

מבני נתונים | 97106

הרצאות | ד"ר גיא כץ וד"ר עמית דניאל

כתביה | נמרוד רק

תשפ"א סמסטר ב'

תוכן העניינים

4	כלים מתמטיים
4	תרגול
6	שבוע II Asymptotic Symbols & The Sorting Problem
6	הרצאה
11	תרגול
13	שבוע III Master Theorem
13	הרצאה
18	תרגול
20	שבוע IV Quick Sort & Comparison Based Sorting
20	הרצאה
23	תרגול
26	שבוע V Introduction to Hashing, Open Addressing & Non-Comparison Sorting
26	הרצאה
29	תרגול
32	שבוע VI Universal Hashing
32	הרצאה
36	תרגול
38	שבוע VII Priority Queue & Heap
38	הרצאה
41	תרגול

42	Binary Search Tree & AVL VII	שבוע
43		הרצאה
47		תרגול
49	Breadth First Search VIII	שבוע
50		הרצאה
53		תרגול
55	Semi-Course Recap IX	שבוע
55		תרגול
55	Depth First Search & Toplogical Sort X	שבוע
55		הרצאה
58		תרגול
59	Minimum Spanning Tree XI	שבוע
59		הרצאה
63		תרגול
64	Weighted Single Source Shortest Path XII	שבוע
64		הרצאה
68		תרגול
70	All Pairs Shortest Path XIII	שבוע
70		הרצאה
74		תרגול
75	Disjoint Set XIV	שבוע
75		הרצאה

כלי מתמטיים

תרגול

נרצה להשתמש באינדוקציה כדי להוכיח כל מיני טענות על המספרים הטבעיים (\mathbb{N}). כדי להוכיח טענה, בשיטת האינדוקציה החלטה, מספיק להוכיח את נכונות הטענה עבור הבסיס ($n = 1$) או כל מספר טבעי אחר שהחל ממנו הטענה אמורה להיות נכונה) וכן עבור הצעד הפירושה להניח שהטענה נכונה עבור אייזה שהוא n ולהוכיח באמצעות הנחה זו שהטענה נכונה גם כן עבור $n + 1$.

$$\text{דוגמה: } \text{נוכיח כי } \sum_{i=1}^n i = \frac{n(n+1)}{2}, \forall n \in \mathbb{N}$$

$$\text{הוכחה: בסיס } (n=1) : \sum_{i=1}^1 i = 1 = \frac{1 \cdot 2}{2}$$

$$\text{צעד } (n \rightarrow n+1) : \text{נניח כי } \sum_{i=1}^n i = \frac{n(n+1)}{2}. \text{ לכן}$$

$$\sum_{i=1}^{n+1} i = \sum_{i=1}^n i + (n+1) = \frac{n(n+1)}{2} + (n+1) = \frac{n(n+1) + 2(n+1)}{2} = \frac{(n+1)(n+2)}{2}$$

■
אינדוקציה חזקה היא דומה לאינדוקציה החלטה, רק שב証明 הצעד נוכל להניח שהטענה נכונה לא רק עבור n , אלא עבור כל המספרים עד $-n$ (1, 2, ..., n) או כל המספרים מהבסיס ועד $-n$).

דוגמה: נתונה טבלת שוקולד שמכילה n קוביית שוקולד. איזי לחלק את הטבלה ל- n קוביות נפרדות, לא משנה איך נחלק אותן.

הוכחה: בסיס ($n = 1$) : לוקח 1 – 1 = 1 צעדים לחלק לקוביות נפרדות שכן קוביה אחת היא כבר לבד.

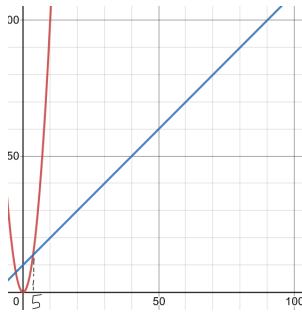
צעד ($1, \dots, n-1 \rightarrow n$) : נסביר את הטבלה איפה שהוא ונקבל שתי טבלאות בגודלים $k, n-k$. מה"א (הנחה האינדוקציה), בגלל ש- $n < k, n-k < k$, יקח לנו לבדוק $1 - k$ שבירות לחלק את הטבלה הראשונה ו- $1 - k$ שבירות עבור הטבלה השנייה. יחד עם השבירה הריאוונה שלנו קיבל סה"כ $(k-1) + (n-k-1) + 1 = n-1 = n$ קוביות נפרדות.
■

עתה נעסוק בסימונים אסימפטוטיים (שהוגדרו [כאן](#)).

$$g(n) = \Omega(f(n)) \text{ וגם כי } f(n) = \mathcal{O}(g(n)). \text{ נוכיח כי } f(n) = n + 10, g(n) = n^2$$

$$\text{הוכחה: } f(n) = n + 10 \leq n + 2n = 3n \leq n \cdot n = n^2 = g(n), \forall n \geq 5$$

$$g(n) = n^2 = n \cdot n \geq 3n = n + 2n \geq n + 10 = f(n), \forall n \geq 5$$



איור 1: f בכחול ו- g באדום, החל מ- $n=5$ “עוקף” את f

מסקנה $f(n) = \Omega(g(n))$ אם $f(n) = \mathcal{O}(g(n))$

דוגמה $p(n) = \mathcal{O}(q(n))$. $p(n) = n^5$, $q(n) = 0.5 \cdot n^5 + n$

הוכחה: ■ $p(n) = n^5 \leq 2 \cdot 0.5n^2 \leq 2(0.5n^2 + n) \stackrel{c=2}{=} c \cdot q(n)$, $\forall n \geq 0$

טענה אם $f(n) = \mathcal{O}(g(n))$ אז $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} < \infty$

הוכחה: נסמן $\epsilon = 1$ (הגבול הוא חיובי כי בהגדרה של הסימונים האסימפטוטיים אמרנו שהפ' חן חיוביות). נבחר L כך ש- $L - 1 < \frac{f(n)}{g(n)} < L + 1$ $\forall n \geq n_0 \in \mathbb{N}$ קלומר (n , $f(n) < (L+1)g(n)$, $L-1 < \frac{f(n)}{g(n)} < L+1$, $L < \frac{f(n)}{g(n)} + 1 < L+2$)

■ ונקבל את ההגדרה של $\mathcal{O}(g(n))$ כרצוי.

מסקנה אם $f(n) = \Theta(g(n))$ אז $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = L > 0$

הוכחה: שני הגבולות קיימים $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)}$, $\lim_{n \rightarrow \infty} \frac{g(n)}{f(n)}$, ולכן $f(n) = \mathcal{O}(g(n))$, $g(n) = \mathcal{O}(f(n))$ ו- $\Theta(g(n))$

$f(n) = \mathcal{O}(g(n)), \Omega(g(n))$

קלומר ($f(n) = \Theta(g(n))$ (השתמשנו בתכונות **מבחן**)). ■

טענה אם $f(n) = \Omega(g(n))$ אז $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \infty$

הערה נשים לב כי $\Theta, \Omega, \mathcal{O}$ הם קבועות, ולכן אם נהייה פונקציות נציג לכתוב ($f(n) \in \mathcal{O}(g(n))$) אבל הסימון המקובל הוא עדיין

$f(n) = \mathcal{O}(g(n))$

טענה ■ $n = \mathcal{O}(2^n)$

הוכחה: באינדוקציה.

בבסיס $n=1 < 2 = 2^1 = 2^n : (n=1)$

■ $n+1 < 2^n + 1 < 2^n + 2^n = 2^{n+1} : (n \rightarrow n+1)$ צעד

טענה יהי $p(n) = \mathcal{O}(q(n))$ פולינום מדרגה $d \leq k$ ויהי $q(n)$ פולינום מדרגה k . אם a_i איזי $p(n) = a_0 + \dots + a_d n^d + \dots + a_k n^k$.

הוכחה: נסמן $\lim_{n \rightarrow \infty} \frac{p(n)}{q(n)} < \infty$. $q(n) = b_0 + \dots + b_d n^d + \dots + b_k n^k$. מספיק שnochich כי $\lim_{n \rightarrow \infty} \frac{p(n)}{q(n)} = 0 < \infty$ איזי $d < k$. $\lim_{n \rightarrow \infty} \frac{p(n)}{q(n)} = \frac{a_k}{b_k} < \infty$ איזי $d = k$. $\lim_{n \rightarrow \infty} \frac{p(n)}{q(n)} = \frac{n^k(a_0 n^{-k} + \dots + a_d n^{d-k})}{n^k(b_0 n^{-k} + \dots + b_d n^{d-k} + \dots + b_k)}$ כל הגורמים הם בחזקת שלילית ולכן מתאפסים באינסוף.

טענה נגדיר $f(n) = \mathcal{O}(n)$ איזי $f(n) = 2f(\lfloor \frac{n}{2} \rfloor) + 1$, $\forall n > 1$ ו $f(1) = 1$

הוכחה:

$$\begin{aligned} f(n) &= 2f\left(\lfloor \frac{n}{2} \rfloor\right) + 1 = 2\left(2f\left(\lfloor \frac{n}{4} \rfloor\right) + 1\right) + 1 = 4f\left(\lfloor \frac{n}{4} \rfloor\right) + 3 \\ &= 4\left(f\left(\lfloor \frac{n}{8} \rfloor\right) + 1\right) + 3 \\ &= \dots = \text{נמשיך } n \text{ פעמים} \\ &= 2^{\log_2 n} f\left(\frac{n}{2^{\log_2 n}}\right) + 2^{\log_2 n} - 1 \\ &= nf(1) + (n-1) = n + n - 1 = 2n - 1 = \mathcal{O}(n) \end{aligned}$$

■

חוקי לוגריתמים

$$\log_a x \cdot y = \log_a x + \log_a y .1$$

$$\log_a \frac{x}{y} = \log_a x - \log_a y .2$$

$$\log_a x^m = m \log_a x .3$$

$$. \frac{\log_a x}{\log_a y} = \log_y x .4 \text{ (נוסחת שינוי הבסיס)}$$

הוכחה: (של נוסחת שינוי הבסיס) נסמן $c = \frac{\log_a x}{\log_a y}$ ולכן $\log_y x = c$ $\log_a y^c = \log_a y^c = \log_a x \cdot y^c = \log_a x \cdot y^{\frac{\log_a x}{\log_a y}}$

שבוע Asymptotic Symbols & The Sorting Problem | II

הרצאה

אלגוריתם (בקשר של מודם"ח) הוא תוכון כדי להגעה ממצב התחלתי (קלט) למצב סופי כלשהו (פלט). אלגוריתם, הוא תיאור אבסטרקטי של האירועים, בנויגוד לתכנית שהוא יותר מדויקת ולא עוסקת ברעיון של הפעולות - הדבר שהוא לנו מעניין בקורס הזה. לכן אנחנו

נעסק באלגוריתמים.

טיפוסי נתונים מפורטים (DT) הם תיאורים פורמליים (רשמיים, מפורטיים) של מאפיינים של מבני נתונים. כאשר מבנה נתונים קונקרטי הוא בעצם מיושן של DT כלשהו.

בහינתם בעיה כלשהי, נרצה לדעת האם קיים פתרון לבעה (לא תמיד, ע"ע בעיית העצירה). אם כן, האם קיים אלגוריתם שפותר את הבעיה באופן יעיל (לא תמיד, ע"ע בעיית האриזה). אם כן, האם האלגוריתם הספציפי שלנו הוא הכי יעיל (לא תמיד, ע"ע מיעון בועות). מהו האלגוריתם הכי יעיל לפתרון הבעיה?

הערה יעילות, במסגרת הקורס זהה, תتبטה בכמה זמן לתקן לאלגוריתם לורץ (סיבוכיות זמן) ובכמה זיכרון הוא צריך לאורכו (סיבוכיות מקום/זיכרון).

בעיה נתונה מפת מטרו ושתי תחנות, מהי הדרך הכי קצרה לעبور בין התחנות?

פתרון נוכל לייצג את המפה בתור גראף ואז למצוא את הדרכ הכי קצרה בין הקודקודים, ונמצא בהמשך דרך לפטור זו את - (n) (וגם נבון מה זה הסימון הזה).

בעיה בהינתן לוח וקבוצת צורות, מלאו את הלוח בצורות (כך שלא יחפפו) כך שיתפסו שטח מינימלי.

פתרון ניציר את כל האפשרויות, נחשב את השטח שכל אפשרות תופסת ונשמר את המקרה הטוב ביותר עד כה. בהינתן n צורות, האלגוריתם המפורט בהמשך יירוץ על 4^n אפשרויות (כל צורה אפשר לסובב ב-4 דרכים). נוכיח בהמשך שאין אלגוריתם יעיל יותר במקרה הגורע ביותר.

1. ניציר את כל האפשרויות

2. נחשב את תפוקת המקום של כל אפשרות

3. נשמר את התוצאה הכי טובה

1: פסויידו קוד לפתרון בעיית האריזה

בහינתם בעיה, נבדוק מה נוכל להניח על הקלט ונחקק אותו לשני מקרים: המקרה הטוב ביותר והמקרה הגרוע ביותר. המקרה הטוב ביותר לא מאד רלוונטי כי המקרה הכללי לרוב לא יהיה "נחמד" אלינו, יותר מעניין אותנו המקרה הגרוע, שהינתן אויב מרושע שנוטן לנו קלטים מרושעים, נוכל לנצל אותו בסיבוכיות הזמן והמקום!

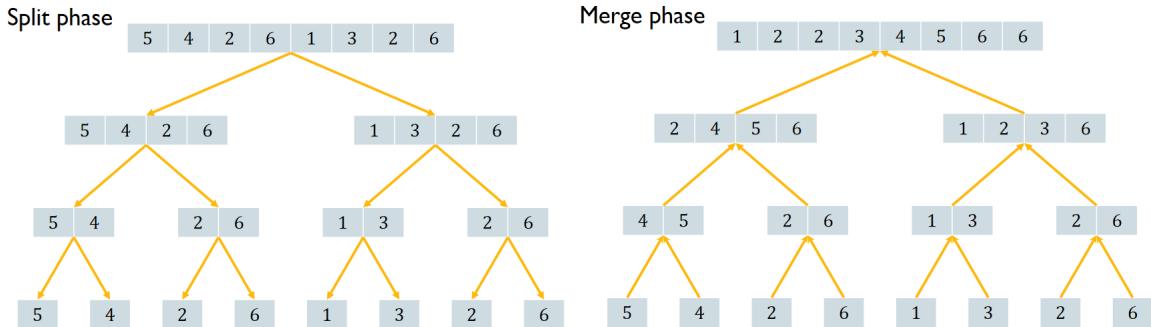
בעיה בהינתן מערך של מספרים בגודל n , נרצה למין את המערך בסדר עולה.

אלגוריתמים למיעון מערכאים

1. **מיון בועות.** נעבור על כל זוג מספרים צמודים, ואם הם מסודרים לא כרצוי (כלומר $A[i] > A[i + 1]$) נחליף אותם. נעבור על המערך באופן זהה כמה פעמים שצריך עד שהיא ממוין (כלומר שלאורך מעבר אחד לא נחליף אף זוג איברים).

ננתן את האלגוריתם הנ"ל. בכל ריצה, נעבור על מערך סה"כ n פעמים, ובכל מעבר במקרה הגרוע ביותר נבצע n פעולות (קצת פחות כי בהכרח שאחרי המעבר הראשון שני האיברים הראשונים יהיו מסודרים), כלומר קיבלו שהאלגוריתם עושה n^2 פעולות.

2. **מיון מיזוג.** נפצל את המערך לשני תת-מערכות, ונעשה זאת שוב ושוב עד שנתקבל מערך אחד, ואז נמזג כל שני מערכות כך שאיבריהם הממזוגים יהיו ממויינים (ראו איור).



איור 2 : שלב הפיצול (משמאל) והמזוג (מימין) של מיון המיזוג

ננתן את מיון המיזוג. מספר הרמות ב”עץ” שדרכו נרד למיון הוא $n \log n$, ושלב המיזוג (ש망תקים בכל רמה) לוקח n פעולות ולכן סה”כ נקבל שבסך $n \log n$ פעולות.

אלגוריתם	זמן ביעות	זיכרון	זמן (הכי טוב)	זמן (הכי גרוע)	זמן (ממוצע)	זמן (נרד)
מיון ביעות	$\Theta(n)$	$\Theta(1)$	$\Theta(n^2)$	$\Theta(n^2)$	$\Theta(n^2)$	$\Theta(n^2)$
מיון מיזוג	$n \log n$	$n \log n$	$n \log n$	$n \log n$	$n \log n$	$n \log n$

טבלה 1 : ניתוח סיבוכיות של מיון ביעות ומיזוג

3. **מיון הכנסה.** נרוץ על כל הרישאות של המערך עי’ הוזת האיבר האחרון ברישא למקום שבו הוא אמור להיות ברישא (בנהנה שככל האיברים ברישא מלבד האחרון כבר מזינו במעבר הקודם). כאשר t_j הוא מספר ההוזות שנצטרכן לעשות כדי לשים את האיבר ה- j

```

1 for j←2 to A.length-1 do
2     key←A[j]
3     i←j-1
4     while i>0 and A[i]>key do
5         A[i+1]←A[i]
6         i←i-1
7     A[i+1]←key

```

מיאון הכנסה

שורה (i)	1	2	3	4	5	6	7
(T_i) מס' הפעמים שהפעולה מתבצעת	n	$n - 1$	$n - 1$	$\sum_{j=2}^n t_j$	$\sum_{j=2}^n (t_j - 1)$	$\sum_{j=2}^n (t_j - 1)$	$n - 1$

טבלה 2 : מס' החזרות על כל שורה במיון הכנסה

במקום הנכוון ברישא שלפניו.

לכן סה”כ זמן הריצה הוא $c_i T_i$ כאשר c_i , T_i הם העלות ומספר הפעמים שמתבצעת השורה ה- i בקוד בהתאם. לכן סיבוכיות הזמן היא

$$T(n) = c_1 n + c_2 (n - 1) + c_3 (n - 1) + c_4 \sum_{j=2}^n t_j + c_5 \sum_{j=2}^n (t_j - 1) + c_6 \sum_{j=2}^n (t_j - 1) + c_7 (n - 1)$$

במקרה הטוב ביותר, $\sum_{j=2}^n t_j = 1$ תלמיד ולכון n

$$T_1(n) = (c_1 + c_2 + c_3 + c_4 + c_7)n - (c_2 + c_3 + c_4 + c_7)$$

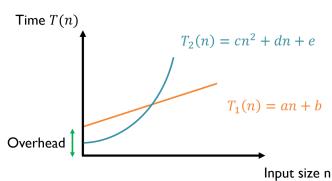
זו פ' לינארית!

במקרה הגרוע ביותר, $\sum_{j=2}^n (t_j - 1) = \frac{n(n+1)}{2}$, $\sum_{j=2}^n t_j = \frac{n(n+1)}{2} - 1$ ולכון $t_j = j$

$$T_2(n) = \frac{(c_4 + c_5 + c_6)}{2} n^2 + \left(c_1 + c_2 + c_3 + c_7 + \frac{c_4 - c_5 - c_6}{2} \right) n - (c_2 + c_3 + c_4 + c_7)$$

זו פ' פולינומיאלית ממעלה 2!

נשים לב כי לא משנה מה הקבועים, אם n מספיק גדול או

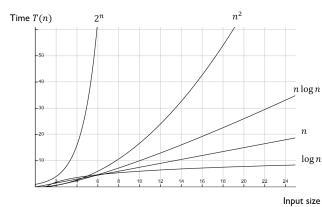


איור 3 : סיבוכיות הזמן במקרה הטוב ביותר והגרוע ביותר

הערה בעיקר אכפת לנו מניתוח אסימפטוטי של הסיבוכיות (כלומר n "מאוד גדול") בהינתן קלט כלשהו (אורץ המערך n במקרה הנ"ל).

הערה כשרצча לנתח את סיבוכיות האלגוריטם, לעיתים ניסוי לא יוזר לנו למצוא את הסיבוכיות שכן מאפיין זה הוא תלוי מערכת, ככלומר הוא מקומי ולא ניתן השוואתו בין אלגוריתמים שונים כמו ש策יך. במקרים זאת, נבצע את ניתוח מפושט (תיאורטי) על פי גודל הקלט בעזרת ייחדות קבועות עבור זמן ומקום. זה אפשר לנו לנתח באופן עמוק ובלתי תלוי את האלגוריתמים השונים.

הערה נסמן $T(n)$ את סיבוכיות הזמן ו- $S(n)$ את סיבוכיות הזיכרון.



איור 4 : כמה פ' סיבוכיות שימושיות

סיכום אסימפטוטיים

יהיו $f, g : \mathbb{N} \rightarrow \mathbb{R}^+$

- **חסם עליון** ($\mathcal{O}(g(n))$ - לכל היותר $f(n) \leq c \cdot g(n)$. קלומר,

$$\mathcal{O}(g(n)) = \{f(n) : \exists c > 0, n_0 \geq 1 : \forall n \geq n_0, f(n) \leq c \cdot g(n)\}$$

- **חסם תחתון** ($\Omega(g(n))$ - לכל הפחות $f(n) \geq c \cdot g(n)$. קלומר,

$$\Omega(g(n)) = \{f(n) : \exists c > 0, n_0 \geq 1 : \forall n \geq n_0, f(n) \geq c \cdot g(n)\}$$

- **חסם הדוק** ($\Theta(g(n))$ - בדיק $f(n) \geq c_1 \cdot g(n) \leq c_2 \cdot g(n)$. קלומר,

$$\Theta(g(n)) = \{f(n) : \exists c_1, c_2 > 0, n_0 \geq 1 : \forall n \geq n_0, c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)\}$$

תכונות של חסמים אסימפטוטיים

$$1. f(n) = \Theta(g(n)) \text{ אם } f(n) = \Omega(g(n)) \text{ וגם } f(n) = \mathcal{O}(g(n)).$$

$$2. (\text{רפלקסיביות}) f(n) = \mathcal{O}(f(n)) \text{ וכך גם עבור } \Omega, \Theta, f(n) = \mathcal{O}(f(n)).$$

$$3. (\text{סימטריה}) g(n) = \Theta(f(n)) \text{ אם } f(n) = \Theta(g(n)).$$

$$4. (\text{טרנזיטיביות}) f(n) = \mathcal{O}(h(n)) \text{ וגם } g(n) = \mathcal{O}(h(n)) \text{ אז } f(n) = \mathcal{O}(g(n)).$$

$$. \mathcal{O}(\mathcal{O}(f(n))) = \mathcal{O}(f(n)) .5$$

$$6. (\text{אדיטיביות}) \mathcal{O}(f(n) + g(n)) = \mathcal{O}(f(n)) + \mathcal{O}(g(n)).$$

$$. \mathcal{O}(f(n) \cdot g(n)) = \mathcal{O}(f(n)) \cdot \mathcal{O}(g(n)) .7$$

$$. \mathcal{O}(\log_a n) = \mathcal{O}(\log_2 n) .8$$

$$. \mathcal{O}\left(\sum_{i=0}^k a_i n^i\right) = \mathcal{O}(n^k) .9$$

תרגול

הוכחת תכונות של סימוני אסימפטוטיים

$$\forall n \geq n_0 = 1, f(n) \leq 1 \cdot f(n). f(n) = \mathcal{O}(n) \quad \bullet$$

$$f(n) = \mathcal{O}(g(n)) \iff g(n) = \Omega(f(n)) \quad \bullet$$

$$\begin{aligned} \exists c > 0, n_0 \in \mathbb{N} : (\forall n \in \mathbb{N} : n \geq n_0 \Rightarrow f(n) \leq c \cdot g(n)) \\ \exists c > 0, n_0 \in \mathbb{N} : \left(\forall n \in \mathbb{N} : n \geq n_0 \Rightarrow \frac{1}{c} f(n) \leq g(n) \right) \iff \\ g(n) = \Omega(f(n)) \iff \end{aligned}$$

$$f(n) = \Theta(g(n)) \iff g(n) = \Theta(f(n)) \quad \bullet$$

$$\begin{aligned} f(n) = \Theta(g(n)) \\ f(n) = \mathcal{O}(g(n)) \wedge f(n) = \Omega(g(n)) \iff \\ g(n) = \Omega(f(n)) \wedge g(n) = \mathcal{O}(f(n)) \iff \\ g(n) = \Theta(f(n)) \iff \end{aligned}$$

הוכיחי/הפרכי

$$\frac{f(n)}{g(n)} = \left| \frac{f(n)}{g(n)} \right| < 1, \forall n \geq n_0 \in \mathbb{N} \text{ נבחר כך ש-} \epsilon = 1 \text{ ולכן } \exists n_0 \in \mathbb{N} \text{ נבחר כך ש-} \epsilon = 1 \text{ ולכן } \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0 \Rightarrow g(n) = \Omega(f(n)) \quad .1 \\ f(n) < g(n)$$

$$2^n < c2^{\frac{n}{2}} \quad .2 \quad \text{לפחות } N \in \mathbb{N}, c > 0. \text{ יי רצח כי } \mathcal{O}(2^{\frac{n}{2}}) \neq 2^n = \Omega(2^{\frac{n}{2}}). f\left(\frac{n}{2}\right) = 2^{\frac{n}{2}}, f(n) = 2^n. f(n) = \Theta\left(f\left(\frac{n}{2}\right)\right) \quad \text{לא נכון!} \\ \text{ולכן } c > 2^{\frac{n}{2}}, \text{ כלומר } \frac{n}{2} \text{ נבחר כך ש-} n > \max\{2 \log c, N\} \text{ ונקבל את הרצוי.}$$

$$\text{ואז } f(n) = \mathcal{O}(g(n)), f' = \mathcal{O}(g(n)). f(n) = \mathcal{O}(g(n)) \Rightarrow f^2(n) = \mathcal{O}(g^2(n)) \quad .3 \\ \text{אם נבחר } f = f', g = g'. f \cdot f' = \mathcal{O}((g \cdot g')(n)) \quad \text{נקבל את הרצוי.}$$

$$2^n \neq \Theta(2^{\frac{n}{2}}) \quad .4 \quad \text{לא נכון!} \quad n = \Theta\left(\frac{n}{2}\right). f(n) = \Theta(g(n)) \Rightarrow 2^{f(n)} = \Theta(2^{g(n)})$$

$$\text{דוגמה: } n! = \prod_{i=1}^n i \leq \prod_{i=1}^n n = n^n. n! = \mathcal{O}(n^n) \quad \text{רוצח להתאמן על יכולות האינדוקציה שלנו ולכן נוכיח זאת שוב פעמיים.} \\ \text{בסיס: } 1! \leq 1^1 : (n=1) \quad \text{צעד: } (n+1)! \leq n^n (n+1) \leq (n+1)^n (n+1) = (n+1)^{n+1} : (n \rightarrow n+1)$$

הערה נראה בתרגיל כי אם $(\log f(n)) = \mathcal{O}(g(n))$ אז $f(n) \xrightarrow{n \rightarrow \infty} \infty$ ולכן אם נבחר $\log n! = \Theta(n \log n)$ אז נקבל $f(n) = n!, g(n) = n^n$ תוצאה חשובה כי משתמש בה כדי להוכיח סיבוכיות של אלגוריתמים שונים).

```

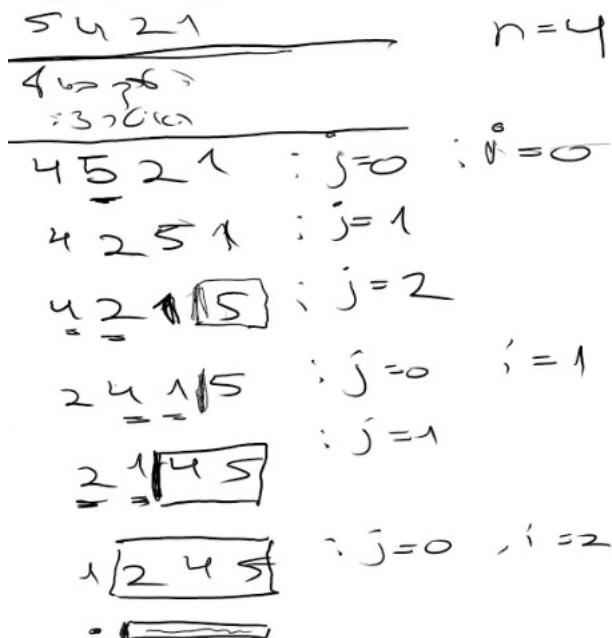
1 for i←0 to A.length-2 do
2   for j←0 to n-2-i do
3     if A[j]>A[j + 1] do
4       swap(A [j] , A [j + 1])

```

3 Algorithm

אחרי כל לולאה פנימית האיבר המקסימלי ברשא "מובעב" לסוף המערך (קדימה). בסוף כל ריצה חיצונית קיבל מערך ממוקן בגודל $i+1$ (ראו הדוגמה).

איור 5 : מיון بواسות בפעולה



תכונות של המערך האיטרציה ה- j -ית של הלולאה הפנימית

$$\forall t \leq j+1, A[t] \leq A[j+1] .1$$

$$A[j+2, \dots, n-1] \text{ לא השתנה.} .2$$

3. קבוצת האיברים בכל A לא השתנתה.

הוכחה: באינדוקציה על i

בסיס ($i=0$) : אחרי ה החלפה $[1], A[0] \leq A[1]$ ולכן $A[0] \leq A[1] \leq \dots \leq A[n-1]$. ואנו ב- $n-1$ מתקיימת. לא נגענו ב- $n-2$ ולכן 2 מתקיימת. לא שינו איברים, רק החלפנו את מיקומם ולכן 3 מתקיימת.

