip List
3. טכניקות להאצת אלגוריתמים
   ← דמני ריצה שכוללים $\sqrt{n}$
   ← אלגוריתמי fingerprinting הסתברותיים

בעיית ה-Range Minimum Query (RMQ)

בהינתן מערך A, מעוניינים לענות על שאילתות מהצורה: מהו $\min A[i,...,j]$? $(i\le j)$

- נבנה מבנה נתונים מתאים (עיבוד מוקדם - מתבצע פעם אחת):

מערך A ← אלג' עיבוד מקדים ← מבנה נתונים $D=(A,B)$

- נבצע שאילתות מעל D (שלב השאילתא - הרבה פעמים):

מבנה נתונים $D=(A,B)$ ו-(i,j) ← אלג' מענה על שאילתא ← $\min A[i,...,j]$

פתרונות אפשריים:

(א) ללא עיבוד מוקדם (RMQ1)

- עיבוד מוקדם: ללא $O(1)$
- שאילתא: החזר את המינימום בקטע $O(n)$
- מקום: שומרים את A $O(n)$

הערה: שמירת הקלט נעשית ב-$O(n)$ בכל המקרים הבאים אך עדיין נניח בהם.

(ב) הרבה עיבוד מוקדם (RMQ2)

- עיבוד מוקדם: נגדיר טבלה B עם $B[i,j]=\min A[i,...,j]$ בעזרת תכנון דינאמי ←

$$B[i,j]\begin{cases}\min\{B[i,j-1],A[j]\} & j>i\\ A[i] & j=i\end{cases} \qquad O(n^2)$$

- שאילתא: החזרת $B[i,j]$ $O(1)$
- מקום: הטבלה B $O(n^2)$

(ג) חלוקה ל-$\sqrt{n}$ בלוקים בגודל $\sqrt{n}$ (RMQ3)

- עיבוד מוקדם: ביצוע החלוקה ולכל בלוק נשמור את המינימום ב-B $O(n)$

המשך RMQ

- שאילתא: מוצאים את הבלוקים שמוכלים במלואם בקטע $[i,j]$, מוצאים את המינימום שלהם בעזרת B ומשווים למינימום של 2 הבלוקים $O(\sqrt{n})$

- מקום: שומרים את המערך המקורי ו-$\sqrt{n}$ ערכים ב-B $O(n)$

(ד) חלוקה לקטעים באורכי חזקות של 2 (RMQ 4)

- עיבוד מוקדם: נגדיר טבלה B באורך $O(n\log n)$ עם $B[i,k] = \min[i, i+2^k-1]$

בעזרת תכנון דינאמי - $B[i,k] \begin{cases} \min\{B[i,k-1], B[i+2^{k-1},k-1]\} & 1 \le k \\ A[i] & k=0 \end{cases}$

$O(n\log n)$

- שאילתא: נסמן את אורך הקטע ב-t, כלומר $t=j-i+1$ ונחזיר את

$$\min\{B[i,\lfloor\log t\rfloor], B[j-2^{\lfloor\log t\rfloor}+1, \lfloor\log t\rfloor]\}$$

← אבחנות לגבי ערכים תחתונים:

$$(*) \quad 2^{\lfloor\log t\rfloor} \le 2^{\log t} = t$$

$$(**) \quad t = 2^{\log t} < 2^{\lfloor\log t\rfloor+1} \Rightarrow t+1 \le 2^{\lfloor\log t\rfloor+1}$$

(שים לב שמ-(**) נובע שיש חפיפה בין הקטעים מכיוון ש: $i+2^{\lfloor\log t\rfloor}-1 \ge j-2^{\lfloor\log t\rfloor}+1$ אמ"מ $t+1 \le 2^{\lfloor\log t\rfloor+1}$)

← מסקנות:

(א) $i+2^{\lfloor\log t\rfloor}-1 \le i+2^{\log t}-1 = i+(j-i+1)-1 = j$ [לא חורגים מ-j]

$O(1)$ (ב) $j-2^{\lfloor\log t\rfloor}+1 \ge j-t+1 = j-(j-i+1)+1 = i$ [לא חורגים מ-i]

- מקום: כל הקטעים באורכי חזקות של 2 $O(n\log n)$

סיכום סיבוכיות הפתרונות:

	עיבוד מוקדם T_p	שאילתא T_q	מקום S
RMQ1	$O(1)$	$O(n)$	$O(n)$
RMQ2	$O(n^2)$	$O(1)$	$O(n^2)$
RMQ3	$O(n)$	$O(\sqrt{n})$	$O(n)$
RMQ4	$O(n\log n)$	$O(1)$	$O(n\log n)$

בעיית ה-Lowest Common Ancestor (LCA)

בהינתן עץ מושרש T, מעוניינים לענות על שאילתות מהצורה: מהו הקדקוד הנמוך ביותר בעץ ש-u ו-v צאצאיו?

[נראה בשבוע הבא שבהינתן פתרון יעיל ל-RMQ ניתן לפתור ביעילות את LCA - ואם להפך! ונראה איך לצאת מהמעגל הרקורסיבי הזה]

פתרונות אפשריים:

(א) סריקה סדרתית (LCA1)

- עיבוד מוקדם: אתחול מערך של flag-ים באורך n — $O(n)$
- שאילתא: נטפס מ-v לשורש ונסמן קודקודים בדרך. לאחר מכן נטפס מ-u ונחזיר את הקודקוד הראשון שסומן $O(n)$
- מקום: מערך ה-flag-ים $O(n)$ — {נשים לב שבסיום השאילתא יש לאפס את מערך ה-flag-ים לטובת השאילתא הבאה}

(ב) סריקה במקביל (LCA2)

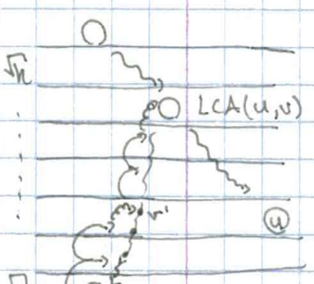
- עיבוד מוקדם: נגדיר את גבהי הקודקודים בעץ במערך L עם DFS $O(n)$
- שאילתא: נטפס מהנמוך מביניהם עד להשוואת רמות ולאחר מכן נטפס במקביל עד להגעה לקודקוד משותף. $O(n)$
- מקום: המערך L. $O(n)$

(ג) חלוקה לקבוצות רמות בגודל $\sqrt{n}$ (LCA3)

- עיבוד מוקדם: לכל קודקוד v נגדיר את $GP[v]$ להיות הקודקוד הנמוך ביותר מקבוצת הרמות הקודמת - [האב הקדמון הנמוך ביותר שלא באותה קבוצה]

$$GP[v]\begin{cases} GP[v.parent] & \text{אם v ואביו באותה קבוצה} \\ v.parent & \text{אחרת} \end{cases} \qquad O(n)$$

- שאילתא:
  - נבצע $O(\sqrt{n})$ "קפיצות גדולות" מהנמוך יותר להשוואת קבוצות הרמות של הקודקודים
  - נבצע "קפיצות קטנות" להשוואת רמת הקודקודים
  - נבצע "קפיצות גדולות" במקביל עד להגעה לקודקוד זהה
  - נבצע "קפיצות קטנות" להגעה לאותו קודקוד משותף $O(\sqrt{n})$



- מקום: לכל קודקוד שומרים את $GP[v]$ $O(n)$

סיכום סיבוכיות הפתרונות:

	T_p	T_q	S
LCA1	$O(n)$	$O(n)$	$O(n)$
LCA2	$O(n)$	$O(n)$	$O(n)$
LCA3	$O(n)$	$O(\sqrt{n})$	$O(n)$

04.11.2019

פתרון בעיות באלגוריתמים שיעור 2

המשך בעיית ה-LCA

(ד) הגדרת האב הקדמון שבמרחק 2^k (LCA4)

- <u>עיבוד מוקדם</u>: נגדיר טבלה B בגודל $n \cdot \log n$ עם:

$B[u,k] =$ האב הקדמון של u במרחק 2^k ממנו

בעזרת תכנון דינאמי ← $B[u,k] = \begin{cases} B[B[u,k-1],k-1] & 0<k \\ u\ parent & k=0 \end{cases}$

בנוסף, גם באמצעות DFS, נשמור את רמות הקודקודים. $O(n\log n)$

- <u>שאילתא</u>: (מימוש של חיפוש בינארי) $2^{\log n-1} = \frac{n}{2}$
  - לכל $k = \log n-1,\dots,0$ - אם $B[v,k]$ נמוך מ-u, עדכן $v = B[v,k]$
    {k יורד מ-$\log n-1$ ל-0}
  - לכל $k = \log n-1,\dots,0$ - אם $B[u,k] \neq B[v,k]$ עדכן $v=B[v,k]$, $u=B[u,k]$
  - החזר את v parent

  $O(\log n)$

- <u>מקום</u>: המערך B שמכיל אבות קדמונים במרחק 2^k לכל v $O(n\log n)$

גם כאן ובמקרים הבאים עלות שמירת הקודים $O(n)$ אבל זניחה ביחס ליתר

בעיית ה-RMQ^*

נגדיר בעיה זו באופן זהה ל-RMQ, למעט ההנחה כי ההפרש בין כל זוג איברים סמוכים במערך הוא ± 1.

באמצעות פתרון יעיל של בעיה זו (שואף ל-$(O(n),O(1),O(n))$) נמצא גם פתרונות יעילים יותר ל-RMQ ו-LCA.

<u>פתרון - ניסיון 1</u>:

- <u>עיבוד מוקדם</u>:

1. נחלק את המערך A לבלוקים בגודל $L = \frac{1}{2}\log n$, ונחשב במערך B את המינימום של כל בלוק.
2. נחשב את מבנה RMQ4 על גבי מערך B (חלוקה לקטעים באורכי חזקות של 2, בשביל המציאה בקריות מינימום של בלוקים מוכלים)
3. לכל בלוק שהגדרנו, נחשב את מבנה RMQ4 עליו (למציאת המינימום של הזנבות)

- <u>שאילתא</u>:

1. הגדר את הקטע $i,\dots,j$ כרצף בלוקים באורך L ומצא את המינימום בהם עם מבנה הנתונים שהוגדר על B.
2. מצא מינימום עבור הזנבות והחזר את המינימלי מבין השלושה.

$O(1)$

המשך *RMQ

נראה שלרב העיבוד המוקדם בפתרון גדול מ-$O(n)$. ננתח:

← עלות שלב 2:

$$O(|B|\cdot\log|B|)=O\left(\frac{n}{L}\log\frac{n}{L}\right)\le O\left(\frac{n}{L}\cdot\log n\right)=O\left(\frac{n}{\frac{1}{2}\cancel{\log n}}\cdot\cancel{\log n}\right)=O(n)$$

← עלות שלב 3:

$$O\left(\frac{n}{\cancel{L}}\cdot\cancel{L}\cdot\log L\right)=O(n\cdot\log\log n)$$

אז עלות העיבוד המוקדם באמת גדולה מ-$O(n)$, אבל לא השתמשנו בהנחה של *RMQ!

תיקון לפתרון:

נגדיר לכל בלוק A את הבלוק הקאנוני המתאים לו, כבלוק שאיברו הראשון הוא 0 וסדרת ההפרשים בו זהה לזו של A

A: | 2 | 3 | 4 | 3 | 2 |

D: | 0 | 1 | 2 | 1 | 0 |

← הבחנות: • $\forall i\ D[i]=A[i]-A[0]$ הבלוק הקאנוני

• $\min A[i,\dots,j]=\min D[i,\dots,j]+A[0]$

• יש 2^{L-1} בלוקים קאנוניים [בגודל לכל איבר הוא ±1 מהקודם]

כעת נתקן את שלב 3 בעיבוד המוקדם (ונוסיף שלב רביעי)

- תיקון עיבוד מוקדם:

1. } לא נגענו
2.

3. לכל בלוק במערך B (שמור מצביע לבלוק הקאנוני שמתאים לו)
4. לכל בלוק קאנוני נגדיר RMQ4

← עלות שלב 3: $O(n)$

← עלות שלב 4:

$$O(2^{L-1}\cdot L\cdot\log L)\le O(2^{\frac{1}{2}\log n}\cdot L^2)=O(\sqrt{n}\cdot(\log n)^2)\le O(n)$$

} $O(n)$

בעצם הטריק הוא שכמה בלוקים מתמפים לאותו בלוק קאנוני וככה חסכנו מלא עבודה בשלבים 3 ו-4

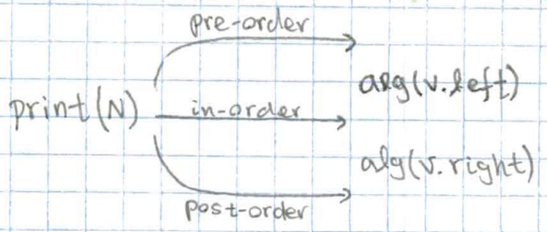
- מקום: המערך A בגודל n המערך B בגודל $\frac{n}{L}$, מבנה RMQ4 על B בגודל $O\left(\frac{n}{L}\log\frac{n}{L}\right)=O\left(\frac{n}{\frac{1}{2}\log n}\log\frac{n}{L}\right)=O(n)$, 2^{L-1} בלוקים קאנוניים ומבנה RMQ4 על הבלוקים גם כן ב-$O(n)$ $O(n)$

← הצלחנו לפתור את *RMQ בסיבוכיות $(O(n),O(1),O(n))$

פתרון יעיל ל-LCA בעזרת RMQ* (LCA5)

<u>תזכורות:</u>

• יש כמה דרכים לסרוק עץ:

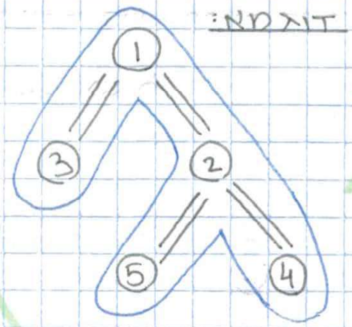


• מסלול אויילר - מסלול שעובר בכל צלעות הגרף בדיוק פעם אחת

← בעץ בינארי נגדיר שתי צלעות בין כל זוג קודקודים, אחרת אין מסלול אויילר בעץ.

← מסלול אויילר כזה הוא בעצם מעגל שמקיף את כל העץ ומתאים לסריקה שלו.

<u>דוגמא:</u>



<u>אבחנה:</u> ניתן לשטח עץ בינארי למערך באמצעות מסלול האויילר 12:20 -

	1	2	3	4	5	6	7	8	9
E^1	1	3	1	2	5	2	4	2	1
E^2	0	1	0	1	2	1	2	1	0

E^2 → מערך הרמות

$\bar{E}$ 1 4 2 7 5 → מיקום קודקוד במסלול

- <u>עיבוד מוקדם:</u>

(1) הגדר את המערך E (הדו מימדי) המתאים לסריקת מסלול אויילר בעץ (2) הגדר מערך $\bar{E}$ ששומר לכל קודקוד את מיקומו הראשון במסלול [כלומר $E^1[\bar{E}[v]] = v$] (3) עיבוד מוקדם של RMQ* על המערך E^2 בעזרת $O(n)$ (גודל המערך הוא $2n-1$ שלכן $O(n)$ והעיבוד המקדים של RMQ* לינארי) } $O(n)$

<u>הבחנות:</u>

✻ {תת מסלול מ-u ל-v שקיים במסלול אויילר שלנו}

• בתת עץ ששורשו w עם v,u קודקודים כעלים, תת✻ מסלול אויילר מ-u ל-v לא עובר בקודקודים הגבוהים מ-w. בפרט כאשר $w = LCA(u,v)$. ← DFS

• ✻תת מסלול אויילר מ-u ל-v עובר בהכרח דרך $LCA(u,v)$ ← בפרט מסלול אויילר הוא מסלול וכל מסלול עובר ב-LCA

• בתת העץ הנ"ל, $LCA(u,v)$ הוא הקודקוד שרמתו נמוכה ביותר.

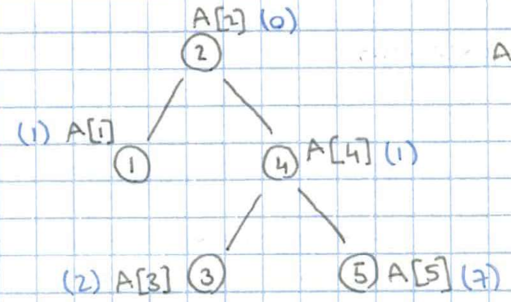
- <u>שאילתא:</u> נחזיר את $E^1[\text{argmin}\, RMQ^*_{E^2}(\bar{E}[u], \bar{E}[v])]$ $O(1)$

- <u>מקום:</u> שמירת המערך E וכן מבנה RMQ* עליו (שלו סיבוכיות מקום לינארית)

$O(n)$

פתרון יעיל ל-RMQ בעזרת LCA (RMQ5)

דוגמא:



	1	2	3	4	5
A:	1	0	2	1	7

- עיבוד מוקדם: בהינתן מערך A נגדיר עץ T המתאים לו ומקיים את התכונות הבאות - # סריקת in-order של שמות הקודקודים בעץ היא 1,...,n
  # ערכו של כל קודקוד קטן מערכי בניו - תכונת הערימה.
  הגדרת עיבוד מוקדם של LCA5 על העץ בעלות O(n)

* בניה כזו ב-O(n)

הבחנות: (כמו קודם כאשר מתייחסים מסדור אי״דר ב-in-order)

- בתת עץ ששורשו w כך ש-v,u קודקודים בו, תת סריקת in-order מ-u ל-v לא עוברת בקודקודים הגבוהים מ-w
- סריקת in-order מ-u ל-v עוברת דרך LCA(u,v)
- בתת העץ הנ"ל, LCA(u,v) הוא הקודקוד שערכו הוא המינימלי

תכונת הערימה

- שאילתא: A[LCA(i,j)] O(1)
- מקום: המערך, העץ ומבנה LCA5 עליו O(n)

בנייה ב-O(n): בהינתן שהכנסנו לעץ את i התאים הראשונים, כדי להוסיף
מהקודקוד התחתון בה!
את התא ה-i+1, נעבור על שרשרת הקודקודים הימנית בעץ ונמצא את המקום המתאים (שמקיים את תכונת הערימה). נוסיף את הקודקוד ונעביר את כל השרשרת שמתחתיו להיות משמאלו. שם הקודקוד יהיה i+1 וערכו A[i+1] (ובכך שמרנו גם את תכונת הסדר).