צעד (j) : נסמן $A[j+2] \geq A[j+1]$ אבל בכל מקרה $A[j+1] < A[j+2]$. נחלף בין $\alpha = A[j+1]$ אחרי ה החלפה ולבסוף $A[t] \leq \alpha \leq A[j+2]$ ולכן $A[j+2] \geq \alpha$. ומכנן תכונה 1 מתקיימת. תכונות 2 ו-3 מיידיות (אותה הוכחה כמו הבסיס). ■

תכונות של המערך אחרי האיטרציה ה- i -ית של הלולאה החיצונית

$$.1. A[n-i-1, \dots, n-1] \text{ ממשין.}$$

$$.2. \forall t \leq n-i-1, A[t] \leq A[n-i-1]$$

3. קבוצת האיברים ב- A לא השתנתה.

הערה נסמן מעתה והלאה $'$, $'2$, $'3$ עבור תכונות של הלולאה החיצונית ו- $1, 2, 3$ עבור תכונות של הלולאה הפנימית.

הוכחה: באינדוקציה על $2 \leq i \leq n-2$

בסיס ($i=0$) באופן ריק. $A[n-1] : (i=0)$ מ-3' ברור מ-3'.

צעד ($i \rightarrow i+1$) : מתכוна 1 עבור $j=n-0-2$, $A[t] \leq A[n-1]$ ו- $\forall t \leq n-i-2, A[t] \leq A[n-i-2]$, $j=n-(i+1)-2=n-i-3$ ולכן $A[t] \leq A[n-i-3]$.

מתכוна 1' עבור i , $A[n-i-2] \leq A[n-i-1, \dots, n-1]$ ממשין, וגם $A[n-i-1, \dots, n-1] \leq A[n-i-2]$ מ-2' עבור i .

ולכן $A[n-(i+1)-1, \dots, n-1] \text{ ממשין}$ כלומר $A[n-(i+1)-1, \dots, n-1] \text{ מ-3' מתקיים}$. ■

טענה מיוון בוועות (כמפורט באלגוריתם הבא) אכן ממיין מערך בסוף הריצה שלו.

הוכחה: נשים לב שכשמציבים $i = n-2$ ($t=0$ (תנאי הסיום של הלולאה) מקבלים כי $A[1, \dots, n-1]$ ממשין (מתכוна 1 של הלולאה החיצונית) וגם כי $A[1] \leq A[t] \leq A[n-1]$ ($t=0$ (מתכוна 2 של הלולאה החיצונית) וזה היחיד שמקיים את זה (שהוא לא 1) הוא $t=0$ ולכן $t=0$ הוא היחיד שמקיים את זה).

$$A[0] \leq A[1] \leq A[2] \leq \dots \leq A[n-1]$$

■ A ממשין.

שבוע | III Master Theorem |

הרצאה

בහינתו T , נרצה למצוא f כך ש-

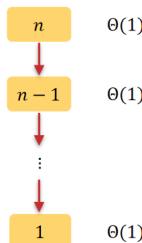
באלגוריתמים שנציג את פועלתו באופן איטרטיבי, נקבל לרוב משהו בסגנון $\sum c_i t_i$. באלגוריתמים ריקורסיביים (הפרד ומשול או אלגוריתמים שאחנו יכולים ליצג כתליים בערכים קודמים) נקבל משהו בסגנון $T(n) = T(n-1) + c$ לשם ניתוח, נניח כמה הנחות.

הנחות לניתוח אלגוריתמים

1. n מאד גדול (לא מספיק לנו מה קורה בהתחלה).
2. $T(1) = \theta(1)$ (קל לפתור מקרים קלים).
3. n הוא יפה (כלומר אם אנחנו מחלקים בנוסחה ב-2, אז n יהיה זוגי או חזקה של 2, כי זה לא ממש משנה).

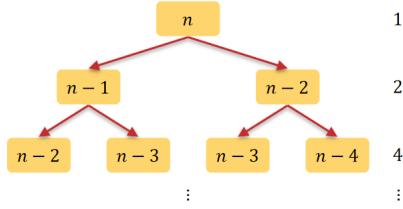
הערה לא תמיד נוכל להתעלם מקרים מיוחדים! נוכל רק לשאוף.

1. חישוב n . $T(n) = T(n-1) + \mathcal{O}(1)$ כי כדי לחשב את n מספיק לחשב את $n-1$ (שזה יקח $\Theta(1)$ ועוד כפול ב- n (שזה יקח $\Theta(1)$ כי אנחנו נמצאים תחת ההנחה שפעולות בסיסיות כמו אריתמטיקה נעשות בזמן קבוע). במקרה הזה נקבל $T(n) = \Theta(n^2)$.
2. אבל נראה דרך פורמלית לפטור דברים כאלה. מבחינה ויזואלית, במקרה הזה נקבל עץ ריקורסיבי שלא מתרחב לעומקו (ראו איור).

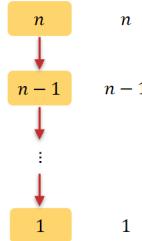


איור 6 : עץ הריקורסיבי בחישוב עצרת

2. חישוב סדרת פיבונאצ'י. נזכיר כי $\text{fib}(1) = \text{fib}(2) = 1$ וגם $\text{fib}(n) = \text{fib}(n-1) + \text{fib}(n-2)$ (נוסחת הנסיגה לחישוב הסדרה היא $T(n) = T(n-1) + T(n-2) + \mathcal{O}(1)$ (ניתן להגיע לכך באמצעות נוסחה הקודמת). הפתרון לשווה זה הוא $\Theta(2^n)$ (באופן היריסטי).
3. מילוי הוכנה. נסתכל עליו בתורו אלגוריתם ריקורסיבי: כדי למין מערך a באורך n , נמיין את הרשימה $-1-n$ -ית ואז נשים את האיבר האחרון במקום הנכון, ככלומר, $T(n) = T(n-1) + \mathcal{O}(n)$. נשים לב שגם סדרה חשבונית ולכן $\Theta(n^2)$. העץ הריקורסיבי במקרה זה הוא גם שrok כמו הדוגמה הראשונה, אבל הפעם העלות בכל רמה היא לנארית ולא קבועה (ראו איור).

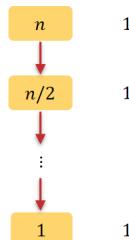


איור 7 : עץ הריקורסיבי בחישוב סדרת פיבונאצ'י



איור 8 : עץ הריקורסיבי במיון הכנסה

4. חיפוש ביןארי במערך ממון. הרעיון הוא להתסכל על האיבר המרכזי, אם הוא גדול מהאיבר הרצוי להסתכל בחצי השמאלי של המערך ואחרות בחצי הימני. בתוך התת-מערך לישוט בדיקת אותו הדבר עד שmaguiim לאיבר הרצוי. במקרה זה, קיבל $T(n) = \mathcal{O}(\log n)$. במקרה הזה עץ הריקורסיב שוב יהיה שroit, עם עלות קבועה אבל מספר הרמות קטן באופן לוגריתמי (ראו איור).



איור 9 : עץ הריקורסיב בחיפוש ביןארי

5. מיון מיזוג (הוזג [כאן](#)). במקרה הזה הנוסחה היא $T(n) = 2T\left(\frac{n}{2}\right) + \mathcal{O}(n) = \Theta(n \log n)$. עץ הריקורסיב הזה הוא גם מתרחב וגם עם עלות ליניארית וגם מספר רמות לוגריתמי (אל תראו איור כי אין כזה, הכי קרוב שיש זה [כאן](#)).

דרכי פתרון נוסחאות נסיגה

נעסק בפתרון בעיות מוחסוג לנו (דברים בסגנון הדוגמאות [הן](#)).

1. **שיטת ההצבה.** נניח פתרון (מאלגנטואיסチ או השראה אלוהית) ונוכיח שהוא מקיים את המשוואה.
2. **שיטת האיטרציה.** “נפרום” את נוסחת הנסיגה עד שנזהה תבנית ואז נשתמש בשיטת ההצבה לחוכחתה.
3. **משפט האב.** משפט שופטר נוסחאות נסיגה כמו קසם.

דוגמה נפתרו את סיבוכיות הזמן של מיזוג באמצעות שיטת הרצבה. $T(n) = \mathcal{O}(n \log n)$. נניח כי $T(n) = 2T\left(\frac{n}{2}\right) + n$.
 $\text{עבור } n \geq 2 \text{ באינדוקציה שלמה.}$
 $c \geq 2$.
 $4 = 2T(1) + 2 = T(2) \leq 2 \log 2 = 2 \cdot c : (n=2)$
 $\text{בבסיס } c \geq 2 \text{ נבחר } 2 = 2T(1) + 2 = T(2) \leq 2 \log 2 = 2 \cdot c : (n=2)$
 $\text{צעד } (2, \dots, n-1 \rightarrow n)$

$$T(n) = 2T\left(\frac{n}{2}\right) + n \stackrel{\text{説明}}{\leq} 2\left(c\frac{n}{2} \log \frac{n}{2}\right) + n \leq cn \log n - cn \log 2 + n = cn \log n - cn + n \leq cn \log n$$

דוגמה נפתרו את סיבוכיות הזמן של מיזוג הכנסה באמצעות שיטת האיתרציה.

$$\begin{aligned} T(n) &= T(n-1) + n = T(n-2) + (n-1) + n \\ &= T(n-3) + (n-2) + (n-1) + n \\ &\vdots \\ &= T(n-k) + \sum_{i=1}^k (n-i+1) \\ &= T(n-k) + nk - \frac{k(k-1)}{2} \\ &\stackrel{k=n-1}{=} T(1) + n(n-1) - \frac{(n-1)(n-2)}{2} \\ &= \frac{n^2}{2} + \frac{n}{2} = \mathcal{O}(n^2) \end{aligned}$$

עתה הסטודנטית המשקיפה תוכיה באינדוקציה כי אכן מתקיים $T(n) = \mathcal{O}(n^2)$.

משפט (משפט האב, Master Theorem).
 $T(1) = \theta(1)$ ו- $T(n) = aT\left(\frac{n}{b}\right) + n^c$ ו- $a, b \geq 1, c \geq 0$ יהיו כך שמתקיים אזי:
 $\log_b a < c$ אם $T(n) = \Theta(n^c)$
 $\log_b a = c$ אם $T(n) = \Theta(n^c \log_b n)$
 $\log_b a > c$ אם $T(n) = \Theta(n^{\log_b a})$

הערה אנחנו מחלקים ל- a -חלקים בגודל $\frac{n}{b}$ ומשלים n^c כדי לאחד את המקרים הללו, כולם:

a משמעו מספר תתי הבעיות.

b משמעו גודל כל תת בעיה.

c משמעו מה העלות בכל חלוקה אחרית שפתרנו את תתי הבעיות.

הערה הבינו חוזה בדוגמאות הלו וזהו את a, b, c וראו כיצד המשפט תואם לתוצאה שקיבנו.

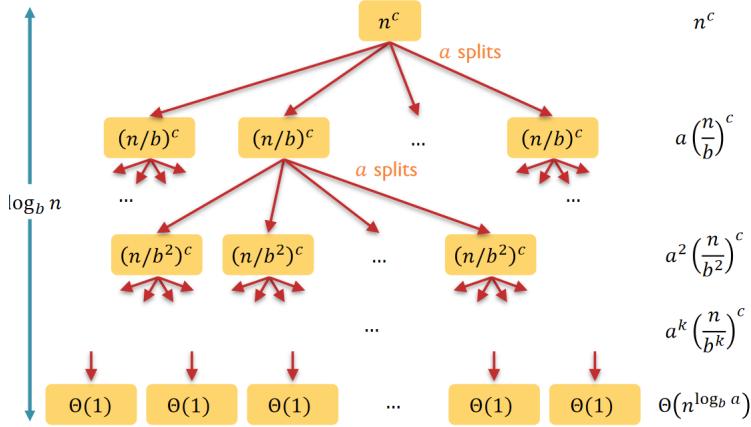
דוגמאות

1. $T(n) = T\left(\frac{n}{2}\right) + n^1$ - כאן a הכי דומיננטי (לא נשלם יותר מדי כל רמה, אבל יהיה לנו הרבה יותר בעיות).

$$\text{כואן } b \text{ דומיננטי (מהר מאוד נגיעה לסוף עץ הריקורסיבי).}$$

$$\text{כואן } n \text{ (מיון מיזוג) - כואן כל הרכיבים תורמים "בערך" אותו הדבר - לא יותר מדי לכיוון אחד.}$$

הוכחה: נבנה את עץ הריקורסיבי.



איור 10 : עץ הריקורסיבי במשפט האב

סביר קצת מה קורה באյור. ראשית, יש לנו n שלבים כי כל פעם אנחנו מחלקים ב- b ותחולנו מ- a . בכל תט בעיה אנחנו משלמים $(\frac{n}{b^k})^c$ כאשר k הוא העומק שלנו בעץ. בכל שלב אנחנו מתפצלים a פעמים ולכן סה"כ בכל רמה נשלם $a^k (\frac{n}{b^k})^c$ על מיזוג תתי הבעיות. נטען כי בرمמה התחתונה ביותר יש $\Theta(n^{\log_b a})$ תט בעיות בעלות (1) Θ כל אחת. למה? כי מספר תט הבעיות בرمמה ה- n הוא $n^{\log_b a} = a^{\log_b n}$ והוא שווה $a^k (\frac{n}{b^k})^c$ שקיים הטענה. נוכיח כי אם נבצע \log_b על שני האגפים ונשתמש בחוקי הלוגריתם נקבל את הדבר ולכן מהיות \log_b פ' חח' עוזי גם הביטויים המקוריים שווים. לכן בכלל שהעלות של כל עלה בرمמה התחתונה היא (1) Θ, אז הערות בرمמה זו היא גם $\Theta(n^{\log_b a})$.

סה"כ העבודה היא

$$\begin{aligned} T(n) &\stackrel{\text{המראת}}{=} a^k T\left(\frac{n}{b^k}\right) + n^c \sum_{i=0}^{k-1} \left(\frac{a}{b^c}\right)^i \\ &\stackrel{\text{סיה"כ}}{=} \Theta(n^{\log_b a}) + \sum_{i=0}^{\log_b n-1} a^i \left(\frac{n}{b^i}\right)^c = \Theta(n^{\log_b a}) + n^c \sum_{i=0}^{\log_b n-1} \left(\frac{a}{b^c}\right)^i \end{aligned}$$

נניח להגיע לכך באמצעות שיטת האיטרציה והוכחה באינדוקציה. המחבר הראשון הוא הערות של העלים והשני זה הערות של כל השאר. נשים לב כי $\frac{a}{b^c}$ הוא ביטוי המפתח. אם הוא קטן מ-1 (כלומר $c < \log_b a$) אז הביטוי השני (עלות המיזוג) היא הדומיננטית, אם הוא שווה ל-1 אז שני הביטויים מאזונים ואם הוא גדול מ-1 ($\log_b a > c$) אז הביטוי הראשון (עלות העלים) הוא הדומיננטי.

עבור המקרה הראשון ($\frac{a}{b^c} < 1$): נשים לב האיבר השני הוא בעצם טור גאומטרי ולכן

$$\sum_{i=0}^{\log_b n-1} \left(\frac{a}{b^c}\right)^i = \frac{\left(\frac{a}{b^c}\right)^{\log_b n} - 1}{\left(\frac{a}{b^c}\right) - 1} = \frac{1 - \left(\frac{a}{b^c}\right)^{\log_b n}}{1 - \left(\frac{a}{b^c}\right)} < \frac{1}{1 - \left(\frac{a}{b^c}\right)} < \text{constant}$$

כלומר הגודל של הביטוי הזה לא תלוי בגודל הקלט. לכן $T(n) = \Theta(n^c)$ והגדר ש-אי איזי ($\log_b a < c$) ממעלה נמוכה יותר מהשני ולכן ”_nb1“ (בביטויי הראשון) הוא פולינום ממעלה נמוכה יותר מהשני ולכן ”_nb2“ (בביטויי הראשון).

עבור המקרה השני ($\frac{a}{b^c} = 1$) נקבל $\sum_{i=0}^{\log_b n-1} 1 = \log_b n$ ולכן ($n^c \log_b n$) $T(n) = \Theta(n^c \log n)$ והראשון הוא פולינום ממעלה שווה לשני ולכן ”_nb3“.

עבור המקרה השלישי ($\frac{a}{b^c} > 1$) $\sum_{i=0}^{\log_b n-1} \left(\frac{a}{b^c}\right)^i = \frac{\left(\frac{a}{b^c}\right)^{\log_b n}-1}{\left(\frac{a}{b^c}\right)-1} = \Theta\left(\left(\frac{a}{b^c}\right)^{\log_b n}\right)$ ולכן $\sum_{i=0}^{\log_b n-1} \left(\frac{a}{b^c}\right)^i = \Theta\left(\frac{n^c}{b^{c \log_b n}} a^{\log_b n}\right) = \Theta\left(\frac{n^c}{n^c} n^{\log_b a}\right) = \Theta(n^{\log_b a})$

■ $T(n) = n^{\log_b a}$. (הביטוי הראשון הוא פולינום ממעלה שווה לשני ולכן ”_nb3“).

תרגול

נותראות הנטיגת שבן אלו עוסוק הן מחלוקתם בעיות כאשר אפשר לפטור עם משפט האב (אבל לא כל המשוואות עונთ על זה, ראו דוגמה פיבונאצ'י).

חזרה על החוכחה של משפט **האב**.

המקרה הראשון במשפט האב הפשט מסתכל על המצב שבו העבודה שנעשתה בעליים גוברת על פעולת האיתוי בכל רמה, ולכן היא הדומיננטית ולכן היא תהיה זמן הריצה. המקרה השני הוא המקרה שבו העבודה בעליים ושורש היא אותו הדבר, ולכן גם (לא נוכיח) בכל הרמות באמצעות זמן הריצה הוא אותו הדבר (אם יש דברים שווים שמאגפים משחו מונווטוני, הכל שווה) ולכן זמן הריצה יהיה מספר הרמות כפול העבודה בכל רמה. המקרה השלישי הוא המקרה שבו עבודת האיתוי בשורש גדולה מהעבודה בעליים ולכן היא זו שתשלוט.

דוגמאות

$$T(n) = \Theta(n^{\log_2 2} \log n) \text{ שכן נשתמש במקרה השני ונקבל } \log_2 2 = 1 = c, a = b = 2, c = 1 . T(n) = 2T\left(\frac{n}{2}\right) + n . 1 . \Theta(n \log n)$$

$$T(n) = \Theta(n^{\log_3 20}) \simeq \Theta(n^{2.726}) \text{ שכן נשתמש במקרה הראשון ונקבל } \log_3 20 > 2 = c . T(n) = 20T\left(\frac{n}{3}\right) + n^2 . 2$$

$$T(n) = \Theta(n^1) = \Theta(n) \text{ שכן נשתמש במקרה השלישי ונקבל } \log_2 1 = 0 < 1 . T(n) = T\left(\frac{n}{2}\right) + n . 3$$

משפט (משפט האב המורחב) תהי $T : N \rightarrow \mathbb{R}^+$ המוגדרת ע”י:

$T(n) = \Theta(n^{\log_b a})$ אם קיימים $c > \epsilon > 0$ כך ש-

$T(n) = \Theta(n^{\log_b a} f(n)) = \mathcal{O}(n^{\log_b a - \epsilon})$ (i) אם $f(n) \leq n^{\log_b a}$ הוא ממש גדול-מ-($n^{\log_b a - \epsilon}$) או

$T(n) = \Theta(n^{\log_b a} \log n) = \Theta(n^{\log_b a})$ (ii) אם $f(n) \geq n^{\log_b a}$ (בבדיקה כמו במקרה השני במשפט האב הפשט) או

(תנאי) $af\left(\frac{n}{b}\right) \leq cf(n), \forall n \geq N$ וקיימים $c < 1, N \in \mathbb{N}$ כך ש- $f(n) = \Omega(n^{\log_b a + \epsilon})$ (iii)

הרגולריות, יש דעיכה כשמתקräבים לעליים) או $T(n) = \Theta(f(n))$

$T(n) = 27T\left(\frac{n}{3}\right) + n^2 \log n$.
 משבט האב הפשט לא יעבוד, אבל أولי המורחב כן. נזכיר כי אסימפטוטית,
 $n^2 \log n = \mathcal{O}\left(n^{3-\frac{1}{2}}\right)$.

$$\lim_{n \rightarrow \infty} \frac{n^2 \log n}{n^{2.5}} = \lim_{n \rightarrow \infty} \frac{\log n}{\sqrt{n}} \stackrel{L'H}{=} \lim_{n \rightarrow \infty} \frac{\frac{1}{n \ln 2}}{\frac{1}{2\sqrt{n}}} = \lim_{n \rightarrow \infty} \frac{2}{\ln 2 \sqrt{n}} = 0$$

ולכן מטענה שראינו בתרגול 1, קיבל את הרצוי ולכן משבט האב המורחב,

$n^{\frac{1}{2}} \stackrel{?}{=} n \log n$. שוב, משבט האב הפשט לא יעבוד אבל أولי המורחב כן.
 $\log_b a = \log_4 2 = \frac{1}{2}$.
 $T(n) = 2T\left(\frac{n}{4}\right) + n \log n$.
 ננסה להשתמש ב מקרה השלישי. נבחר $0 < \epsilon < \frac{1}{2}$ ו $\Omega(n^{\frac{1}{2}+\epsilon}) = \Omega(n)$ אפשר להוכיח את זה. נראה את תנאי הרגולריות. נרצה למצוא $c \in (0, 1)$ כך ש-

$$\begin{aligned} 2\frac{n}{4} \log \frac{n}{4} &\leq cn \log n \\ \frac{n}{2} (\log n - \log 4) &\leq cn \log n \\ 0 &\leq n ((c - \frac{1}{2}) \log n + 1) \end{aligned}$$

כמעט תמיד. נשים לב כי כל $c < \frac{1}{2}$ מקיים זאת. לכן משבט האב המורחב קיבל

$T(n) = 3T\left(n^{\frac{1}{3}}\right) + \log \log n$.
 פה לא נוכל להשתמש סתם במשפט האב, אבל כן נוכל להחיליף משתנה. נבחר $n^m = \log n$, כלומר $m = \log \log n$.
 $S(m) = 3T\left(\frac{m}{3}\right) + \log m$ ו $S(m) = T(2^m) = 3T\left(2^{\frac{m}{3}}\right) + \log m = 2^m$.
 $S(m) = \Theta(m^1)$.
 $\log m = \mathcal{O}\left(m^{1-\frac{1}{2}}\right)$.
 משם נסיק כי $m^1 \stackrel{?}{=} \log m$.
 $\log_3 3 = 1$
 ולכן $T(n) = T(2^m) = S(m) = \Theta(m) = \Theta(\log n)$

$n^{\log_3 9} = n^2 \stackrel{?}{=} n^2 \log n$.
 $T(n) = 9T\left(\frac{n}{3}\right) + n^2 \log n$.
 משבט האב המורחב לא יעבוד כאן בכלל!
 נניח בשליליה כי $n^2 = \mathcal{O}(n^{2-\epsilon})$ כלומר: במקרה הראשון לא יעבוד.
 $n^2 \log n \neq \Theta(n^2)$ ולכן גם במקרה השני לא יעבוד.
 $\lim_{n \rightarrow \infty} \frac{n^2 \log n}{2^{2+\epsilon}} = \lim_{n \rightarrow \infty} \frac{\log n}{n^\epsilon} \stackrel{L'H}{=} \lim_{n \rightarrow \infty} \frac{1}{\epsilon \ln 2} n^{\epsilon-1} = 0$.
 $\epsilon > 0$.
 במקרה השלישי.
 נשתמש בשיטת האיטרציה.

$$\begin{aligned} T(n) &= 9T\left(\frac{n}{3}\right) + n^2 \log n \\ &= 9 \left(9T\left(\frac{n}{3^2}\right) + \left(\frac{n}{3}\right)^2 \log \frac{n}{3}\right) + n^2 \log n \\ &= T\left(\frac{n}{3^2}\right) + \sum_{i=0}^1 9^i \left(\frac{n}{3^i}\right)^2 \log \frac{n}{3^i} \end{aligned}$$

מכאן נחש כי לכל n מתקיים $T(n) = T\left(\frac{n}{3^m}\right) + \sum_{i=0}^m 9^i \left(\frac{n}{3^i}\right)^2 \log \frac{n}{3^i} \leq m \leq \log_3 n$ ובסוף נציב

$$T(n) = \dots = \Theta(n^2 \log^2 n) \text{ (העומק המקסימלי בעץ, המקרה הסופי) ואז קיבל } m = \log_3 n$$

$$\text{טענת עזר אם } \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0 \text{ אז } f(n) \neq \Omega(g(n))$$

שבוע 1 | Quick Sort & Comparison Based Sorting

הרצאה

בבעיית המינון נתונינו לנו n מספרים במערך, והמטרה שלנו היא למיין אותם לערך עם ערכיהם בסדר עולה. מיון הוא בעיה מאוד חשובה כי היא יסודית (משתמשים בה בהרבה אלגוריתמים אחרים), מוגנת יש הרבה אלגוריתמים, נרצה לדעת מי מהם הכى "טוב" ובנוסח ידועה הדריך הכי מהירה למיין.

ישנם שני סוגים מיונים: מיון משווים ומיונים לא משווים. עד כה כל מה שראינו היו מיון משווים. נראה בהמשך כי בהכרח שמיון הכנסה הוא $S(n) = \Omega(n)$ וגם $T(n) = \Omega(n \log n)$. מיונים ללא השוואה נקבע כי $S(n) = \Omega(n)$.

בחרצאה זו עוסוק בעיקר במיון השוואתי אחד - QuickSort - שהומצא ב-1961 ועדין משתמשים בו הרבה עד היום. מיון זה הוא מיון במקומות, ככלומר הוא לא דורש עוד זיכרון מעבר למערך עצמו (עד כדי מיקומיים קבועים) ולכן הוא נחשב $S(n) = \Theta(n)$. זמן הריצה במקרה הגרוע שלו הוא $\Theta(n^2)$ ובמקרה הממוצע הוא $\Theta(n \log n)$.

הרענון מיון מהיר הוא מיון בשיטת הפרד ומשלול (כמו מיון מיזוג ודמויים). החלק של ה"הפרד" הוא לחלק את המערך לשני חלקים, שהמונה שמספריד ביניהם הוא ה-Pivot (הציר). נרצה ציר כך שכל הערכים משמאל לציר יהיו קטנים מכל האיברים מימין לציר. החלק של ה"משלול" הוא אחורי שמיקמו את הцентр, נמיין באופן ריקורטיבי כל צד בנפרד.

```

1 QuickSort(A,l,r) is
2     if l < r do
3         m ← Partition(A,l,r)
4         QuickSort(A,l,m-1)
5         QuickSort(A,m+1,r)
6     enddo
7     return m
  
```

Algorithm 4: מיון מהיר, עבור $i = l$ נעצור את המיון. $A[m : n]$ יישאר במקומות כי הוא הцентр.

אלגוריתם Partition האלגוריתם ירוץ על כל המערך, יבחר ציר שרירותי וימקם אותו במקום הנכון כך שהאיברים מימינו יהיו גדולים מהאיברים משמאלו. האלגוריתם עצמו (בקצת יותר פירוט) הוא:

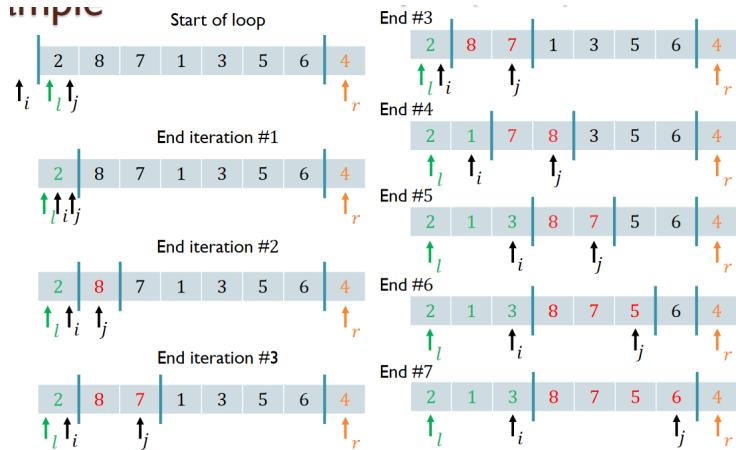
נחלק את המערך לארבעה חלקים, Left, Right, Pivot, Unseen בהתחלה נבחר את ה-Pivot להיות האיבר הימני ביותר ו-Left, Right, Unseen יהיה כל המערך. בסוף Unseen יהיה ריק וכל האיברים חוץ מהцентр יהיו ב-Right או ב-Left. כל איטרציה עוסקת באיבר אחד שטרם עברנו עליו. האינדקסים j, i הם אלו שיקבעו את Left ו-Right. אם האיבר החדש גדול מהцентр, נזוז את j אחד קדימה ובקצ' נכניס את האיבר Left. אם הוא קטן מהцентр, נזוז את i ו- j אחד קדימה ונחליף בין האיבר הנוכחי לאיבר האחרון Left. בסוף נמקם את הפיבוט במקום המקורי באמצעות החלפה בין האיבר i -i לאיבר האחרון.

```

1 Partition(A,l,r) is
2     i←l-1
3     for j←l to r-1 do
4         if A [j] ≤ A [r] do
5             i←i+1
6             Exchange(A[i],A[j])
7     Exchange(A[i + 1],A[r])
8     return i+1

```

אלגוריתם ה-Partition



איור 11 : דוגמת ליריצת אלגוריתם ה-Partition

ניתוח האלגוריתם אם כל האיברים יוצאים ממשאל, נקבל תתי בעיות בגודל $1 - n - 0$. אם האיברים יוצאים בדיקות חצי חצי בשמאלי ובמימין

או נקבל תתי בעיות בגודל $\frac{n}{2}$ ובמקרה הכללי נקבל תתי בעיות בגודל $q - 1 - 1 = q - 2$.

אם המערך כבר ממוקן או ממוין הפוך נקבל $T(n) = T(0) + T(n-1) + \Theta(n)$ (כי אנחנו מחשבים תת בעיה בגודל 0, בעיה בגודל

$n-1$ ומשקיעים ב-Partition(n) $\Theta(n)$ כי אנחנו חייבים לróż על הכל והפתרו לנוסחה זו היא $T(n) = \Theta(n^2)$ וזה המקרה הרע.

במקרה שהציר תמיד נמצא במרכז נקבל $T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n)$ וזה משפט האב הוא $T(n) = \Theta(n \log n)$

במקרה הגרוע ביותר ($n = 1$) $T(n) = \max_{0 \leq q \leq n-1} \{T(q) + T(n-q-1)\} + \Theta(n)$ ווכיח בשיטת ההצבה כי

$T(n) = \mathcal{O}(n^2)$.

הוכחה: בסיס (1) $T(1) = 1 \leq c \cdot 1^2$:

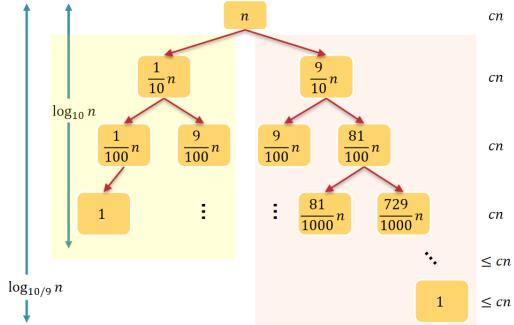
צעד $(1, \dots, n-1 \rightarrow n)$

$$\begin{aligned}
 T(n) &= \max_{0 \leq q \leq n-1} \{T(q) + T(n-q-1)\} + \Theta(n) \\
 &\leq \max_{0 \leq q \leq n-1} \left\{ cq^2 + c(n-q-1)^2 \right\} + \Theta(n) \\
 &= c \max_{0 \leq q \leq n-1} \left\{ q^2 + (n-q-1)^2 \right\} + \Theta(n^2) \\
 &\stackrel{(*)}{\leq} c(n-1)^2 + \Theta(n) = cn^2 - c(2n-1) + \Theta(n) \\
 &\leq cn^2
 \end{aligned}$$

■ (*) זו בעצם פרבולה מחייבת בקטע סגור ולכון היא מקסימלית בקצוות.

באוטו האופן ניתן להוכיח כי $\Omega(n^2) = T(n)$ ומשם נסיק כי $\Theta(n \log n)$ במקרה ההפוך (עליו נדבר בהמשך) הוא המקרה הכי טוב, כמובן.

דוגמה נניח כי ה-Partition מחלק בכל שלב את המערך למערך בגודל $\frac{9}{10}$ מהמערך המקורי. נקבל משהו כזה



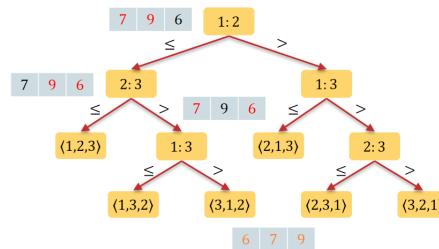
איור 12 : עץ רקורסיבי של מיון מהיר עבור חלוקה של 1-9

סה"כ נקבל $(n \log n) + \Theta\left(\frac{n}{10}\right) + \Theta\left(\frac{9n}{10}\right)$. הענף השמאלי הרובה יותר קטן מהענף השני, הרקורסיה נגמרה ב- $\frac{9}{10}n$ ובכל רמה נעשה $\Theta(n \log n)$. זה נכון לכל חלוקה פרופורציונית (כלומר שחלוקת את המערך לחלקים עם יחס ביניהם ולא אחד הצדדים קבוע או משווה בסגנון כמו במקרה ההפוך ביותר).

הערה בחירת ה策יר להיות איבר אקראי כל פעם מוגדר לטרפז ניסיון של תוקף להאט את המיון באמצעות השמת האיברים הגדולים ביותר.

משפט כל מיון מבוסס השוואות הוא $\Theta(n \log n)$.

הוכחה : נתאים לכל אלגוריתם עץ החלטה שמסתכל על ההשוואת של האלגוריתם. נביט בדוגמה



איור 13 : עץ החלטה של אלגוריתם מיון כלשהו

כآن $j : i$ מסמן את האלגוריתם משווה בין האיבר באינדקס ה- i לאיבר באינדקס ה- j . בדוגמה זו, אנחנו משווים בהתחלה את 9, 7, 6 ונכנסים ל מקרה שבו האיבר הראשון קטן-שווה השני. וזה במקרה שהאיבר הראשון גדול מהשלישי וכו'. זה עץ ביןاري שהקודקודים הפנימיים שלו הם שני אינדקסים והעליהם הם פרמטריזציות של המערך המקורי והענפים הם התוצאה של ההשוואה בקודקוד מעיליהם. הרצה של האלגוריתם היא בסך הכל מסלול דרך העץ.

כל עץ החלטה של אלגוריתם תקין חייב לכלול לפחות את כל הpermotaciyot של המערך המקורי (כי כל מערך חייב להתמיין נכון). יכול להיות שיש יותר עליים מכלל הpermotaciyot במקרה שהאלגוריתם עבר על משהו פשוט או עשו שהוא לא יעיל, אבל יש לפחות את כולם. לכן לעץ חייב להיות לפחות $n!$ עליים.

החסם התיכון על גובה העץ הוא החסם התיכון על מספר הצעדים הדרושים לשם מיזון המקרה הגרוע ביותר עבור כל אלגוריתם מיזון מבוסס השוואות.

נסמן את עומק העץ ב- d , לכן יש לו לפחות $\log n! \leq d$. חיבר להתקיים בסוף כי $n! \geq (n \log n)^d$ ולכן $\log n! = \Theta(n \log n)$ ומשם נסיק כי $\Omega(n \log n) = \Omega(\Theta(n \log n)) = \Omega(n \log n)$ ולבסוף $\Omega(n \log n)$ הוא ערך קבוע.

$$\log n! = \Theta(n \log n) \text{ ואמנם } \log n \leq \sum_{i=1}^n \log i \stackrel{\text{באיינדוקציה}}{\leq} \sum_{i=\frac{n}{2}}^n \log \frac{n}{2} = \frac{n}{2} \log \frac{n}{2} \quad (*)$$

$$\blacksquare \quad \log n! = \log \prod_{i=1}^n i = \sum_{i=1}^n \log i \stackrel{\text{באיינדוקציה}}{\geq} \sum_{i=\frac{n}{2}}^n \log \frac{n}{2} = \frac{n}{2} \log \frac{n}{2} \quad (*)$$

Algorithm	Space	Worst Case	Best Case	Average Case	Random Case
Bubble Sort	$\Theta(n)$	$\Theta(n^2)$	$\Theta(n)$	$\Theta(n^2)$	$\Theta(n^2)$
Insertion Sort	$\Theta(n)$	$\Theta(n^2)$	$\Theta(n)$	$\Theta(n^2)$	$\Theta(n^2)$
Merge Sort	$\Theta(n \log n)$	$\Theta(n \log n)$	$\Theta(n \log n)$	$\Theta(n \log n)$	---
Quick Sort	$\Theta(n)$	$\Theta(n^2)$	$\Theta(n \log n)$	$\Theta(n \log n)$	$\Theta(n \log n)$

- Lower bounds: $T(n) = \Omega(n \log n)$ and $S(n) = \Omega(n)$

טבלה 3 : טבלת סיכום של מיזונים מבוססי השוואות

תרגול

מושגי יסוד

1. מרחב הסתברות זו Ω כאשר Ω הוא מרחב מדגם ו- P היא פונקציית הסתברות.

2. מרחב מדגם הוא קבוצה Ω המתארת את כל המאורעות האפשריים בניסוי מסוים.

3. מאורע הוא תת קבוצה של מרחב מדגם Ω .

4. אוסף כל המאורעות האפשריים יוסמן ב- 2^Ω (מספר תת קבוצות של Ω).

5. מאורע אטומי מאורע עם איבר אחד (יחידו).

6. מאורעות נקראים זרים אם $A \cap B = \emptyset$ עבור $A, B \subseteq \Omega$.

.7. ה' $P : 2^\Omega \rightarrow [0, 1]$ היא פונקציית הסתברות המקיים:

$$\begin{aligned} & \cdot \sum_{w \in \Omega} P(\{w\}) = 1 \text{ (i)} \\ & .P(A) = \sum_{w \in A} P(\{w\}), \forall A \subseteq \Omega \text{ (ii)} \end{aligned}$$

דוגמה בהטלת קובייה, $\Omega = \{2, 4, 6\}$. מאורע יכול להיות לדוגמה הטלת של מספר זוגי, כלומר $\{2, 4, 6\}$.

$P(\{i\}) = \frac{1}{6}$ הם זרים. עבור קובייה הוגנת נוכל להגיד פ' הסתברות $i \in \{1, 3, 5\}$

$$.P([4]) = \sum_{i=1}^4 P(\{i\}) = \frac{2}{3}$$

(הסתברות איחוד). ההסתברות שיצא מספר קטן ממש מ-5 היא

תכונות

יב. $\langle \Omega, P \rangle$ מרחב הסתברות אזי:

$$.P(\emptyset) = 0 .1$$

$$.2. (\text{אדייטיביות}) \text{ לכל אוסף סופי של מאורעות זרים } A_1, \dots, A_n \text{ מתקיים}$$

$$.P(A) \leq P(B), \text{ אם } A \subseteq B \text{ מקיימים}$$

$$.4. \text{ לכל מאורע } A \text{ מתקיים } P(A) \leq 1$$

$$.5. \text{ לכל מאורע } A \text{ מתקיים } P(A) + P(A^c) = 1 \text{ כאשר } A^c = \Omega \setminus A \text{ המאורע המשלים.}$$

$$.6. (\text{תת-אדייטיביות}) \text{ לכל מאורעות } A, B \text{ מתקיים } P(A \cup B) \leq P(A) + P(B)$$

■ $.P(A) \leq P(B) + \underbrace{P(B \setminus A)}_{\geq 0} = P(B)$ הוכחה: נוכיח את תכונת המונוטוניות.

הגדרה הסתברות של A בהינתן B (כבר קרה, מה הסיכוי שגם A עכשו יקרה) מסומן ב-

הערה האינטואיטיבית לנוסחה זו היא שאחרי ש- B התרחש, ניכנס לעולם שכלו B , ואו הסיכוי שגם A יקרה הוא החיתוך של A עם B עם נרמול להסתברות של B .

דוגמה סכום שני מספרים שהתקבלו בהטלת זוג קוביות הוגנות גדול ממש מ-10 (B). מה ההסתברות שבקובייה הראשונה יתקבל מספר גדול

$$.A = \{6\} \times [6], B = \{(6, 5), (5, 6), (6, 6)\}, \Omega = [6] \times [6] \text{ ממש מ-5 (A). במקרה זה}$$

$$P(A|B) = \frac{P(\{(6, 6), (6, 5)\})}{P(\{(6, 5), (5, 6), (6, 6)\})} = \frac{\frac{2}{36}}{\frac{3}{36}} = \frac{2}{3}$$

הגדרה מאורעות A, B יקראו בלתי תלויים אם מתקיים $P(A \cap B) = P(A) \cdot P(B)$ או באופן שקול

אחרת מאורעות אלו יקראו תלויים.

דוגמה בהטלה שתי קוביות, נגידר את A להיות המאורע בו יצא בהטלה הראשונה 6 ו- B המאורע בו יצא בהטלה השנייה 6.

$$P(A) \cdot P(B) = \frac{1}{6} \cdot \frac{1}{6} = \frac{1}{36} = P(A \cap B)$$

לכן האירועים האלה (מעבר לאינטואיטיבית) הם בלתי תלויים.

עם זאת, אם A הוא המאורע בו בהטלה הראשונה יצא 6 ו- B הוא המאורע בו סכום ההטלות הוא 5, קיבל שםם כן תלויים. נראה זאת. $P(A) \cdot P(B) = \frac{1}{6} \cdot \frac{4}{36} = \frac{4}{6 \cdot 36} \neq 0 = P(A \cap B)$

הגדרה משתנה מקרי (מ"מ) הוא פונקציה $X : \Omega \rightarrow \mathbb{R}$

הערה מ"מ פשוט מצמידה לכל איבר במרחב המדגם מספר ממשי שיישמש לה כנצח.

נסמן $P(X \in A) = P(\{w \in \Omega : X(w) \in A\})$, כלומר $P(X = x) = P(\{w \in \Omega : X(w) = x\})$.
שorthand המשניתה המקרי יהיה שווה לאיושחו נציג $x \in A$, היא היחסות של כל האיברים במרחב המדגם כך שהוא משלם הוא מספר ב- \mathbb{R} . $A \subseteq \mathbb{R}$.

דוגמה בהטלה קוביה, נגידר i לחלוון, נכון להגדר מ"מ מצינו (אינדיקטור) $P(\{1, 2\}) = P(X \leq 2) = \frac{2}{6} = \frac{1}{3}$ והיחסות להטיל מספר קטן שווה ל-2 הוא $P(X = 2) = \frac{1}{6}$ היחסות להטיל 2 היא $P(Y = 1) = \frac{1}{2}$.

הגדרה ימי מ"מ X אזי התוחלת של X היא

$$E[X] = \sum_{w \in \Omega} X(w) P(w) = \sum_{x \in \mathbb{R}} x \cdot P(X = x) = \sum_{x \in \text{Im } X} x \cdot P(X = x)$$

הערה אינטואיטיבית, התוחלת היא הממוצע של המ"מ.

דוגמה הטלת קוביה עם $E[X] = \sum_{i=1}^6 i P(X = i) = \frac{1}{6} \sum_{i=1}^6 i = 3.5$. $\forall i \in [6], X(i) = i$ אינדיקטטור.

$$E[Y] = 0 \cdot P(X = 0) + 1 \cdot P(X = 1) = \frac{1}{2}$$

נסיק כי $E[X] = P(X = 1)$ עבור X מ"מ אינדיקטטור.

תבונות

1. (מוניוטוניות) אם $X \leq Y$ (כלומר $X(w) \leq Y(w), \forall w \in \Omega$) אזי $E[X] \leq E[Y]$.

2. (lienarיות) $E[\alpha X + \beta Y] = \alpha E[X] + \beta E[Y]$.

3. (תוחלת באזיאת תלות) אם X, Y בת"ל אז $E[X \cdot Y] = E[X] \cdot E[Y]$.

$$. E[X] = \sum_{k=1}^{\infty} P(X \geq k) \text{Im} X \subseteq \mathbb{N}_0$$

דוגמה בפארק שעשויים משחקים משחק. מולנו n כדורים ו- m תאים. ההסתברות שכדור יכנס לתא ה- j -י היא $\frac{1}{m}$ (הסתברות אחידה). נגידר מ"מ X_j^i המציין את המאורע "כדור ה- i נכנס לתא ה- j " (כלומר יש לנו הרבה מ"מ מצינימ שווים).

מה הסיכוי שהכדור הראשון נכנס לתא שמספרו לכל היותר k ? נסתכל על $1 \leq j \leq k$, לכן

$$P\left(\sum_{j=1}^k X_j^1 = 1\right) = P\left(\bigcup_{j=1}^k (X_j^1 = 1)\right) = \sum_{j=1}^k P(X_j^1 = 1) = \sum_{j=1}^k \frac{1}{m} = \frac{k}{m}$$

שבוע 1 | IV Introduction to Hashing, Open Addressing & Non-Comparison Sorting

הרצאה

נעסוק במבנה נתונים חדש - טבלאות גיבוב. הרבה פעמים נדרש להיות יכולים לגשת לתוכנים בזמן ממוצע ב-(1) \mathcal{O} מAGER מAGER מAGER גAGER גAGER של מידע. בנוסף, לא בהכרח של מפתחות שביצורים ניגש לאיברים הללו יש סדר מסוים.

דוגמה מאגר מידע של לקוחות בנק לפי ת"ז - יש הרבה קטלה ונרצה לשלו מהר משתמש. בדוגמה זו בעיקר עוסוק בשננות ונשווה טבלאות גיבוב למבני נתונים אחרים.

נרצה להיות יכולים לחפש, להכניס ולהוציא איברים בזמן ממוצע של (1) \mathcal{O} . בנגדוד ליותרם שבתים המدد שלנו היה המקרה הגורע ביותר, פה נתעניין במקרה הממוצע.

מערכות לא עוזרים ממש במקרים קטלה לא ממש עוזר, אפילו שניגים לאיברים בהם ב-(1) \mathcal{O} , כי יש 10^9 ת"ז ומערך בגודל הזה הוא לא יעיל מבחינת זיכרון.

ኖכל במקומות זאת לקבל מפתח, להעביר אותו לפ' (Hash Function) ואז לשים את הערך באינדקס שנתקבל מפ' הגיבוב.

פונקציות גיבוב

1. כתוב ישיר - $(k) = h$. פ' זו לא פותרת לנו את הבעיה בשימוש במערך שכן זו בדיקת הערך שפה ניגשים לאיברים במערך.
2. החזורת חלק מהערך המושם. לדוגמה, בת'ז, $(k_1 \dots k_9) = k_8 k_9 \dots k_1$. פ' זו עוזרת מבחינת מקום, שכן תדרוש רק 100 תאים ולא מיליארד, אבל יכוליםים להיווצר ממנה הרבה התנשויות (כל שתי ת"ז שנגמרו באוותן שתי ספרות).

כליל את המושג. טבלה גיבוב היא מבנה נתונים המבוסס על מערך $A[0 \dots m - 1]$, כאשר כל מפתח k מופנה למיקום $-A$ [k], כאשר h היא פ' גיבוב. במקרה זה, מספר המיקומים בטבלה יהיה פרופורציוני למספר האיברים - אבל לא קשור לטווח שלהם.

הערה פ' הגיבוב לא צריכה בהכרח להיות חח', נוצר במודע ליצור מנוגנים שיאפשרו להתמודד עם התנשויות.

דוגמה אם רצחה לאחסן לוחות רישוי בחניון, יוכל להשתמש ב- $1000 \text{ mod } h = x$ וכן להגביל את גודל הפלט, שיתאים לגודל המערך.

נסמן את קבוצת כל המפתחות האפשריים ב- U (ואיבריה לשם פשוטות נסמן ב- $\{0, \dots, |U| - 1\}$) ואת קבוצת המפתחות שאליים נוצרת להתייחס באמצעות $U \subseteq K$. בנוספ', נסמן $n = |K|$ ונקרא לטבלת הגיבוב T , כאשר $|U| \leq m = |T|$. פ' הגיבוב היא יסומן ב- NIL או $.$

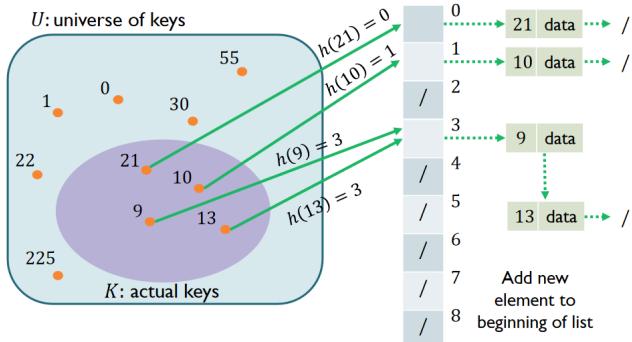
רצחה שפ' גיבוב שתפזר את התנשויות באופן אחד. בנוספ', פ' גיבוב שמחזירה רק ערכים זוגיים היא לא טובה. נעשה שימוש בהנחה התפלגות הגיבוב הפשוטה, שלפיה למפתח רנדומלי יש סיכוי להתגבע אל כל ערך מ בין האינדקסים האפשריים באופן שווה. זו הנחה אידיאלית, שעוררת לנתח פ' גיבוב.

שיטות גיבוב

1. **גישה ישירה**. טבלת גיבוב שהיא בעצם בדיקת מערך שנסמן $[0 \dots m - 1] = T$. שיטה זו יעילה כאשר מספר המפתחות יחסית קטן ו גם שהמפתחות עצמים קטנים, כך שיישמשו אינדקסים. במקרה זה לא יהיו התנשויות כי כל מפתח שונה מהאחרים. שיטה זו לא יעילה אם m גדול וזה יכול להיות בזבזני בזיכרון.

2. **שרשור**. ניקח פ' גיבוב שתמפה מפתחות לאינדקסים $b-T$, אבל לא חשוב שהוא תהיה חח'. כל פעם שיש התנשויות בערכים של פ' הגיבוב, נכניס את האיבר החדש לרשות הרשימה מקושרת בתא המיעוד לו, כאשר אברי המערך (טבלת הגיבוב) בעצם יהיו רשימות הקשורות. שיטה זו עובדת טוב כאשר $|U| < m$ הכנסה של מפתח חדש היא במקרה הגורע ביוטר (1) \emptyset כי תמיד נכניס אותו לרשות הרשימה. מצד שני, חיפוש יכול להיות הרבה יותר גורע. היתרונו המרכזי של שיטה זו הוא שהטבלה אף פעם לא מתמלאת, תמיד אפשר להוסף עוד איברים לרשותה. כדי להתמודד עם הסיבוכיות של חיפוש, משתמש בהנחה התפלגות הגיבוב הפשוטה. נגידיר את מוקדם העומס להיות $\frac{n}{m} = \alpha$ כאשר m הוא מספר התאים בטבלה ו- n הוא מספר האיברים בטבלה. בעזרת ההנחה הנ'ל, הסיבוכיות המומוצעת של חיפוש בטבלת גיבוב מסוירת היא $(1 + \alpha)$, ויש שם אחד כי צריך לחשב את ערך פ' הגיבוב קודם. נשים לב כי אם $n = \Theta(1)$ אז סיבוכיות החיפוש היא $\Theta(m)$.

3. **גיבוב פתוח**. השתמש רק בזכרון של הטבלה, בלי זיכרון חיצוני, וכך לא נזבז זיכרון בכלל שווה יותר טוב במערכות שבהם אין אפשרות סתם ככה ועוד וזה זיכרון לרישומות מקשורות. הרעיון הוא שברגע שיש התנשויות, נמצא תא פניו ונשים בו את הערך. ניתן לפרק זאת בעזרת חוספת אינדקס חיפוש לפ' הגיבוב, כלומר $[0 \dots m - 1] \rightarrow [0 \dots m - 1] \times h : T = \Theta(n)$ ואז החיפוש יתבצע ע"י חישוב $(k, 0), (k, 1), \dots, (k, m - 1)$. הכנסה תתבצע ע"י חיפוש עד שיימצא תא פניו, ואם אחריו m ניסיונות לא נמצא תא פניו אז הטבלה מלאה. נשים לב שיכול להיות מצב שבו כשםחיק איבר כלשהו ואז ננסה למחוק עוד איבר שערק פ' הגיבוב על שניהם זהה, דומה שנקבל שגיאה כי לכאהר כבר מחקנו את האיבר. אבל כדי להתמודד עם זה צריך פשטוט לסטמן את התא אחרי שמחקם אותו בסימון מיוחד (השונה מ- NIL) שמצויב על כך שהוא שם איבר שנמחק.



איור 14 : דוגמה של טבלת גיבוב עם שירשו

<pre> Hash-Insert(T, k): 1. $i = 0$ 2. repeat 3. $j = h(k, i)$ 4. if $T[j] == NIL$ 5. $T[j] = k$ 6. return j 7. else $i = i + 1$ 8. until $i == m$ 9. error "overflow" </pre>	<pre> Hash-Search(T, k): 1. $i = 0$ 2. repeat 3. $j = h(k, i)$ 4. if $T[j] == k$ 5. return j 6. $i = i + 1$ 7. until $T[i] == NIL$ or $i == m$ 8. error NIL </pre>
---	---

איור 15 : אלגוריתמי החיפוש וההכנסה בטבלת גיבוב פתוחה

שיטות חיפוש במילון פתוחה

1. **חיפוש לינארי.** $h(k, i) = (h'(k) + i) \mod m$ כאשר $h'(k, i)$ פ' גיבוב בלתי תלויות באינדקס. המשמעות של זה היא פשוט

לlect כל פעם איבר אחד עד שנגיע לתא ריק. זה יכול להיות די ארוך אם יש הרבה ערכים שונים בlij איבר ריק בינם (Primary Clustering). במקרה שיש לנו i תאים מלאים ואז תא ריק, ההסתברות שהתא הריק יתמלא הוא $\frac{i+1}{m}$, שהוא מגדיל את הזמן הממוצע.

2. **חיפוש ריבועי.** $h(k, i) = (h'(k) + c_1i + c_2i^2) \mod m$ כאשר $h'(k, i)$ פ' גיבוב בלתי תלויות באינדקס. במקרה הזה הבעיה היא שני ערכים עם אותו ערך גיבוב יקבלו בדיקת אותו מסלול חיפוש (Secondary Clustering).

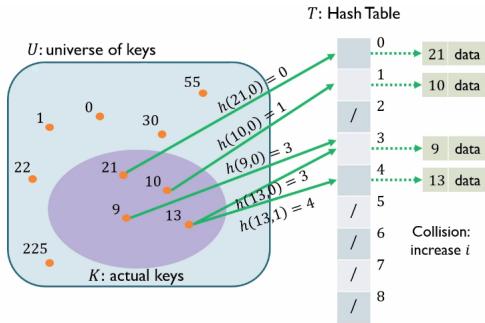
3. **גיבוב כפול.** $h(k, i) = (h_1(k) + i \cdot h_2(k)) \mod m$ כאשר h_1, h_2 הן פ' גיבוב בלתי תלויות באינדקס. שיטה זו יותר יעילה מהקודמות כי למפתחות עם ערכי h_1 זרים עדין לרובם יקבלו מסלול אחר.

משמעות בהנחת הגיבוב הפשוט, בהינתן טבלת גיבוב פתוחה עם מקדם עומס של $1 < \frac{n}{m} = \alpha$, מספר החיפושים בחיפוש לא שנכשל הוא לכל היותר $\frac{1}{1-\alpha}$ ומספר החיפושים בחיפוש שהצליח הוא לכל היותר $\ln \frac{1}{1-\alpha}$.

הערה ביטויים אלו חוסמים את זמן החיפוש, וכאשר α קבוע נקבע סיסוביות הזמן לחיפוש היא $(1/\alpha)$.

מה נחשבת פ' גיבוב טובה? פ' גיבוב טובה תפזר את המפתחות באופן אחיד על פני האינדקסים הנתונים. ישנן שתי גישות מרכזיות לפ' גיבוב. הראשונה - היררכית, הכולמת לשימושו שונראה אליו הוא עובד טוב בלי להוכיח בהכרח שהוא אכן כך, אבל בסופו של דבר הביצועים שלו אכן טובים. השניה, היא לבחור פ' גיבוב אקראית מבין מגוון פ' גיבוב וכך ניתן לנוסף להגן מפני מתקיפים שניסו להאט ולהגביר את איקות הגיבוב שלנו.

שיטת אחת פופולרית היא שיטת החלוקה, שהיא $h(k) \mod m$ הוא לא חזקה של 2 (אם כן, פ' הגיבוב פשוט תיקח רק את הביטים הנמוכים) אלא מספר ראשוני מרחוק מחזקה של 2 (לא ניכנס לזה, אבל בעזרת הנחה זו ניתן להגיע לכל אינדקס וההתפלגות היא דיאדית).



איור 16 : דוגמה של טבלת גיבוב עם גיבוב פתוח

דוגמא נניח שיש לנו $n = 2000$ ורצתה $c \leq 3$ התנגשויות בכל תא. מה יהיה m ? נחשב $m = \frac{n}{c} = 666$ ונחפש מספר ראשוני הקרוב ל-666 אבל לא חזקה של 2, לדוגמה של 701, $701^2 = 49201$.

תרגול

בהתנחת הנחות מסוימות על הקלט, נוכל להימנע ממילוי מבוסס השוואות ולקבל ביצועים יותר טובים מ- $\Theta(n \log n)$, ככלומר לינאריים,

הגדרה מיוון יציב הוא מיוון כך שאם $y < x$ ו- x קודם ל- y במערך המקורי אז x קודם ל- y גם במערך הממוין.

מיון מנייה ההנחה: כל המספרים במערך הם מותוק $\{0, \dots, k\}$ וכן

הרענון: נשמר מערך בגודל $1 + k$ שיכיל את ה"דיאלוגים" עבור $0, \dots, k$.

זמן ריצה: לולאה 1-3 יקחו (k) Θ ולולאה 2 ו-4 יקחו (n) Θ ולכן סה"כ קיבל (n) $= \Theta(2n + 2k)$.

הוא המערך הלא ממויין, B הוא מערך הפלט ו- k הוא טווח המספרים. הרעיון בספירה הוא שברגע שנדע כמה איברים קטנים מ-

יש במערך, נדע מיד איפה לשbz את i (באינדקס ה- i -י וההמשך עד שנגמרים ה- i -ים במערך המקורי).