זמן הריצה נובע מכך שבכל פעם ש"חותכים" חלק מהעץ העברנו אותו מתת עץ ימני להיות בן שמאלי - ולא ניגע בו עוד.

11.11.2019

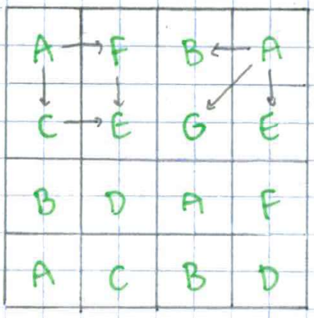
פתרון בעיות באלגוריתמים שיעור 3

אלגוריתמים שזמן ריצתם כולל $\sqrt{n}$

① **פירוק שורש** - בטכניקה זו מחלקים את הנתונים ל-$\sqrt{n}$ קבוצות בגודל $\sqrt{n}$ (או פרמטר אחר) **בשלב שלאחר מכן** מבצעים את החישוב ברמת המאקרו (אוסף בלוקים) או ברמת המיקרו (בלוק נתון)

← דוגמא: LCA3 ,RMQ3

② **שילוב אלגוריתמים** - טכניקה זו שימושית למקרים בהם לבעיה נתונה קיים אלגוריתם שטוב עבור קלטים מסויימים ולא טוב עבור אחרים (או עבור חלקים מהקלט). ננסה למצוא אלגוריתם אחר שטוב עבור הקלטים הבעייתיים של אלג' 1 ונשלב את שניהם לאחר מכן.



← דוגמא: נתונה טבלה עם n אותיות ומעוניינים למצוא את זוג התאים הקרובים ביותר שמכילים את אותה אות.

המרחק מוגדר כ: # המרחק בגרף המתאים

מרחק מנהטן בשריג: $|x_1-x_2|+|y_1-y_2|$

אלגוריתם 1 - לכל אות c_i עם k_i מופעים נחשב את המרחק המינימלי בין כל הזוגות

מס' סוגי האותיות

← זמן ריצה: $O(n\log n + \sum_{i=1}^{m} k_i^2)$. במקרה הגרוע: $O(n^2)$.

מיון

זוגות

למשל אות שמופיעה הרבה פעמים

אלגוריתם 2 - נריץ BFS במקביל מכל אות עד להגעה להתנגשות שתעיד על המרחק המינימלי עבור אות זו.

← זמן ריצה: $O(n\log n + n\cdot m)$. במקרה הגרוע: $O(n^2)$

למשל כשכל אות מופיעה פעם אחת

שילוב האלגוריתמים: לכל אות c_i עם $k_i < k$ מופעים נריץ את אלג' 1 ועבור היתר את 2.

← זמן ריצה:

• עבור אותיות עם מעט מופעים:

$$O\left(\sum_{i=1}^{m} k_i^2\right) = O\left(\sum_{i=1}^{m} k_i\cdot k_i\right) \le O\left(k\cdot\sum_{i=1}^{m} k_i\right) = O(n\cdot k)$$

• עבור אותיות עם הרבה מופעים - ישנן לכל היותר $\frac{n}{k}$ אותיות כאלו:

$$O(n\cdot m)\le O\left(n\cdot \frac{n}{k}\right) = O\left(\frac{n^2}{k}\right)$$

כלומר סה"כ $O\left(n\log n + nk + \frac{n^2}{k}\right)$

המשך שיפור אלגוריתמים

כעת נרצה למצוא את ה-k שיגיע לערך מינימלי:

$$\frac{d\left(nk+\frac{n^2}{k}\right)}{dk} = n - \frac{n^2}{k^2} = 0 \Longrightarrow \frac{n^2}{k^2} = n \Rightarrow k^2 = n \Rightarrow \boxed{k=\sqrt{n}}$$

נבחין כי $O\left(nk+\frac{n^2}{k}\right) = O\left(\max\left\{nk, \frac{n^2}{k}\right\}\right)$, ועבור $nk = \frac{n^2}{k}$ מתקבל המינימום - כלומר זה אכן $k=\sqrt{n}$.

← לפיכך זמן הריצה הסופי הוא $O(n\sqrt{n})$.

③ אלגוריתם Mo - טכניקה זו שימושית בבעיות אופליין, בהן השאילתות נתונות מראש ומדובר בשאילתות טווח.

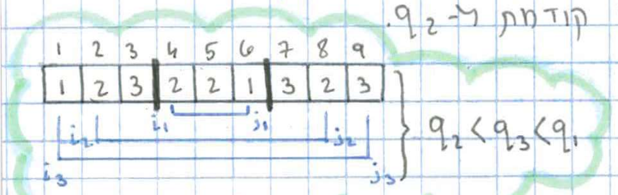
← **הסכימה:** [חילקנו את המערך לבלוקים בגודל $\sqrt{n}$]

1. נמיין את השאילתות כך שלכל זוג $q_1=(i_1,j_1)$ ו- $q_2=(i_2,j_2)$:

אם $\left\lfloor \frac{i_1}{\sqrt{n}} \right\rfloor < \left\lfloor \frac{i_2}{\sqrt{n}} \right\rfloor$, אז q_1 קודמת ל-q_2. באיזה בלוק מתחיל הטווח

אם שני הטווחים מתחילים באותו בלוק, נמיין לפי הראשון שמסתיים

אם $\left\lfloor \frac{i_1}{\sqrt{n}} \right\rfloor = \left\lfloor \frac{i_2}{\sqrt{n}} \right\rfloor$ ו- $j_1 < j_2$, אז q_1 קודמת ל-q_2.



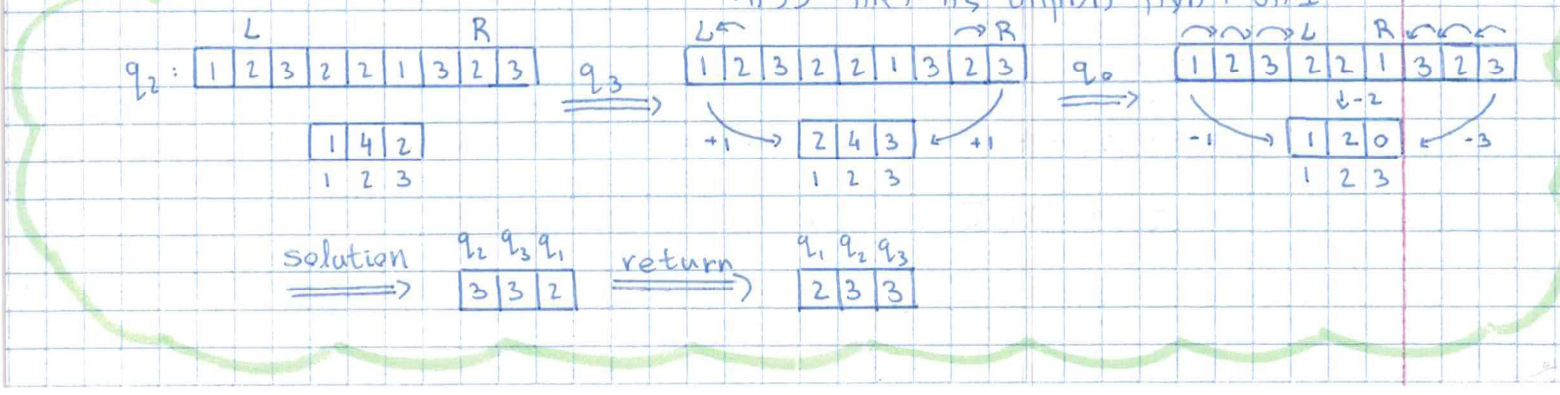
2. נמפה את השאילתות למיקומן במערך הממוין. כך נדע איפה התשובות לכל q_i

3. נחשב את תוצאות השאילתות אחת אחרי השניה לפי המיון כשמעבר משאילתא אחת לעוקבת ננהל ונזיז שני מצביעים - L (קצה שמאלי) ו-R (קצה ימני) ונשתמש בחישובי השאילתא הקודמת לטובת הבאה תוך הכנסה והוצאה של נתונים למבנה נתונים נפרד.

← **דוגמא:** שאילתות מהצורה "כמה איברים שונים יש בטווח?"

נניח כי איברי המערך בטווח $1,\dots,n$. נגדיר טבלה שלכל איבר שומרת את מספר ההופעות שלו ברגע נתון. עם כל תזוזה של R ימינה / L שמאלה נגדיל את הערך המתאים במערך ב-1, עם תזוזות הפוכות (R שמאלה / L ימינה) נקטין ב-1. אם ערך מסוים קטן מ-1 ל-0, נקטין את המונה מהשאילתא הקודמת, ואם גדל מ-0 ל-1, נגדיל את המונה.

ביחס למערך ממקודם זה יראה ככה:



11.11.2019

המשך אלגוריתם Mo

← זמן ריצה

m מספר השאילתות

- $O(n+m)$ - מיון מניה
- $O(m)$ - סריקת המערך הממוין
- $O(n\sqrt{n})$ - עלות הזזת R: כאשר L נמצא בבלוק נתון, R זז $O(n)$ צעדים ימינה ואחר כך $O(n)$ צעדים שמאלה כש-L עובר לבלוק הבא.
- $O(n+m\sqrt{n})$ - עלות הזזת L: המעבר של L על פני כל הבלוקים בעלות $O(n)$. כאשר L בבלוק נתון הוא זז $\sqrt{n}$ צעדים.

כל המערך

לכן סה"כ זמן הריצה הוא:

מיון, סריקה, הזזת R, הזזת L, פעולה אטומית

$$O(n+m+m+(n\sqrt{n}+n+m\sqrt{n})t) = O((n+m)\sqrt{n}\,t)$$

4 המרכיבים השונים בסכום טבעי

הטכניקה שימושית באלגוריתמים שזמן ריצתם כולל סכום וניתן לייעל אותם ע"י החלפת סריקת איברי הסדרה בסריקה רק של האיברים הייחודיים בה (שמספרם כמספר האיברים השונים בסדרה)

טענה: בהינתן סדרה $a_1,\dots,a_n$ של מספרים טבעיים, מספר האיברים השונים בה הוא $O\left(\sqrt{\sum_{i=1}^{n} a_i}\right)$.

← **הוכחה:** נסמן ב-t את מספר האיברים השונים ואת סכום הסדרה ב-K. אזי:

$$K=\sum_{i=1}^{n} a_i \geq \sum_{i=1}^{t} a_i \geq \sum_{i=1}^{t} i = \Theta(t^2) \Rightarrow t=O(\sqrt{K})$$

נניח של-t הראשונים שונים

← **דוגמא:** נניח כי נתונה סדרת מטבעות $k_1,\dots,k_n$ ואנו מעוניינים לחשב את מספר הסכומים השונים שניתן להרכיב מהם

נסיון 1 (תכנון דינאמי נאיבי) - נגדיר:

$$F(i,j)=\begin{cases} T & \text{ניתן להגיע לסכום } j \text{ עם } i \text{ המטבעות הראשונים} \\ F & \text{אחרת} \end{cases}$$

ובאופן רקורסיבי:

$$f(i,j)\begin{cases} f(i-1,j) \vee f(i-1,j-k_i) & 0<i \\ T & i=0, j=0 \\ F & i=0, 0<j \end{cases}$$

← **זמן ריצה:** זמן המילוי של טבלה בגודל $O\left(n\cdot \overbrace{\sum_{i=1}^{n} k_i}^{K}\right)$. במקרה הגרוע מדובר ב- $O(K^2)$.

$n \leq K$

המשך המרכיבים השונים בסכום טבעי

<u>ניסיון 2 (מוצלח)</u> - נגדיר מערך A של סוגי המטבעות הנתונים ומערך B

שסופר בתא ה-i את מספר ההופעות של המטבע מסוג A[i]

עבור 1,2,2,7:

$A = \boxed{1}\boxed{2}\boxed{7}$

$B = \boxed{1}\boxed{2}\boxed{1}$

כעת נגדיר פונקציה $f(i,j)$ שמחזירה tuple:

$$f(i,j) = \begin{cases} \boxed{\frac{T}{*}} & \text{ניתן להגיע לסכום } j \text{ עם } i \text{ סוגי המטבעות הראשונים תוך שימוש ב-* מטבעות מהסוג ה-} i \\ \boxed{\frac{F}{*}} & \text{אחרת} \end{cases}$$

ניכר להפריד לכמה מקרים:

$$f(i,j) = \begin{cases} \boxed{\frac{T}{0}} & f(i-1,j) = \boxed{\frac{T}{*}} \\ \boxed{\frac{T}{1}} & f(i-1,j) = \boxed{\frac{F}{*}} \text{ ואם } \boxed{\frac{T}{*}} = f(i-1, j-A[i]) \\ \boxed{\frac{T}{p+1}} & f(i-1,j) = \boxed{\frac{F}{*}} \text{ ו } f(i-1, j-A[i]) = \boxed{\frac{F}{*}} \text{ ו } f(i, j-A[i]) = \boxed{\frac{T}{p}} \quad p<B[i] \\ \boxed{\frac{F}{0}} & \text{שלושת התנאים הקודמים לא מתקיימים} \\ \boxed{\frac{T}{0}} & i=0\ ,\ j=0 \\ \boxed{\frac{F}{0}} & i=0\ ,\ 0<j \end{cases}$$

מקרי בסיס

← זמן ריצה: $O(k\cdot\sqrt{k}) \geq O(k \cdot \#\text{איברים שונים})$.

[וכן עלות הגדרת המערכים A ו-B הדורשים מיון, קרי $O(n\log n)$

והרי $O(k\cdot\sqrt{k}) > O(k\cdot\log k) \geq O(n\log n)$ ולכן עלות זו זניחה]

18.11.2019

פתרון בעיות באלגוריתמים שיעור 4

עץ קטעים

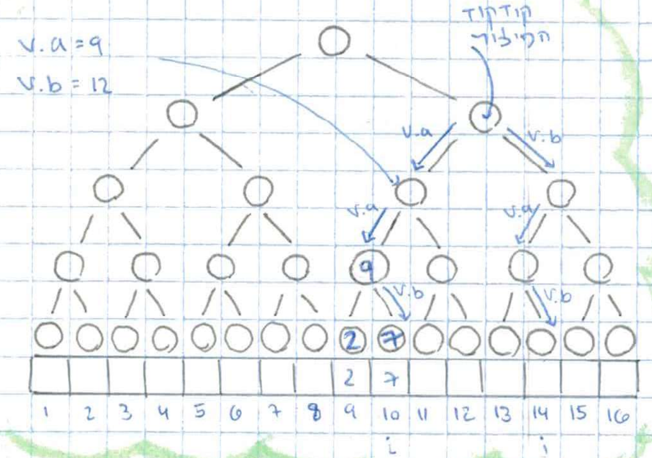
עץ קטעים לשאילתות טווח: מבנה נתונים דינאמי (= תומך בעדכונים) שמייצג סדרת איברים $a_1,\dots,a_n$ ותומך בפעולות:

שאילתת טווח (i,j) - החזר את $f(a_i,\dots,a_j)$ } נדרוש סיבוכיות

עדכון ערך (i,x) - עדכן את האיבר a_i להיות x } $O(\log n)$

← מבנה הנתונים:

בהינתן סדרה $a_1,\dots,a_n$ נגדיר עץ שלם n עלים ($2n+1$ קודקודים) כך שכל קודקוד בעץ "שולט" על קטע רציף במערך.
למשל במקרה של f = תומך נ"ל סכום כל קודקוד יחזיק את סכום האיברים עליו הוא שולט.



דוגמאות: סכום, כפל, הרכבה, GCD, כפל מטריצות

הדרישה היא ש-f מוגדרת לפי אופרטור בינארי אסוציאטיבי $\circ : \tilde{A}\times\tilde{A}\longrightarrow\tilde{A}$

אין דרישה לקומוטטיביות או לקיום איבר הופכי

כלומר $\forall x,y,z: (x\circ y)\circ z = x\circ(y\circ z)$

f מוגדרת לפי אופרטור כזה אם: $f(a_1,\dots,a_n)=a_1\circ a_2\circ\dots\circ a_{n-1}\circ a_n$

← מימוש הפעולות (עבור אופרטור הסכום):

(1) אתחול (A) - בהינתן סדרה $a_1,\dots,a_n$ נגדיר עץ בינארי שלם בעל n עלים (= $2n+1$ קודקודים) באופן הבא:

- כל קודקוד v יחזיק שדות v.parent, v.left, v.right
- בנוסף, הוא יחזיק שדות v.a ו- v.b שמייצגים כי הוא שולט על [v.a, …, v.b].
- לבסוף, כל קודקוד ישמור את הערך שלו בשדה v.value כך ש- $v.val = a_i$ אם $v = leaf_i$, ואחרת $v.val = v.left.val + v.right.val$

← מכאן מקבלים: $v.val = \sum leaf_i.val$

$leaf_i$ עלה בתת העץ ש-v שורשו

בנוסף, נגדיר מערך P כך ש-P[i] מכיל מצביע לעלה ה-i.