1 CountingSort(A, B, k) is

```

2   for i=0 to k:
3     do C[i] = 0 // zero all rankings
4   for j=1 to A.length:
5     do C[A[j]] = C[A[j]+1] + 1 // now C contains the number of elements equal to i
6   for i = 1 to k:
7     do C[i] = C[i] + C[i-1] // C contains the number of elements  $\leq i$ 
8   for j = A.length downto :1
9     do B[C[A[j]]] = A[j] // place elements
10    do C[A[j]] = C[A[j]] - 1 // reduce count by 1

```

Algorithm 6: CountingSort

בהתחלת	A	5 (1)	2 (2)	3 (3)	1 (4)	4 (5)	5 (6)	3 (7)	3 (8)
C אחרי האיפוס (הלואה הראשונה)		0 0	0 1	0 2	0 3	0 4	0 5		
C אחרי ספירת האיברים (הלואה השנייה)		0 0	1 1	1 2	3 3	1 4	2 5		
C אחרי איסוף ספירת האיברים (הלואה השלישייה)		0 0	1 1	2 2	5 3	6 4	8 5		

את השלב האחרון לא נדגים כי הוא מאד ארוך, אבל לדוגמה האיטרציה הראשונה תקח את האיבר האחרון של A (3), וזה לבצע עכשו אחד ה-3-ים במקומות הנכון וכך גם עבור שאר האיברים.

טבלה 4 : דוגמת ריצה של מילון מניה

מיון בסיסי הhnacha: n מספרים עם לכל היוטר d ספרות.

הרענון: נמיין את המספרים כל פעם לפי ספרה אחת.

זמן ריצה: $T(n) = d\Theta(n) = \Theta(n)$.

נניח כי המילון היציב הוא מילון מניה.

1 RadixSort(A, d) is

2 for $i=0$ to d :

3 do use a stable sort to sort array A on digit i

אלגוריתם 7: RadixSort

המקורי	אחדות	עשרות	מאות	329
355	329	355	457	
436	436	436	657	
457	839	457	839	
657	355	657	436	
720	457	329	720	
839	657	839	355	

טבלה 5 : דוגמת ריצה של מילון בסיסי

משפט מילון נכון ויציב.

הוכחה: נגדיר x_i להיות ה- i -ספרות הראשונות של x . קלומר אם $x = 2345$, $x_1 = 45$, $x_2 = 5$ או $x_1 = 234$ ווכו.

וכlich באינדוקציה על i שהמערך ממיון נכון לפי x_i וכן שהוא יציב.

בבסיס ($i = 1$): עבור x_1, y_1 , אם $y_1 < x_1$, RadixSort ימוך את y_1 גובה יותר מ- x_1 , אם $y_1 > x_1$ אז האלגוריתם ימוך את x_1 גובה יותר

מ- y_1 . אם $x_1 = y_1$ אז RadixSort לא ישנה את הסדר של y_1 , שכן המילון בו הוא משתמש הוא יציב.

צעד : $(1, \dots, d-1 \rightarrow d)$

מקרה 1 : $x_d < y_d$ כי y_d נניח בה “ c ”

תת מקרה א' (המספרה ה- d של y גדולה מהמספרה ה- d של x). האלגוריתם ימיין את y_d גובה יותר מאשר x_d שכן הוא משתמש במילוי

תקין.

תת מקרה ב' (המספרה ה- d של y שווה למספרה ה- d של x). האלגוריתם לא ישנה את הסדר של x_d, y_d באיטרציה ה- d אבל מה “ a עברו

$x_{d-1}, y_{d-1}, x_{d-1}, y_{d-1}$ מוקם גובה יותר מאשר x_{d-1} ולכן y_d ישאר במקום גובה יותר מאשר x_{d-1} .

מקרה 2 : $(d_d = y_d)$ מוקמו באותו סדר ובאייטרציה ה- d , x_d, y_d ימוקמו גם באותו הסדר שכן האלגוריתם משתמש במילוי יציב. ■

מיון דליים ההנחה : n מספרים בטוחה $[1, \dots, 0]$ בהתפלגות אחידה.

הРЕУИН : נפצל את הטוחה $[0, 1]$ ל- n דליים $([0, \frac{1}{n}], [\frac{1}{n}, \frac{2}{n}], \dots, [\frac{n-1}{n}, 1])$ ונמיין כל דלי בנפרד.

זמן ריצה : במקרה הטוב ביותר בכל דלי יש איבר אחד ואז נקבל (n) . במקרה הגרוע ביותר כולם הלוו לאותו דלי ואז יחד עם מיון

הכנסה יקח סה “ c ” (n^2) . במקרה המוצע יש מספר קבוע של איברים בכל דלי ולכן יקח סה “ c ” (n) .

מיון דליים הוא יציב אם המיון הפנימי של הדליים הוא יציב. במקרה שלנו מיון הכנסה הוא יציב ולכן גם מיון הדליים יציב.

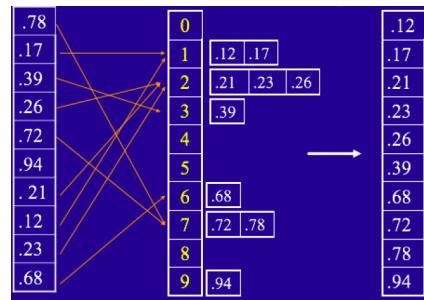
1 BucketSort(A) is

```

2     initialize B as a list of n lists
3     for i = 0 to n:
4         do insert A[i] into list B[ $\lfloor nA[i] \rfloor$ ]
5     for i = 0 to n-1:
6         do InsertionSort(B[i])
7     Concatenate lists B[0], ..., B[n - 1] in order

```

Algorithm 8: BucketSort



דגם ריצה של מיון דליים 17 Figure

אינוריאנטה סוג של אינדוקציה המתאים להוכחת קיום תכונות באלגוריתמים מסוימים.

1. **אתחול.** הטענה נכונה לפני האיטרציה הראשונה ("בסיס").
2. **שמעור.** אם הטענה נכונה לפני איטרציה כלשהי, אז היא תישמר נכון גם לפני האיטרציה הבאה ("צעד").
3. **סיום.** כשהוללה האחרונה מסיום, האינווריאנטה מוגנת לנו תוכנה שיעזרת להראות כי האלגוריתם נכון.

דוגמה הוכחה של היציבות והנכונות של מילון בסיס היא הוכחה באינווריאנטה.

```

1 let j = 9
2 for (let i = ; 0 i > ; 10 i++)
3     j = j-1

```

Algorithm 9: אלגוריתם להוכחת נכונות באינווריאנטה

האינווריאנטה יכולה להיות ש- $j = i + 9$. וכך נוכיח בכך את נכונות האלגוריתם (המrtleק במיוחד זהה).

שבוע | Universal Hashing | ו

הרצאה

כדי למנוע את המקרה הגורע בו כל המפתחות מותאמים לאוטו תא ($\Pr[n \geq \frac{1}{m} | U] \geq \frac{1}{m}$), נדריל פ' גיבוב רנדומלית, וכך יוכל למנוע את המקרה בו מספר גדול מאוד של מפתחות יותאמו לאוטו התא (במקרה הממושע). בכלל התקנה של טבלת גיבוב (יצירת אובייקט מסווג זה) נדריל פ' גיבוב אקרטי (הדבר היחיד שבאמת אקרייז או ההגלה הזה) בהסתברות אחידה לכל פ'. נרצה להגיע למצב של הנחת הנא' (הגיבוב האחד) והפשותו) בלי ההנחה האידיאלית, שכן היא לא מתקינה למציאות. כאמור, שההסתברות ש- $h(k_1) = h(k_2)$ תהיה $\frac{1}{m}$.

תהי H קבוצת סופית של פ' גיבוב המפותה מפתחות מ- U ל- $\{0, \dots, m-1\}$.

הגדירה נאמר כי H אוניברסלית אם לכל $k_1 \neq k_2$ מספר פ' הגיבוב כך ש- $h(k_1) = h(k_2)$ הוא לכל היותר $\frac{|H|}{m}$.

הסיבה שבחרנו את $\frac{|H|}{m}$ היא כי אז ההסתברות להתנגשות היא $\frac{1}{m}$ (מהסתברות אחידה), ובגלל ש- h נבחרת אקרטי, הסיכוי לבחירה רעה הוא קטן שווה ל- $\frac{1}{|H|}$.

משפט תהי h פ' גיבוב מותוך קבוצת פ' גיבוב אוניברסלית. נגיד $\alpha = \frac{n}{m}$ מקדם העומס. אז סבוכיות הזמן בתוכנת בטבלת גיבוב עם שרשור

היא:

α אם $k \leq b - T$

$T + \alpha$ אם $b - T \leq k \leq b$

הערה התוצאה זו היא האורך המוצפה של הרשימה בה k יגובב. בנוסף זה זהה לchlוטין למשפט מההרצאה הקודמת עם הנחתת הגא"פ.

דוגמה אם הטבלה תהיה בסופו של דבר חצי מלאה, אז $\alpha = \frac{1}{2}$ ואז האורך של הרשימה (בשערו) הוא קטן מ-1 אם המפתח לא שם וקטן מ-2 אם המפתח כן שם.

הוכחה: הסיבוכיות המוצפה היא הtotolat, שכן מוחוק המספרים הגדולים הממוצע מתכנס לתוחלת עבור מספיק ניסויים.

נסמן n אורך הרשימה בתא i - j . האורך הצפוי של הרשימה בכל תא (שתיי בבחירה פ' גיבוב, לא מפותחות). נגידר מ"מ אינדיקטור המציין האם יש התנגשות בין המפתחות $P(h(k) = h(l)) \leq \frac{1}{m} \cdot k \cdot l$. $X_{kl} = \begin{cases} 1 & h(k) = h(l) \\ 0 & \text{otherwise} \end{cases}$ אוניברסלית ועבור מ"מ אינדיקטוריים ההסתברות והתוחלת שוויים, ולכן

$$E[h(k) = h(l)] = P(h(k) = h(l)) \leq \frac{1}{m}$$

נגידר מ"מ המציין את מספר ההתנגשויות של k עם מפתחות אחרים שונים $.Y_k = \sum_{l \neq k} X_{kl}$

$$E[Y_k] \stackrel{\text{לפיו}}{=} \sum_{l \neq k} E[X_{kl}] \leq \sum_{l \neq k} \frac{1}{m}$$

מקרה 1 ($k \notin T$) : $E[Y_k]$ הוא מספר האיברים בתא אליו ילק k והוא כהה תאים יהיו בתא אליו ילק k בממוצע (בתוחלת). ככלומר

$$| \{l : l \in T \wedge l \neq k\} | = n \cdot n_{h(k)} = Y_k$$

$$E[n_{h(k)}] = E[Y_k] \leq \sum_{l \neq k} \frac{1}{m} = \frac{n}{m} = \alpha$$

מקרה 2 ($k \in T$) : $E[n_{h(k)}] \leq \frac{n-1}{m} + 1 = \alpha - \frac{1}{m} + 1 \leq \alpha + 1$ ומכאן באותו האופן כמו המקרה הקודם $n_{h(k)} = Y_k + 1$.

מסקנה בהינתן טבלת גיבוב עם שרשור וגיבוב אוניברסלי בעלת m תאים, הזמן שיקח לבצע n פעולות הכנסה/חיפוש/מחיקה הוא $\Theta(n)$ (בהתה שנעשה $\mathcal{O}(n)$ הכנסות).

הוכחה: לאורך כל הפעולות, $\alpha = \frac{n+\mathcal{O}(m)}{m} = \alpha + \mathcal{O}(1)$ ולכן מהמשפט הנ'יל יקח לנו α זמן לעשوت כל אחת מה- n פעולות, ככלומר $\Theta(n)$ פעולות.

משפט תהי h פ' גיבוב מותך קבוע פ' גיבוב אוניברסלי. נגידר $1 < \frac{n}{m} = \alpha$ מקדם העומס (בגיבוב פתוח אי אפשר להכנס יותר איברים ממספר התאים). אז סבוכיות הזמן בתוחלת היא :

$$\frac{1}{1-\alpha} \text{ אם } k \text{ לא ב-}T$$

$$\frac{1}{\alpha} \ln \frac{1}{1-\alpha} \text{ אם } k \text{ ב-}T$$

דוגמה בטבלת גיבוב חצי מלאה $\frac{1}{2} = \alpha$ ואז בחיפוש לא מוצלח יידרש $2 \ln 2 = \frac{1}{1-\frac{1}{2}}$. צעדים.

הוכחה: נגדיר X מ"מ להיות מספר הצעדים בחיפוש לא מוצלח. $i \geq X$ אם כל התאים עד התא $-i$ תפוסים. ההסתברות לכך היא $P(X \geq i) = \sum_{n=m}^{\infty} P(X \geq i)$, שכן כל פעם שהשאלה היא מה הסיכוי שמספר יכנס לתא ספציפי, וזהו בדיקת החישוב. התוחלת של X הוא בדיקת ממוצע מספר הצעדים שיידרשו לחיפוש לא מוצלח.

$$\begin{aligned} E[X] &= \sum_{i=1}^{\infty} P(X \geq i) \\ &= \sum_{i=1}^m P(X \geq i) + \sum_{i=m+1}^{\infty} P(X \geq i) \\ &\stackrel{(*)}{=} \sum_{i=1}^m \alpha^{i-1} + 0 \\ &\leq \sum_{i=1}^{\infty} \alpha^{i-1} = \sum_{i=0}^{\infty} \alpha^i = \frac{1}{1-\alpha} \end{aligned}$$

(*) כי אחרי m צעדים בהכרח נעצור כי זה אומר שהטבלה מלאה.

■ לא נוכיח את המקרה של החיפוש המוצלח.

מסקנה גם במיון פתו וגם בששרור קיבלנו שבעזרת משפחה אוניברסלית של \mathbb{F} קיבל את אותן הביצועים כמו בעזרת הנחת הגא" \mathbb{F} , כלומר $\mathcal{O}(1)$.

בנייה קבוצת \mathbb{F} גיבוב אוניברסלית.

יהי $m > p$ ראשוני הגדל מכל אייר ב- U . נגדיר \mathbb{F} גיבוב

$$h_{a,b}(k) = ((ak + b) \mod p) \mod m$$

וכו

$$H_{p,m} = \{h_{a,b} : a, b \in Z_p \wedge a \neq 0\}$$

נבחר אקראיית את a, b מתוך Z_p בהסתברות אחידה של $p(p-1)$

טענה $H_{p,m}$ היה אוניברסלית.

הוכחה: יהיו $k_1 \neq k_2 \mod p$. נסמן $r = (ak_1 + b) \mod p, s = (ak_2 + b) \mod p$, אז $r \neq s$ (מתורת המספרים לא יוצר התנגשויות). מעבר לכך, כלבחירה $\langle a, b \rangle$ מחזירה זוג אחר של $\langle r, s \rangle$. לעומת זאת, אם נבחר a, b אקראיית, אז $\langle r, s \rangle$ לא נסימן את הטענה.

גיבוב מושלים מגיע לביצוע של (1) \mathcal{O} במקרה הגרוע ביותר, אבל הוא עושה זאת בעורת ההנחה (המחלישה) ש- U סטטי, כלומר הוא ידוע מראש. הרעיון הוא שנבחר F , גיבוב מתוך קבוצה אוניברסלית כך שלא נקבל התנשויות, בידיעה מה הן המפתחות האפשריים.

משפט תהי טבלת גיבוב שונרזה לשומר בה n איברים בגודל $n = m$ בעורת F , גיבוב הנבחרת באקראי מתוך קבוצה אוניברסלית. אז הסתברות של התנשויות כלשהי היא קטנה מ- $\frac{1}{2}$.

מסקנה כדי למצוא F בלי התנשויות, פשוט נגידיל אקראיית F עד שנקבל אחת בלי התנשויות ולפי המשפט הנ"ל יש הסתברות גבוהה לכך. אבל הבעיה היא שזו דרוש הרבה זיכרון.

ונכל לעשות זאת גם עם (n) \mathcal{O} באמצעות גיבוב דו שכבותי. כמובן, כל תא בטבלת הגיבוב הראשית יהיה בעצם טבלת גיבוב עם F , גיבוב עצמו עצמו. הגודל של טבלאות הגיבוב המשניות יהיה מספר האיברים בהן בריובוע. עדיין, בטבלה המשנית אין התנשויות וכלן המקרא הגרוע ביותר הוא גישה בזמן קבוע.

ננתח את הגיבוב המושלים עם גיבוב דו שכבותי. בשלב הראשון, גודלה של טבלת הגיבוב הוא $n = m$ וגודלה של כל טבלת משנה T_j הוא m_j ולכן סה"כ נדרש $\sum n_j^2 + n$. מה היחס בין גודל של סה"כ טבלאות הגיבוב?

משפט תהי טבלת גיבוב מושלים ذو שכבותי שונרזה לשומר בה n איברים בגודל $n = m$ בעורת F , גיבוב אקראיית מתוך קבוצה אוניברסלית. אז $E \left[\sum_{j=0}^{m-1} n_j^2 \right] < 2n$

הוכחה:

$$\begin{aligned} E \left[\sum_{j=0}^{m-1} n_j^2 \right] &= E \left[\sum_{j=0}^{m-1} \left(n_j + 2 \binom{n_j}{2} \right) \right] \\ &= E \left[\sum_{j=0}^{m-1} n_j \right] + 2E \left[\sum_{j=0}^{m-1} \binom{n_j}{2} \right] \\ &= E[n] + 2E \left[\sum_{j=0}^{m-1} \binom{n_j}{2} \right] \\ &\stackrel{(*)}{=} n + \frac{n-1}{2} < 2n \end{aligned}$$

(*) נשים לב כי $\binom{n_j}{2}$ הוא מספר הזוגות של מפתחות המותנשימים בתא $-j$ ומתקנות הגיבוב האוניברסלית, הסיכוי להtanשויות בין שני מפתחות הוא $\frac{1}{m}$ ולכן $E \left[\sum_{j=0}^{m-1} \binom{n_j}{2} \right] = \binom{n}{2} \frac{1}{m} = \frac{n(n-1)}{2m} = \frac{n-1}{2}$

לסיכום, בעורת גיבוב נוכל בהינתן מפתחות דינמיים להגיע בזמן גישה של (1) \mathcal{O} בממוצע וזכרו של (n) \mathcal{O} במקרה ובהינתן קבוצת מפתחות סטטית נוכל לעשות זאת בזמן גישה (1) \mathcal{O} במקרה הגרוע ביותר וזכרו של (n) \mathcal{O} במקרה הגרוע ביותר.

מסקנה בעורת גיבוב מושלים ذو שכבותי, בתוחלת נקבל צרכית זיכרון של (n) \mathcal{O} עם גישה בזמן קבוע, אידיאלי!

תרגול

דוגמה יהי p ראשוני, $x \in U$ ו- $a \in \{0, \dots, p-1\}^k$. $U = \{0, \dots, p-1\}^k$. $k \in \mathbb{N}$.
 $H = \left\{ h_a : a \in \{0, \dots, p-1\}^k \right\}$. $h_a(x_1, \dots, x_k) = \sum_{i=1}^k a_i x_i \pmod{p}$

טענה H היא אוניברסלית.

הוכחה: יהיו $j \in \{1, \dots, k\}$. קיימים $P(h_a(x) = h_a(y)) \leq \frac{1}{p} \leq \frac{1}{m}$. נבחר $m \geq p$ ראשוני ונוכיח כי $\forall x, y \in U$ $x \neq y \Rightarrow h_a(x) \neq h_a(y)$.

$$\begin{aligned}
 h_a(x) = h_a(y) &\Updownarrow \\
 \sum_{i=1}^k a_i x_i = \sum_{i=1}^k a_i y_i \pmod{p} & \\
 \Updownarrow \\
 \sum_{i=1}^k a_i (x_i - y_i) = 0 \pmod{p} & \\
 \Updownarrow \\
 \sum_{i \neq j} a_i (x_i - y_i) + a_j (x_j - y_j) = 0 \pmod{p} & \\
 \Updownarrow \\
 \sum_{i \neq j} a_i (x_i - y_i) = a_j (y_j - x_j) \pmod{p} & \\
 \Updownarrow \\
 \frac{\sum_{i \neq j} a_i (x_i - y_i)}{y_j - x_j} = a_j \pmod{p} &
 \end{aligned}$$

לכן

$$P(h_a(x) = h_a(y)) = P\left(\frac{\sum_{i \neq j} a_i (x_i - y_i)}{y_j - x_j} = a_j \pmod{p}\right)$$

עתה נניח שכבר חישבנו את הסכום משמאלו (שכן הוא לא תלוי במשתנה החסתברותי a_j) ונחשב אותו $p \bmod{p}$. ההסתברות שהיא שוויה ל- $\frac{1}{p}$ בין הסכום $\sum_{i \neq j} a_i (x_i - y_i)$ והוא בדיקת $y_j - x_j$.

דוגמה הצעו אלגוריתם המקבל מערך A ובודק אם קיימת בו שלשה אריתמטית בזמנו $\mathcal{O}(n^2)$ (שלשה אריתמטית בזmeno $\mathcal{O}(n^3)$). הטענה היא שלישיה סדרה שהמפרק בין כל שני איברים עוקביהם הוא איזוה שווה d .

פתרון נבנה טבלת גיבוב עם כל האיברים במערך. נעבור על כל זוג איברים ב- A - A ונבדוק האם $b + (b - a)$ נמצא בטבלה. אם כן, הרি לנו שלשה פיתגורית. אם עברנו על הכל ולא מצאנו אף שלשה, אז אין במערך שלשה פיתגורית.

זמן ריצה: ראשית ניציר את הטבלה, שיקח לנו $\mathcal{O}(n^2)$. נעבור על כל זוג $(1, 2)$ ונבצע חיפוש $\mathcal{O}(1)$ כלומר מה'כ נקלט $\mathcal{O}(n^2)$.

בשיעור ראיינו כי אנו מחפשים ומוחקים איברים בטבלת גיבוב עם שירשור ב- $\mathcal{O}(1 + \frac{n}{m})$ בתוחלת. מה נעשה במקרה ש- $m > n$?

טבלאות גיבוב בגדים דינמיים

1. נגידר רף $m = \frac{3}{4}n$ וכשנחצה אותו, נבנה טבלה בגודל $2m$ ונעביר את האיברים מהטבלה המקורית לחדשה. הבעה היא שההעתקה תיקח $\mathcal{O}(n)$.

2. נזכיר שני מערכים A, B כך ש- B - A גדול פי שניים מ- A . נזכיר i שהייתה המיקום התפוס ב- A ו- j שהייתה המיקום התפוס ב- B . כל פעם שנכניס איבר, נכנסו אותו גם ב- A וגם ב- B ונגדיל את j , i , בהתאם. ברגע שנגיע לרף, נקרה ל- B , ונבנה מערך גדול פי שניים מ- A הישן, נקרה לו B ונשנה האינדקסים בהתאם. oczywiście, כל פעם שנכניס איבר חדש, נכנס את האיבר הרצוי גם ב- A וגם ב- B ובנוסך עותק עוד איבר מ- A ל- B . כך, כשנגיע שוב לרף של A , בדוק נסימן להעתיק את כל האיברים מ- A ל- B ואז נשנה את אותו שינוי שמות ואינדקסים שעשינו במהלך הראשון.

כדי למחוק איבר פשוט נמחק אותו מהמערכות ולא נעדרנו את j, i , וכך נשמר את הנכונות של האלגוריתם. במקרה שנרצת לשומר על סדר בטבלת הגיבוב, נבנה טבלת גיבוב של פוינטורים שתציביע לטבלה ממויינת.

הגדרה יהיו \mathbb{N} משפחת פוקנציות $[m] \rightarrow U$. נאמר כי H היא k -אוניברסלית אם לכל $x_1, \dots, x_k \in U$ מתקיים כי לכל $P(h(x_1) = i_1 \wedge \dots \wedge h(x_k) = i_k) \leq \frac{1}{m^k}, (i_1, \dots, i_k) \in [m]^k$

טענה אם H היא 2-אוניברסלית אז היא אוניברסלית.

הוכחה: יהיו $h \in H, x \neq y, x, y \in U$ ונבחר a אקראית.

$$P(h(x) = h(y)) = P(h(x) = h(y) = 1 \vee \dots \vee h(x) = h(y) = m) \leq \sum_{i=1}^m P(h(x) = h(y) = i) \leq \sum_{i=1}^m \frac{1}{m^2} = \frac{1}{m}$$

■

משפט (אי שוויון מרקוב) יהיו X מ"מ אי שלילי אזי $0 < t < E[X]$ מתקיים $P(X \geq t) \leq \frac{E[X]}{t}$

תרגיל תהיו T טבלת גיבוב בשירשור המשמשת במשפחה 2-אוניברסלית עבור פ'. $[m] \rightarrow U$. נניח כי אנו מכנים m איברים ל- T . הראו כי ההסתברות שקיים תא עם $\leq 2\sqrt{m}$ איברים קטנה-שווה ל- $\frac{1}{2}$.

פתרון נגידר מצין ש- x, y נמצאים בתא ה- i . נגידר n_i להיות מספר האיברים בתא ה- i . מספר הזוגות בתא ה- i הוא C_{xy}^i וההסתברות המתבקשת היא

$$\begin{aligned} P(n_i \geq 2\sqrt{m}) &= P(n_i^2 \geq 4m) \\ &= P\left(\sum_{(x,y)} C_{xy}^i \geq 4m\right) \end{aligned}$$

ואם נחשב את $E \left[\sum_{(x,y)} C_{xy}^i \right]$

$$\begin{aligned}
E \left[\sum_{(x,y)} C_{xy}^i \right] &= \sum_{(x,y)} E [C_{xy}^i] \\
&= \sum_{(x,x)} E [C_{xy}^i] + \sum_{(x,y):x \neq y} E [C_{xy}^i] \\
&\stackrel{\text{אינדיוקטורי}}{=} \sum_{(x,x)} P(h(x) = h(y) = i) + \sum_{(x,y):x \neq y} P(h(x) = h(y) = i) \\
&= \dots
\end{aligned}$$

שבוע VII | Priority Queue & Heap

הרצאה

נרצה לעשות מכירה פומבית, שבזמנה ניתן להוסיף, להוריד ולעדכן הצעות ושלכל אורכה נרצה את ההצעה המקסימלית.

תור קדימות לטור קדימות (Priority Queue) S יש ארבע פעולות:

Max(S) שתחזיר את הערך המקסימלי.

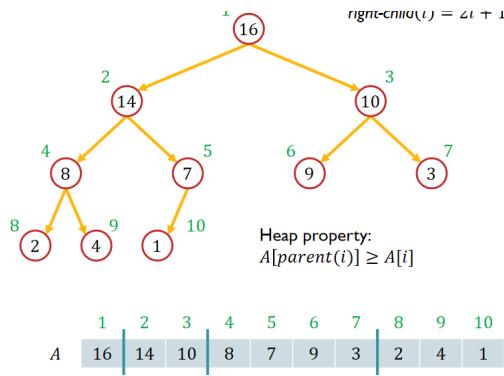
Insert(S, x) שתכניס את הערך x.

Extract-Max(S) תחזיר ותוציא את הערך המקסימלי.

Increase-Key(S, x, k) להעלות את הערך של x ב-k.

נמשיך זאת עם עירימה (Heap).

עירימה מבנה נתונים הממומש ע"י עץ בינארי שלם, כלומר שכל הרמות של העץ מלאות, מלבד אולי האחרונה (שמתמלאת משמאלי לימין). העץ הבינארי הוא בעצם מערך (ראו דוגמה). נשים לב כי בהתעלם מהתוכן, יש לבדוק עץ ביןארי שלם אחד גדול n. בנוספ', נדרש כי האב יהיה גדול מכל הילדים שלו, כלומר $A[\text{parent}(i)] \geq A[i] \quad \forall i > 1$. נקבע את השורש שלו אין אב. לכן מההגדרה, השורש הוא האיבר הגדול ביותר (וכך גם בכל שאר תתי העצים). הגובה של העץ הוא $n = \log_2 h$ (עבור n גודל המערך). בرمה ה-i (במפרק i מהשורש) יש 2^i צמתים וסה"כ יש $2^h - 1$ צמתים פנימיים ולכל היוטר 2^h עליים בرمה התחטונה. לכן מתקיים בرمה ה-h (במפרק h מהשורש) יש 2^h צמתים וסה"כ יש $2^{h+1} - 1$ העליים של העירימה הן מציאות המקסימום והכנסה/הוצאתו/עדכו. כדי לשמור על תנאי המונוטוניות השתמש בפ' עוז Max-Heapify. הפ' הוזע איבר חדש שנרצה להכניס למיטה כל עוד האיבר קטן מהילד שלו, ונעשה זאת ריקורסיבית. בכלל ש- $\lfloor \log n \rfloor$ אוזי סיבוכיות הזמן היא $\mathcal{O}(\log n)$. הזמן שימושקו על עץ בגודל n הוא $\Theta(1)$ ואז קריאה ריקורסיבית על הילד של



איור 18 : הדוגמה של מיימוש של עירימה במערך

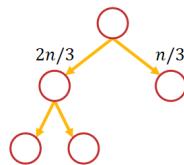
```

1 Max-Heapify(A, i)
2     l = left(i)
3     r = right(i)
4     if l ≤ heapsize(A) and A[l] < A[i]
5         largest = l
6     else
7         largest = i
8     if r ≤ heapsize(A) and A[r] < A[largest]
9         largest = r
10    if largest ≠ i
11        Exchange(A[i], A[largest])
12        Max-Heapify(A, largest)

```

Algorithm 10: Max-Heapify

מה הגודל של תת העץ (הילד של i)? במקרה המכון מאוון נקבל שיש $\frac{2n}{3}$ בחזי השמאלי ו- $\frac{n}{3}$ בחזי הימני (ראו איור). לכן במקרה הגרוע החסם התיכון הוא $n \log n$.



איור 19 : עץ שלם לא מאוון

וגם העליון הוא $n \log n$ ולכן $T(n) = \Theta(\log n)$

עתה נרצה לבנות את העירימה. נשתמש בפ' Build-Max-Heap אשר מקבלת מערך ויוצרת ממנו מערך תקין. הרעיון הוא שאנו מניחים שהרמה התיכונה כבר במקומו, ואז נקרא ריקורסיבית ל-`MaxHeapify` מלמטה למעלה על כל שאר האיברים שעוד לא עברו עליהם.

טענה BuildMaxHeap תקינה. עירימה יוצרת מערך

הוכחה: נוכח באינוקריינטה. האינוקריינטה שלנו במקרה זהה תהיה שבאיטרציה ה- i , כל האיברים מימיין ל- i ($i+1, \dots, n$) הם שורשים של עירימות תקינות.

אתחלו: $\lfloor \frac{n}{2} \rfloor = i$ כל $i > i'$ הוא עלה ולכן ראש של עירימה בגודל 1 באופן ריק.

```

1 BuildMaxHeap(A)
2     for i = ⌊A.length / 2⌋ downto 1
3         MaxHeapify(A, i)
4

```

Algorithm 11: BuildMaxHeap

שימור : כל הילדים של $[i]$ הם ראשיהם של עריומות ואחרי Max-Heapify נקלט כי $[i]$ גדול משני הילדים שלו והילדים שלו ישארו ראשיהם תקינות.

■ סיום : הריצה נגמרה ב- $i = 1$, כלומר ראש של עריימה תקינה.

מבחינה נאיבית, אנחנו מראים פעולה של $n \log n$. זה נכון אבל לא אדווק. כי רצ' יותר זול ככל שהוא יותר יורד יותר למיטה.

נשתמש בטענה שימושית, שאומרת לכל גובה בין $\lceil \log_2 n \rceil$ ל- 0 , מספר הקודקודים בגובה h הוא לכל היותר $\lceil \frac{n}{2^{h+1}} \rceil$.

לכל קודקוד העבודה פרופורצנית לגובה. עבור גובה h , העבודה היא $\mathcal{O}(h)$ ולכן

$$T(n) \leq \sum_{h=0}^{\lfloor \log n \rfloor} \lceil \frac{n}{2^{h+1}} \rceil \mathcal{O}(h) = \mathcal{O}\left(n \sum_{h=0}^{\lfloor \log n \rfloor} \frac{h}{2^h}\right) = \mathcal{O}\left(n \sum_{h=0}^{\infty} \frac{h}{2^h}\right) \stackrel{(*)}{=} \mathcal{O}(n)$$

$$\cdot \sum_{k=0}^{\infty} kx^k = \frac{x}{(1-x)^2}$$

כלומר קיבנו שлокח לבנות את העריימה ב- $\mathcal{O}(n)$.

נראה כיצד ניתן למיין במקומות בסיבוכיות $\mathcal{O}(n \log n)$. נבנה את העריימה בהתחלה ואז נמצא את האיבר המקסימלי, נשים אותו בטוף המערך (המקום הנכון שלו בסופו של דבר), נתקן את שאר המערך (כל זה באמצעות Max-Heapify) וונפעיל ריקורסיבית. לוקח (n) לייצר את העריימה בהתחלה ואז אנחנו קוראים n פעמים ל- $\mathcal{O}(\log n)$ Max-Heapify (ולומר מה'כ' $\mathcal{O}(\log n)$ ובערך $\mathcal{O}(n \log n)$ למשוך את העריימה בהתחלה ואז נמצא את האיבר המינימלי).

```

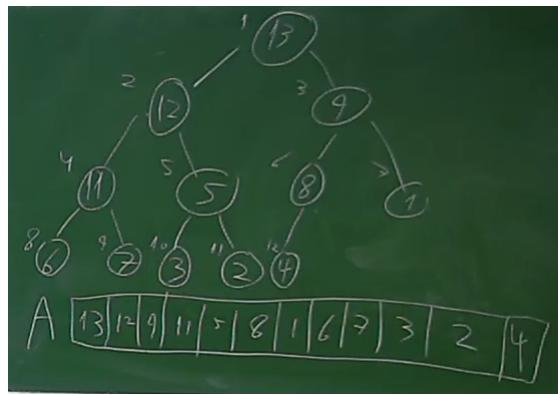
1 HeapSort(A)
2     BuildMaxHeap(A)
3     heapsize(A) = length(A)
4     for i = length(A) downto 2
5         Exchange(A[i], A[1])
6         heapsize(1)-
7         MaxHeapify(A, i)
8

```

Algorithm 12: HeapSort

שזה מבוסס השוואות או זה $\Theta(n \log n)$.

נרצה לממש את Max-Heapify. נעשה זאת ע"י מציאת האיבר המינימלי בעץ, נציב אותו בשורש ונקרא ל- HeapExtractMax והוא כבר יסדיר את השאר.



איור 20 : דוגמה ליישום של ערימה במערך

תרגול

בערימה האב גדול מכל היצאים שלו. העלים בערימה נמצאים ב- $\lceil \frac{n}{2} \rceil$ עד n .

כדי לתקן את הערימה נפעע את האיבר למיטה עד שלא תהיה בעיה עם השורש שלו או מסתכלים.

כדי למחוק את האיבר המקסימלי נמחק את האיבר האחרון בערימה ונשים את ערכו בשורש ונרצה את `.MaxHeapify`

```

1 HeapExtractMax(A)
2     if heapsize(A) > 1
3         error "heap underflow"
4     max = A[1] // 1 is the first index
5     A[1] = A[heapsize(A)]
6     heapsize(A)-
7     MaxHeapify(A, 1)
8     return max

```

Algorithm 13: HeapExtractMax

כדי להעלות את ערכו של קודקוד מסוים, נשנה את ערכו ונפעע אותו למעלה עד שנקבל ערימה תקינה.

```

1 IncreaseKey(A, i, key)
2     if key > A[i]
3         error "new key is smaller"
4     A[i] = key
5     while i < 1 and A[parent(i)] > A[i]
6         Exchange(A[i], A[parent(i)])
7         i = parent(i)

```

Algorithm 14: IncreaseKey

כדי להכניס איבר נכניס את ∞ – כאיבר האחרון ואז נבצע `IncreaseKey` לו בערך הרצוי.

כדי למחוק איבר נחליף את $[i]$ עם האיבר האחרון. נמחק את האיבר האחרון ונבצע `i`.

ערימות מינימום ערימה סימטרית לערימת מקסימום המשמרת את התכונה (ההופוכה) $\forall i \in [n], A[i] \geq A[\text{parent}(i)]$. לערימה זו בינה

פעולות מקבילות ואנלוגיות לערימות מקסימום.

```

1 HeapInsert(A, i, key)
2     heapsize(A)++
3     A[heapsize] = -∞
4     IncreaseKey(A, heapsize(A), key)

```

Algorithm 15: HeapInsert

הגדירה חציון הוא מספר השיך לקבוצה של מספרים כך שמס' האיברים שגדולים ממנו וקטנים ממנו שווים. לצורךינו, אם בקבוצה מס' זוגי של איברים, נבחר את הגדול מבין השניים.

ערימות חציון נגידר שתי ערים, A -עירימת מקסימום, B -עירימת מינימום ונדרوش כי כל האיברים ב- A קטנים מכל האיברים ב- B וגם כי $n_A = n_B$ (אם יש מס' כולל זוגי של איברים) או $1 < n_A = n_B$ (אם יש מספר אי-זוגי של איברים). אז כדי להחזיר את החציון

פושט נחזיר את השורש של B . עבור הכנסת איבר יש ארבעה מקרים :

. $B.insert(v) \geq rootA$ ו- $n_A = n_B$ (i)

. $B.insert(v) < rootA$ ו- $n_A = n_B$ (ii)

. $B.insert(v) \geq rootA$ ו- $n_A = n_B - 1$ (iii)

. $B.insert(v) < rootA$ ו- $n_A = n_B - 1$ (iv)

טבלת יאנג טבלת יאנג היא מטריצה בגודל $n \times m$ כך שכל שורה ממויינת (משמאל לימין) על וכל עמודה ממויינת מלמעלה למטה. חלק מהאיברים הם ∞ ונחשבים כאיברים לא-קיים. ככל שהולכים ימינה או למטה גדלים בערך.

2	3	4	∞
5	12	14	∞
8	16	∞	∞
9	∞	∞	∞

טבלה 6 : דוגמה לטבלת יאנג

כדי להוציא את האיבר המינימלי, נשים במקומו ∞ ואז נפעע אותו למטה (וימינה) עד למיקום הנכון. זה יקח $(n+m)\mathcal{O}$. את הפעוע נעשה פשוט באמצעות בדיקה של האם האיבר נמצא במקום הנכון מבחינת סדר ביחס לאיברים מימיינו מתחתיו. אם כן, אז הוא במקומות הנכון. אחרת, נחליף אותו עם האיבר המקסימלי ונחזיר להתחלה.

כדי לשנות את ערכו של איבר נשנה את ערכו ואז נפעע אותו למיטה באמצעות האלגוריתם הסימטרי לפיעוף למטה.

כדי להכניס איבר נשנה את ערכו של האיבר האחרון $[m, Y]$ (רק אם האיבר האחרון הוא ∞ , אם הוא לא זה אומר שהטבלה מלאה) לערך הרצוי ונפעע אותו למיטה כמו בשינוי ערכו של איבר.

כדי לMIN מערך בגודל n^2 , נכניס את כל האיברים אל תוך טבלת היאנג ואז נבצע n^2 פעמים ונקבל מערך ממויין. הסיבוכיות היא $\mathcal{O}(n^3) = \mathcal{O}(n^2(2n))$, אבל אנחנו ידעים כי מיפויים על מערך בגודל n^2 הם

$$\Omega(n^2 \log n^2) = \Omega(n^2 2 \log n) = \Omega(n^2 \log n)$$

ולכן האלגוריתם הזה לא נותן לנו ביצועים טובים כפי שהיינו רוצים.

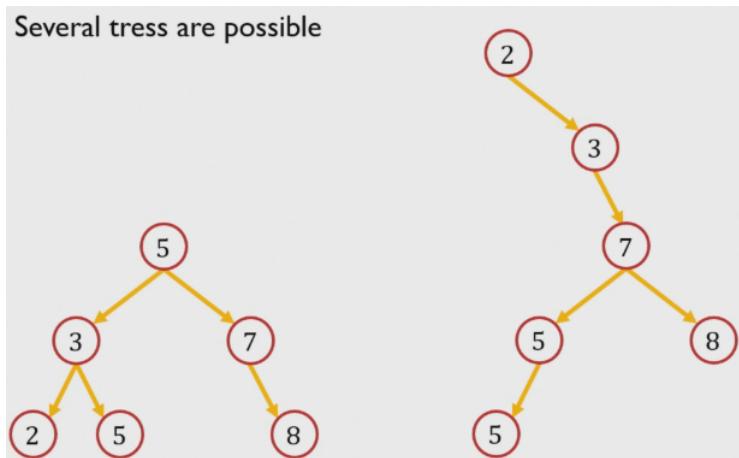
שבוע | VII | Binary Search Tree & AVL

הרצאה

נרצה לבנה נתונים שנוכל להכניס ולהוציא ממנו איברים (לכן מערך סטטי לא יעבד) וגם לשמר על סדר ביןיהם (לכן טבלאות גיבוב לא יעבדו). נרצה למש חיפוש, הוצאה והכנסה, מינימום/מקסימום ועקב/קודם. נשתמש בשביל זה בעצים בינאריים.

בעץ בינארי יש שורש, קודקודים פנימיים, עם לפחות היותר 2 ילדים ועלים. לכל קודקוד יש ארבע שדות, Key, Left, Right, Parent, ואם הקודקוד הוא עלה או שנייה Left, Right. התכונה המרכזית של עץ חיפוש בינארי היא שלכל קודקוד l בתת העץ השמאלי של x מתקיים $l.key \leq x.key$ ולכל קודקוד r בתת העץ הימני של x מתקיים $r.key \geq x.key$. אם כן, תת עץ של עץ חיפוש בינארי גם הוא עץ חיפוש בינארי.

דוגמה $S = (2, 3, 5, 5, 7, 8)$



איור 21 : עצים בינאריים שונים על אותו אוסף

Iterative-Tree-Search(x, k)

1. while $x \neq null$ and $k \neq x.key$ Tree-Search(x, k)
 1. if $x = null$ or $k = x.key$
 2. then return x
 3. if $k < x.key$
 4. then return Tree-Search($x.left, k$)
 5. else return Tree-Search($x.right, k$)

Algorithm 16: אלגוריתמי חיפוש בעץ חיפוש בינארי (ריקורסיבי ואיטרטיבי)

סיבוכיות אלגוריתם החיפוש ואלגוריתם מציאת המינימום והמקסימום הם ($O(h)$) כאשר h הוא הגובה של העץ. הסיבה לכך שהמינימום נמצא לפני מושך שמאל היא של כל קודקוד אחר או שהגענו אליו בעזרת פניה ימינה או שיש עוד איבר מתחתיו קטן-שווה לו. סיבוכיות אלגוריתמי הדרטה היא ($O(n)$) (או עבורים בדיקות אחת על כל איבר). בగלול ש-InorderTreeWalk(n) והוא מדפיס את האיברים לפי הסדר, אזי בהכרח שלבנות את העץ יקח ($n \log n$) (כי זהה בעץ שיטה למינון מערכיים).

```

Tree-Minimum( $x$ ):
1. while  $x.\text{left} \neq \text{null}$ 
2.   do  $x \leftarrow x.\text{left}$ 
3. return  $x$ 

```

```

Tree-Maximum( $x$ ):
1. while  $x.\text{right} \neq \text{null}$ 
2.   do  $x \leftarrow x.\text{right}$ 
3. return  $x$ 

```

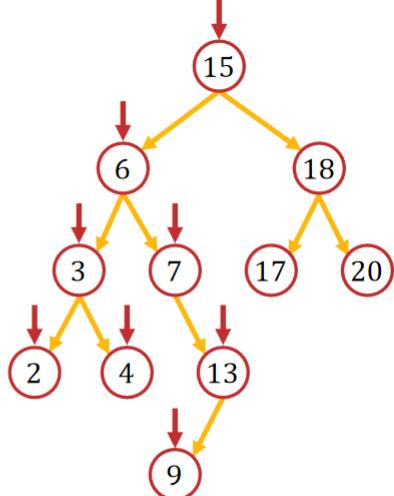
17 Algorithm: אלגוריתמי מציאת מינימום ומקסימום בעץ חיפוש בינארי

Preorder-Tree-Walk(x)

1. If ($x \neq \text{null}$) then
2. print(x)
3. Preorder-Tree-Walk($x.\text{left}$)
4. Preorder-Tree-Walk($x.\text{right}$)

Postorder-Tree-Walk(x)

1. If ($x \neq \text{null}$) then
2. Postorder-Tree-Walk($x.\text{left}$)
3. Postorder-Tree-Walk($x.\text{right}$)
4. print(x)



18 Algorithm: אלגוריתמי הדפסה בעץ חיפוש בינארי

הגדולה העוקב של x הוא y עם $y.\text{key}$ המינימלי הגדל מ- $x.\text{key}$.

אם x הוא המקסימום או העוקב שלו הוא `null`. אם יש ל- x תת עץ ימני אז זה האיבר המינימלי שם. אחרת, זה האב הקדמון הנמוך ביותר כך שבנו השמאלי הוא x או אב קדמון של x , כלומר בעיקורו פשוט מעלה עד שנמצא מקום בו נוכל לפנות שמאלה ונחזיר את הערך הזה. רעיון האלגוריתם הוא כל פעם לעלות זוג עד שמוצאים זוג שהוא לא אב-בן אלא אב-בן שמאלי.

טענה: `TreeSuccessor` הוא אלגוריתם שモזיא את העוקב של x .

הוכחה: אם ל- x אין יلد ימני אז העוקב של x הוא האב הקדמון הראשון של x שהילד השמאלי שלו הוא גם אב קדמון של x ואם אין אב קדמון, אז x מקסימלי ולכן העוקב הוא `null`. כדי להראות ש- y הוא אכן העוקב של x , נctrיך להוכיח כי $y.\text{key} > x.\text{key}$ וכן ש- y הוא המקסימום בין כל האיברים הקטנים מ- y . מעלה על הילדים הימניים החל מ- x עד שנעצור. נסמן ב- z את הקודקוד בו עצרנו ו- $y = z.parent$. נשים לב כי x הוא המקסימום של תת העץ ששורשו ב- x (שכן הוא הכי ימני בתת העץ). מתקיים $y.\text{key} \geq x.\text{key}$ שכן x נמצא בתת-העץ הימני של y . x הוא המקסימום של כל האיברים הקטנים מ- y כי כל האיברים הקטנים מ- y מצויים בתת העץ הזה והוא המקסימום שלו. נרד כל הדרך למטה לפי היחס של x לשאר הערכים בעץ ואז נכנסים אותו בצד ההפוך ביחס לאיבר האחרון אליו הגיעו וזה יהיה מקומו.

```

Tree-Successor( $x$ )
1. If  $x.\text{right} \neq \text{null}$ 
2.   then return Tree-Minimum( $x.\text{right}$ )
3.  $y \leftarrow x.\text{parent}$ 
4. while  $y \neq \text{null}$  and  $x = y.\text{right}$ 
5.   do  $x \leftarrow y$ 
6.      $y \leftarrow y.\text{parent}$ 
7. return  $y$ 

```



Algorithm 19: אלגוריתם מציאת העוקב

```

Tree-Insert( $T, z$ )
1.  $y \leftarrow \text{null}$ 
2.  $x \leftarrow T.\text{root}$ 
3. while  $x \neq \text{null}$ 
4.   do  $y \leftarrow x$ 
5.     if  $z.\text{key} < x.\text{key}$ 
6.       then  $x \leftarrow x.\text{left}$ 
7.     else  $x \leftarrow x.\text{right}$ 
8.    $z.\text{parent} \leftarrow y$ 
9.   if  $y = \text{null}$  then  $T.\text{root} \leftarrow z$ 
10.  else if  $z.\text{key} < y.\text{key}$ 
11.    then  $y.\text{left} \leftarrow z$ 
12.    else  $y.\text{right} \leftarrow z$ 

```

Algorithm 20: אלגוריתם הכנסת איבר לעץ

עבור המchèקה יש שלושה מקרים :

אם $l-z$ אין ילדים, פשוט נמחק אותו.

אם $l-z$ יש ילד אחד, נמחק את z וונחבר את הילד לאב של z .

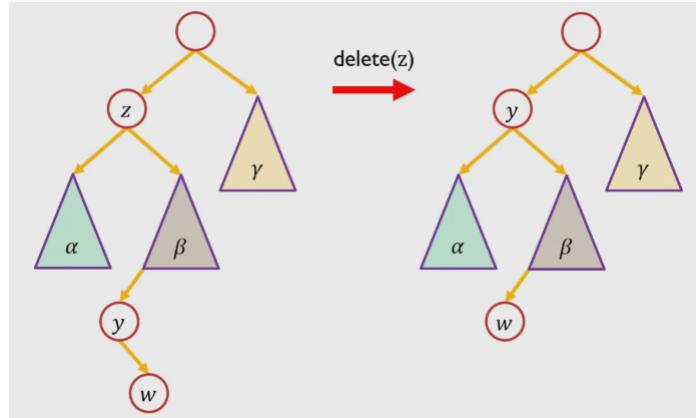
אם $l-z$ יש שני ילדים, נרצה להשתמש בעוקב כדי למחוק באופן תקין. נחליף את z בעוקב שלו ונתקן את מה שקרה כמשמעותו את העוקב (זהה קל כי יש לו לכל היותר ילד אחד, כי אם היו לו שניים אז היינו יכולים לרדת עוד ימינה בתת העץ של z ואז זה לא היה העוקב).

בעזרת אלגוריתם זה, נשמר על התוכונה של עצי חיפוש ביןאריים.

כדי למשש את האלגוריתם הזה נגדיר פ' Transplant אשר תחליף תת-עץ אחד אחר. הסיבוכיות של מחיקה במקרים 1 ו-2 היא $\Theta(1)$ ובמקרה 3 היא $\Theta(h)$.

כדי שהסיבוכיות תהיה מינימלית, נרצה ש- h יהיה מינימלי, כלומר $n \log$.

נרצה שהעץ יהיה מאוזן אבל לא מאוזן כלותין כי זה מאד יקר. נשתמש בעציו AVL שדורשים שהפרש הגבהים בין תת-עצים יהיה לכל יותר 1. זה מבטיח כי $h = \lfloor \log_2 n \rfloor$ ולכן פעולות יعلו ($\log n$) Θ . כדי לשומר על תוכנה זו, נאוזן מחדש את העץ כל פעם שצריך.



איור 22 : המקרה שבו יש ל�ודקוד הנמק שמיילדים

`Transplant(T, u, v)`

```

1. if  $u.parent = null$ 
2.   then  $T.root \leftarrow v$ 
3. else if  $u = u.parent.left$ 
4.   then  $u.parent.left \leftarrow v$ 
5. else  $u.parent.right \leftarrow v$ 
6. if  $v \neq null$ 
7.   then  $v.parent \leftarrow u.parent$ 

```

21 Algorithm: אלגוריתם החלפת תת-עצים

הגדירה עץ חיפוש ביןארי הוא AVL אם T אינו ריק, אם תת-עץ הימני והשמאלי של T גם הם עצים AVL ואם הפרש הגבהים בין העץ השמאלי לימני הוא $-1, 0, 1$.

נגידר לכל קודקוד את הפרש הגובה בין תת-עץ שלו ($hd(x)$) לבין התת-עץ האין לו ילדים ו- 0 אם הוא ריק. בעץ AVL $hd(x)$ הוא תמיד $-1, 0, 1$. הסימן אחרית. לכן הגובה של עץ ריק הוא 1 .

משפט הגובה של עץ AVL הוא $\Theta(\log n)$ (ובפרט הוא לכל היותר $1.44 \cdot \log_2(n+2) - 0.328$).

נכיס וונציא איברים כמו בעץ חיפוש ביןארי וזו נתקן את הפרשי הגובה לאחר מכון. מספר תת-העצים המושפעים הם ($O(h)$) בהכנסה (מהעלה ועד לשורש) וגם במחיקה (מחוקק שמחק ועוד לעץ במרקם 1 ו-2 וגם תת-העצים של האיבר שמחק במרקם 3).

הכנסה משנה את הגבהים בכלל יותר. נחשב מחדש את הפרשי הגבהים ($O(h)$). אם הגבהים תקינים נסיק. אחרת, נזוז מחדש את העץ באמצעות רוטציה. ישנו ארבעה מקרים כמפורט לאZN את x אחרי הכנסה איבר חדש:

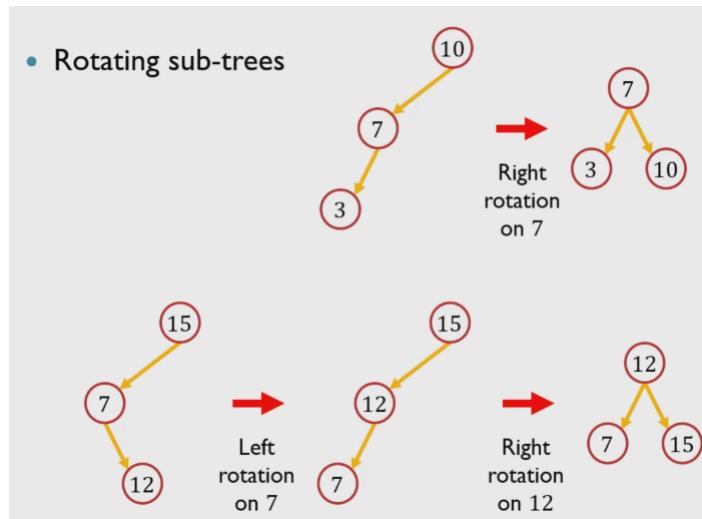
1. הכנסה בתת-עץ השמאלי של הילד השמאלי של x .
2. הכנסה בתת-עץ הימני של הילד השמאלי של x .
3. הכנסה בתת-עץ השמאלי של הילד הימני של x .
4. הכנסה בתת-עץ הימני של הילד הימני של x .

```

Tree-Delete( $T, z$ )
1. If  $z.\text{left} = \text{null}$ 
2.   then Transplant( $T, z, z.\text{right}$ )
3. Else if  $z.\text{right} = \text{null}$ 
4.   then Transplant( $T, z, z.\text{left}$ )
5. Else  $y \leftarrow \text{Tree-Minimum}(z.\text{right})$ 
6.   if  $y.\text{parent} \neq z$ 
7.     then Transplant( $T, y, y.\text{right}$ )
8.      $y.\text{right} \leftarrow z.\text{right}$ 
9.      $y.\text{right.parent} \leftarrow y$ 
10. Transplant( $T, z, y$ )
11.  $y.\text{left} \leftarrow z.\text{left}$ 
12.  $y.\text{left.parent} \leftarrow y$ 

```

Algorithm 22: אלגוריתם מחיקת איבר מעץ חיפוש בינארי



איור 23 : דוגמה לסיבוב העץ לשם איזוון

מקרים 4, 1 הם סימטריים וידרשו סיבוב אחד וקיימים 2, 3 גם הם סימטריים אך ידרשו שני סיבובים. אחרי הכנסה וסיבוב אחד, תת העץ הוא באותו הגובה כמו המקורי ולכן שאר העץ עדין AVL וזרנו לאיזוון.

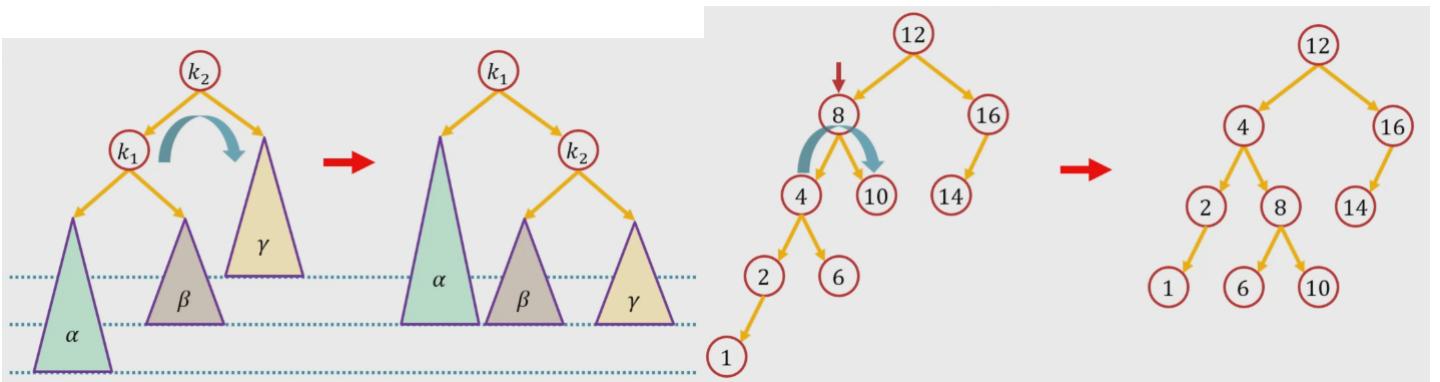
משקגה מספיק לעשות רוטציה רק על האיבר הראשון (מהעלת לשורש) שבו האיזוון מופר.

רוטציה לוקחת $\Theta(\log n)$ ומציאת הקודקוד לוקחת $\Theta(1)$.

אחרי מחיקה יתכן שנctrיך לבצע $\Theta(\log n)$ רוטציות לאורך הדרכ שבין השורש לעלה אבל עדין אותן רוטציות פשוטות.

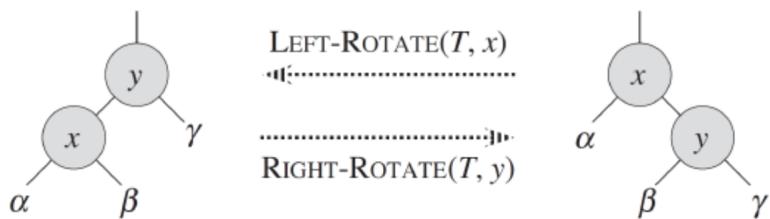
תרגול

גולם האיזוון של קודקוד v הוא $.hd(v) = \text{left.height} - \text{right.height}$



איור 24 : סיבוב במקהה 1

כדי לאזן עץ AVL, נבצע פעולות רוטציה על הקודקודים. להלן שתי הפעולות הבסיסיות.



איור 25 : רוטציות בעץ AVL

נשוכנו כי: (כל האיברים ב- γ) \leq y $<$ (כל האיברים ב- β) \leq x $<$ (כל האיברים ב- α).

סוגי הפרות איזון בעץ AVL

.1 : העץ נוטה כULO שמאליה, $hd(v.left) = 1, 0, hd(v) = 2$. כדי לאזן מחדש נסובב את העץ ימינה.



איור 26 : הפרה מסוג LL ותיקונה באמצעות רוטציה ימינה

.2 : העץ נוטה שמאליה אבל תת העץ של הבן השמאלי נוטה ימינה, $hd(v.left) = -1, hd(v) = 2$. אם נסובב את כל העץ ימינה.