(2) עדכון (i,x) - הצב ב- P[i] את הערך x ולכל אב קדמון של P[i] עדכן: $v.val = v.left.val + v.right.val$.

המשך - עץ קטעים לשאילתות טווח

(3) <u>שאילתא (i,j)</u> -

1. נמצא את הקודקוד v הנמוך ביותר בעץ ש- $i,\dots,j$ מוכל בקטע השליטה שלו.
2. נחשב את הסיפא של v.left שמתחילה ב-i
3. נחשב את הרישא של v.right שמסתיימת ב-j

<u>חישוב הסיפא (v)</u> - כל עוד $v \neq null$:

1. אם $v.a = i$, נוסיף את v.val לסכום ונחזיר אותו.
2. אם i נמצא בקטע השליטה של v.left, נוסיף לסכום את v.right.val ונעדכן $v = v.left$
3. אם שני התנאים לא התקיימו, נעדכן $v = v.right$.

← חישוב הרישא עבור v.right סימטרי.

← <u>ניתוח הסיבוכיות</u>:

אתחול בעלות $O(n)$ כהגדרת עץ והרצת DFS עליו.

עדכון בעלות עדכון הערכים של האבות הקדמונים של P[i], כלומר $O(\log n)$ קודקודים בעץ.

שאילתא בעלות מציאת קודקוד הפיצול (חיפוש בינארי) וחישוב הרישא/הסיפא בעלות מס' הרמות בעץ = מס' הקודקודים שאיחוד קטעי השליטה שלהם הוא $i,\dots,j$. ← $O(\log n)$.

<u>עץ קטעים לעדכוני טווח</u> (עץ קטעים דואלי): מבנה נתונים דינאמי התומך ב:

<u>שאילתת אינדקס (i)</u> - החזר את a_i

<u>עדכון טווח (i,j,x)</u> - לכל $i \le k \le j$ מעדכן את a_k להיות $x \circ a_k$.

← <u>מבנה הנתונים</u>:

עבור סכום

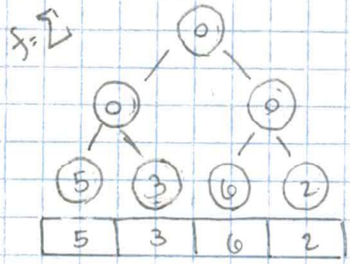
נזכיר כי תכונת עץ הקטעים לשאילתות היא $v.val = \sum leaf_i.val$

$leaf_i$ עלה בתת העץ ש-v שורשו

כלומר, v מחזיק את $f(a_i,\dots,a_j)$ אם הוא שולט על הקטע $i,\dots,j$.

תכונת עץ הקטעים הדואלי היא שהערך a_i מתקבל כ- $v_k.val \circ \dots \circ v_1.val$ כאשר v_k שורש העץ ו- $v_1,\dots,v_k$ הוא המסלול משורש העץ עד לעלה שמתאים ל- a_i.

המוצר - עץ קטעים לעדכון טווח



→ מימוש הפעולות:

(1) אתחול (A) - בדה לאתחול עץ קטעים רגיל למעט כך שלכל עלה $leaf_i$ נגדיר $leaf_i.val = a_i$ ולשאר הקודקודים $v.val = id$ [במקרה של סכום $id=0$]

(2) שאילתא (i) - נמצא את העלה שמתאים ל- a_i ב- P[i] ונחשב את הרכבת הערכים המתאימה עד לשורש.

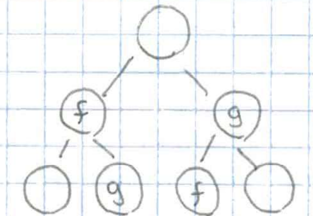
(3) עדכון (x,i,j) -

ניסיון 1: בדה לפעולת השאילתא בעץ קטעים רגיל - מוצאים קודקודים שאיחוד קטעי שריגתם הוא j,...,i ונוסיף לכולם את הערך x

→ תקלה! מוגבל לאופרטורים קומוטטיביים:

עדכון (f,1,3)
עדכון (g,2,4)

עדכון (g,2,4)
עדכון (f,1,3)

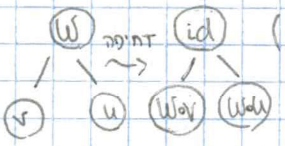


id	id	id	id
f	f	f	
f	$g\circ f$	$f\circ g$	g

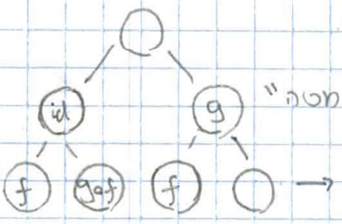
id	id	id	id
	g	g	g
f	$g\circ f$	$f\circ g$	g

ובכלל רצינו לקבל: $f, g\circ f, g\circ f, g$ $\quad f, f\circ g, f\circ g, g$

ניסיון 2: נמצא את קבוצת הקודקודים הרלוונטית כמו בניסיון 1, למעט כך שעם סריקת קודקוד v נבצע פעולת דחף מטה (v)



← כעת עבור העדכון הראשון (קודם f) (קבל אחרי הדחף-מטה)



→ $f, g\circ f, g\circ f, g$

כמו שרצינו.

→ ניתוח הסיבוכיות:

אתחול בדה לאתחול עץ קטעים רגיל - $O(n)$.

שאילתא - כמו עדכון בעץ קטעים רגיל - $O(\log n)$.

עדכון - כמו שאילתא בעץ קטעים רגיל למעט הוספת "דחף מטה" אך מכיוון שבדה ב- $O(1)$, עדיין נשמרת הסיבוכיות של $O(\log n)$.

25.11.2019 שיעור 5

פתרון בעיות באלגוריתמים

פתרון שאלה 1 - תרגיל 1

אלגוריתם 1 $O(n^2 \log n)$

(1) הגדר עץ AVL שממש set והכנס אליו את הנקודות. [לטובת חיפוש הנקודות ב-$\log n$]

(2) מיין את הנקודות לפי קואורדינטות ה-X.

(3) עבור על כל הזוגות שבאותה עמודה - אותו ערך X - $(x_1,y_1),(x_1,y_2)$ ובדוק האם ניתן להשלים לריבוע מימין, כלומר אם $(x_1+|y_1-y_2|, y_1), (x_1+|y_1-y_2|, y_2)$ קיימות. אם כן, הגדל מונה ב-1.

<u>בעיון הנכונות:</u> כל ריבוע רלוונטי אפשר לייצג כזוג נק' עם אותה קואורדינטת X וזוג נוסף מימין להן המשלים לריבוע. האלגוריתם סופר בדיוק את כל הייצוגים הללו ובפרט לא סופר כפילויות.

<u>סיבוכיות:</u> (1), (2) בעלות של $O(n \log n)$ ו-(3) בעלות של לכל היותר מעבר על כל הזוגות - $O(n^2)$ ובדיקת קיום הנקודות המשלימות ב-set בעלות $O(\log n)$. לפיכך זמן הריצה הכולל הוא $O(n^2 \log n)$.

אלגוריתם 2 $O(n\sqrt{n} \log n)$

(1) הגדר עץ AVL כמו באלגוריתם 1, מיין לפי קואורדינטה X. בנוסף, לכל נקודה הוסף שדה שערכו TRUE $\iff$ היא בעמודה "קצרה", כלומר בעמודה בה יש פחות מ-k נקודות.

(2) לכל זוג נקודות בעמודה קצרה, בדוק אם קיימות הנקודות המשלימות לריבוע (גם מימין וגם משמאל), ואלא אם המשלימות הן על עמודה קצרה משמאל, הוסף 1 למונה (אם הן אכן קיימות).

(3) נמחק את כל העמודות הקצרות ממבנה הנתונים. כל עמודה ארוכה - אורכה גדול מ-k. נבחין כי יש לכל היותר $\frac{n}{k}$ עמודות ארוכות. לפיכך נקבל כי אורך כל שורה מהשאריות הוא לכל היותר $\frac{n}{k}$.

החל משלב המיון כולל

(4) נתרגם כל נק' (x,y) ל-(y,x) ונפעיל את אלגוריתם 1 על העמודות. (נשים לב שכעת נספור רק השלמה מימין כי כל העמודות קצרות).

<u>בעיון הנכונות:</u> כל ריבוע שמקיים שלפחות צלע אחת שלו על עמודה קצרה נספר בשלב (2), תוך כדי שמנענו ספירת כפילויות. יתר הריבועים - שתי צלעותיהם על עמודות ארוכות והם יספרו בשלב (4).

המשך אלגוריתם 2

s_i הוא אורך העמודה הקצרה ה-i

סיבוכיות: עבור עמודות קצרות - $O(\sum_i s_i^2) \leq O(k\cdot\sum_i s_i) \leq O(n\cdot k)$

d_i הוא אורך השורה ה-i מבין השארית - לפני תרגום עמודות ארוכות

ועבור עמודות ארוכות - $O(\sum_i d_i^2) \leq O(\frac{n}{k}\sum_i d_i) \leq O(\frac{n^2}{k})$

בשני המקרים נכפול את הביטויים ב-$\log n$ - עלות חיפוש הנקודות המשלימות ונקבל

$$O(\underbrace{n\log n}_{\text{מיון ואתחול}} + \underbrace{n\cdot k\cdot\log n}_{\text{עמודות קצרות}} + \underbrace{\frac{n^2}{k}\cdot\log n}_{\text{עמודות ארוכות}})$$

← נמצא k אופטימלי: $n\cdot k\log n = \frac{n^2}{k}\log n \Rightarrow n\cdot k = \frac{n^2}{k} \Rightarrow k=\sqrt{n}$

לכן זמן הריצה הכולל הוא $O(n\sqrt{n}\log n)$.

אלגוריתם 2' - הערכה $O(n\sqrt{n}\cdot\sqrt{\log n})$

נוכל לשכלל את המעבר על העמודות הארוכות בכך שנסרוק כל זוג עמודות כאלה - x,x' - עם שני מצביעים כאשר בכל פעם נקדם רק את הנמוך מביניהם. אם הגענו לערך y זהה, נזכור אותו. בכל פעם שנוסיף ערך y חדש - y' - נבדוק האם כבר שמרנו ערך y כך ש- $|x-x'|=|y-y'|$ (כלומר השלמנו ריבוע). אם כן, נמחק את הנקודה הנמוכה יותר ונגדיל את המונה.

הסיבוכיות:

$$\overbrace{\frac{n}{k}\cdot n}^{\text{עמודות ארוכות}} + \overbrace{k\cdot n\cdot\log n}^{\text{עמודות קצרות}} \Rightarrow k^2 = \frac{n^2}{n\cdot\log n} = \frac{n}{\log n} \Rightarrow k=\sqrt{\frac{n}{\log n}}$$

אלגוריתם 3 $O(n\sqrt{n})$

רעיון כללי: אילו כל הנקודות היו בטווח $[1,\dots,n]\times[1,\dots,n]$ היינו יכולים להגדיר לכל זוג נקודות n שאילתות - כלומר n זוגות פוטנציאלים ולבדוק אם אחת השאילתות קיבלה ערך TRUE. אם נוכל לענות על השאילתות ביעילות (n שאילתות ב-$O(n)$) נצליח להחזיר פתרון ב-$O(n\sqrt{n})$.

← תחת ההנחה הזו נוכל לפעול באופן הבא:

(1) נמיין את הנק' ע"פ ערך ה-x ונשמור לכל נקודה אם היא בעמודה קצרה או ארוכה.
(כל נק' הקלט)

(2) לכל זוג נק' שנמצאות בעמודה קצרה נכניס למערך את כל הנקודות מימין ומשמאל
($2n$ נקודות השאילתא)

(3) לכל מקבץ של n שאילתות נוסיף למערך את n הזוגות הפוטנציאליים שעלולים להשלים לריבוע.

(4) נבצע counting sort ב-$O(n)$ על המערך המעורב של נק' ושאילתות.

אם מצאנו כפילויות, סימן שקיימת נקודות שעונות על שאילתא כלשהי.

(5) אם עבור שאילתא מסוימת שתי הנקודות סומנו כקיימות - נגדיל את המונה ב-1

← נחשוב גם האם נק' השאילתא היא על עמודה קצרה או ארוכה כדי לדעת האם לספור את הריבוע או לא (עמודה קצרה משמאל)

המשך אלגוריתם 3

← דוגמא

הדוגמא מסתכלת רק על
השלמה מימין - היינו צריכים
להסתכל לשני הכיוונים ולספור
בהתאם

$P : [(1,1),(1,2),(1,3),(2,1),(2,2)]$

$Q_1 : [(2,1),(2,2),(2,2),(2,3),(3,1),(3,3)]$

המערך הממויין $\Rightarrow [(1,1),(1,2),(1,3),(2,1)^{q_1},(2,1),(2,2)^{q_1},(2,2)^{q_2},(2,2),(2,3)^{q_2},(3,1)^{q_3},(3,3)^{q_3}]$

נספור ונתייג								
	1	1	1	2	3	1	1	1
	F	F	F	T	T	F	F	F

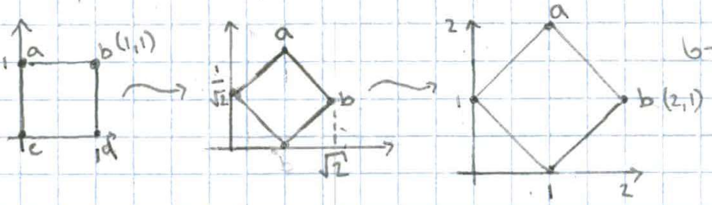
נבדוק את השאילתות $\Rightarrow q_1 = TT \ , \ q_2 = TF \ , \ q_3 = FF \Rightarrow \#\text{squares} = 1$

<u>האלגוריתם במקרה שלנו</u> (איך נתגבר על כך שהערכים שלנו לא בטווח)

(1) לכל נקודה (x,y) נגדיר את התווית $(x, y, x+y, y-x)$

← הערה - למה דווקא $x-y$, $y+x$?

באלגוריתמים שקשורים למרחקי מנהטן על GRID, סיבוב של ה-GRID ב-$45°$ והכפלה בפקטור $\sqrt{2}$ מאפשרים לנו לקבל באמצעות מידע על קואורדינטה אחת מידע על אחרת, ובפרט מתקיים שקואורדינטת ה-x של b לאחר הטרנספורמציה



היא הסכום של שתי הקואורדינטות שלו לפני הסיבוב. ככה רק מידיעת את a ו-c אנחנו יכולים להסיק את המידע שאנחנו צריכים על b ו-d. [ואיזה מידע נקבל ביחידות]

(2) ניתן לקואורדינטות ה-x שמות בטווח n-1 וכך גם לקואורדינטות ה-y ולזוגות $(x+y)$ ולזוגות $(y-x)$.

(3) נשפר את אלגוריתם 2, כשההבדל הוא בחישוב השאילתות:

לכל זוג נקודות (a,b,c,d) (a',b',c',d') [נניח שאלו השמות החדשים שנתנו להן]

נגדיר את השאילתות $((*,b,c',*),(*,b',*,d))$, $((*,b,*,d'),(*,b',c,*))$

המתאימות לזוגות הנקודות המשלימות משמאל ומימין בהתאמה.

בכל איטרציה נגדיר n שאילתות שעליהן נוכל לענות ב-$O(n)$ {בדומה לרעיון שהוצג בעמ' הקודם}

← <u>הערה</u>: נוסיף סימונים לזיהוי עמודות קצרות/ארוכות ונוודא שכל ריבוע נספר רק פעם אחת.

רכיבי דו-קשירות בגרף לא מכוון

הגדרה - גשר: בהינתן גרף $G=\langle V,E\rangle$, צלע $e=\{u,v\}$ תיקרא "גשר"
בגרף אם הסרתה מהגרף מובילה לכך שאין מסלול מ-u ל-v.
הסרת גשר מגדילה את מספר רכיבי הקשירות בגרף.

טענה (הגדרה אלטרנטיבית לגשר): צלע $e=\{u,v\}$ היא <u>לא</u> גשר
אמ"מ היא חלק ממעגל בו.

← <u>הוכחה:</u>

אם e היא חלק ממעגל, הסרתה מהגרף לא גורמת לכך שאין מסלול
מ-u ל-v $\Leftarrow$ e איננה גשר

נניח כי e איננה גשר. נסיר אותה מהגרף ונקבל כי עדיין יש מסלול כלשהו
מ-u ל-v. נוסיף למסלול את e ונקבל מעגל. ∎

02.12.2019

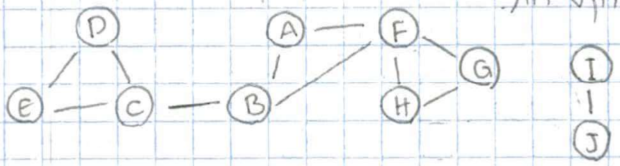
פתרון בעיות באלגוריתמים שיעור 6

רכיבי הדו-קשירות בגרף לא מכוון

תזכורת מהשיעור הקודם:

גשר בגרף לא מכוון היא צלע $e=\{u,v\}$ שהסרתה מהגרף גורמת לכך שאין מסלול בין u ל-v בו. במילים אחרות, הסרתה מגדילה ב-1 את מספר רכיבי הקשירות.

← דוגמא:



$\{B,C\}$ ו-$\{I,J\}$ גשרים.

← טענה (הגדרה אלטרנטיבית): צלע $e=\{u,v\}$ בגרף לא מכוון אינה גשר $\Longleftrightarrow$ היא חלק ממעגל.

[הוכחנו בשיעור הקודם]

בהינתן $G=\langle V,E\rangle$ גרף לא מכוון, נגדיר את $G_{B1}=\langle V,E_{B1}\rangle$ להיות הגרף שצלעותיו הן $E_{B1}=\{e\in E \mid \text{e אינה גשר ב-G}\}$

טענה: ב-G_{B1} אין גשרים.

← הוכחה: תהי $e\in E_{B1}$. לפי הגדרה e אינה גשר ב-G ולכן לפי האיפיון האלטרנטיבי לגשר, e חלק ממעגל ב-G. מכיוון שכל הצלעות במעגל אינן גשרים, המעגל נמצא גם ב-G_{B1}. מכאן ש-e אינה גשר ב-G_{B1}.

רכיבי הקשירות ב-G_{B1} יקראו רכיבי הדו קשירות של הגרף G. בהינתן u ו-v ב-G נאמר כי הם דו-קשורים אם הם באותו רכיב קשירות ב-G_{B1}.

משפט (מובא ללא הוכחה): בהינתן $G=\langle V,E\rangle$ ו-$u,v\in V$, התנאים הבאים שקולים:

(1) u ו-v דו-קשורים ב-G

(2) יש 2 מסלולים זרים בצלעות בין u ל-v

(3) u ו-v הם חלק ממעגל ב-G

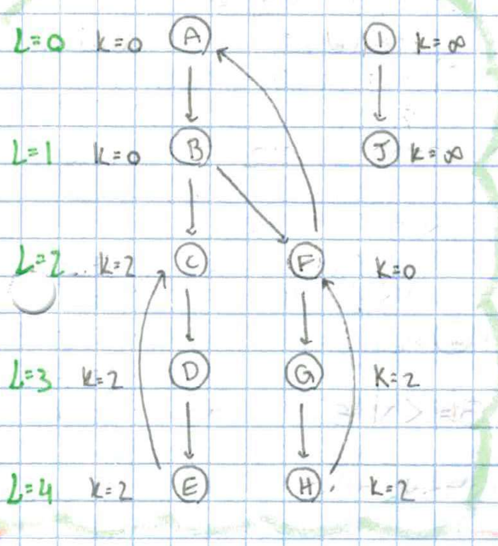
מציאת רכיבי הדו-קשירות בגרף לא מכוון: נתאר אלגוריתם שמוצא את הגשרים ב-G. בהינתן מציאת הגשרים נוכל למצוא את רכיבי הדו-קשירות ביעילות, בזמן לינארי בקלט, תוך הרצת DFS על G_{B1} שמתקבל ממחיקת הגשרים, ומציאת רכיבי הקשירות של G_{B1}.

[המשך - מציאת רכיבי דו-קשירות]

ניסיון 1 (נאיבי): לכל $e=\{u,v\}$, נסיר את e מ-G ונבדוק אם יש מסלול בין u ל-v

← זמן ריצה: $O(|E|(|V|+|E|))$ - ריבועי בקירוב

נרצה למצוא פתרון שמבצע ריצה בודדת של DFS (ולא $|E|$ כאלה) למציאת כל הגשרים



הבחנה: מספר הגשרים הוא לכל היותר מספר צלעות העץ שכן צלעות אחוריות הן תמיד חלק ממעגל.

הגדרות: בריצת DFS על G, צלע $e=\{u,v\}$ שנסרקת בהרצה ומובילה מ-u ל-v, כאשר v קודקוד שלא נסרק קודם, תקרא עץ. ↓

יתר הקשתות יקראו קשתות אחוריות ↑

לכל קודקוד $v\in V$ נגדיר את הערך $k(v)$ ע"י

$$k(v)=\min\{L(w') \mid \text{w צאצא של v בגרף ה-DFS ו-(w,w') קשת אחורית}\}$$

כאשר $L(w')$ היא הרמה של w' בגרף ה-DFS.

טענה: לכל $G=\langle V,E\rangle$ גרף לא מכוון, נתונן בסריקת DFS כלשהי של G. תהי $e=\{u,v\}$ צלע ב-G שהיא קשת עץ בגרף ה-DFS. נניח כי u אב של v בגרף זה, אז e איננה גשר $\iff k(v)<L(v)$

← הוכחה: $(\Rightarrow)$ נניח כי $k(v)<L(v)$. לפיכך מתקיים גם $k(v)\le L(u)$. $k(v)<L(v)$ רק אם קיים ל-v צאצא w שמוביל בקשת אחורית לאב קדמון w' של u. לפיכך יש מסלול $\overbrace{w \rightsquigarrow v \to u \rightsquigarrow w'}$ ובפרט $e=\{u,v\}$ היא חלק ממעגל. מכאן נובע ש-e אינה גשר.

$(\Leftarrow)$ נניח כי $L(v)\le k(v)$, כלומר מכל הצאצאים של v ניתן להגיע בקשתות אחוריות רק לקודקודים נמוכים מ-v ובפרט לא לאבות קדמונים של u. מכאן שהסרת e מהגרף גורמת לכך שאין מסלול מ-v ל-u $\Leftarrow$ e גשר. ∎

נבחין:
$$k(v)=\min\left\{\left\{k(v') \mid \text{v' בן של v}\right\}\cup\left\{L(w') \mid \text{\{v,w'\} קשת אחורית}\right\}\right\}$$

← בעזרת הנוסחה הרקורסיבית נוכל לבנות פתרון יעיל

02.12.2019

המשך - מציאת רכיבי דו-קשירות

<u>אלגוריתם 2:</u> בצע סריקת DFS של הגרף תוך סימון רמות כל הקודקודים בו וחישוב ערכי ה-k של הקודקודים בהינתן הנוסחא הרקורסיבית.
נשווה את $k(v)$ ל-$L(v)$ לכל קודקוד ונאפיין את הגשרים לפי הטענה הקודמת.

← <u>זמן ריצה:</u> $O(|E|+|V|)$ [עלות הרצת DFS]

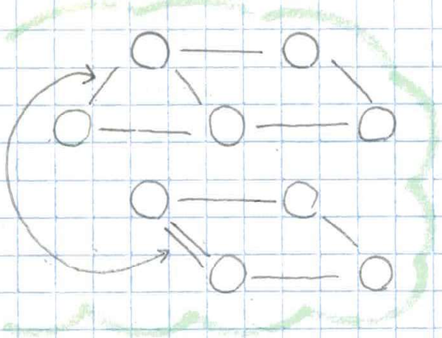
← <u>הערה:</u> ניתן גם להגדיר רכיבי דו-קשירות ע"י הגדרת קודקודים מפרידים.
רעיון דומה לרעיון הנ"ל מוצא קודקודים כאלה בזמן לינארי.

<u>מציאת חתך מינימלי בגרף</u>

<u>הגדרה:</u> מולטי-גרף $G=\langle V,E\rangle$ זהה בהגדרתו לגרף למעט כך שייתכנו כמה צלעות בין זוג קודקודים נתון (E היא מולטי-קבוצה)

<u>הגדרה:</u> חתך בגרף הוא קבוצת צלעות שהסרתן מגדילה את מספר רכיבי הקשירות בו.

נרצה לתאר אלגוריתם הסתברותי שמוצא חתך במולטי-גרף בהסתברות גבוהה.



<u>האלגוריתם:</u>

(1) עד להיוותרות שני קודקודים בגרף:
א. הגרל צלע בגרף בהתפלגות אחידה
ב. חבר את שני הקודקודים בצלע זו לקודקוד יחיד שצלעותיו הן כל הצלעות היוצאות משניהם, פרט לצלעות ביניהם

(2) החזר את מספר הצלעות ביניהם.

<u>הוכחה:</u>

<u>חוקיות:</u> נניח כי נותרו הקודקודים v_S ו-$v_{S'}$. הצלעות ביניהם הן הצלעות שבין הקודקודים מ-S לקודקודים מ-S', ואלו הצלעות היחידות ביניהם.
לכן צלעות אלה מהוות חתך ב-G

<u>אופטימליות:</u> נתבונן בחתך מינימלי C ב-G. נסמן $|C|=k$. נבחין כי: $|E|=\frac{1}{2}\sum_v deg(v) \geq \frac{n\cdot k}{2}$. C מינימלי ולכן לכל v, $k\leq deg(v)$, שכן אם הייתה קטנה מ-k, ניתן היה לקבל חתך קטן יותר ע"י הסרת הצלעות של v

09.12.2019 שיעור 7

פתרון בעיות באלגוריתמים

מציאת חתך מינימלי בגרף

תזכורת: בשיעור הקודם ראינו אלגוריתם הסתברותי למציאת חתך מינימלי בגרף שבכל איטרציה מגריל צלע ומחבר את שני הקודקודים בה לקודקוד יחיד שצלעותיו הן כל הצלעות שיוצאות משני הקודקודים שאיחדנו פרט לצלעות ביניהם. כשנשארים 2 קודקודים הגרף עוצר ומחזיר את מס' הצלעות ביניהם. הוכחנו חוקיות בשיעור שעבר (שבאמת חוזר חתך) וניתר להראות שהחתך אכן מינימלי בהסתברות גבוהה.

אופטימליות (בהסת' גבוהה): נסמן ב-C חתך מינימלי בגרף שגודלו k. נסמן ב-A_i את המאורע לפיו באיטרציה ה-i לא נבחרה צלע מ-C.

אנו מעוניינים לדעת את ההסתברות שהחזרנו חתך מינימלי, כלומר:

$$\mathbb{P}\begin{pmatrix}\text{הוחזר חתך}\\ \text{מינימלי בגרף}\end{pmatrix} \leq \mathbb{P}\begin{pmatrix}\text{לא הוסרה אף}\\ \text{צלע מ-C}\end{pmatrix} = \mathbb{P}\left(\bigcap_{i=1}^{n-2} A_i\right) = \mathbb{P}(A_1)\cdot\mathbb{P}(A_2|A_1)\cdot\mathbb{P}(A_3|A_1,A_2)\dots$$

נחשב בעבור שתי האיטרציות הראשונות ומשם נקבל את היתר באופן אינדוקטיבי.

מתקיים:

$$\# \; \underbrace{\mathbb{P}(\bar{A}_1)}_{\text{k צלעות ב-C שנבחרו בהסת' אחידה}} = \frac{k}{|E|} \leq \frac{k}{\frac{1}{2}nk} = \frac{2}{n} \implies \mathbb{P}(A_1) \geq 1-\frac{2}{n}$$

$$|E| = \frac{1}{2}\sum_v \deg(v) \geq \frac{1}{2}nk \qquad |V|=n$$

$$\# \; \mathbb{P}(\bar{A}_2|A_1) \leq \frac{k}{\frac{1}{2}k(n-1)} = \frac{2}{n-1} \implies \mathbb{P}(A_2|A_1) \geq 1-\frac{2}{n-1}$$

ולכן אינדוקטיבית:

$$\# \; \mathbb{P}\left(\bigcap_{i=1}^{n-2} A_i\right) \geq \prod_{i=1}^{n-2}\left(1-\frac{2}{n-i+1}\right) = \prod_{i=1}^{n-2}\frac{n-i-1}{n-i+1} = \frac{n-2}{n}\cdot\frac{n-3}{n-1}\cdot\ldots\cdot\frac{1}{3}$$

$$= \frac{(n-2)!}{\frac{n!}{2}} = \frac{2}{n\cdot(n-1)} \geq \frac{2}{n^2}$$

זו כמובן לא הסתברות מספיק טובה אבר מה שנוכל לעשות זה להריץ את האלגוריתם $\frac{n^2}{2}$ פעמים ב"ת. ההסתברות שבכל ההרצות התקבלה טעות היא לכל היותר: $\left(1-\frac{2}{n^2}\right)^{\frac{n^2}{2}} < \frac{1}{e}$

נמשיך להכפיל בקבועים ונקבל הסתברות מעולה.

כלומר האלגוריתם צודק בהסתברות $\geq 1-\frac{1}{e}$ וזה כבר סבבה. ∎

אנטי-דוגמא: נראה כעת אלגוריתם דומה ברעיון שלא מחזיר את הקוצאה.

← במקום לבחור צלע בהסת' אחידה נבחר זוג קודקודים כשהם בהסת' אחידה ונאחד ביניהם.

המשך - אנטי-דוגמא

נתבונן בגרף בעל n קודקודים בו 2 קליקות זרות בגודל $\frac{n}{2}$ וצלע יחידה שמחברת בין הקליקות.  [יש פה גרפים כאלה]

כמובן שחתך מינימלי בגרף זה הוא 1, אבל בכל מקרה שבו שני הקודקודים שנגריל יהיו משתי קליקות שונות, כשנאחד אותם נחזיר חתך גדול יותר.

נחשב את ההסתברות שהאלג' מחזיר פתרון אופטימלי:

נסמן ב-A_i את המאורע דלא נבחרו באיטרציה ה-i שני קודקודים "בעייתיים"

$$\mathbb{P}\left(\bigcap_{i=1}^{\frac{n}{100}} A_i\right) = \mathbb{P}(A_1)\cdot\mathbb{P}(A_2|A_1)\cdot\mathbb{P}(A_3|A_1,A_2)\dots$$

מתקיים:

$$\mathbb{P}(A_{i+1}|A_1,\dots,A_i) \le \frac{\binom{\frac{n}{2}}{2}+\binom{\frac{n}{2}}{2}}{\binom{n-i}{2}} \underset{i\le\frac{n}{100}}{\le} \frac{2\binom{\frac{n}{2}}{2}}{\binom{\frac{99n}{100}}{2}} \le \frac{4\binom{\frac{n}{2}}{2}}{\left(\frac{99n}{100}-1\right)^2}$$

$$\left[\frac{99n}{100}-1 \ge \frac{9n}{10} \iff 99n-100 \ge 90n \iff 9n \ge 100 \implies n \ge 12\right] \underset{12\le n}{\le} \frac{\frac{n^2}{2}}{\left(\frac{9n}{10}\right)^2} = \frac{\frac{1}{2}}{\frac{81}{100}} = \frac{50}{81}$$

מכאן:

$$\mathbb{P}\left(\bigcap_{i=1}^{n-2} A_i\right) \le \mathbb{P}\left(\bigcap_{i=1}^{\frac{n}{100}} A_i\right) \le \left(\frac{50}{81}\right)^{\frac{n}{100}} \underset{0<q<1}{=} q^n = \frac{1}{2^{\Omega(n)}}$$

אכן הסתברות גרועה. ∎

פתרון שאלה 1- תרגיל 2

<u>קלט:</u> סדרת הריצות V_{final}, V_{max}

<u>פלט:</u> t ו- V_{init} מקס' מר"ם

<u>הבחנות:</u>

1. $V_{final}-n \le V_{init} \le V_{final}+n$ כי יש בדיוק n ריצות

2. זמני ההשהיה t הרלוונטים לבדיקה הם בדיוק אלו שמתקבלים מסדרת הפרשי הזמנים של הריצות.

כלומר אם t מסוים בסדרה זו לא מניב פתרון חוקי, אך הערך שלפניו בסדרה מניב כזה (והוא הראשון שעבורו זה מתקיים), הערך המתאים הוא ה-t.

<u>אלגוריתם 1</u> $O(n^3)$

(1) מיין את סדרת הפרשי זמני הריצות החל מ 1 ושמור את התוצאה במערך A יחד עם הערך ∞.

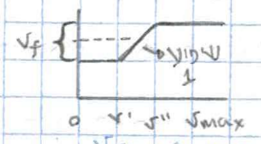
המשך - אלגוריתם 1

(2) לכל i מ-|A| עד 1, לכל j מ-(v_f+n) ל-(v_f-n) הרץ את המכשיר עם $v_{init}=j$, $t=A[i]$ ואם התקבל v_f החזר $(A[i], j)$.

<u>זמן ריצה</u>: הגדרת A בעלות $O(n\log n)$ ושלב (2) בעלות $O(n^3)$.

<u>אלגוריתם 2</u> $O(n^2)$

<u>הבחנה</u>: בהינתן t_0 קבוע נגדיר $f_{t_0}(v)$ להיות הווליום שמתקבל עם $t=t_0$ ו-$v_{init}=v$. נבחין כי f נראית:



כלומר מהרגע שהגענו ל-v_{max} הפונקציה תראה אותו דבר לכל v_{init}

ציר ה-x הוא v_{init} וציר ה-y הוא v_{final}

כלומר, כדי לבדוק אם ניתן להגיע ל-v_f נבדוק לכל t אם

$$(*)\quad f_t(0)\le v_f\le f_t(v_{max})$$

<u>האלגוריתם</u>:

(1) מיין את סדרת הפרשי זמני הלחיצות פחות 1 ושמור את התוצאה ב-A יחד עם הערך ∞

(2) לכל i מ-|A| עד 1 נבדוק אם (*) מתקיים ואם כן נחזיר את v_{init} המתאים, למשל באמצעות חיפוש בינארי.