נקבל פשוט RL שזה לא יעזור לנו. לכן קודם נעשה רוטציה שמאליה ליד השורש ואז רוטציה ימינה לכל העץ.

.3 : העץ נוטה כULO ימינה, $hd(v.right) = -1, 0, hd(v) = -2$. כדי לאזן מחדש נסובב אותו שמאליה (בדיקה באופן

סימטרי ל-LL).

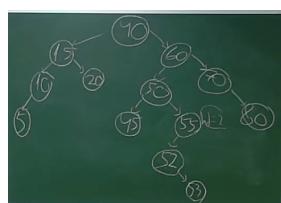


איור 27 : חפירה מסוג LR ותיקונה באמצעות רוטציה ימינה

4. RL : העץ נוטה ימינה אבל תחת העץ הימני נוטה שמאליה, $-2 = hd(v.right) - 1 = hd(v)$. כדי לאוזן מחדש את העץ נסובב אותו ימינה ואו שמאלה (בדיקה באופן סימטרי ל- LR).

הפרת איזון בפעולות על העץ

- **הכנסה.** נעה ממקום הרכניטה במעלה העץ עד שנמצא חפירה של האיזון ואו נסובב בהתאם לסוג החפירה רק את תחת העץ הזה ואו נקבל עץ AVL תקין שוב.



איור 28 : חפירה מסוג LR בהכנסת 53 לעץ

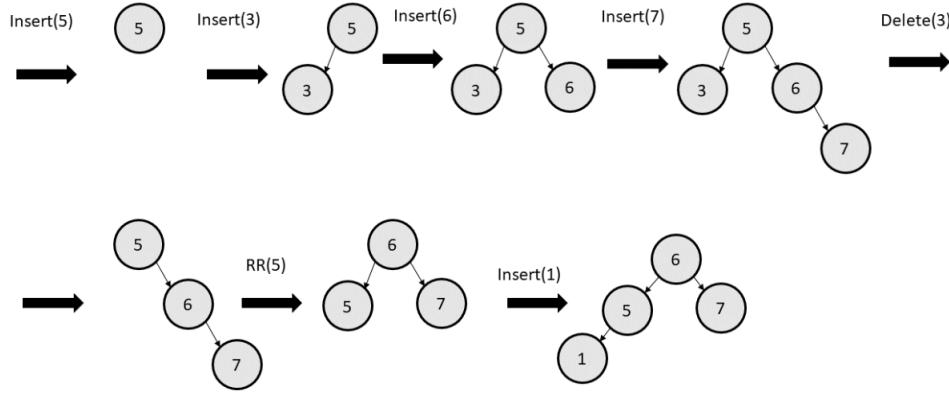
- **מחיקה.** נעה בוגה העץ ובכל נקודה בה יש הפרת איזון נתקו עד שנגיע לשורש.



איור 29 : חפירה מסוג RR במחיקת 40 מהעץ

הערה אפילו שבמחיקה אנחנו עושים יותר פעולות מאשר הרכניטה באיזון חדש, שניהם $\mathcal{O}(\log n)$ יחד עם האיזון החדש.

דוגמה נבצע על עץ ריק את הפעולות הבאות.



איור 30 : דוגמת ריצעה של פעולות על עץ AVL

Breadth First Search | VIIII

הרצאה

הגדרות

1. גרף מכון הוא $G = (V, E)$ כך ש- $V = \{v_1, \dots, v_n\}$ היא קבוצת הקודקודים ו- $E = \{e_1, \dots, e_n\}$ כאשר $e_k = e_{ij} = (v_i, v_j)$ הוא זוג סדורי.

2. גרף לא מכון הוא $G = (V, E)$ כך ש- $V = \{v_1, \dots, v_n\}$ היא קבוצת הקודקודים ו- $E = \{e_1, \dots, e_n\}$ הוא קבוצה בגודל 2 $\{v_i, v_j\}$.

3. בגרף לא מכון, הדרגה של קודקוד v ($d(v)$) היא סך הצלעות שיוצאות מ- v .

4. הגודל של גרף הוא $|G| = |V| + |E|$.

5. בגרף מכון, דרגת הכניסה ($d_{in}(v)$) היא סך הצלעות שנכנסות ל- v ובאותו הזמן דרגת היציאה ($d_{out}(v)$) היא סך הצלעות שיצואות מ- v .

6. מסילה מקודקוד u לקודקוד v היא n -ייה (v_0, v_1, \dots, v_n) כך ש- $v_0 = u, v_k = v$ ולכל i $v_i, v_{i+1} \in E$. אורך המסלילה הוא k . המרחק בין u ל v , $\delta(u, v)$, הוא האורך המינימלי של מסילה בין u ל- v .

7. גраф קשרי הוא גраф לא מכון שיש בו מסילה בין כל שני קודקודים. גраф קשר חזק אם לכל זוג קודડודים יש מסילה מכוונת מ- u ל- v וגם מ- v ל- u .

8. גраф G' הוא תת גרף של $G = (V, E)$ אם $V' \subseteq V$ ו- $E' \subseteq E$. רכיבי הקישור של G הם תנוי הגרפים הגדולים ביותר בהם הם קשורים (כלומר שככל קודקוד שנוסף להם יתנו לנו תת-graf לא קשרי).

9. עץ הוא גраф קשרי חסר מעגלים, והוא מקיים $|E| = |V| - 1$.

10. גרף ממושקל הוא גרף שלכל צלע שלו יש משקל $0 < w_i, v_j >$. שני קודקודים ללא צלע המחברת ביניהם הם בעלי משקל צלע במושקל ∞ . המשקל של מסילה (v_0, \dots, v_k) הוא סכום המשקלים, $\sum_{i=1}^k w(v_{i-1}, v_i)$

$$\text{משפט (לחיצות הידיים)}: \sum d(v_i) = 2|E|$$

$$\text{מסקנה: } |E| \leq |V|(|V| - 1) = \mathcal{O}(|V|^2)$$

$$\text{משפט (לחיצות הידיים עבור גרף ממושקל)}: \sum d_{in}(v_i) = \sum d_{out}(v_i) = |E|$$

2. דוגמאות ליניאריזציה

1. רשיימת סמיוכיות: לכל קודקוד v יש רשיימה מקוושרת L_v של השכנים שלו. גודל ההצגה בזיכרון הוא $\Theta(|V| + |E|)$. הצגה זו עדיפה במקרה של גרף עם מעט צלעות.

2. מטריצת סמיוכיות: מטריצה מסדר $|V| \times |V|$ בה צלע $e = (v, u)$ מציינת ערך השונה מאפס (המשקל במקרה של גרף ממושקל) במקומות (v, u) במטריצה. גודל ההצגה בזיכרון הוא $\Theta(|V|^2)$. הצגה זו עדיפה במקרה של גרף דחוס בצלעות. במקרה של גרף לא ממושן, $A = A^T$.

בעיה (בעיית המסלול הקצר ביותר) בהינתן גרף ושני קודקודים, נרצה לדעת מה המסלילה הקצרה ביותר בין שני הקודקודים הללו.

תתי בעיות של בעיית המסלול הקצר ביותר

1. מסלול יחיד: בהינתן שני קודקודים, נרצה לדעת מה המסלול הקצר ביותר בין שני הקודקודים.

2. מקור יחיד: בהינתן קודקוד, נרצה לדעת המסלול הקצר ביותר לכל קודקוד אחר בגרף.

3. כל הזוגות: בהינתן גרף, נרצה למצוא את המסלול הקצר ביותר בין כל שני קודקודים.

אנו נ עסק בעיית המקור היחיד. השתמש באלגוריתם BFS (חיפוש רוחב תחילה). אלגוריתם זה יסתכל על הקודקוד, ויסמן ברמה 0. לאחר מכן, יעבור על כל השכנים של הקודקוד ויסמן ברמה 1. בצעד $-1 + i$, הוא יסמן את כל הקודקודים השכנים לקודקודים ברמה $-i$. שרטם ערכנו עליהם ברמה $-1 + i$. הרמה של קודקוד היא המרחק שלו מהקודקוד המקורי.

נחלק את הקודקודים לשולש סוגים: (הקודקוד וכל הקודקודים שיש להם מסילה אליו שכבר נבדק), current (הקודקודים בחזיות שעכשיו נבדק) ו-visited (הקודקודים שעוזר לא גילינו). את ה-current-ים נשמר בתור.

כל קודקוד נשמר עוד שלושה שדות מעבר לשכנים: label (הסוג של הקודקוד, אחד מהשלשה הנ"ל), dist (הרמה של הקודקוד, המרחק מהמקור) ו-π (הקודם של הקודקוד בחיפוש). ננתח את זמן הריצה של BFS. נשים לב כי אין מורידים קודקוד מהטור בדיקות פעם אחת, ולכל קודקוד האלגוריתם עובר על כל הקודקודים השכנים לו ומציע מספר קבוע של פעולות בסיסיות. העובודה על כל קודקוד v בሪיצה על השכנים היא $d(v)$ ולכן סך הפעולות של האלגוריתם שרצות ב- $\mathcal{O}(1) |E| = 2|E|$. בנוסף, בתחילת הריצה אנחנו Überרים על כל הקודקודים בגרף ומএפסים אותם.

$$\text{לכן סה"כ, זמן הריצה הוא } \mathcal{O}(|V|) + \mathcal{O}(|E|) = \mathcal{O}(|V|^2)$$

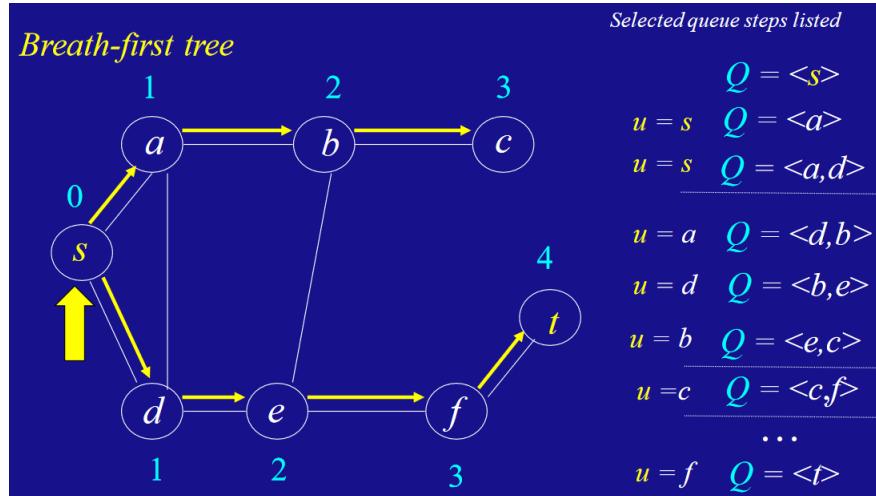
The BFS algorithm

```

BFS( $G, s$ )
 $s.label \leftarrow current; s.dist \leftarrow 0; s.\pi \leftarrow null$ 
for all vertices  $u$  in  $V - \{s\}$  do
     $u.label \leftarrow not\_visited; u.dist \leftarrow \infty;$ 
     $u.\pi = null$ 
EnQueue( $Q, s$ )
while  $Q$  is not empty do
     $u \leftarrow DeQueue(Q)$ 
    for each  $v$  that is a neighbor of  $u$  do
        if  $v.label = not\_visited$  then
            {  $v.label \leftarrow current$ 
               $v.dist \leftarrow u.dist + 1; v.\pi \leftarrow u$ 
              EnQueue( $Q, v$ )}
     $u.label \leftarrow visited$ 

```

איור 31 : אלגוריתם ה-BFS



איור 32 : דוגמת ריצת אלגוריתם ה-BFS

$$\delta(s, v) \leq \delta(s, u) + 1, \forall (u, v) \in E$$

הוכחה: נסמן $\delta(s, v) = k$. קיימת מסילה (v_0, \dots, v_k, v) מ- s ל- v ולכן

$$\delta(s, v) \leq k + 1$$

$$v.d \geq \delta(s, v), \forall v \in V$$

הוכחה: באינוריאנטה (אינדוקציה) עם השמורה שאחרי עדכוני השדה $dist$ מתקיימת הטענה.

בסיס: לאחר האתחול מתקיימים עבור $v \neq s$ $v.d = 0 = \delta(s, s)$, $v = s$

צעד : נניח שמעדכנים את $V \in v$ אזי לאחר העדכון,

$$v.d = u.d + 1 \stackrel{\text{ה''ג}}{\geq} \delta(s, u) + 1 \stackrel{\text{טענה 1}}{\geq} \delta(s, v)$$

■

לכן בסוף ריצת האלגוריתם נשמר התנאי הרצוי.

טענה 3 אם במהלך ריצת האלגוריתם מתקיים $Q = (v_1, \dots, v_r)$ אזי מתקיים :

$$v_i.d \leq v_{i+1}.d \quad \forall i \in [r-1] \quad (i)$$

$$v_1.d \leq v_r.d + 1 \quad (ii)$$

תרגום

שאלות

1. נתונים k מערכות ממוגנים במרחב n כל אחד. הציגו אלג'יעיל ככל שתוכלו ליצירת מערך ממון המכיל את כל איברי המערכות.

אם נצמיד את כל המערכות וממיינם כרגיל, נוכל לומר $\Theta(nk \log nk)$ בミון מבוסס השוואות.

אם נמיין בעורת לקיחת המינימום מכל המערכות כל פעם, זה יקח $\mathcal{O}(nk^2)$.

אם נעשה הפרד ומשול (כלומר נאחד כל זוג, ואז כל זוג של זוגות מאוחדים וכן הלאה) כמו ב-MergeSort, אז נעשה $\mathcal{O}(\log k)$ איחודים רוחביים שייעלו $\mathcal{O}(nk \log k)$ סה"כ (כל פעם מאחדים כל מערך עם מערך אחר) ולכן זה יקח $\mathcal{O}(\log k)$, שזה אופטימי.

2. נתונים n איברים בתחום $[m], \dots, [0]$. הצביעו מבנה נתונים שיאפשר לענות בזמן (1) על השאלה "אם x הוא בין n האיברים ו"מהו האיבר ה- k בגודלו" שזמן הבנייה שלו יהיהiesel ככל שאפשר.

נשים את האיברים בטבלת גיבוב במושלים $(U, \dots, [m], [0])$, ובשאילתת הראשונה נבצע פשוט Find שזה $\mathcal{O}(1)$ במקורה הגרוע. הבנייה שלו תיקח $\mathcal{O}(n)$ פעולות הכנסה.

נמיין את n האיברים בטבלת גיבוב במושלים $(U, \dots, [m], [0])$, ואחרת בעורת מיוון מבודס השוואות הפועל ב- $\mathcal{O}(n \log n)$, ונענה על השאלה השניה באמצעות גישה לאיבר ה- k במרחב הממוין.

3. האם נוכל לבנות מבנה נתונים עם הפ' הבאות?

$$\mathcal{O}(n \log n) \text{-ב-} \text{init}(S)$$

$$\mathcal{O}(1) \text{-ב-} \text{exists}(x)$$

$$\mathcal{O}(1) \text{-ב-} \text{the-kth}(k)$$

$$\mathcal{O}(1) \text{-ב-} \text{insert}(x)$$

את שלושת הפעולות הראשונות כבר עשינו בתרגילים הקודמים, השאלה היא האם נוכל למשה הכנסה ב- (1) יחד עם המבנית הזה. התשובה היא לא! נניח בשלילה שאפשר לבנות D כזו. יהיו מערך A של מספרים. נביט באלגוריתם הבא זהו אלג'חרץ ב- (n) הממיין מערך בלי הנחות נספנות עלייו (כלומר מבוסס השוואות) בסתיויה לחסם $(n \log n) \Omega$ שראינו בהרצאה.

```

1 D.init( $\emptyset$ )
2 for a $\in$ A
3     D.insert(a)
4 for i=1 to n
5     A[i] = D.the-kth(i)

```

Algorithm 23: אלגוריתם מיוון מבוסס השוואה בזמן לינארי

הערה כל הפעולות האריתמטיות במודולו p מעטה.

טענה יהי p ראשוני- -1 . $H_p = \{h_{a,b} : a, b \in F_p\}$, $h_{a,b}(x) = ax + b$ נגידר $a, b \in F_p$. $F_p = \{0, \dots, p-1\}$. $\forall a, b \in F_p$ קיימת $x, y \in F_p$ מושלמת מ- 2 לאוניברסלית מ- 1 .

הוכחה: עבור $x \neq y \in F_p$ ועבור $\gamma, \beta \in F_p$, בבחירה אקראית של $h_{a,b} \in H_p$

$$\begin{aligned}
P(h_{a,b}(x) = \gamma \wedge h_{a,b}(y) = \beta) &= P(ax + b = \gamma \wedge ay + b = \beta) \\
&\stackrel{x \neq y}{=} P\left(a = \frac{\gamma - \beta}{x - y} \wedge b = \frac{\beta x - \gamma y}{x - y}\right) \\
&\stackrel{\text{מארעיה בת"ל}}{=} P\left(a = \frac{\gamma - \beta}{x - y}\right) \cdot P\left(b = \frac{\beta x - \gamma y}{x - y}\right) \\
&= \frac{1}{p} \cdot \frac{1}{p} = \frac{1}{p^2}
\end{aligned}$$

בגיבוב מושלים, נוכל להניח הנחה מקלה שהיא שיש לנו k מפתחות סטטיים שהם ידועים מראש שאוותם כניט לטבלה. כדי לשמר על $(O(1))$ במקורה הגורע, נבחר h ראשי לטבלת הגיבוב אקראית ואז נחשב לכל תא, כמה מפתחות זה "כ" מומופים אליו ואז נבנה לכל אינדקס i טבלת גיבוב בגודל $h^{-1}(i)$ ובעזרת הסתבותות נוכל להבטיח שזה יהיה $(O(1))$. כדי להשתמש בשיטה לינארי, נרצה שהאיברים יפרסו באופן די אחיד כך שsek טבלאות הגיבוב החדשות יהיו בגודל לינארי.

תרגיל הדגימו הרצה של מציאת p ' גיבוב עם גיבוב מושלים כאשר

$$h(x) = ((ak + b) \mod p) \mod m$$

היא הפ' הראשית וגם

$$h_j(k) = ((a_j k + b_j) \mod p) \mod m$$

הפ' המשנית.

פתרונות שלב ראשון - מציאת h ראשית. נגריל a, b ונסמן ב- n_i את מספר האיברים כך ש- $i = h(x) \leq i \leq m-1$. נבדוק האם אם כן נסיים, אחרת נגריל מחדש a, b וחזור חלילה.

שלב שני - מציאת h -ים משניים. לכל i , נגיד h_i הממפה לטבלת גיבוב מגודל i^2 , ואם לא- h_i אין התנשויות בתת טבלת הגיבוב נסרים, אחרת נגיד שוב h_i וחזר חלילה.
הוכחנו בהרצאה כי בتوزלת יקח פחות מפעמים למציאת f' גיבוב מתאימה לכל טבלה.

שבוע | IX | Semi-Course Recap

תרגול

בתרגול הוגדרו עוד כמה הגדרות שנוספו [להגדרות בהרצאה](#) ובנוסף נעשתה דוגמת ריצה של אלג'י BFS וה-DFS.
ב-DFS, תת גרפ' הקודמים, G_π , הוא הגרף הכלול את כל הקודקודים ש-BFS עבר אליהם יחד עם הצלע (המכוונת) שדרך הגענו מכל v ל- u . הוא גם עץ רוחב, ככלומר שהוא מכיל את כל המסלילות הקצרות ביותר בין s לכל קודקוד אחר בגרף (שאליו יש לו מסילה בכלל).

שבוע | X | Depth First Search & Toplogical Sort

הרצאה

משפט (נכונות של BFS) בסיום ריצת האלגוריתם, מתקיים כי $\forall v \in V$:

$$v.d = \delta(s, v) \quad (i)$$

אם $\delta(s, v) < \infty$ אז קיים מסלול קצר ביותר מ- s ל- v שהקודקוד הפניו אחרון שלו הוא $\pi.v$. $\quad (ii)$

הוכחה: באינדוקציה על $\delta(s, v)$:

בסיס: ברור, $s = v$ ומתקיים $v.d = \delta(s, v)$.

צעד: יהי u הקודום ל- v במסלול הקצר ביותר מ- s . נביט בრיצת האלגוריתם כאשר u מוצא מהתור.

אם ביקרנו את v , אז $v.d \leq u.d + 1 = \delta(s, v)$.

אם לא, נעדכו $v.d = u.d + 1 = \delta(s, v)$.

■

החלק השני נובע די ישירות ונשאר לסטודנטית המשקיעה להוכיח אותו.

chiposh leu'mek tchilah, DFS هو אלגוריתם שבודם כל כניסה عمוק לגרף, והוא משמש שלב מקדים לפתרת בעיות אחרות. בעזרת האלגוריתם ניתן ליצור סידור לינארי של גרפים (אלגוריתם מיון) וגם לפטור את MST ומציאת רכיבי קשריות קשורים חזק.

האלג' מתחילה מ- s ובוחר שכן אחד שלו. וחופר כמה שיותר עמוק. כשהוא מסיים, הוא חוזר קודקוד אחד אחרה, סורק את הקודקוד הבא שלו שטרם נסרק ואז כשהוא מסיים הוא שוב הולך אחרה וכו'.

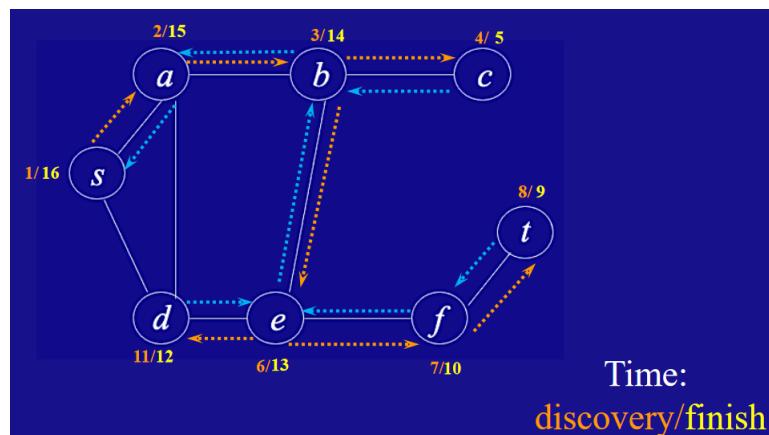
הפלט של האלגוריתם הוא (V, E_π) , כאשר $E_\pi = \{(v.\pi, v) : v \in V \wedge v.\pi \neq null\}$ שבסופו של דבר נוצרת מהאלג'.

עבור כל קודקוד, נשמר לכל קודקוד 4 שדות : π (הקודקוד הקודם הבודק השכן לו שדרכו הגיעו לקודקוד), d (זמן ה גילוי) ו- f (זמן בו סיימנו לחקר את כל השכנים של הקודקוד).

```

DFS( $G, s$ )
for each vertex  $u$  in  $V$  do
     $u.d \leftarrow null$ ;  $u.f \leftarrow null$ ;  $u.\pi \leftarrow null$ ;  $u.label \leftarrow not\_visited$ ;
     $time \leftarrow 1$ 
DFS-Visit( $s$ )
for each vertex  $u$  in  $V$  do
    if  $u.label = not\_visited$  then DFS-Visit( $u$ )
DFS-Visit( $u$ )
 $u.label \leftarrow current$ ;  $u.d \leftarrow time$ ;  $time \leftarrow time + 1$ ;
for each  $v$  that is a neighbor of  $u$  do
    if  $v.label = not\_visited$  then
         $v.\pi \leftarrow u$ 
        DFS-Visit( $v$ )
     $u.label \leftarrow visited$ ;  $u.f \leftarrow time$ ;  $time \leftarrow time + 1$ ;
```

איור 33 : פסויידו-קוד של DFS



איור 34 : דוגמת ריצה של DFS

הערות

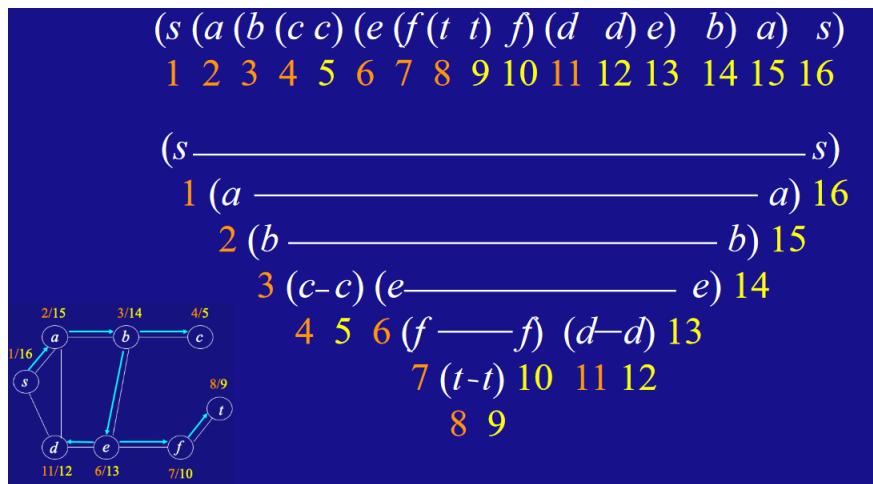
1. הפלט של האלג' תלוי באופן שבו אנחנו בוחרים שכנים.

2. במקום ריקורסיה, אפשר להשתמש במחסנית (כמו התור ב-SFS) כדי לעשות את זה איטרטיבית.

$\{u, v\} \in G_\pi$ אםDFS-Visit(u)-מ לקרא DFS-Visit (v) .3

. G_{π} בזמן ריצת $\text{DFS-Visit}(u)$ אם v הוא צאצא של u ב-

כל ריצת אלגוריתם יוצרת מחרוזות סוגריים. נוכחים כי היא תקינה.



איור 35 : דוגמה למחוזות סוגריים של DFS

משפט הסוגרים (Parenthesis Theorem) לכל שני קודקודים x, y מתקיימת אחת מהאפשרויות הבאות:

$[u.d, u.f] \cap [v.d, v.f] = \emptyset$ (i) (כמו בדוגמה).

כמו a, s בדוגמה). $[u.d, u.f] \subseteq [v.d, v.f]$ (ii)

$\cdot [v.d, v.f] \subseteq [u.d, u.f]$ (iii)

משפט (משפט המסלול שנסרק, DFS Theorem) (Visited Path Theorem).
בזמן $d.u$, ניתן להציג מסלול המורכב מקודוקדים שטרם בוקרו.

הובחת: \Leftarrow : נראה שההמיסלה מ- a ל- b ב- G_{π} מכילה קודקודים w עם $w.d < w.d.u$ מושפעת הסוגרים לכך w טרנס נבדק.

\Rightarrow נניח בשליליה ש- v לא צaczא של u ב- G_π . בה"כ v הוא הראשון במסילה שאינו צaczא של u . יהיו w העוקב של v במסלול لكن הוא צaczא של u , ולכן $w.f < u.f$. בנוסף, $u.d > v.d$ שכן $v.d < u.d$ אבל הוא לא צaczא שלו. לכן $w < v.d$ ולכן כ- v מסקר בתוקן DFS-Visit(w) ■

מיון טופולוגי הוא מיון שמקבל "גרף קדימיות", שבו כל קודקוד הוא משימה, וצכלעו מ- u משמעה שיש לבצע את u לפני v . אם יש מעגלים בgraf לא נוכל למצוא סידור של המשימות (כי נקבל מעגל אינסופי של דרישות קדם). בgraf מכובן קשר חסר מעגלים (DAG) תמיד ניתן למצוא סידור טופולוגי-לינארי כזה.

- 1 TopologicalSort(G):
- 2 Call DFS(G)
- 3 Return the list of vertices v_1, \dots, v_n such that $v_1.f > \dots > v_n.f$

Algorithm 24: אלגוריתם מילון טופולוגי

משפט TopologicalSort מפיק סידור טופולוגי (כלומר שנוכל לבצע את המשימות בסדר הנtent ולקבל סדרת פעולות תקינה) של G בזמן

$$\text{lienari}(|V| + |E|)\Theta.$$

הוכחה: תהי צלע (v, u) . נוכיח כי $f.v < f.u$ ונשים את הוכחת נכונותה.

אם $d.v < d.u$, אז ימשפט המסלול שנערך v הוא יצא של u ב- G_π ולכן ממשפט הסוגרים $f.v < f.u$.

אם $d.u > d.v$ אז אם אין מסלול מ- v ל- u , שכן זה היה יוצר מעגל. לכן, ממשפט המסלול שנערך u הוא לא יצא של v ב- G_π ולכן ממשפט

הסוגרים $f.u < f.v < f.d < f.u$.

■

תרגול

הגדלה מילון טופולוגי (TS) של גרף מכון חסר מעגלים (G) הוא סידור קודקודי G כך שאם v נמצא לפני u אז לא קיימת ב- G צלע $m-u$.

תרגיל כתבו אלג' המוצא בזמן $O(|V|)$ האם קיימים מעגל בגרף לא מכון.