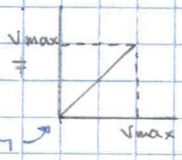
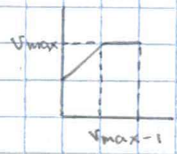
<u>זמן ריצה</u>: שלב (1) בעלות $O(n\log n)$, שלב (2) בעלות $O(n^2)$ ומציאת v_{init} המתאים - פעם אחת בלבד - בעלות $O(n\log n)$

<u>אלגוריתם 3</u> $O(n\log n)$

<u>הבחנות</u>:

1. באופן דומה להגדרת $f_t(v)$ נגדיר $f_t^{(i,j)}(v)$ כווליום שמתקבל עבור תת הסדרה בין i ל-j. לדוגמא, בעבור לחיצת (+):

סדרה באורך 1

$f_t^{(i,i)}(v)=$  או 

לחיצה לא אפקטיבית = פחות.

לחיצת + אפקטי,

גם כאן, ציר ה-x הוא v_i וה-y v_f

2. נבחין כי לכל t,i,j,k מתקיים:

$$f_t^{(k+1,j)}(v)\circ f_t^{(i,k)}(v)=f_t^{(i,j)}(v)\Rightarrow f_t^{(n,n)}(v)\circ\dots\circ f_t^{(1,1)}(v)=f_t^{(n,1)}(v)=f_t(v)$$

$\Leftarrow$ קל לראות שזהו אופרטור אסוציאטיבי שמתאים לשימוש עם עץ קטעים!

המשך- אלגוריתם 3

האלגוריתם:

(1) מיין את סדרת הפרשי זמני ההתחלות תחת 1 ושמור את התוצאה ב-A יחד עם הערך ∞

(2) נגדיר מערך B שאיבריו הם: $\boxed{f_\infty^{(1,1)}(v) \mid \cdots \mid f_\infty^{(n,n)}(v)}$

(3) נגדיר עץ קטעים C מעל המערך B

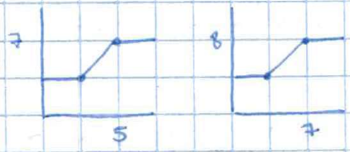
(4) לכל i מ-$|A|$ עד 1: נבדוק אם $f_{A[i]}(v)$ מקיים את (*). אם כן, נמצא את v_{init} המתאים עם חיפוש בינארי.

אחרת, נעדכן את כל הערכים בסדרה הרלוונטים ל-$A[i-1]$ כך שיכילו את הערך המתאים.

<u>זמן ריצה</u>: שלב (1) בעלות $O(n \log n)$, שלבים (2) ו-(3) בעלות $O(n)$ כל אחד. שלב (4) עדכון $\leq n$ ערכים ב-$O(\log n)$ לכל ערך. לאחר מציאת t מתאים, מציאת v_{init} בעלות $O(\log n)$ [כי הפונקציות נתונות]

הערה: כל פונקציה $f_t^{(i,j)}(v)$ ניתנת לייצג באמצעות שתי הנקודות של קצוות השיפוע שלה, וחישוב ההרכבה בעלות $O(1)$ כמספר המקרים הרלוונטים לבדיקה.

חישוב ההרכבה ב-$O(1)$ [העשרה]



$f_1(v)$ $f_2(v)$ $f_2 \circ f_1(5) = 8$

#במקרים עבורם f_1 קבועה $\Leftarrow$ ההרכבה תהיה קבועה, וזה מוגדר לפי נקודות קצוות השיפוע

#ב-v-ים כך ש-$f_1(v)$ מניבה ערכים בהם f_2 קבועה, ההרכבה תהיה קבועה

= נוכל למצוא את f_t^i באמצעות שיקוף נקודות קצוות השיפוע

23.12.2019 שיעור 8

פתרון בעיות באלגוריתמים

גיאומטריה חישובית

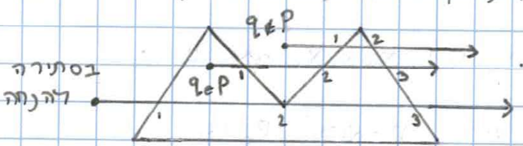
התחום עוסק בקשר בין אובייקטים במרחב או במישור: קטעים, נקודות, מצולעים, כדורים...
פעמים רבות, בעיות שנראות קלות לתיאור ידרשו הבחנות מתמטיות עדינות שיניבו פתרון / הוכחה עבורן.

דוגמא: בהינתן מצולע P במישור הנתון כסדרת נקודות $p_1,\dots,p_n$ כך שצלעותיו $[p_1,p_2],\dots,[p_{n-1},p_n],[p_n,p_1]$ ונניח כי המצולע פשוט (צלעותיו לא נחתכות), מעוניינים לדעת אם $q\in P$ או שלא. (בהינתן נקודה q, האם היא בתוך המצולע)

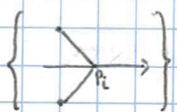
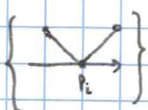
הנחה: $q.y \neq p_i.y$ $\forall i\in[n]$ (ערך ה-y שונה)

האלגוריתם: נעביר קרן מ-q, מקבילה לציר ה-x ובכיוון החיובי. אם חצתה את צלעות המצולע מס' אי-זוגי של פעמים (או ש-q על צלע של P), נחזיר כי $q\in P$, אחרת $q\notin P$

נכונות: נניח כי $q\in P$. כל חציה של צלע המצולע לאורך הקרן משנה את מצבה של הנקודה, כלומר מהפנים לחוץ או להיפך (נובע מכך שהקרן לא חוצה את המצולע בנקודות בהן צלעותיו נחתכות ← בגלל ההנחה על y)
מכיוון שבסיום התהליך הנקודה מחוץ למצולע, נסיק כי מספר החציות היה אי-זוגי.
אם $q\notin P$, באופן סימטרי מספר החציות זוגי.



זמן ריצה: בדיקה אם q על צלע של P בעלות $O(1)$ לכל צלע $\Leftarrow O(n)$.
בדיקת מס' החציות בעלות בדיקת חיתוך הקרן עם $n-1$ צלעות, כל בדיקה בעלות $O(1) \Leftarrow O(n)$. לכן סה"כ $O(n)$.

התייחסות להנחה: אם ההנחה לא מתקיימת, בכל פעם שנחצה קודקוד p_i של P נבדוק אם p_{i-1}, p_{i+1} אחת משמאל ואחת מימין ל-p_i  נספור חציה זו (או לא נספור פעמיים) ואחרת  לא נספור (או נספור פעמיים)

השוואת אוריינטציה ובדיקת כיוון פניה

בהינתן אוסף של n נקודות ו-3 מהן: p_1,p_2,p_3, מעוניינים לדעת אם p_3 משמאל ל-p_2 ביחס ל-p_1 (ואז נחזיר 1), מימין לה (ואז נחזיר -1) או ללא שינוי כיוון ההסתכלות (ואז נחזיר 0).
כלומר האם ההילוך $p_1\to p_2\to p_3$ הוא עם פניה שמאלה/ימינה/ללא פניה

המשך השוואת אוריינטציה

← האלגוריתם: החזר את: $\text{sign}\left(\det\begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{bmatrix}\right)$

← זמן ריצה: $O(1)$ בעבור מס' קבוע של פעולות אלמנטריות

← נכונות: נגדיר את הישר שעובר ב- $[P_1,P_2]$ - $y=mx+n$ כאשר

$$m=\frac{y_2-y_1}{x_2-x_1} \quad ; \quad n=y_1-\frac{y_2-y_1}{x_2-x_1}\cdot x_1$$

מתקיים:

אם $x_1<x_2$, P_3 משמאל ל- P_2 כש- $y_3>mx_3+n$, ולכן:

$x_1<x_2$ ולכן לא הופכים את הסימן

$$y_3>mx_3+n \iff y_3>\frac{y_2-y_1}{x_2-x_1}\cdot x_3+y_1-\frac{y_2-y_1}{x_2-x_1}\cdot x_1 \iff$$

$$y_3(x_2-x_1)>x_3(y_2-y_1)+y_1(x_2-x_1)-x_1(y_2-y_1) \iff$$

$$y_3(x_2-x_1)-y_1(x_2-x_1)+x_1(y_2-y_1)-x_3(y_2-y_1)>0$$

$$\underbrace{(x_2-x_1)(y_3-y_1)-(x_3-x_1)(y_2-y_1)}_{\circledast}>0$$

ואם נסתכל על הדטרמיננטה:

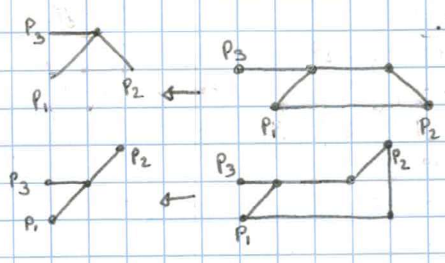
$$\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}=\begin{vmatrix} x_1 & y_1 & 1 \\ x_2-x_1 & y_2-y_1 & 0 \\ x_3-x_1 & y_3-y_1 & 0 \end{vmatrix}=\begin{vmatrix} x_2-x_1 & y_2-y_1 \\ x_3-x_1 & y_3-y_1 \end{vmatrix}=\circledast$$

הסימן של הדט' מתאים גם פה

אם $x_2<x_1$ במעבר השני צריך להפוך את הסימן ומכיוון ש-P_3 משמאל ל-P_2 כש- $y_3<mx_3+n$ כשנכפול ב-(x_2-x_1) ונהפוך את הסימן נקבל את אותו פתרון

* באופן דומה ניתן לנתח את המקרה $x=k$ ונקבל תוצאה דומה.

* בבעיית ההימצאות של נק'



המוטיבציה לשימוש בדטרמיננטה:

① הפעלת טרנספורמציה לינארית על הנקודות תשפיע על האלגוריתם בתיוג הדטרמיננטה של מכפלת המטריצות.

② הדטרמיננטה בערך מוחלט היא פעמיים שטח המשולש $\Delta P_1P_2P_3$

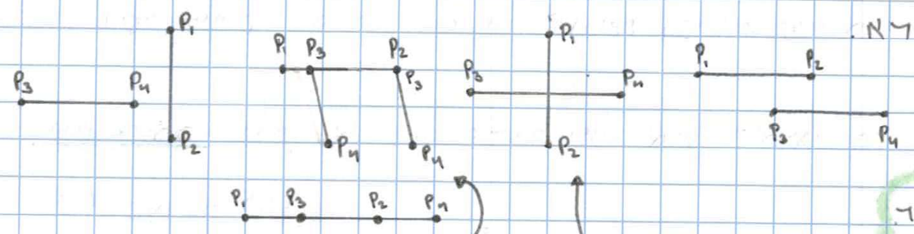
③ ניתן להכליל באמצעות השימוש בדטרמיננטה. למשל סימן הדטרמיננטה הבאה קובע האם P_4 בעיגול המוגדר ע"י P_1,P_2,P_3:

$$\begin{bmatrix} x_1 & y_1 & x_1^2+y_1^2 & 1 \\ x_2 & y_2 & x_2^2+y_2^2 & 1 \\ x_3 & y_3 & x_3^2+y_3^2 & 1 \\ x_4 & y_4 & x_4^2+y_4^2 & 1 \end{bmatrix}$$

המשך השוואת אוריינטציה

<u>תרגיל</u>: בהינתן אוסף של n נקודות ומהן $p_1, \dots, p_4$ מעוניינים לקבוע אם $[p_3, p_4]$, $[p_1, p_2]$ נחתכים או לא.

← מקרים מעניינים:



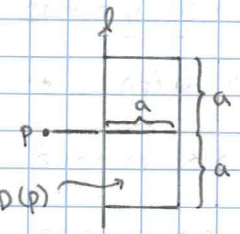
את המקרים הללו ניתן לבדוק ע"י בדיקת הימצאות נקודה על קטע ב-$O(1)$ באמצעות השוואת הישר

נוכל לבדוק באמצעות השוואה של כיווני פניה - האם p_4 מימין ל-p_2 ו-p_3 משמאל ל-p_2 ביחס ל-p_1? וכך גם עבור הקטע השני.

ניתן לפתור את התרגיל ב-$O(1)$ כפי שלמדנו בבדיקת חיתוך קרן עם קטע בבעיה הראשונה בשיעור

מציאת המרחק בין זוג הנקודות הקרוב ביותר

בהינתן n נקודות במישור רוצים להחזיר את המרחק המינימלי בין שתיים מהן כלומר: $\min_{j<i} \|p_i - p_j\|_2$.



<u>הבחנה</u>: בהינתן נקודה p, ישר l וקבוע $a>0$ מהצורה נסמן ב-$D(p)$ את המלבן שמימין ל-l ובו נמצאות נקודות במרחק של לכל היותר a מ-p. נסמן את קבוצת הנקודות של-$D(p)$ ב-$N(p)$.

<u>האלגוריתם</u>:

נניח כי הקלט H ממוין לפי קואורדינטות ה-x.

(1) אם $|H| \leq 3$, נמצא את המרחק המינימלי ביניהן ונחזיר ב-a

(2) אחרת, הגדר ישר $x=c$ עם $H\left[\lfloor \frac{n}{2} \rfloor\right] \leq c \leq H\left[\lfloor \frac{n}{2} \rfloor + 1\right]$

(3) חשב את ריצת האלגוריתם על $H_1 = [1, \dots, \lfloor \frac{n}{2} \rfloor]$ ועל $H_2 = [\lfloor \frac{n}{2} \rfloor + 1, \dots, n]$ ונשמור את המינימום ב-a

(4) הגדר את כל הנקודות מ-H_1 שמרחקן מהישר $\leq a$, מיין לפי קואורדינטת ה-y שלהן ושמור ב-V_1

(5) בצע פעולה דומה עבור H_2 ושמור ב-V_2.

(6) לכל $p \in V_1$, נבדוק אם ב-$N(p)$ קיימת נק' שמרחקה מ-p קטן מ-a, ונעדכן את a אם כן. נשמור את הנק' התחתונה מ-$N(p)$ לטובת האיטרציה הבאה.

(7) החזר את a

המשך מציאת המרחק הקרוב ביותר בין זוג נקודות

<u>נכונות</u>: נובעת אינדוקטיבית עבור שני החצאים. עבור קטעים חוצים - נובעת מההבחנה הקודמת לפיה רק הנקודות הרלוונטיות נבדקות בכל איטרציה.

<u>זמן ריצה</u>: ננתח את זמן הריצה לפי שלבי האלגוריתם

(1) בעלות $O(1)$

(2) בעלות $O(n)$

(3) בעלות $2\cdot T(\frac{n}{2})$

(4) בעלות $O(n\log n)$, עלות מיון של $O(n)$ נקודות

(5) כמו שלב (4) - $O(n\log n)$

(6) בעלות $O(n)$ לסריקת V_1 ו- $O\left(n+\sum_p |N(p)|\right)$ ל- V_2.

← ב- $N(p)$ יש לכל היותר 8 נק' שכן כל ריבוע קטן שצלעו $\frac{a}{2}$ מכיל לכל היותר נק' אחת שכן המרחק המקסימלי שניתן לקבל ממנו הוא $\frac{a}{\sqrt{2}}$ (ובפרט קטן מ-a) בסתירה לכך שהאלג' החזיר ערך $\leq a$ בקריאה הרקורסיבית.

לכן סה"כ שלב (6) בעלות $O(n)$

(7) בעלות $O(1)$

לפיכך קיבלנו בסה"כ: $T(n)=2\cdot T(\frac{n}{2})+O(n\log n)$ וממשפט האב

$\Leftarrow T(n)=O(n\log^2 n)$.

<u>הערות</u>:

כדי להתגבר על כך ש-H לא ממוין, ניתן למיין מראש ולקבל $F(n)=T(n)+O(n\log n)$

כדי להימנע ממיון V_1 ו-V_2 בכל איטרציה, נוכל להטמיע מיון מיזוג בריצת האלג':

בשלב הבסיס נמיין את הנק' לפי y, נחזיר את a (המינימום) V (הממוין עפ"י y)

בצעד, קיבלנו מהקריאות הרקורסיביות את V_1 ו-V_2 ממוינים לפי y ובסיום הקריאה נחזיר את a ו-V כאשר V הוא מיזוג של V_1 ו-V_2

← אזי נקבל:

$$T(n)=2\cdot T(\tfrac{n}{2})+O(n)=O(n\log n),\quad F(n)=T(n)+O(n\log n)$$

וסה"כ: $F(n)=O(n\log n)$.

30.12.2019

פתרון בעיות באלגוריתמים שיעור 9

נתקן את האלגוריתם מהשיעור הקודם כך שנקבל זמן ריצה של $O(n\log n)$:

נניח כי הקלט H ממוין לפי x

(1) אם $|H| \leq 3$, נמצא את המרחק המינימלי ביניהן ונחזיר ב-a וכן נחזיר את H (עותק שלו) ממוין לפי y.

(2) אחרת, הגדר ישר $x=c$ עם $H[\lfloor \frac{n}{2} \rfloor] \leq c \leq H[\lfloor \frac{n}{2} \rfloor +1]$.

(3) חשב את ריצת האלגוריתם על $H_1 = [1,...,\lfloor \frac{n}{2} \rfloor]$ ועל $H_2 = [\lfloor \frac{n}{2} \rfloor +1,...,n]$

ושמור את המינימום ב-a וכן את המערכים הממויינים ב-V_1 ו-V_2

שהתקבל מהקריאה הרקורסיבית

(4) הגדר את כל הנקודות מ-~~H_1~~ V_1 שמרחקן מהישר $\leq a$ ~~מיין לפי y~~ ושמור ב-$\hat{V}_1$.

(5) פעל באופן דומה על ~~H_2~~ V_2 ושמור ב-$\hat{V}_2$.

(6) לכל $p \in \hat{V}_1$ נבדוק אם ב-$N(p)$ קיימת נק' שמרחקה מ-p קטן מ-a ונעדכן את a אם כן.

נשמור את התחתונה מ-$N(p)$ לטובת האיטרציה הבאה

(7) החזר את a ואת המיזוג של V_1 ו-V_2

{השינויים הללו מטמיעים את מיון המיזוג בריצת האלגוריתם ומשפרים את זמן הריצה}

← נשים לב ששילוב של האלגוריתם לחיתוך קטעים והאלגוריתם לאוריינטציית פנייה נותנים לנו גם פתרון לבעיית הימצאות נקודה במצולע מאחר שגם במקרה הזה נוכל לחלק למקרים שמתאימים למקרים המיוחדים בהם דנו ב2'20 בעיית חיתוך הקטעים.

מציאת קמור של קבוצות במישור

הגדרה: מצולע במישור הוא סדרת נק' $p_0, p_1, ..., p_n$ כך שצלעותיו $[p_0, p_1], ..., [p_{n-1}, p_n], [p_n, p_0]$.

מצולע יקרא פשוט אם צלעותיו לא נחתכות. עבור מצולעים כאלו, "הליכה" על צלעות המצולע (נגד כיוון השעון) תראה כי הפנים של המצולע משמאל והחוץ מימין.

המעטף - קמור

הגדרות:

(1) קבוצה קמורה במישור, $A\subseteq\mathbb{R}^2$ היא קבוצה כך שלכל $p,q\in A$, הקטע $[p,q]$ מוכל ב-A

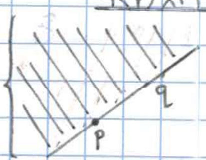
(2) בהינתן $A\subseteq\mathbb{R}^2$ סופית, הקמור של A הוא הקבוצה הקמורה המינימלית (ביחס להכלה) שמכילה את A.

(3) בהינתן $p,q\in\mathbb{R}^2$, נסמן:

חצי מישור שמאלי המתאים לישר העובר מ-p ל-q :
$$l^+_{p,q}=\{r\in\mathbb{R}^2 \mid \underset{\text{פניה שאינה ימינה}}{p\to q\to r}\}$$

חצי מישור ימני המתאים לישר העובר מ-p ל-q :
$$l^-_{p,q}=\{r\in\mathbb{R}^2 \mid \underset{\text{פניה שאינה שמאלה}}{p\to q\to r}\}$$

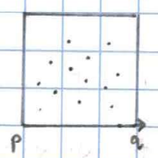
← דוגמא:



$l^+_{p,q}$, $l^-_{q,p}$ {זה תמיד נכון שחצי מישור שמאלי בכיוון אחד הוא חצי מישור ימני בכיוון השני}

(4) בהינתן $A\subseteq\mathbb{R}^2$ סופית, כאשר $p,q\in A$, הקטע $[p,q]$ ייקרא קטע תומך של A אם: $A\subseteq l^+_{pq}$ או $A\subseteq l^-_{pq}$, כלומר כל הנקודות בה מתקבלות מפניה שמאלה או שכולן מפניה ימינה.

← דוגמא:



בטענות ובאלגוריתמים הבאים נניח כי $A\subseteq\mathbb{R}^2$ סופית, ללא שלושה של נקודות על אותו ישר וכי ב-A יש נקודה אחת נמוכה ביותר (בעלת שיעור y נמוך ביותר)

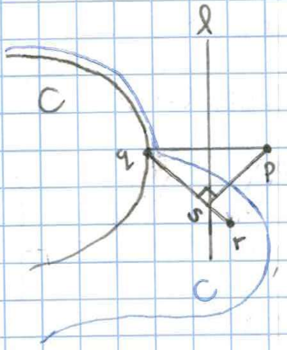
← {ההנחות הללו נחוצות לנכונות האלגוריתמים שנראה.}

משפט - אפיון הקמור: תהי $A\subseteq\mathbb{R}^2$ סופית ויהי C הקמור של A, ו-P המצולע הפשוט שצלעותיו הן קטעים תומכים של A. אז $P=C$

← הוכחה:

$P\subseteq C$ - תהי $p\notin C$, נוכיח כי $p\notin P$. נסמן ב-q את הנקודה ב-C שקרובה ביותר ל-p. נעביר קטע ביניהן וישר l המאונך לקטע זה. נוכיח כי p מימין לישר l ו-C משמאלו (או להיפך).

המשך הוכחת אפיון הקמור



נניח כי קיימת $r\in C$ שמימין לישר ונעביר אנך מ-p ל-$[q,r]$

נסמן ב-s את נקודת המפגש בין האנך מ-p לישר $[q,r]$.

מכיוון ש-$r\in C$ ו-C קמורה, גם $[q,r]$ ב-C ובפרט

$s\in C$. והרי s יותר קרובה ל-p מאשר q בסתירה

לבחירה של q.

מכאן לא יתכן שקיימת $r\in C$ מימין לישר l ולכן p מימין ו-C משמאל

ולכן גם כל התומכים ב-P נמצאים משמאל ולא יתכן $p\in P$.

$C\subseteq P$ - ראשית נבחין כי חיתוך קבוצות קמורות הוא קמור:

תהי I קבוצת קבוצות קמורות, נראה כי $\bigcap_{\tilde{C}\in I}\tilde{C}$ קבוצה קמורה.

יהיו p,q נקודות בחיתוך. אז:

$$\forall \tilde{C}\in I\quad p,q\in\tilde{C} \Rightarrow [p,q]\subseteq \tilde{C} \Rightarrow [p,q]\subseteq\bigcap_{\tilde{C}\in I}\tilde{C}$$

כעת נראה כי $l^+_{p,q}$ היא קבוצה קמורה:

פניות שאינן ימינה $r,s\in l^+_{p,q} \Rightarrow p\to q\to r,\ p\to q\to s$

לכל $t\in[r,s]$: $p\to q\to t$ פניה שאינה ימינה ולכן

$r,s\in l^+_{p,q} \Rightarrow [r,s]\subseteq l^+_{p,q}$ כלומר $l^+_{p,q}$ היא קבוצה קמורה.

נסיק כי: $C\subseteq \bigcap_{[p,q]\ \text{צלע של}\ P} l^+_{p,q} = P$

אם $p\notin P$ כלומר p בחוץ של P אז היא מימין לצלע כלשהי ולכן לא בחיתוך של חצאי המישורים שמשמאל לצלעות. כמו כן אם p לא בחיתוך, היא מימין לקטע תומך, אך A משמאל לקטע הזה ובפרט P משמאלו ולכן $p\notin P$

החיתוך הוא של קבוצות קמורות ולכן קמור, ו-A מוכלת בו. C קבוצה קמורה מינימלית ש-A מוכלת בה ולכן גם C מוכלת בחיתוך הנ"ל.

מכאן שאכן $C=P$. ∎

הערות: למעשה $C=\bigcap\tilde{C}$

$\tilde{C}$ קבוצה קמורה המכילה את A

נראה כעת כי גם האובייקט P קיים, ומתקבל מהפעלת האלג' הבא לחישוב קמור.

אלגוריתמים לחישוב קמור

אלגוריתם 1 - עטיפת מתנה (Jarvis March)

① מצא ב-A את הנקודה הנמוכה ביותר p_0.

② הגדר רשימה L שמאותחלת ל-p_0. נאתחל $p = p_0$.

③ עד לסגירת p_0:

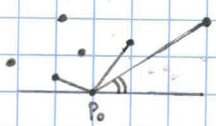
(i) מצא נקודה ימנית ביותר ב-A ביחס ל-p. נסמנה p'.

(ii) הוסף את p' ל-L ועדכן $p = p'$.

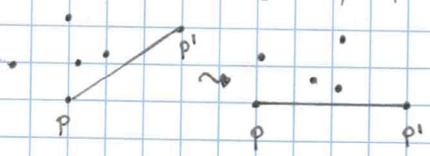
④ החזר את הרשימה L

← נכונות:

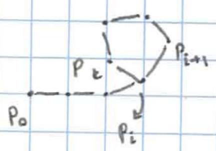
א. נבחין כי בחירת הנקודה הימנית ביותר מוגדרת היטב: עבור p_0, מכיוון שהיא הנמוכה ביותר, הימנית ביותר היא זו שבצעת הזווית החדה ביותר עם הישר שמקביל לציר ה-x ועובר דרך p_0:



בהתקדמות ל-p' הראשונה, מכיוון ש-p' ימנית ביותר, אם נסובב את הנק' כך ש-p ו-p' יהיו על ישר שמקביל לציר ה-x, נקבל כי אין נקודה נמוכה מישר זה ולכן משיקול דומה יש נק' ימנית ביותר ביחס ל-p'



ב. נבחין כי האלגוריתם אכן מגיע ל-p_0 (כלומר "סוגר" מצולע ועוצר):



נניח בשלילה שהאלגוריתם לא חוזר ל-p_0.

אזי משוקך היונים נקבל מעגר $p_i, p_k, \dots, p_{i+1}, p_i, p_{i-1}, \dots, p_0$

נתבונן במעגל הראשון הנ"ל (בו לפי הנחה $p_i \neq p_0$).

מהמינימליות, נקבל כי $p_{i-1} \neq p_k$. כמו כן, מכיוון ש-p_{i+1} ימנית ביותר ביחס ל-p_i, לא יתכן כי היא שווה ל-p_{i-1} או ל-p_k.

מכיוון ש-$[p_i, p_{i-1}]$, $[p_i, p_{i+1}]$, $[p_i, p_k]$ קטעים תומכים (כפי שנראה), מצב זה לא יתכן. זאת מכיוון שנקודה יחידה לא יכולה להיות קצה של שלושה קטעים תומכים. (כל סידור של הקטעים יוביל לסתירה להגדרתם).

← ניתן גם לתקן את האלגוריתם כך שנעצור כשהגענו לנקודה שכבר היינו בה.

ג. הקטעים המתקבלים הם קטעים תומכים של A ואינם נחתכים ולכן ממשפט אפיון הקמור מוחזר הקמור של A [מלווע העונה לתנאי המשפט]

המשך אלגוריתם עטיפת מתנה

← <u>זמן ריצה</u>: שלב ① בעלות $O(n)$ - מציאת מינימום. הלולאה מתבצעת $|C|$ איטרציות - כמספר קודקודי הקמור, וכל איטרציה בעלות $O(n)$ למציאת מינימום כאשר ההשוואות ב-$O(1)$. לכן עלותה: $O(n\cdot|C|)$.

שלב ④ בעלות $O(n)$ $\Leftarrow$ לכן סה"כ $O(n\cdot|C|) \leq O(n^2)$

← <u>הערה</u>: במקרים רבים ההתפלגויות עבורן נבחרות הנקודות יתנו $|C| = O(\log n)$ ואז האלגוריתם פועל ב-$O(n\log n)$.

פתרון בעיות באלגוריתמים

שיעור 10 06.01.2020

אלגוריתמים לחישוב קמור (המשך)

אלגוריתם 2 - הסריקה של גרהם (Graham's Scan)

① מצא נקודה נמוכה ביותר ב-A. נסמנה ב-p_0

② מיין את כל הנקודות ב-$A\setminus\{p_0\}$ מהימנית ביותר ביחס ל-p_0 ועד לשמאלית ביותר

③ אתחל מחסנית: $S = \begin{array}{|c|}\hline p_2 \\ \hline p_1 \\ \hline p_0 \\ \hline \end{array}$

④ לכל i מ-3 עד n:

(א) כל עוד $p_i \to S[top] \to S[top-1]$ היא פנייה ימינה, הסר את $S[top]$

(ב) הוסף את p_i ל-S

⑤ החזר את S מתחילתה לסופה.

← זמן ריצה: שלב ① בעלות $O(n)$, שלב ② בעלות מיון כאשר השוואה עולה $O(1)$ (השוואת אוריינטציה) ולכן עלותו $O(n \log n)$. שלב ③ בעלות $O(1)$. בשלב ④ מכניסים למחסנית n קודקודים ומסירים לכל היותר n כאלה ולכן עלותו $O(n)$. שלב ⑤ בעלות $O(n)$ $\Longleftarrow$ סה"כ $O(n \log n)$.

← נכונות: נוכיח באינדוקציה על i כי בסיום האיטרציה ה-i המחסנית מכילה את קודקודי הקמור של $\{p_0,\dots,p_i\}$.

בסיס - המחסנית מאותחלת ל-$\{p_0,p_1,p_2\}$ וזה משולש המהווה קמור.

צעד - נניח נכונות ל-i ונוכיח ל-i+1.

באיטרציה זו, במחסנית: $(p_0 = p_{t_0}, p_{t_1}, \dots, p_{t_k}, p_{i+1})$

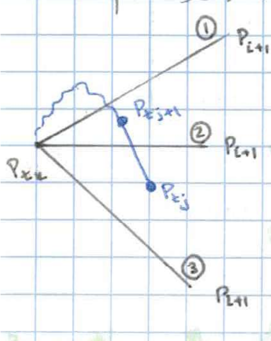
א. הקטע $[p_0, p_{i+1}]$ קטע תומך כי p_{i+1} שמאלית ביותר ביחס ל-p_0.

ב. לכל $1 \le j \le k-1$ הקטע $[p_{t_j}, p_{t_{j+1}}]$ קטע תומך:

כל הנקודות $\{p_0,\dots,p_i\}$ משמאל לקטע כזה - מהנחת האינדוקציה.

נראה כי גם p_{i+1} משמאל לו.

נניח בשלילה כי קיים $[p_{t_j}, p_{t_{j+1}}]$ עם j מקסימלי עבורו זה לא נכון. יש מספר מקרים:



• נבחין כי מקרה ① לא יתכן כי j מקסימלי

• מקרה ② לא יתכן כי $p_{i+1} \to p_{t_k} \to p_0$ פניה ימינה לפי המיון

וכך גם $p_{t_k} \to p_{t_{j+1}} \to p_0$ וכן $p_{t_{j+1}} \to p_{t_j} \to p_0$, כלומר p_0

נקודה פנימית במשולש $\Delta p_{t_j}, p_{t_{j+1}}, p_{t_k}$ ולכן לא יכולה להיות קצה של

קטע תומך כי תמיד 2 מבין ← יהיו אחד מימינו ואחד משמאלו.

המשך הסריקה של גרהם הוכחת נכונות

• באופן דומה, מקרה ③ לא יתכן כי אז p_0 במשולש $\{p_{t_k}, p_{t_{j+1}}, p_{t_j}\}$ וכך קטע תומך שנעביר דרכה, 2 מבין $\{p_{t_k}, p_{i+1}, p_{t_j}, p_{i+1}\}$ יהיו אחד מימין ואחד משמאל.

מכאן שאכן $\{p_0,\dots,p_{i+1}\}\setminus\{p_{t_j}, p_{t_{j+1}}\}$ משמאל לקטע $[p_{t_j}, p_{t_{j+1}}]$ ולכן הוא קטע תומך ב- $\{p_0,\dots,p_{i+1}\}$

ג. נותר להראות כי $[p_i, p_{i+1}]$ קטע תומך. מסעיף א' $[p_{i+1}, p_0]$ קטע תומך ולכן יודעים כי p_{i+1} קודקוד של הקמור.

לא יתכן כי הוא מצוות בלבד לקודקוד במחסנית שאינו p_i כי קודקודים אלה מצוותים כבר לשני קודקודים אחרים.

לא יתכן כי הוא מצוות לקודקוד שכבר הוסר מהמחסנית באיטרציה קודמת כי קודקודים כאלה הם נקודות פנימיות בקמור של $\{p_0,\dots,p_{i+1}\}$.

כמו כן, לא יתכן כי הוא מצוות לנקודות שהוסרו באיטרציה הזו כי נק' כאלו ק"מ $p_{i+1} \to p'' \to p'$ פניה ימינה ואז $p' \to p_{i+1} \to p''$ פניה ימינה בסתירה לכך שזהו קטע תומך.

מכאן אכן $[p_i, p_{i+1}]$ קטע תומך וטענת האינדוקציה מתקיימת, קרי באיטרציה ה-i המחסנית מכילה את קודקוד הקמור של $\{p_0,\dots,p_i\}$ ובשעה טובה סיימנו. ∎

אפליקציות של מציאת הקמור

דוגמא מכלכלה: קל למצוא חיתוך של קבוצות קמורות ומסתבר שגם גדר שמתקיים $c(A)+c(B)=c(A+B)$ לכן גם ניתן קירוב טוב במקרה הלא קמור

← Shapley Folkman

דוגמאות מאנליזה:

• Gauss Lucas

• הטווח הנומרי של מטריצות - הקמור של הע"ע ניתן את $W(A)=\left\{\frac{x^*Ax}{x^*x} \mid x\in\mathbb{C}^n\right\}$

דוגמא מ-vision - מציאת TTC

להרחבה

כעת נדון באפליקציה נוספת של מציאת הקמור הפעם מתחום הגיאומטריה החישובית.

זוג הנקודות הרחוקות ביותר { הדגמת שימוש בחישוב הקמור לצורך מימוש אלג' אחר }

הבחנות:

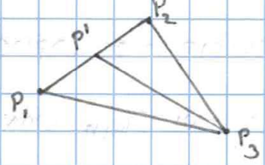
① בהינתן A סופית, כל נקודה ב-A נמצאת במשולש שקודקודיו הם קודקודים של הקמור של A.