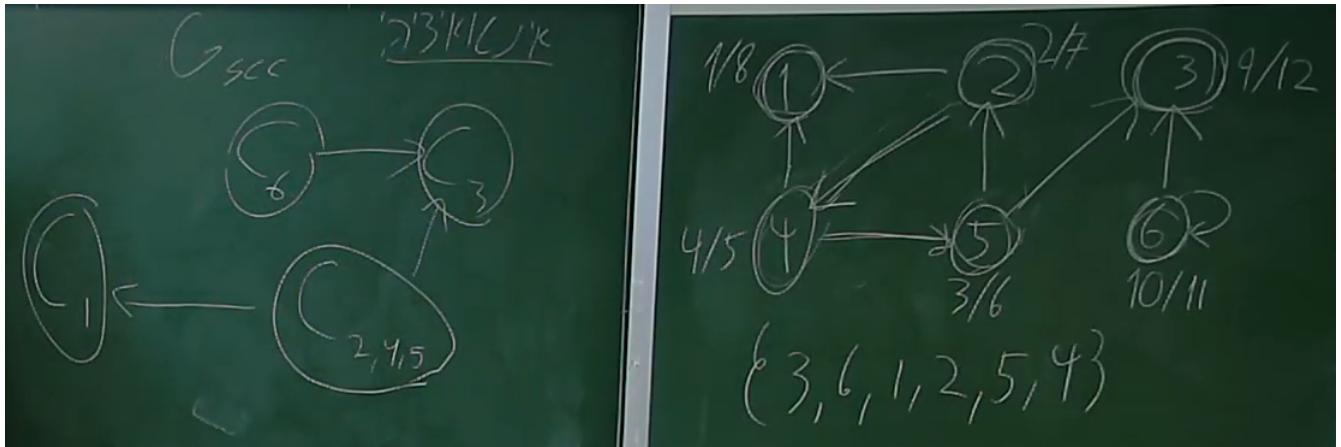
פתרון נבנה וריאציה לאלג' DFS כך שבכל שלב נבדוק האם הקודקוד הבא הוא visited ולא עברנו בצלע אליו, אם כן, נחזיר true, אחרת, false.

אבל נזכיר כי DFS הוא $O(|V| + |E|)$, אם כן, מהו האלג' שהצינו הוא $\mathcal{O}(|V|)$? אם אין מעגלים בגרף, אז יש לכל היוטר $1 - |V|$ צלעות בגרף ולכן הזמן הוא $\mathcal{O}(|V| + |V|) = \mathcal{O}(2|V|)$. אם יש במעגל גראף או אחריו $|V|$ צלעות נקבע שסדרנו גראף עם יותר צלעות מאשר קודקודים, כלומר שיש בו מעגל ולכן כבר נסיים את הריצעה עד אז ולכן זה עדין $\mathcal{O}(|V|)$.

סביר את האינטואיציה לאלג' SCC. לשם כך, נגידיר לכל גראף G את הגראף G_{SCC} שהוא גראף שקיים קשיות החזקה של G וצלעותיו יהיו הצלעות שמחברות בין רכיבי הקשיות הללו. גם ב- G וגם ב- G^T , רכיבי הקשיות החזקה נשאים אותו הדבר, ככלומר מתקיים $((G^T)_{SCC})^T = G_{SCC}$ ולבן ב- G יחד ב- G^T , DFS יסתובב לבדוק ברכיבי הקשיות החזקה. זאת משומש שב- G הוא יבודק שיש דרך מכל קודקוד לאחר בכיוון אחד ובריצה על G^T הוא יבודק שיש גם בכיוון הפוך. הריצה עם הסידור הספציפי היא כדי לוודא את שהבדיקה נעשת על אותם המסלולים של הקודקודים.

הגדרה יהיו G גראף לא מכון וקשרי. מפרק (Articulation Point) הוא קודקוד שם נסיר אותו מהגרף (ואת הצלעות המוחוברות אליו) נקבל גראף לא קשרי. גשר הוא צלע שהסרה שלה מהגרף תהפוך אותו לא קשרי.

נרצה לכתוב אלג' שיפארו לנו למצוא מפרקים וקשרים. הרעיון: בניית וריאציה על DFS ונשמר לכל קודקוד את $d.v$ (זמן הכניסה) וגם $v.earliest$ (זמן הכניסה הנמוך ביותר של קודקוד שניית להגעה אליו מ- v).



איור 36 : דוגמה ל- G_{SCC}

כדי לעשות זאת, בכל הגעה לקודקוד חדש נבדוק את הצלעות השכנות לו ונקבע את שדה $earliest$ שלו להיות הקודקוד עם הזמן הכנינה המינימלי שיש לו צלע אליו שלא נוספת כבר ליער ה-DFS. בדרך חוזר (ב-Backtracking) שוב נבדוק לכל קודקוד האם יש לו קודקוד שכן עם $earliest$ נמוך יותר, ואם כן נשנה אותו להיות כן.

טענה u הוא מפרק אם אחד מהbabisms מתקיימים :

(i) u הוא שורש העץ ביער ה-DFS ויש לו לפחות שני ילדים.

(ii) u אינו שורש עצם ביער ה-DFS ויש לו בן v כך שלא ניתן להגעה ממנו לאב קדמון של u .

לאחר ריצת ה-DFS, נעבור על כל קודקוד u . אם u הוא שורש עם שני בניים או יותר - נוסיפו לרשימה נק' המפרק. אם עבור $E \in E$ (u, v) הוא ריצת ה-DFS, נעבור על כל קודקוד v . אם v הוא שורש עם שני בניים או יותר - נוסיף לו לרשימה נק' המפרק. אם עבור $u.d < v.earliest$ מתקיימים $u.d - v.earliest$ נוסיף את u לרשימה נק' המפרק.

שבוע XII | Minimum Spanning Tree

הרצאה

הגדרה יהי $G = (V, E)$ גראף מכוכו. הרכיבים הקשורים חזק של G הם תת-הגרפים המקסימליים כך שיש מסילה בין כל קודקוד לאחר מכן (בשני הכוונים).

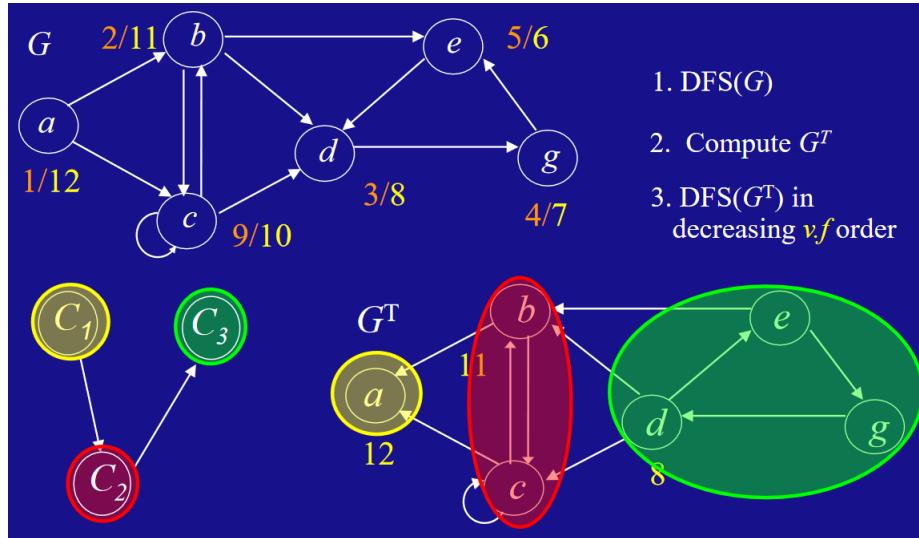
העזרה הרכיבים הקשורים חזק של G מייצרים חלוקה של G עם היחס שקיים מסלול דו צדדי בין כל שני קודקודיים.

הגדרה יהי $G = (V, E)$ גראף מכוכו. הgraft המושכלן של G הוא $G^T = (V, E^T)$ כאשר $E^T = \{(u, v) : (v, u) \in E\}$.

משפט יהיו C, C' שני רכיבי קשרות שונים של G . תהי $f(C) > f(C')$ כאשר f היא פונקציה המושכלנת על כל רכיב קשרות חזק. המשמעות של זה היא שנסים לעסוק ב- C רק אחרי שנסים לעסוק בכל C' .

- 1 SCC(G):
- 2 DFS(G) to compute v.f, $\forall v \in V$
- 3 Compute G^T a
- 4 DFS(G^T) in the order of decreasing v.f
- 5 Return sets of vertices grouped by the DFS trees created in DFS(G^T)

Algorithm 25: G all algorithm for finding the strong components of G



איור 37 : דוגמת הרצה של SCC, כאשר הסימוניים הצהובים ב- G^T הם $v.f$ עבור הקודקודים שאנו מתחילה מthem DFS

מסקנה לכל צלע $f(C) < f(C')$, כאשר $u \in C, v \in C'$, $u, v \in E^T$ מתקיים $(u, v) \in E^T$.

משפט אלגוריתם SCC מחשב את הרכיבים הקשורים חזק של גראף מכובן G .

הוכחה: באינדוקציה על k ש- k -העצים הראשונים מהאלגוריתם הם אכן ורכיבים קשורים חזק.

בסיס ($k = 0$): ברור.

צעד ($k + 1$): נניח כי העץ מושרש בקודקוד u (שמננו אנחנו מתחילה את ריצת ה-DFS אחריו מציאת k הרכיבים הקשורים חזק הקודדים).

כאשר u נבדק ב-DFS(G^T), אף אחד מהקודודים האחרים עוד לא נבדק, ולכן כל הקודודים הם בעזםCACIM של u בעז-DFS (משמעות המסלול שנטרך).

מה'א, כל רכיב קשירות חזק C' שעוזר לא התגלה מקיים $f(C') < f(C)$ (כי אנחנו סורקים את G^T בסדר f . u יורד) ולכן לא קיימת צלע G^T -מ- C ל- C' , ולכן כל צלע שיווצאת מ- C ל- C' הולכת לרכיב קשירות חזק שכבר סרקנו אבל מה'א אין כלהה ולכן אין אף קודקוד שלא נמצא ב- C שהואCACIM של u , ולכן רכיב קשירות אחד.

בעיה עץ פורש מינימלי (MST) הוא תת הקבוצה של צלעות של גראף לא מכובן ממושקל כך המשקל של סך הצלעות הוא מינימלי אבל עדין יש מסלול בין כל שני קודודים בגרף. לדוגמה, ברישות אתרים נרצה להציג תשתיית במחירים מינימלי (פר מרחק) כך שנגיע לכל האטררים הרצויים.

הערה עבור $T \subseteq E$, בגרף ממושקל, נגיד $w(T) = \sum_{e \in T} w(e)$.

הגדרה יהיי $G = (V, E)$ גרף ממושקל, לא מכובן וקיים. עץ פורש של G הוא תת קבוצה $T \subseteq E$ של צלעות מכובנות כך שהגרף G' $= (V, T)$ גראף ממושקל, לא מכובן וקיים. הוא קשור וחסר מעגלים (ולכן הוא עץ).

הגדרה עץ פורש מינימלי הוא עץ פורש עם משקל מינימלי.

הערה לגרף מסויים ניתן יותר מעץ פורש מינימלי אחד.

אסטרטגיה חמדנית: נתחילה עם $\emptyset = T$. נוסיף כל פעם צלע e כך ש- $\{e\} \cup T$ הוא עדין תת קבוצה של עץ פורש מינימלי (זו תהיה האינוריאנטה שלנו). צלע המקיים תנאי זה נקראת צלע בטוחה. נוסיף צלעת עד שנתקבל גראף קשור.

הגדרה חתק של גראף לא מכובן $G = (V, E)$ הוא חלוקה של V לשתי קבוצות זרות, ככלומר $C = (U, V \setminus U)$.

הגדרה צלע e נקראת צלע קלה אם $e \in U$ וגם $v \in V \setminus U$.

הגדרה חתק $C = (U, V \setminus U)$ מכבד את קבוצת הצלעות $A \subseteq E$ אם אין צלע של A שחوتכת את C , ככלומר כל צלע המכילה קודקודים שנשנים ב- U או ב- V .

הגדרה צלע e נקראת צלע קלה אם היא הצלע עם המשקל המינימלי של הצלעות שחותכות את החתק.

משפט יהיו $G = (V, E)$ גראף לא מכובן, ממושקל וקיים, T המוכלת בעץ פורש מינימלי C של $A \subseteq E$ החתק שמכבד את צלע קלה, אזי $e = (u, v)$ היא צלע בטוחה.

הוכחה: יהיו T עץ פורש מינימלי שמכיל את A . אם T לא מכיל את e או סימנו. אחרת, $e \notin T$. בגלל ש- u, v נמצאים בצדדים שונים של החתק, יש לפחות צלע אחת $(u', v') = e' \in T$ שנמצאת על מסילה שחותכת את החתק (ומחברת בין u ו- v). כי $e' \notin A$ החתק מכבד את הקבוצה. לכן, A מוכלת ב- $\{e'\}$, וכך מספיק להראות ש- T' הוא עץ פורש מינימלי (כי ברור ש- $e' \in T'$).

בגלל ש- e' הוא על המסלול היחיד בין u ל- v , מוכיחת מהפרק את T' לשני רכיבי קשרות. הוספה $e = (u, v)$ מ לחברת מחדש את רכיבי הקשרות ויוצרת עץ פורש T' .

מהיות $e = (u, v)$ הינה צלע קלה החותכת את T' , מוכיחת מהפרק את $w(T') \leq w(T) + w(u, v)$ וולכן $w(T') \leq w(T) + w(u, v) \leq w(u', v') + w(u, v) = w(T)$.

$$w(T') = w(T) - w(u, v) + w(u', v') \leq w(T)$$

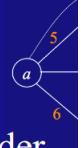
■ אבל T' עץ פורש מינימלי וולכן $w(T') \leq w(T)$ וולכן גם T הוא עץ פורש מינימלי.

אלגוריתמים המחשבים עצים פורשים מינימליים

1. פרים (Prim): נשמר קבוצה A שהיא עץ והצלע הבטוחה שנוציא היא הצלע בעלת המשקל המינימלי שמחברת קודקוד שאינו ב- A (ככלומר החתק הוא קודקודים בעץ מול כל שאר הקודקודים).

2. קורסקל (Kruskal) : נשמר קבוצה A שהיא עץ והצלע הבטוחה שנוסף היא הצלע בעלת המשקל המינימלי שמחברת בין שני רכיבי קשרות שונים של A .

```
MST-Kruskal( $G$ )
 $A \leftarrow \emptyset$ 
for each vertex  $v \in V$  do Make-Set( $v$ )
Sort edges  $e \in E$  in non-decreasing weight order
for each edge  $e = (u,v) \in E$  do
    if Find-Set( $u$ )  $\neq$  Find-Set( $v$ ) the trees are distinct
    then
         $\{A \leftarrow A \cup \{e\}$ 
        Union( $u,v$ )} combine two trees
return  $A$ 
```



איור 38 : מימוש של Kruskal

במימוש זה, מתחילה מהקבוצה הריקה ויוצרים קבוצת סינגלטון לכל קודקוד, שזה יהיה הקבוצה שלו ביער. נעבור על הצלעות בסדר עולה לפי משקלן ובכל פעם נבדוק האם הצלע ממחברת בין שני רכיבים שונים. אם כן, נוסיף את הצלע לקבוצה ונאחד את הרכיבים של הקודקודים שהוא מחברת ביניהם.

זמן הריצה של מימוש זה יגთח בהמשך. במימוש זה, Q הוא ערימת מינימום. ראשית נעבור ונקבע לכל קודקוד את המפתח שלו, שהוא

```
MST-Prim( $G=(V,E),root$ )
 $Q \leftarrow V \setminus \{root\}$ 
 $A \leftarrow \emptyset$ 
for each vertex  $v \in Q$  do
     $v.key \leftarrow w(root,v); v.\pi \leftarrow root$ 
while  $Q$  is not empty do
     $u \leftarrow \text{Extract-Min}(Q)$ 
    Add ( $u.\pi, u$ ) to  $A$ 
    for each  $v$  that is a neighbor of  $u$  do
        if  $v \in Q$  and  $w(u,v) < v.key$ 
        then  $v.\pi \leftarrow u$ 
             $v.key \leftarrow w(u,v)$ 
Return  $A$ 
```

איור 39 : מימוש של Prim

המשקל של הצלע שמחברת אותו לשורש (ובהמשך המשקל של הצלע הקלה ביותר שתחבר אותו לעץ A) ואת הקודם שלו להיות השורש. נעבור על ערימת המינימום עד שככל הקודקודים יהיו בעץ ובכל פעם נוסיף את הקודקוד עם המשקל המינימלי לעץ ונעבור על כל השכנים שלו ונדכן להם את המפתח שלהם להיות המשקל הנמוך ביותר החדש (אם השתנה) לחיבור לעץ.

זמן ריצה : בניית העירימה היא $\mathcal{O}(|V|)$ וכן גם איפוס הערכיה בלולאה הראשונה. הלולאה השנייה (החיצונית) רצה $|V|$ פעמים והוצאה איבר מערימת המינימום זורשת $\mathcal{O}(\log |V|)$. הלולאה הפנימית $\mathcal{O}(|E|)$ פעמים והבדיקות שנעשות בה צורכות (1) \mathcal{O} למעט שינוי הערך של המפתח של השכן שדורש $\mathcal{O}(\log |V|)$. לכן סה"כ זמן הריצה הוא

$$\mathcal{O}(|V| + |V| + |V|(\log |V| + |E| \log |V|)) = \mathcal{O}(|E| \log |V|)$$

תרגול

משפט (למה החתך) יהיו $G = (V, E, w)$ גרף ממושקל, C חתך של הגרף, F תת-גרף ללא מעגלים ולא צלעות בחתך (צלעות בין קודקודים מקבוצות החתך) כך ש- e ב- F קיים עפ"מ T' כך ש- $T' \cup \{e\} \subseteq T$, כאשר e היא צלע מינימלית בחתך.

באלג' של שקורסקל, נמיין את הצלעות לפי משקלן בסדר עולה. לעבור על כל הצלעות לפי הסדר, אם הצלעה הנוכחית לא סוגרת מעגל, נוסיף אותה לצלעות העפ"מ. נשתמש במנג'ת $Union - Find$ (amortized) $\mathcal{O}(1)$ Union, FindSet .
זמן הריצה : מיוון יקח $\mathcal{O}(|E| \log |E|)$ ואז המעבר על כל הצלעות יקח (amortized) $\mathcal{O}(|E| \log |E|)$, כלומר סה"כ

$$\mathcal{O}(|E| \log |E|) = \mathcal{O}(|E| \log |V|^2) = \mathcal{O}(|E| \log |V|)$$

אלג' פרים דומה ל-BFS. במקומות תור Q נחזיק עירימת מינימום, ובכל שלב נתקדם לאיבר עם הערך המינימלי וنعשה Extract-Min .
נשים לב כי העירימה מגדרה לנו חתך והאלג' כל פעם מוציאה את הצלע הקטנה ביותר בחתך. בגלל נקודות המבט הזו נוכל להשתמש בلمota החתך להוכחת הנוכנות ולקבל הוכחה טריוויאלית (בעזרת אינדוקציה).

משפט יהיו $G = (V, E, w)$ גרף ממושקל וקשר e ב- E כך שכל משקל הצלעות שונים זה מזה. אזי לגרף יש עפ"מ יחיד.

הוכחה : נניח בשלילה שקיים שני עפ"מים שונים T_1, T_2 נגידר את e' להיות קבוצת הצלעות שנמצאת רק ב- T_1 או רק ב- T_2 . מההנחה $e' \in T_1 \setminus T_2$ ולכן נבחר את e' הצלע עם המשקל המינימלי ב- T_2 . בה"כ $e' < e_j \in T_2 \setminus T_1$.

נוסיף את e' ל- T_2 . בהכרח נסגור מעגל, $(e', e_1, \dots, e_k, e')$. במעגל בהכרח קיימת צלע $e_j \in T_2 \setminus T_1$ (כי כל המעלג ב- T_1 , שהוא עז), כלומר $e' < e_j$. מינימליות e'

נבייט ב- $T_2 \setminus \{e_j\}$ שווה T' $= (T_2 \setminus \{e_j\}) \cup \{e'\}$ והוא מתקיים

$$w(T') = w(T_2) - w(e_j) + w(e') < w(T_2)$$

ובנוסף ב- T' יש $|V| - 1$ צלעות ואין בו מעגלים ולכן הוא עצם פורש, וכך מצאנו עצם פורש עם משקל קטן ממש מושך, כלומר T_2 כולל איינו עפ"מ, סטירה.

נביט במימוש נוסך לאלג' קורוסקל. נעבור על צלעות G לפי משקלן בסדר יורד (מהגדול לקטן), אם צלע שyiיכת למעגל- G , נמחק אותה מ- G . זה בדיקת הפוך מקורוסקל, במקרה להוסיף צלעות מותאיימות, אנו מוחקים צלעות לא מותאיימות.

משפט (שראיינו בדיסקרטית) הטענה של צלע מסוימת קשירה לא מכון הופעתה אותה ללא קשר או פותחת מעגל כלשהו.

לכן, כדי למש את האלג' המוזכר לעיל, מה שבעצם נעשה זה לעבור על הצלעות לפי סדר יורד של משקלן, להסיר את הצלע, להריץ DFS/BFS על הגרף עם הצלע המוסרת, ואם נגלה מהרצאה זו שהגרף כבר לא קשיר, נחזיר את הצלע, ואם הוא נשאר קשיר, נסיר אותה ונעבור לצלע הבאה. מהמשפט הקודם הקודם האלג' יחזיר בסופו של דבר עפ"מ.

סיבוכיות האלג': המינון דרוש $\mathcal{O}(|E| \log |V|)$, ואו הרצת BFS/DFS למציאת הקשיירות דורשת $\mathcal{O}(|E| \log |E|)$ ונו עושים זאת $|E|$ פעמים, ככלומר סה"כ

$$\mathcal{O}(|E| \log |V| + |E|(|E| + |V|)) = \mathcal{O}(|E|^2)$$

שזה פחות טוב מקורוסקל המקורי.

שבוע XII | שבוע XII | Weighted Single Source Shortest Path

הרצאה

עד כה, פתרנו את בעיית המיקור האחד והמסלול האחד עם גרפים לא משוקלים. עתה נפתר את הבעיה עם משקלות.

הגדרה משקל המסלול הקצר ביותר מוגדר ע"י δ .

הערה במסילות קצרות ביותר אין מעגלים, כי נוכל פשוט להיפטר מהמעגל ולהתקבל מסלול קל יותר.

הערה כל תת מסילה של מסילה קצרה ביותר גם היא זמנה קצרה ביותר.

הגדרה עצם מסלול קצר ביותר המושרש ב- s הוא עצם פורש המושרש ב- s כך שהמסילה בין s לכל קודקוד אחר היא המסילה הקצרה ביותר בgraf המקורי.

האלג' שנדונו בהם יחוירו את עצם המסלול הקצר ביותר בעזרת השיטה π.u, ככלומר G_π יהיה עצם מסלול קצר ביותר המושרש ב- s .

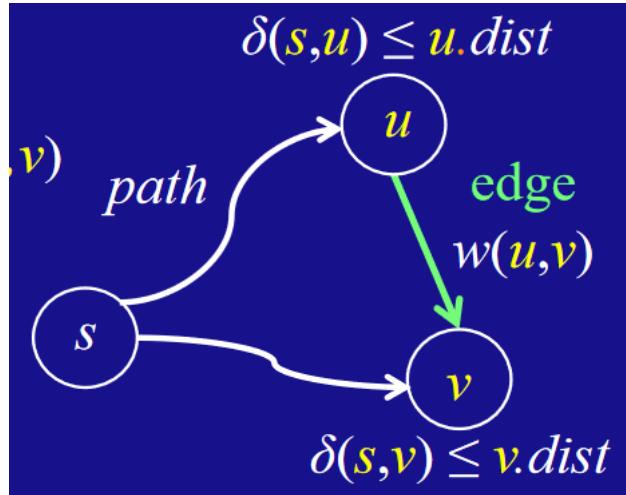
באתחול, נקבע $v.dist = \pi.v$ לכל הקודזדים, $s.dist = 0$ ו- $v.dist = \infty$ לכל $s \neq v$. אחרי האתחול, נסנה את $v.dist$, $\pi.v$ רק באמצעות פ' Relaxation. הרעיון באלג' הוא שמצאנו קודקוד שהמעבר דרכו הוא יותר קל מהמסלול הנוכחי.

```

1 Relax(u,v):
2     if v.dist < u.dist + w(u,v):
3         v.dist = u.dist + w(u,v):
4         v.π = u

```

Algorithm 26: Relax-



איור 40 : אינטואיציה מאחוריו Relax

תחת האלגוריתם, נוכל להסביר כמה דברים.

.1 אם $v.\pi \neq \text{null}$ אז $(v.\pi, v)$ היא צלע ב- G .

.2 $v.dist \geq \delta(s, v)$ כי אחרי עדכון $v.dist$

$$v.dist = u.dist + w(u, v) \geq \delta(s, v) + w(u, v) \geq \delta(s, v)$$

.3 ואם אין מסלול מ- s ל- v אז $v.dist = \infty$

$v.dist = \delta(s, v)$ רק יורד לאורך האלגוריתם.

טענה (תכונת ההתקנסות) יהי $v \rightarrow u$ המסלול הקצר ביותר ב- G בין u, v . אם $u.dist = \delta(s, u)$ הצלע (u, v) כל הזמן לאחר מכן.

הוכחה:

$$v.dist = u.dist + w(u, v)$$

$$= \delta(s, u) + w(u, v)$$

$$= \delta(s, v)$$

ולא ניתן כי נקבע את הערך מתחילה לכך בהמשך.

טענה (תכונת תת-עץ הקודמים) אם $\forall v \in V, v.\pi = s$ אז לכל $v \in V$, $v.dist = \delta(s, v)$.
 או במלילים אחרות, G_π הוא עץ מסלול קצר ביותר.

האלגוריתם הראשון שנראה הוא Djikstra.

```
Dijkstra( $G=(V,E),s$ )
   $s.dist \leftarrow 0; s.\pi \leftarrow null$ 
  for all vertices  $u$  in  $V-\{s\}$  do
     $u.dist \leftarrow \infty; u.\pi = null$ 
   $S = \emptyset$ 
   $Q \leftarrow V$ 
  while  $Q \neq \emptyset$  do
     $u \leftarrow \text{Extract-Min}(Q)$ 
     $S \leftarrow S \cup \{u\}$ 
    for each neighbor  $v$  of  $u$  do
      Relax( $u,v$ )
```

איור 41 : האלגוריתם של Djikstra

באלג' זהה, Q הוא ערימת מינימום עם המפתח $.dist$. הרעיון של האלג' הוא ראשית את התחול, ואז לבחור כל פעם את הקודקוד בעל ה- $.dist$ המינימלי ולעשות Relax לכל הצלעות שיצאות ממנו וכך שוב ושוב עד שעברנו על כל הקודקודיים.

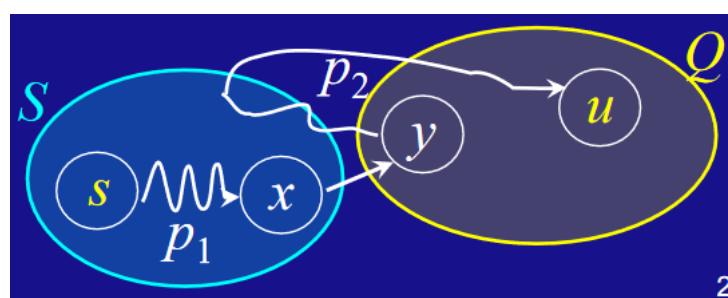
נשים לב כי זהה בסך הכל וריאציה על BFS, כאשר כאן אנחנו מתייחסים גם למשקל.
 $L-S$ אין משמעות (מלבד להיות $V \setminus Q$ לאורך כל הרצאה, ולווער באנגלית).

משפט בסוף הרצאה של האלגוריתם של Djikstra מתקיים $\forall u \in V, u.dist = \delta(s, u)$.

הוכחה: נוכיח כי כאשר u מוצא מ- Q , $u.dist = \delta(s, u)$.

בבסיס: כש- s מוצא, $s.dist = 0$.

צעד: נניח כי u הוצא מ- Q . יש מסלול מ- s ל- u (אחרת $u.dist = \infty$). ייְהי p המסלול קצר ביותר מ- s ל- u . יהי y הקודקוד המוקדם ביותר ב- p כך שה- y אינו ב- S כאשר (לפניהם) u הוצא ו- x הקודקוד שלפני y . לכן מתקאים u



איור 42 : אינטואציה לצעד האינדוקציה

נוכיח כי $y.dist = \delta(s, y)$ כאשר y הוצאה מ- Q . מה"א, מתקיים $x.dist = \delta(s, x)$ כאשר x הוצאה. הרצנו את Relax על x אחרי ש- x הוצאה ולפני ש- y הוצאה. לכן מתכונת ההתכונות, $y.dist = \delta(s, y) \leq \delta(s, x)$ כי $y = s$.

נוכיח כי $u.dist = \delta(s, u)$. בגלל ש- y הוא לפני u במסלול קצר יותר בין s ל- u , אז $y.dist \leq u.dist$

$$y.dist = \delta(s, y) \leq \delta(s, u) \leq u.dist$$

אבל בgal Sh- y ו- u שניהם היו ב- Q כשל- u הוצאה ולכן u בא לפני y בתור הקדימות ולכן $u.dist \leq y.dist$ וגם על הדרך הוכחנו כי $y = u$.

מסקנה מטענה שראינו לעיל (שנווכה בתרגום), זה גורר שם G_π הוא אכן עז מסלול קצר ביותר המושרש ב- s .