(מתקבל אינדוקטיבית ע"י חלוקת הקמור למשולשים)

לפיכך, כל נקודה $p \in A$ היא צירוף קמור: $\circledast\ \lambda_1 p_1 + \lambda_2 p_2 + \lambda_3 p_3$

של p_1, p_2, p_3 קודקודי הקמור, כאשר $0 \le \lambda_1, \lambda_2, \lambda_3$ וסכומם 1.

$$\circledast = (\lambda_1 + \lambda_2) \cdot \underbrace{\left(\frac{\lambda_1}{\lambda_1+\lambda_2} p_1 + \frac{\lambda_2}{\lambda_1+\lambda_2} p_2 \right)}_{p' \text{ שעל הקטע } [p_1, p_2]} + \lambda_3 p_3$$



② (המסקנה) יהיו $p, q \in A$ ונחשב:

$$\|p-q\| = \left\|\sum_i \lambda_i p_i - q\right\| = \left\|\sum_i \lambda_i p_i - \sum_i \lambda_i q\right\| = \left\|\sum_i \lambda_i (p_i - q)\right\| \le \sum_i \lambda_i \|p_i - q\|$$

$$\le \sum_i \lambda_i \max_j \|p_j - q\| = \max_j \|p_j - q\|$$

כלומר, המרחק המקסימלי מתקבל על הקמור!

③ אם p ו-q נקודות רחוקות ביותר על הקמור, הן אנטיפודליות { שתי נק' על המצולע שניתן להעביר דרכן ישרים מקבילים שלא חותכים את המצולע }

נימוק: נניח ש-p ו-q רחוקות ביותר ונעביר את הקטע המחבר ביניהן וישרים l_1, l_2 מאונכים לו. אם l_1 (או l_2) חותכים את המצולע נקבל כי יש נקודה מעבר להן שהיא במצולע ורחוקה יותר

האלגוריתם: נניח כי יש ב-A נק' גבוהה ביותר ואין קטעים תואמים מקבילים

(1) חשב את הקמור C של A

(2) מצא נק' נמוכה ביותר ב-A p_{min} וגבוהה ביותר p_{max}. אתחל: $p_{min} \to p$, $p_{max} \to q$

(3) בצע עד ש-p ו-q משלימים כל אחד הקפה מלאה

(א) אם המרחק בין p ל-q גדול מהמקסימלי עד כה, עדכן אותו

(ב) אם p' (העוקבת ל-p) ו-q' (העוקבת ל-q) מקיימות ש- p-p' ימנית יותר מ-q-q'

ביחס לראשית, נעדכן $p \leftarrow p'$ ואחרת $q \leftarrow q'$

(4) החזר את הערך המקסימלי שנמצא

זמן ריצה: $O(n\log n)$ - עלות חישוב הקמור. יתר השלבים בעלות לינארית.

המשך זוג הנקודות הרחוקות ביותר

נכונות: בכל איטרציה, האלגוריתם בוחר אם להתקדם מ-p ל-p' או מ-q ל-q' לפי ההשוואה המוגדרת בו. אם $p'-p$ ימנית יותר כל "משיק" בנקודה q' יחתוך את הישר שעובר בצלע $[p,p']$, ולכן p ו-q' אינן אנטיפודליות, וכך גם עבור הנקודות שאחרי q'.

תכונה זו נובעת מכך ש-p ו-q אנטיפודליות וממבנה הקמור.

זוג הנקודות באיטרציה שלאחר ההתקדמות גם הן אנטיפודליות, וכך הלאה.

כך, בסיום כל איטרציה, כל הנקודות שבין P_{min} ל-p (לא כולל) הושוו אנטיפודלית לכל הנקודות שאחרי P_{max} ועד ל-p לפי סדר הסריקה, וכל הנקודות שבין P_{max} ל-q (לא כולל) הושוו אנטיפודלית לכל הנקודות שאחרי P_{min} ועד ל-q.

13.01.2020

פתרון בעיות באלגוריתמים שיעור 11

בעיית מנוע החיפוש

בהינתן מחרוזת (ארוכה) T ($|T|=n$), מעוניינים לענות על שאילתות מהצורה "האם P ב-T?" כאשר P מחרוזת שאילתה (קצרה). נסמן $|P|=k$

פתרון נאיבי: לכל תא ב-T בודקים אם P מתחילה בו, ואז בודקים את ההמשך. פתרון זה לא כולל עיבוד מוקדם ופועל ב-$O(n \cdot k)$ לשאילתה.

פתרון 1 $(O(n^2 \log n), O(k \log n))$

- עיבוד מוקדם: מיין את מערך מחרוזות הסיפות לפי סדר לקסיקוגרפי ושמור ב-B. (כאשר נגדיר שאם סיפא אחת היא רישא של סיפא מסויימת היא תופיע מאוחר יותר במיון).

← דוגמא: $T=ABCABD \quad P=ABD$

$B=[ABCABD, ABD, BCABD, BD, CABD, D]$

- שאילתא: בצע חיפוש בינארי של המחרוזת P במערך B והחזר כן אמ"מ מצאנו את P כרישא של איזושהי סיפא.
- זמן ריצה: העיבוד המוקדם בעזרת מיון מיזוג: $T(m)=2 \cdot T\left(\frac{m}{2}\right)+O(m \cdot n)$ וממשפט האב - $T(m)=O(n \cdot m \log m)$ ולכן בהצבת $m=n$ $O(n^2 \log n)$ השאילתא בעזרת חיפוש בינארי שהיא כמספר שלבי כפול עלות כל שלב ובמקרה זה $O(k \cdot \log n)$

פתרון 2 $(O(n), O(k+\log n))$

הגדרה: מערך הסיפות של T שיסומן $SA(T)$ הוא המערך ש-i באינדקס ה-i נמצא המיקום של הסיפא ה-i (לפי סדר לקסיקוגרפי כמו שהגדרנו קודם - עוקבת $S_1,...,S_m$ ל-$S_1,...,S_{m'}$ כאשר $m<m'$)

הגדרה - Longest Common Prefix: בעבור מחרוזות S_1, S_2 נגדיר את $LCP(S_1,S_2)$ כגודל הרישא המשותפת המקסימלית שלהן.

← דוגמא: $LCP(ABC, ABD)=2$

[נרצה למצוא מימוש של חיפוש בינארי שמוצא את P ביעילות ב-$SA(T)$ באמצעות שימוש ב-LCP-ים של הסיפות]

המשך פיתרון 2

← דוגמא: נניח $P = BDDACB$ וכי מערך מחרוזות הסיפא מכיל:

$$[\ldots, \underset{1}{BDDABC}, \underset{2}{BDDABE}, \ldots, \underset{3}{BDDACB}, \ldots, \underset{4}{BDEACB}, \ldots]$$

נסתכל על ה-LCP-ים של המחרוזות הללו ושל P

\# $LCP(P,1)=4$, $LCP(1,2)=5$ → תזוזה ימינה

\# $LCP(P,2)=5$, $LCP(1,4)=2$ → תזוזה שמאלה

\# $LCP(P,2)=4$, $LCP(2,3)=4$ → באמצע!

אם $LCP(P,x)=P$ מצאנו!

• שאילתא (המימוש של החיפוש הבינארי)

① אתחל max נוכחי $c=0$ ו-argmax נוכחי $i=n+1$

② כל עוד הקטע עדיין מתפשט באורך חיובי:

(א) אם $LCP(T[SA(mid),\ldots,n], T[SA(i)\ldots,n]) < c$, מתקרבים ל-$i$

(ב) אם 〃 $> c$ מתרחקים מ-i

בהתאם לקטעים המתקבלים בחיפוש בינארי

(ג) אם 〃 $= c$ מצא את התו

הראשון ב-P שונה מ-$T[SA(mid),\ldots,n]$ עם השוואות החל מהאינדקס

ה-c בשניהם.

מעדכנים את i ו-c בהתאם ומתקדמים ימינה/שמאלה בשניהם, לפי האות השונה.

אם $c=|P|$, החזר "כן"

③ החזר "לא"

← רעיון הנכונות

\# אם (א) מתקיים, המחרוזת המתחילה ב-$SA(mid)$ רחוקה מדי מהמקסימלית עד כה, ויש להתקרב אליה.

\# אם (ב) מתקיים, מחרוזת זו "דומה/קרובה" מדי למקסימום הנוכחי, ויש להתרחק ממנה.

\# אם (ג) מתקיים, אז גם ה-LCP של P עם הנוכחית הוא לפחות המקסימלי שנמצא, לכן הנוכחית מקסימלית וניתן להמשיך ולסרוק מהאותיות הרלוונטיות עד מציאת ערך מקס' נוכחי.

למעשה, בכל איטרציה c הוא LCP המקסימלי עם P שנמצא עד כה, והוא מתקבל במיקום ה-i.

המשך תרגול 2 - שאילתא

← זמן ריצה: עלות צעדי החיפוש הבינארי היא $O(\log n)$ ובנוסף עלות סריקת P מתחילתה לסופה לאורך כל האיטרציות היא $O(k)$

סה"כ זמן הריצה של העיבוד המוקדם $O(k + \log n)$.

• עיבוד מוקדם

כדי לאפשר מענה יעיל על השאילתא עלינו להכין:

(1) מערך הסיפות $SA(T)$

(2) מבנה נתונים המאפשר מענה על שאילתות LCP על סיפות של T ב-$O(1)$

← הערה: מערך הסיפות הוא קונספט שימושי ומגניב ומשתמשים בו לדחיסה (L-Z) ובריצוף למשל, ויש דרך לבנות אותו ב-$O(n)$ אבל לא נראה אותה.


ביולוגית חישובית

נתמקד ב-(2):

הגדרה: בהינתן מחרוזת T, מערך ה-LCP של T, המסומן $LCPA(T)$ הוא

$$LCPA[i] = LCP(SA[i], SA[i+1])$$

טענת פונקציית ה-LCP:

$$\forall\ 1 \le i < j \le n: \quad LCP(SA[i], SA[j]) = \min_{i \le k < j} \underbrace{LCP(SA[k], SA[k+1])}_{LCPA[k]}$$

← הוכחה: נסמן $m = \min_{i \le k < j} LCP(SA[k], SA[k+1])$. רוצים להראות

(1) $T[SA[i] \ldots SA[i]+m-1] = T[SA[j] \ldots SA[j]+m-1]$

(2) $T[SA[i]+m] \ne T[SA[j]+m]$ (המחרוזות שמתחילות באינדקסים האלה שונות)

(1) מתקבל כי:

$$T[SA[i] \ldots SA[i]+m-1] = T[SA[i+1] \ldots SA[i+1]+m-1] = \ldots = T[SA[j] \ldots SA[j]+m-1]$$

(2) לפי המיון:

$$T[SA[i] \ldots n] \le T[SA[i+1] \ldots n] \le \ldots \le T[SA[j] \ldots n]$$

$$\Downarrow$$

$$T[SA[i]+m] \le \ldots \le T[SA[j]+m]$$

ומבין אי השוויונות הללו יש אי-שוויון חזק ובפרט $T[SA[i]+m] < T[SA[j]+m]$

כאמור הן שונות. ∎

המשך תרגול 2 - עיבוד מוקדם

← מסקנה: ניתן להגדיר RMQ5 על LCPA ולקבל מענה על שאילתות ב-O(1):

$$LCP(i,j) = LCP(SA[SA^{-1}[i]], SA[SA^{-1}[j]]) = \min_{SA^{-1}[i] \le k < SA^{-1}[j]} LCP(SA[k], SA[k+1])$$

טענת פונ' ה-LCP

כלומר $LCP(i,j) = LCPA[k]$

נראה כיצד ניתן לבנות את LCPA ב-O(n)

טענת מערך ה-LCP:

$$\forall i:\ LCPA[SA^{-1}[i+1]] \geq LCPA[SA^{-1}[i]] - 1$$

[כלומר יש סוג של מונוטוניות, כשהאיבר הבא גדול, שווה או קטן ב-1]

← הוכחה: נסמן $m = LCPA[SA^{-1}[i]]$. נניח כי $1 \le m$, אחרת אי השוויון טריוויאלי.

נגדיר $j = SA[SA^{-1}[i]+1]$ ונקבל: (המחרוזת הבאה במערך הסיפות.)

$$LCPA[SA^{-1}[i]] = LCP(\underbrace{SA[SA^{-1}[i]]}_{i}, \underbrace{SA[SA^{-1}[i]+1]}_{j})$$

מתקיים:

$$T[i \ldots i+m-1] = T[j \ldots j+m-1] \wedge T[i+m] \neq T[j+m]$$

ולכן:

$$T[i+1 \ldots (i+1)+(m-1)-1] = T[j+1 \ldots (j+1)+(m-1)-1]$$

$$\wedge\ T[(i+1)+(m-1)] \neq T[(j+1)+(m-1)]$$

מכאן $LCP(i+1, j+1) = m-1$. ($T[i+1, \ldots, n]$) נסיק כי:

$$LCPA[SA^{-1}[i+1]] \geq \min_{SA^{-1}[i+1] \le k \le SA^{-1}[j+1]} LCPA[k] = LCP(i+1, j+1) = m-1 = LCPA[SA^{-1}[i]] - 1$$

טענת פונק' LCP

20.01.2020 שיעור 12 # תקרון בעיות באלגוריתמים

תזכורת - בעיית מנוע החיפוש: הראינו כי בהינתן מענה יעיל ($O(1)$) על "כמה קרובות הסיפות ה-i וה-j לפי המיון?" ניתן לענות על שאילתות ב-$O(k + \log n)$. מה ה-LCP שלהן

← הגדרנו: $LCPA[i]$ - כמה קרובות המחרוזות ה-i וה-i+1 לפי המיון?

פורמלית - $LCPA[i] = LCP(SA[i], SA[i+1]) = LCP(T[SA[i],...,n], T[SA[i+1],...,n])$

← טענת פונקציית ה-LCP: כמה קרובות המחרוזות ה-i וה-j לפי המיון? נסתכל על כמה קרובות המחרוזות i ו-i+1, כמה קרובות i+1 ו-i+2, ..., כמה קרובות j-1 ו-j וניקח את הערך המינימלי

פורמלית: $LCP(SA[i], SA[j]) = \min_{\substack{i \le k < j \\ (i<j)}} \underbrace{LCP(SA[k], SA[k+1])}_{LCPA[k]}$

← טענת מערך ה-LCP: כמה קרובה הסיפא $T[i+1,...,n]$ לעוקבת לה לפי המיון? לפחות m-1 כאשר m = כמה קרובה הסיפא $T[i,...,n]$ לעוקבת לה לפי המיון

פורמלית: $LCPA[SA^{-1}[i+1]] \geq LCPA[SA^{-1}[i]] - 1$

← דוגמא: T = ABCABD

i	1	2	3	4	5	6
מיון	ABCABD	ABD	BCABD	BD	CABD	D
SA(T)	1	4	2	5	3	6

← מסקנה: בחישוב $LCPA[SA^{-1}[i]]$ ($LCP([T[SA[i],...,n]$, העוקבת לה לפי המיון$)$) מספיק להשוות אותיות החל מהתו ה-$\max\{1, LCPA[SA^{-1}[i]]-1\}$

האלגוריתם (חישוב LCPA)

(1) לכל i מ-1 עד n: חשב את $LCP[SA^{-1}[i+1]]$ תוך ביצוע השוואות רק החל מהתו ה-$\max\{1, LCPA[SA^{-1}[i]]-1\}$

(2) החזר את LCPA

← נכונות: נובעת מיידית מהמסקנה לעיל

← זמן ריצה: עיקר פעולת האלגוריתם היא כמספר הזזות המצביע על האותיות המושוות. מצביע זה זז לכל היותר n צעדים ימינה, תוך קידום (אפשרי) על הזזה שמאלה לכל איטרציה => $O(n)$ [נעצר מהסריקה ימינה עד האינדקס ה-n + $O(n)$ תיקונים שמאלה]

המשך בעיית מנוע החיפוש

נסכם את שלבי העיבוד המוקדם:

(1) חשב את $SA(T)$

(2) חשב את $SA^{-1}(T)$

(3) חשב את $LCPA(T)$

(4) הגדר מבנה נתונים 5 RMQ על $LCPA(T)$

(5) התחל את המבנים שחושבו

← זמן ריצה: (1) ניתן לביצוע ב-$O(n)$ (2) גם ב-$O(n)$ תוך סריקת $SA(T)$ (3) ב-$O(n)$ כפי שהראינו כעת ו-(5) ב-$O(n)$ כפי שהראינו בשיעור 2.

← סה"כ הצלחנו למצוא פתרון ב-$O(n)$ לעיבוד המוקדם ו-$O(k+\log n)$ לשאילתות.

רשימת דילוגים

הגדרה: מערך אסוציאטיבי (dictionary, map) הוא מבנה נתונים אבסטרקטי (כלומר ניתן למימוש ע"י מבנה נתונים קונקרטי) שמייצג רשימות מהצורה (key, value) כך שהערך המתאים למפתח key הוא value ונרצה שערכי ה-key יהיו שונים (אין 2 value-ים שונים לאותו key)

הערות:

המבנה שנציג הוא מבנה הסתברותי, כזה שמחזיר תמיד תוצאה נכונה ותוחלת זמן ריצתו טובה. מאקראיות המבנה, נקבל כי עבור שתי סדרות זהות של הכנסת איברים למבנה, עשוי להתקבל מבנה (פרט) אחר.

מימוש מוכר למילון הוא עץ חיפוש בינארי מאוזן, למשל עץ AVL. מבנה הנתונים שנציג לעומתו, הוא כאמור הסתברותי, ולו יתרונות אחרים לעומת עץ AVL.

התוחלת בניתוח הפעולות היא על הביטים הרנדומליים המוגרלים בהן, ולא על פני התפלגות כלשהי מתוך נלקחים הקטעים.

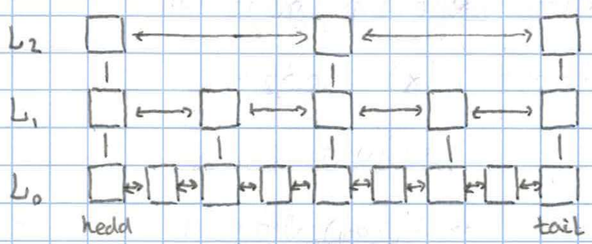
← לדוגמא, ניתן להראות כי עבור עץ חיפוש בינארי לא בהכרח מאוזן, בחירת פרמוטציה מקרית על $[n]$ והכנסתה לעץ, תניב עץ שגובהו בתוחלת $O(\log n)$ אך למשל הפרמוטציה $\{1,...,n\}$ תיצור עץ בגובה $O(n)$ [נרצה מבנה שלא תלוי בקלט]

המשך רשימת דילוגים

הדרישות ממבנה הנתונים מילון:

1. חיפוש (key) - החזר איטרטור לרשומה $(\tilde{key}, value)$ כך ש-$\tilde{key}$ הוא המקסימלי שאינו גדול מ-key.
2. הכנסה (key,value) - הוסף למבנה את הרשומה (key, value) והחזר איטרטור לה.
3. הסרה (key) - הסר את הרשומה עם המפתח key (אם קיימת) מהמבנה
4. עוקב (iterator) - החזר את האיטרטור לרשומה העוקבת לזו שעליה iterator מצביע
5. קודם (iterator) - החזר את האיטרטור לרשומה הקודמת לזו שעליה iterator מצביע

(אנחנו מגדירים מילון ממוין)



ניסיון 1 - רשימת דילוגים דטרמיניסטית:

המבנה הזה מוגדר כאוסף רשימות דו-כיווניות כמתואר באיור לעיל ומקיים:

(1) כל איברי המבנה נמצאים ברשימה התחתונה ביותר. כלומר L_0 מכילה את $(head = x_0, x_1, x_2, \dots, x_{n-1}, x_n = tail)$

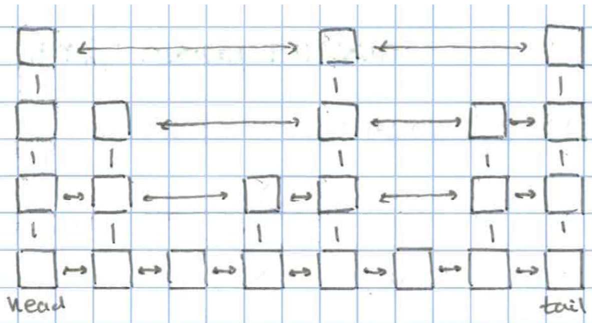
(2) כל האיברים ברשימה L_i נמצאים גם ברשימה L_{i-1}. קונקרטית:

- ב-L_1 נמצאים $(x_0, x_2, x_4, \dots, x_{n-2}, x_n)$
- ב-L_2 נמצאים $(x_0, x_4, x_8, \dots, x_{n-4}, x_n)$
- ב-L_k, כאשר $k = \Theta(\log n)$, נמצאים $(x_0, x_{\lfloor \frac{n}{2} \rfloor}, x_n)$

פעולת החיפוש במבנה מתחילה מה-head של L_k ומתקדמת ימינה עד למציאת x_j מקסימלי שאינו גדול מה-key. לאחר מכן, מתקדמים מה-x_j ב-L_{k-1} ימינה עד למציאת x'_j מקסימלי שלא גדול מה-key, וכן הלאה.

← החיפוש בעלות $O(\log n)$ כי יש $O(\log n)$ רשימות, ובכל רשימה סורקים מס' קבוע של איברים.

← הבעיה: פעולת ההכנסה עלולה לגרום לשינוי כל המבנה. לכן היא פעולה יקרה.



המשך רשימת דילוגים

ניסיון 2 - רשימת דילוגים:

המבנה יוגדר כאוסף רשימות דו-כיווניות, כמתואר באיור, כאשר בכל מעבר אוסף האיברים ברשימה מעל נבחר רנדומית.

תכונות:

(1) כמו בדטרמיניסטית, L_0 מכילה את כל האיברים.

(2) איברי L_{i+1} מוכלים ב-L_i

(3) לכל איבר x מגדר גובהו $h(x)=t$ להיות מספר הרשימות שהוא נמצא בהן: $L_0,\dots,L_{t-1}$, אך לא ב-L_t ומעלה.

(4) בנוסף, קיימים איברי דמה (זקיפים) - head, tail - שלא מכילים ערכים וגובהם כ-$h(x)$ המקסימלי.

מימוש הפעולות:

1. חיפוש (key) - זהה לחיפוש במבנה הדטרמיניסטי למעט כך שהחיפוש יתחיל מהרשימה L_m כאשר $m=\log_{\frac{1}{1-p}}(n)$ ו-$p\in(0,1)$ מוגדר מראש

2. הוספה (key, value) - מחפשים את הרשימה עם המפתח key במבנה. אם לא קיימת, מוסיפים אחרי הרשימה שנמצאה את (key, value).
ההוספה תוך הגרלת $h\sim Geo(p)$ ונוסיף את x בגובה זה לרשימות הרלוונטיות. אם h גדול מהמקסימלי עד כה, הגדל את הזקיפים.
ביצוע פעולת חיפוש נוספת מאפשרת לקשר את האיבר החדש לקודמים ולעוקבים לו.

3. הסרה (key) - מצא את key במבנה והסר אותו תוך קישור קודמו לעוקבו

4,5. קודם/עוקב (iterator) - החזר איטרטור המצביע לאיבר ב-L_0 אחרי מצביע איטרטור הקלט

סיבוכיות:

מקום: סך כל הופעות האיברים ← סכום גובהם + גובה הזקיפים. בתוחלת:

$$\mathbb{E}\Big(\sum_{j=1}^{n} h(x_j) + \text{גובה הזקיפים}\Big) \le \mathbb{E}\Big(3\cdot\sum_{j=1}^{n} h(x_j)\Big) = 3\cdot\sum_{j=1}^{n}\mathbb{E}(h(x_j)) = 3\cdot n\cdot\frac{1}{p} = \Theta(n)$$

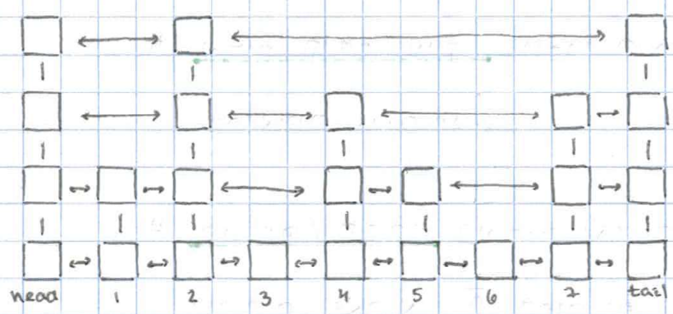
בהנחה ש-p קבוע.

27.01.2020

פתרון בעיות באלגוריתמים

שיעור 13

המשך רשימת דילוגים

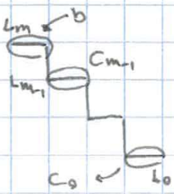


תזכורת: בשיעור שעבר הגדרנו את המבנה הבא ←

פירטנו את מימוש הפעולות מעליו

וניתחנו את סיבוכיות המקום - $O(n)$

ננתח כעת את סיבוכיות הפעולות שהגדרנו.



\# ניתוח זמן הריצה של פעולת החיפוש: מעוניינים למצוא את התוחלת של $b+\sum_{i=0}^{m-1} c_i$ כאשר b הוא מס' האיברים שנסרקו ב-L_m ו-c_i-1 מס' האיברים שנסרקו ב-L_i.

(1) ננתח את $E(b)$. נבחין כי $b \le |L_m|+2$ (עם כולל זקיפים) וכן כי כל איבר ב-L_m מופיע באופן ב"ת באחרים ובהסתברות $(1-p)^m$ כאשר $1-p$ היא ההסתברות לאי-הצלחה של המשתנה הגיאומטרי, כלומר $P(h(x))=m=(1-p)^m$.

לכן $|L_m| \sim Bin(n,(1-p)^m)$ ומכאן:

$$E(|L_m|) = n\cdot(1-p)^m = n\cdot\frac{1}{\left(\frac{1}{1-p}\right)^m} = n\cdot\frac{1}{\left(\frac{1}{1-p}\right)^{\log_{\frac{1}{1-p}}(n)}} = n\cdot\frac{1}{n} = 1$$

ולכן $E(b)\le 3$, ממונוטוניות התוחלת.

(2) ננתח כעת את $E(L_i)$ לכל i. נבחין כי $c_i=1 \iff$ האיבר האחרון שנסרק ב-L_i הופיע גם ב-L_{i+1}. ההסתברות לכך היא $P(x\in L_{i+1} \mid x\in L_i)=1-p$.

חוסר זיכרון של משתנה גיאו' → $P(x\ge i+1 \mid x\ge i) = P(x\ge 1) = 1-p$

באופן דומה $c_i=2 \iff$ האיבר x האחרון שנסרק לא הופיע ב-L_{i+1} אך x' שנסרק לפניו הופיע. כלומר:

$$P(x\in L_{i+1}, x'\notin L_{i+1} \mid x,x'\in L_i) = (1-p)\cdot p$$

כי הפעם היתה הצלחה.

באופן כללי, נקבל:

$$P(c_i=k) = \underbrace{p^{(k-1)}}_{k-1\text{ הצלחות}}\cdot\underbrace{(1-p)}_{\text{אי הצלחה באיבר האחרון}}$$

L_{i+1} □ x x x

L_i □ □ □ □

זאת בהנחה ש-k קטן מספיק, או במילים אחרות - בהנחה שמתעלמים מהזקיפים.

מכאן ש-$c_i \sim Geo(1-p)$ תחת הנחה זו.

באופן כללי וללא ההנחה, c_i גיאומטרי עם זנב קטום. לכן נוכל לחסום את התוחלת של c_i בתוחלת של מ"מ גיאומטרי: $E(c_i) \le E(Geo(1-p)) = \frac{1}{1-p}$

נסיק כי תוחלת זמן הריצה:

$$E\left(b+\sum_{i=0}^{m-1} c_i\right) \le 3+\sum_{i=0}^{m-1} E(c_i) \le 3+m\cdot\frac{1}{1-p} = 3+\log_{\frac{1}{1-p}}(n)\cdot\frac{1}{1-p} = O(\log n)$$

כאשר p קבוע

המשך רשימת דילוגים ניתוח סיבוכיות הפעולות

\# עלות קודם/עוקב במבנה היא $O(1)$ בכל מקרה, כי זו העלות של הפעולות ברשימה דו כיוונית.

\# הוספה/הסרה בעלות שתי פעולות חיפוש ועוד עלות גובהו של האיבר שמוסיפים/מסירים, שהיא $\frac{1}{p}$ בתוחלת $\Leftarrow$ $O(\log n)$ בתוחלת כאשר p קבוע.

ניתוח תוחלת גובה המבנה

טענה: עבור רשימת דילוגים עם n איברים, תוחלת גובהה היא $O(\log n)$.

במילים אחרות: $X_1,\dots,X_n \sim Geo(p) \Leftarrow \mathbb{E}(\max_i X_i) \le O(\log n)$

← הוכחה: נגדיר אינדיקטור Y_i שערכו 1 אמ"מ יש איבר ברשימה L_i.

נחשב את $\mathbb{E}\left(\sum_{i=0}^{\infty} Y_i\right)$. נבחין כי $Y_i \le 1$ [בהיותו אינדיקטור] וכן $Y_i \le |L_i|$ (*)

והראנו כי $\mathbb{E}(|L_i|) = n\cdot(1-p)^i$.

כל הלוגים בסכום $\frac{1}{1-p}$

נקבל:

$$\mathbb{E}\left(\text{גובה המבנה}\right) = \sum_{i=0}^{\infty}\mathbb{E}(Y_i) = \underbrace{\sum_{i=0}^{\log n}\mathbb{E}(Y_i)}_{Y_i\le 1 \Rightarrow \text{הסכום} \le (\log n+1)\mathbb{E}(1)} + \sum_{i=\log n}^{\infty}\mathbb{E}(Y_i) \le \log n + 1 + \underbrace{\sum_{i=\log n}^{\infty} n\cdot(1-p)^i}_{(*)\ Y_i \le |L_i| \to \mathbb{E}(Y_i)\le \mathbb{E}(|L_i|)}$$

$$\le \log n + 1 + \underbrace{\sum_{i=1}^{\infty}(1-p)^i}_{\text{טור גיאומטרי שמתכנס לקבוע}} = O(\log n)$$

הראנו בניתוח סיבוכיות החיפוש

פתרון הבונוס תרגיל 3.

בהינתן סדרת שאילתות $\left((u_i,v_i)\right)_{i=1}^{n}$ נרצה להחזיר "כן" אם ניתן לכוון את הגרף ולקבל מסלול מ-u_i ל-v_i לכל i, ולא אחרת.

האלגוריתם:

1. אם קיימים u,v בשאילתות ברכיבי קשירות שונים, החזר "לא".
2. הגדר את הגרף בו כל קודקוד הוא רכיב דו-קשירות בגרף המקורי (איחוד מעגלים)

   ← רדוקציה של הבעיה לעץ, ואף לעץ יחיד כי נוכל לבדוק כל אחד בנפרד
3. הגדר מבנה LCAS על כל עץ שהתקבל (אחרי שנשרש אותו!), ולכל זוג קודקודים בשאילתא מצא את ה-LCA שלהם.
4. לכל קודקוד v בעץ נשמור: $v.s$ - כמה קודקודי מקור יש בתת העץ ש-v שורשו

   $v.t$ - כנ"ל עבור קודקודי יעד

   $v.m$ - כנ"ל עבור LCAS

המשך שאלה ו תרגיל 3 - האלגוריתם

כלומר נדרש לכיון את אותה צלע בשני הכיוונים.

⑤ לכל קודקוד r בעץ, ובנו r':

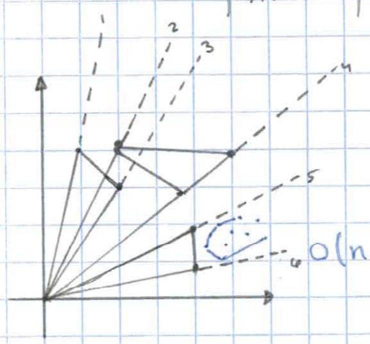
← אם $r'.s > r'.m$ ו- $r'.t > r'.m$, החזר "לא"

⑥ החזר "כן"

נותר רק להסביר למה שלב ② נכון, כלומר למה בתוך רכיב דו-קשירות ניתן לעבור מכל קודקוד לכל קודקוד ולכן ניתן לאחד אותם.

וזה נובע מהטענה הבאה מדיסקרטית:

אם לכל קבוצות A,B זרות יש צלע מ-A ל-B ⇐ הגרף קשיר חזק.



פתרון הבונוס תרגיל 4

אלגוריתם 1:

① מיון קודקודי המשולשים לפי שיפוע ושמירה במערך L_1, $O(n\log n)$

② מיון k הנקודות לפי שיפוע ושמירה במערך L_2 $O(k\log k)$

③ לכל קונוס המוגדר ע"י זוג נקודות מ-L_1: $O(n)$ פעמים

(א) נמצא את החלק הגדול של הקמור שמתאים לו

מציאת החלק הגדול | חישוב הקמור

$\underbrace{\log k_i + k_i \log k_i}_{O(k\log k)}$ הסכום על כל i יהיה

(k_i קבוצת הנק' הגדולות לקונוס i)

$\sum_{i=1}^{O(n)} k_i = k = O(n\sqrt{n})$

(ב) לכל משולש המתאים לזוג זה, נבדוק אם הנקודה הקרובה ביותר מאותו

חלק גדול נמצאת ב2 $O(n\cdot\log k)$

④ החזר משולשים מסומנים $O(1)$

⇐ סה"כ זמן הריצה:

$$\overset{①}{O(n\log n} + \overset{②}{k\log k} + \overset{③(א)}{k\log k} + \overset{③(ב)}{n\cdot n\log k)} = O(n^2\log n)$$

אלגוריתם 2:

$T(k) = 2T\left(\frac{k}{2}\right) + O(k\log k) \Rightarrow$

① הגדר עץ קטעים לחלק הגדול של הקמור לנק' $O(k\log^2 k)$

② מיון הנקודות לפי שיפוע $O(k\log k)$

③ לכל משולש: $O(n)$

(א) מצא את הקודקודים בעץ שמכילים נק' k רלוונטית $O(\log k)$

$O(\log^2 k)$

(ב) לכל (חלק גדול של) קמור זה, מצא את הנקודה הקרובה ביותר ובדוק אם במשולש

④ החזר משולשים מסומנים $O(1)$

⇐ סה"כ זמן הריצה:

$$O(k\log^2 k + k\log k + n + \log k + \log^2 k) = O(n\sqrt{n}\log^2 n)$$

$$< O(n^2)$$