נניח את זמן הריצה של האלגוריתם של Dijkstra . האתחול של העירימה דרוש $\mathcal{O}(\log |V|)$. בתוך הלולאה השליפה דורשת $\mathcal{O}(|V|)$. או $\mathcal{O}(|V|^2 \log V)$ פעמים סה"כ. את פ' ה-Relax אנחנו מרים פעם אחת על כל צלע, ובහינתן שאחנו אכן מעודכנים את שדה $dist$ - s . העדכו בעירימה ידרוש $\log |V|$. לכן סה"כ נקבל

$$\mathcal{O}(|V|) + \mathcal{O}(|V| \log |V|) + \mathcal{O}(|E| \log |V|) = \mathcal{O}((|V| + |E|) \log |V|)$$

נוכל למש את העירימה אחרת. אם נעשה זאת באמצעות מערך נתו, אז עדכון ערכים יקח $\mathcal{O}(1)$ ומציאת המינימום לוקחת $\mathcal{O}(|V|)$ ולכן הסיבוכיות תרד ל- $\mathcal{O}(|V|^2 \log V)$ ובקשה ש- $\mathcal{O}(|E|) = \Omega(|V|^2)$ זה עדיין.

עתה נרצה גם לאפשר שימוש במשקולות שליליים (או 0). הגדרת משקל המסילה נשארת זהה. אם יש מעגל שלילי בגרף אז אם יש קודקודים שיש ביניהם מסלול שעובר דרכו אז המרחק ביניהם יהיה $-\infty$. כדי לפטור זאת, נגיד

$$\delta(u, v) = \min \left\{ w(p) : u \xrightarrow{p} v \right\} > -\infty$$

תתי מסילות של מסילות קצרות ביותר, וכן קיימים מסילות קצרות ביותר, וכן קיימים מסלול קצר ביותר ללא מעגלים (בעצם, במקרה שהוא שמי שליליים, ההגדרות שקולות).

עם זאת, אם u בא לפני v במסלול קצר ביותר מ- s ל- v אז לא בהכרח מתקיים $\delta(s, v) \leq \delta(s, u)$ ולכן גם האלגוריתם של Dijkstra לא יעבד.

בגלל שאחנו עדיין משתמשים ב- Relax , נשמר על כך ש- $v.dist \geq \delta(s, v)$ רק יורד לאורך הריצה. בנוסף תוכנות ההתכונות ותכונות הקודדים גם הן ימשיכו להתקיים.

נזכיר עתה את האלגוריתם של Bellman-Ford

נכונות האlg' לא תלויות בסדר המעברים על הקודדים/צלעות.

```

Bellman-Ford( $G, s$ )
  Initialize ( $G, s$ )
  for  $i \leftarrow 1$  to  $|V| - 1$  do
    for each edge  $(u, v) \in E$  do
      Relax( $u, v$ )
    for each edge  $(u, v) \in E$ 
      if  $v.dist > u.dist + w(u, v)$ 
      then return “negative cycle”
  return “no-negative cycle”

```

איור 43 : האלגוריתם של Bellman-Ford

משפט בסוף ריצת האלגוריתם של Bellman-Ford, $v.dist = \delta(s, v)$ ו אם אין בגרף מעגלים שליליים, אז קיבל G_π עץ מסלול קצר ביותר המושרש ב- s ואם יש בגרף מעגלים אז האלג' יחזיר שיש מעגלים שליליים.

הוכחה: ראשית נניח כי אין מעגלים שליליים. יהיו v קודקוד ויהי v_0, \dots, v_k מסלול קצר ביותר בין s ל- v שאינו בו מעגל וכלן בפרט מתקיים $.k \leq |V| - 1$

נוכיח באוינדוקציה כי לאחר שהאלגוריתם עבר i פעמים על כל הצלעות, המרחק של v_i אכן, כמובן, קבוע, כלומר, $.v_i.dist = \delta(s, v_i)$.

בבסיס : ברור.

צעד : מה "א", $v_i.dist = \delta(s, v_i), (v_{i-1}, v_i)$. מתכונת ההתכונות, ומהעובדה שהרצינו את Relax על v_{i-1} . $v_{i-1}.dist = \delta(s, v_{i-1})$. האיטרציה ה- i .

בפרט, לאחר האיטרציה ה- i , $v.dist = \delta(s, v)$. מתכונת הקודמים, $v.dist = \delta(s, v)$. מעתה, כאמור, v עץ מסלול קצר ביותר ב- s .

■ בתרגול נוכיח כי האלג' יחזיר "מעגל שלילי" אם s יש מעגל שלילי.

ננתן את זמן הריצה של האלג'. אנחנו מביצאים $\mathcal{O}(|V| \cdot |E| \cdot |E|)$.

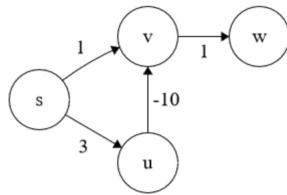
תרגול

נזכר באלג' של Djikstra: האלג' יבוד בקרה דומה לאלג' פרימ, רק שבמוקום לעשות ExtractMin למשקל הצלע הנמוך בערימה, נעשה ExtractMin למשקל המסלול ביותר מ- s שנמצא בערימה. נשמר לכל v מושנה $dist(v)$ שהיא שינה הערך המרחק המינימלי מ- s ו- (v) π שיהיה הקודם של v במסלול הקצר ביותר מ- s ל- v (בסוף של דבר). במעבר על צלע (u, v) השתמש בפ' העזר Relax, שתעדכן את שדות v אם הם יותר אידיאלים מאשר הנוכחיים שלו.

שאלות

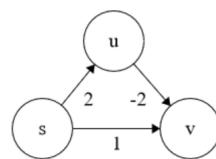
- אם אלג' Djikstra עובד כאשר יש צלעות שליליות בגרף?
- לא! בדוגמה המאוירת, נעבור קודם על (s, v) , נעדכן את $dist(v)$ להיות 1 ומבצע Relax על w , כך שבסופה של דבר $2.dist(w) = 2$

לאחר מכן, נעבור על (s, u) ו- (s, v) ונעדר את $dist$, כראוי עבור s, v, u . אבל w לא יעדכן שכן לא ניתן שוב לשכניו של v כי כבר עברנו עליו, ולכן נקבל שהמסלול הקצר ביותר מ- s ל- w הוא במשקל 2, אפילו שהוא במשקל 6.



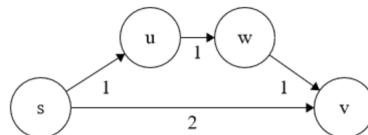
איור 44: דוגמה נגדית ל-Dijkstra עם צלעות בעלות משקל שלילי

2. עתה, נוסיף לכל קודקוד את $|e'|$ כאשר e' היא בעלת המשקל המינימלי בגרף ואז נרים על זה e' . האם אז נוכל להתמודד עם צלעות שליליות? לא! עבור הגרף המאויר, לאחר הוספת 2 לכל הצלעות, נקבל שהמסלול הקצר ביותר מ- s ל- u הוא $((s, v), (v, u))$, במשקל 3 (בגרף החדש), אך בעצם המשקל המינימלי הוא $((s, u), (u, v))$.



איור 45: דוגמה נגדית ל-Dijkstra עם צלעות בעלות משקל שלילי והוספת המשקל המינימלי בגרף

3. האם כל מסלול קצר ביותר בין שני קודקודים מוכל בעפ"מ של גרף? לא! בgraf המאויר, עפ"מ הוא $\{ (s, u), (u, w), (w, v), (s, v) \}$ אבל המסלול הקצר ביותר מ- s ל- v הוא $(s, u), (u, v)$ ולא נכלל בעפ"מ הנ"ל.



איור 46: דוגמה נגדית ל-Dijkstra עם צלעות בעלות משקל שלילי והוספת המשקל המינימלי בgraf

משפט (תכונת שיפור מסלול) יהיו $(s = v_0, \dots, v_k = v)$ מסלול קצר ביותר בין s ל- V_k ב- G . אם מתבצעות פעולות Relax על הצלעות לפי הסדר (משמאלה לימין) או בסימון מתקיים $dist(v) = \delta(s, v)$.

עתה נביט באילג' בלמן-פורד: האילג' יכול לקבל הצלעות עם משקלים שליליים. לכל קודקוד נשמר ערך $dist(v)$ שהוא המרחק מ- s (כמו ב-Dijkstra). נרים את הצלעות Relax לכל הצלעות במשך $1 - |V|$ איטרציות. נסה לעדכן את הצלעות עוד פעם אחת. אם נצליח לעדכן צלע, משמע שיש מעגל שלילי בgraf.

טענה יהי G גראף ממושקל עם קודקוד מקור s כך ש- G לא מכיל מעגלים שליליים. אז לאחר $1 - |V|$ איטרציות של הלולאה המרכזית בבלמן-פורד, $\forall v \in V, dist(v) = \delta(s, v)$.

הוכחה: יהי קודקוד s וכי $(s = v_0, \dots, v_k = v) = p$ מסלול קצר ביותר מ- s ל- v . אין מעגלים שליליים ולכן מסלול קצר ביותר הוא בהכרח מסלול פשוט. לכן p מכיל לכל היותר $1 - |V|$ צלעות. לכל $1 - |V|$ האיטרציות מתבצעת פעולה Relax על כל הצלעות ב- E .
 $\forall i \in [k]$ באיטרציה ה- i של הלולאה מתבצעת פעולה Relax על (v_{i-1}, v_i) , כלומר מtbody{לען} פעולות Relax על הצלעות של p לפי הסדר (אוili עם פעולות Relax נוספת). לכן מתכונת שיפור המסלול, לאחר האיטרציה ה- k , מתקיים $dist(v) = \delta(s, v) = \delta(s, v)$ וראינו בהרצתה כי ■

הערה אינואטיבית, אם החלטנו לאחר האיטרציה ה-1 $- |V|$ לשפר שוב את המסלולים בגרף זה אומר שיש מעגל שלילי, כי אפשר לרדת מתחת $\delta(s, v)$ (כפי שתואר בהוכחה הנ"ל, רק שם לא ירדנו מתחת לו).

תרגיל הצעו אלג' הרץ בזמן לינארי ב- $|V|$ ומחשב לבודקoid s את מרחקו מכל קודקוד v בגרף שניתן כי הוא עז.

פתרון נבנה וריאציה על אלג' BFS/DFS ונדכן את משקל המסלול המctrבר מ- s במעבר על כל צלע, בדומה לאlg' Djikstra. בערך אין מעגלים ולכן יש מסלול יחיד מכל קודקוד לכל קודקוד, והוא בפרט המינימלי. זמן הריצה הוא $\mathcal{O}(|V| + |E|) = \mathcal{O}(|V|)$ כי זהו עץ ולכן $|E| = |V| - 1 = \mathcal{O}(|V|)$.

All Pairs Shortest Path | XIII

הרצאה

נפתרו את בעיית המסלול הקצר ביותר בגרף מכון ממושקל, הפעם בגרסת כל הזוגות.

הפתרון הכני פשוט הוא להריץ את בעיית המקור היחיד לכל קודקוד, שזה דורש

$$\mathcal{O}((|E| \cdot |V|) \cdot |V|) = \mathcal{O}(|V|^4)$$

מעבר (Bellman-Ford) או

$$\mathcal{O}((|E| \log |V|) |V|) = \mathcal{O}(|V|^3 \log |V|)$$

מעבר Djikstra במקרה שאין משקלות שליליים).

עם זאת, נוכל להגיע לפתרון כללי ב- $\mathcal{O}(|V|^3 \log |V|)$ בעזרת "כפל מטריצות" ו- Floyd-Warshall.

נתעלם מהקרה שיש בו מעגלים שליליים הפעם.

נניח מעתה בה “ $\exists i \in [n], \forall j \in [V] \text{ such that } w(i,j) = \infty$ ”, כלומר $w(i,j) = \infty$ עבור כל $j \neq i$.

$$[W]_{ij} = \begin{cases} 0 & i = j \\ w(i,j) & i \neq j \wedge (i,j) \in E \\ \infty & i \neq j \wedge (i,j) \notin E \end{cases}$$

הפלט של האלגוריתם יהיה מטריצת מסלולים קצרים, L , שבעורמה $[L]_{ij}$ מכיל את המשקל המינימלי של מסלול בין i ל- j . בנוסף, נחזיר מטריצת קודמים Π , שבעורמה $[\Pi]_{ij}$ מכיל את הקודם של הקודם j במסלול הקצר ביותר מ- i ל- j , או $null$ אם $j = i$. כדי למצוא את המסלול הקצר ביותר בין i ל- j , נסתכל על $[W]_{ij}$, ואז הקודם שלו וכן הלאה וכן הלאה עד שנגיע ל- j . נסתכל מראש על אלגוריתם “הכפלת המטריצות”.

נגיד $L^{(m)}$, מטריצת המסלולים הקצרים ביותר המוגבלת, ע”י מטריצת מרחקים עבור מסלולים עם לכל היוטר m קודמים (המשקל המינימלי של כל מסלול באורך לכל היוטר m בכלל תא).

נשים לב כי $L^{(1)} = W$. אם אין מעגלים שליליים, או $|V| \geq m$. א. $L^{(1)} = L$. ב. אם $k \geq \log_2 |V|^3$ ומשם נסיק כי ניתן לחשב את $L^{(2^k)}$ ועבורו $\mathcal{O}(L^{(2^k)})$ ומשם נסיק כי ניתן לחשב את $L^{(q+r)}$ ועבורו $\mathcal{O}(L^{(q+r)})$. נקבל כי ניתן לחשב את $L^{(q+r)}$ ב- $\mathcal{O}(|V|^3 \log |V|)$.

למה לכל V $[L^{(q+r)}]_{ij} = \min_{1 \leq k \leq n} [L^{(q)}]_{ik} + [L^{(r)}]_{kj}$, $i, j \in V$.
הערה אינטואטיבית, נוכל לפרק כל מסלול בין i ל- j באורך לכל היוטר q ומסלול באורך לכל היוטר r . לכן נוכל לבחור את המינימליים מהפירוק ונקבל שזה המינימלי בכללי.

הוכחה: נוכח כי $[L^{(q+r)}]_{ij} \leq \min_{1 \leq k \leq n} [L^{(q)}]_{ik} + [L^{(r)}]_{kj}$.
מספיק שנוכח כי $[L^{(q+r)}]_{ij} \leq [L^{(q)}]_{ik} + [L^{(r)}]_{kj}$, $\forall k \in [n]$. קיימות מסילות p_1 בין i ל- k ובין k ל- j באורך q ו- p_2 בין i ל- j באורך r . חיבור p_1, p_2 ייתן לנו מסלול באורך לכל היוטר $q+r$ בין i ל- j עם $w(p) = w(p_1) + w(p_2)$.

$$w(p) = w(p_1) + w(p_2) = [L^{(q)}]_{ik} + [L^{(r)}]_{kj}$$

$$\text{ולכן } [L^{(q+r)}]_{ij} \leq [L^{(q)}]_{ik} + [L^{(r)}]_{kj}$$

$.s \leq q + r$ ו- $w(p) = [L^{(q+r)}]_{ij}$ במסילה בין i ל- j עם משקל $p = (v_0, \dots, v_s)$.
נוכח כי $[L^{(q+r)}]_{ij} \geq [L^{(q)}]_{ik} + [L^{(r)}]_{kj}$ ו- $t \leq q + r - t \leq r$ עם $p_2 = (v_t, \dots, v_s)$ ולכן קיבלנו פרק את p במסילה (v_0, \dots, v_t) .

$$[L^{(q+r)}]_{ij} = w(p) = w(p_1) + w(p_2) \geq [L^{(q)}]_{ik} + [L^{(r)}]_{kj} \geq \min_{1 \leq k \leq n} [L^{(q)}]_{ik} + [L^{(r)}]_{kj}$$

■

מסקנה כדי לחשב את $\mathcal{O}(n^3)$, וזה יקח $[L^{(q)}]_{ik} + [L^{(r)}]_{kj}$, עבור על כל זוג i, j , ונחשב ב- (n) את המינימום של $L^{(q)}, L^{(r)}$ מ- $L^{(q+r)}$

```

Extend-Shortest-Paths( $A, B$ )
 $n \leftarrow A.rows$ 
Let  $C = (c_{ij})$  be an  $n \times n$  matrix.
for  $i \leftarrow 1$  to  $n$  do
    for  $j \leftarrow 1$  to  $n$  do
         $c_{ij} \leftarrow \min_k (a_{ik} + b_{kj})$ 
return  $C$ 

```

איור 47 : אלגוריתם הרחבת המסלולים הקצרים ביותר

הערה הסיבה שהאלג' דומה לכפל מטריצות היא שהוא החלוטין למעט השורה בה אנו מישמים ב- c_{ij} , שם בכפל מטריצות נשים את $[A]_{ik} [B]_{kj}$ ולא את המינימום על הסכום שלהם.

```

Faster-All-Shortest-Paths( $W$ )
 $n \leftarrow \text{rows}[W]$ 
 $L^{(1)} \leftarrow W$ 
 $m \leftarrow 1$ 
while  $m < n-1$  do
     $L^{(2m)} \leftarrow \text{Extend-Shortest-Paths}(L^{(m)}, L^{(m)})$ 
     $m \leftarrow 2m$ 
return  $L^{(m)}$ 

```

איור 48 : אלג' מציאת מסלולים קצרים ביותר בין כל זוג קודקודים ב- $\mathcal{O}(|V|^3 \log |V|)$

ונכל לחשב את Π מתוך L באמצעות החישוב הבא : $[\Pi]_{ij} = k$ שuboרו k הוא בעל ערך מינימלי (מסלול + משקל כולל).

חישוב זה דורש $\mathcal{O}(n)$ לכל זוג קודקודים, כולם סה"כ. ווכל בנוסף לחשב את Π בזמן העדכנים של $L^{(k)}$, אבל בזה לאណון בהרצאה (אלא בתרגול).

עתה נטוק ב-[Floyd-Warshall](#).

הגדירה קודקוד פנימי של מסילה $v_l = (v_0, \dots, v_l)$ הוא כל קודקוד ב- p שאינו v_0 או v_l .

מטריצת המסלולים הקצרים המוגבלת, תוגדר ע"י $[D^{(k)}]_{ij}$ הוא המשקל המינימלי של מסילה מ- i -ל- j שקודקודיה הפנימיים מוכלים ב- $\{1, \dots, k\}$.

אם כן, $D^{(n)} = L \cdot D^{(0)} = W$

$\mathcal{O}(|V|^3 \cdot D^{(n)}) = L \cdot \mathcal{O}(|V|^2 \cdot D^{(k-1)} \cdot D^{(k)})$ וואז אם נרץ את העדכנים n פעמיים, נקבל את $D^{(n)}$ וראה כיצד נחשב את

הערה נסמן $P_{ij}^{(k)}$ את אוסף כל המסלולים מ- i -ל- j עם קודקודים פנימיים ב- $\{1, \dots, k\}$

$$\left[D^{(k)} \right]_{ij} = \min \left\{ d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)} \right\}$$

הערה בכל מסלול ב- $P_{ij}^{(k)}$ יש את k כקודקוד פנימי, או שייכות ל-. המשקל המינימלי של מסילה מהסוג הראשון הוא $d_{ij}^{(k-1)}$ כי זה האיחוי של המסלול עד k ו- $(k-1)$ ואילו עבור הסוג הראשון זה פשוט $d_{ij}^{(k-1)} + d_{ij}^{(k-1)}$.

הוכחה:

$$\begin{aligned} d_{ij}^{(k)} &= \min_{p \in P_{ij}^{(k)}} w(p) \\ &= \min \left\{ \min_{p \in P_{ij}^{(k-1)}} w(p), \min_{p \in P_{ij}^{(k)} \setminus P_{ij}^{(k-1)}} w(p) \right\} \\ &= \min \left\{ d_{ij}^{(k-1)}, \min_{p \in P_{ij}^{(k)} \setminus P_{ij}^{(k-1)}} w(p) \right\} \end{aligned}$$

עתה נוכיח כי

$$d_{ik}^{(k-1)} + d_{kj}^{(k-1)} = \min_{p \in P_{ij}^{(k)} \setminus P_{ij}^{(k-1)}} w(p)$$

יהי p_{ij} מסלול קצר ביותר ב- $P_{ij}^{(k)} \setminus P_{ij}^{(k-1)}$, כלומר p_{ij} בעל משקל מינימלי מבין כל המסלולים מ- i ל- j עם קודקודים פנימיים ב- $\{1, \dots, k\}$ ו- k -קודקוד פנימי.

$$\text{מספיק להוכיח כי } w(p_{ij}) = d_{ik}^{(k-1)} + d_{kj}^{(k-1)}$$

נפרק את p_{ij} ל- p_{ik} מ- i ל- k ו- p_{kj} מ- k ל- j . הוא בעל משקל מינימלי מבין המסלולים מ- i ל- k עם קודקודים פנימיים ב- $\{1, \dots, k-1\}$. כלומר, p_{kj} בעל משקל מינימלי מבין המסלולים מ- i ל- k עם קודקודים פנימיים ב- $\{1, \dots, k-1\}$.

$$w(p_{ij}) = w(p_{ik}) + w(p_{kj}) = d_{ik}^{(k-1)} + d_{kj}^{(k-1)}$$



```

Floyd-Warshall( $W$ )
 $n \leftarrow W.\text{rows}$ 
 $D^{(0)} \leftarrow W$ 
for  $k \leftarrow 1$  to  $n$  do
    for  $i \leftarrow 1$  to  $n$  do
        for  $j \leftarrow 1$  to  $n$  do
             $d_{ij}^{(k)} = \min(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)})$ 
Return  $D^{(n)}$ 

```

איור 49: אלג' Floyd-Warshall

הערה מכאן נוכל לחשב את Π_B - $D^{(n)}$ מ- \mathcal{O} בלי לפגוע בסיבוכיות הזמן.

תרגום

¹. APPSP (All Pairs Shortest Paths) – שתהיה מטריצת המרחקים הקצרים ביותר בין כל זוג של נקודות.

בהרצתה כי $W \bullet L^{(m)} = L^{(m-1)}$ כאשר לא מדובר בכפל מטריצות, אלא בפעולה

$$\left[L^{(m-1)} \bullet W \right]_{ij} = \min_{1 \leq k \leq n} \left\{ \left[L^{(m-1)} \right]_{ik} + [W]_{kj} \right\}$$

היעון הוא להסתכל על כל המסלולים מ- v לכל הנקנים האפשריים, שיהיו הקודם ל- v במסלול הקצר ביותר, ולבזוק איזה מהם יתנו מסלול אידיאלי. האלגוריתם יתחיל מ- $W = \emptyset$ וונעלת עד $L = L^{(1)} = \{v\}$ שזויה המטריצה הרצוייה. כל פעולה כפולה דורשת $\mathcal{O}(|V|^3)$ וnocheshב $|V| - 1$ פעולות כפולה וכן "כחסיבויות שנΚבל היא" $\mathcal{O}(|V|^4)$, שזה כמו הפתרון הנאיי עם Bellman-Ford.

2. FAPSP (Faster APSP). נוכל לחשב בכל שלב $L^{(i+j)} = L^{(i)} \bullet L^{(j)}$ (הוכיחנו נכונות בהרצאה). הרעיון בחישוב הזה הוא להכליל את הנציג שהיה הקודם במסלול קצר יותר בחישוב $W \bullet L^{(m-1)}$ לסדרה נציגים שיהיו בעצם המסלול שיופיע לפני j במסלול קצר $.L^{(1)} \bullet L^{(\frac{n}{2})} \bullet L^{(n)}$... מ- m -ה בחישוב $|V|^3 \log |V|$ -ה. כך נוכל לשפר את הסיבוכיות מ-

3. Floyd-Warshall. הפעם נביט ב- $D^{(k)}$ שההיה מטריצת המרחקים הקצרים ביותר עבור מסלולים שקודקודייהם הפנימיים הם ב- $\{1, \dots, k\}$.

$$\left[D^{(k)} \right]_{ij} = \min \left\{ \left[D^{(k-1)} \right]_{ij}, \left[D^{(k-1)} \right]_{ik} + \left[D^{(k-1)} \right]_{kj} \right\}$$

כלומר שהמסלול הקצר ביותר מ- i ל- j שקודקודיו ב- $\{1, \dots, k\}$ הוא המסלול הקצר יותר מבין המסלולים שכבר מצאו (ש קודקודיים ב- $\{1, \dots, k-1\}$) או אלו שעורבים דרכו k באמצעותם.

ונכל באופן דינמי לחשב את גרען הקודמים בכל עדכון של $D^{(k)}$, ע"י

$$\left[\Pi^{(k)} \right]_{ij} = \begin{cases} \left[\Pi^{(k-1)} \right]_{kj} & \left[D^{(k-1)} \right]_{ij} \leq \left[D^{(k-1)} \right]_{ik} + \left[D^{(k-1)} \right]_{kj} \\ \left[\Pi^{(k-1)} \right]_{ij} & \text{otherwise} \end{cases}$$

נוסחה זו נובעת מהנוסחה ל- $D^{(k)}$, כאשר העדכון פשוט מתייחס למסלול ההיפותטי הנוחchip שאנחנו עוקבים אחריו כרגע ב-

הגדירה יהי $G = (V, E^*)$ גרען מכובן. הסגור הטרנזיטיבי של G הוא הגרף $(V = \{v_1, \dots, v_n\}, E)$ כך ש-

$$E^* = \{(v_i, v_j) : G \text{ בערך } v_i \text{ ל-} v_j\}$$

הסגור הטרנזיטיבי עוזר לנו להבין אילו קודקודים ישנים מאילו קודקודים.

כדי ליחס את הסגור הטרנזיטיבי, נבנה וריאציה על פלואיד-ורשא�.

במקום $D^{(k)}$, נזכיר מטריצת $T^{(k)}$ המוגדרת ע"י

$$\left[T^{(k)} \right]_{ij} = \begin{cases} 1 & \{v_1, \dots, v_k\} \text{ עם קודקודים המוכלים ב-} \\ & \text{מסלול מ-} v_i \text{ ל-} v_j \\ 0 & \text{otherwise} \end{cases}$$

ונכל ליחס את $T^{(k)}$ באופן דומה לפלואיד-ורשא� מ-.

ונכל למצוא באופן יותר עילית. נרץ BFS על כל הקודקודים וכל ה

שבוע | XIV Disjoint Set

הרצאה

נזכר כי באlg' של קרויסקל השתמשנו במבנה'ת שאפשר לנו ליצג של קבוצות זרות עם פעולות (amortized) $\mathcal{O}(1)$.

הבעיה שאויה פוטר המבנה'ת Disjoint Set היא כדילקמן. בהינתן n איברים שונים, $X = \{x_1, \dots, x_n\}$, נספק אוסף דינמי של k קבוצות זרות $\{S_1, \dots, S_k\}$ כך ש- $i \neq j \Rightarrow S_i \cap S_j = \emptyset$. נרצה לתמוך בפעולות:

• MakeSet(x) יוצר קבוצה חדשה S_x שאיברה היחיד הוא x . נניח כי x לא נמצא באף אחת מהקבוצות הקיימות ב-

• Union(x_i, x_j) שמחליף את שתי הקבוצות S_i ו- S_j המכילות את x_i ו- x_j בהתאמה באיחוזן. נניח כי x_i ו- x_j נמצאים בשתי קבוצות שונות.

שMOVED ומחזיר את הנציג של S_x , הקבוצה ב- x נמצאת.

המטרה שלנו היא למצוא מבני'ת שתומך ב- m שאלות UnionFind על קבוצות זרות באופןיעיל. כלומר, לא מטריד אותנו בהכרח כמה זמן תיקח כל שאלה בנפרד, אלא מה הזמן הממוצע בהינתן סדרה של m פעולות כללה. ניתוח כזה נקרא סיבוכיות זמן amortized.

באופן נאיבי, נוכל לשמר רשימה לכל קבוצה. תדרוש $\mathcal{O}(n)$ FindSet, $\mathcal{O}(1)$ MakeSet (לעבור על כל הרשימות) ולכן גם Union תדרוש $\mathcal{O}(n)$, כלומר m פעולות יקחו במרקחה הגרוע $\mathcal{O}(m \cdot n)$.

ריאנו שני אלג' שמשתמשים ב-UnionFind. כפי שראינו בעבר, Prim עוזה זאת. אך נוכל גם להשתמש זהה כדי למצוא רכיבי קשירות בגרף לא מכון (כל קבוצה זורה תהיה רכיב קשירות).

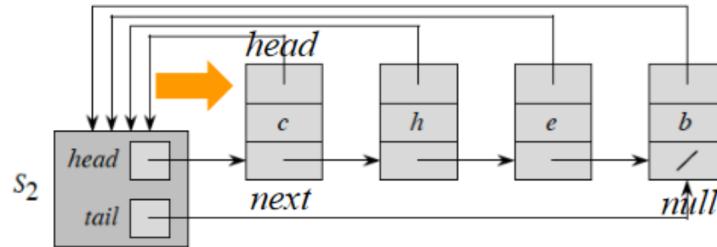
```
Connected-Components( $G=(V,E)$ )
for each vertex  $v$  in  $V$ 
    do { Make-Set( $v$ ) };
for each edge  $(u,v)$  in  $E$ 
    do {
        if Find-Set( $u$ )  $\neq$  Find-Set( $v$ )
            then Union( $u,v$ ) }
```

איור 50 : אלג' למציאת רכיבי קשירות בגרף לא מכון באמצעות UnionFind

סיבוכיות הזמן של האלג' היא $\mathcal{O}(f(|V|, 3|E| + |V|))$ כאשר $f(n, m)$ היא הסיבוכיות של ביצוע m פעולות על n איברים.
 $\mathcal{O}(|E| \log |E| + f(|V|, 3|E| + |V|))$ הסיבוכיות של פרימ תהיה

מימושים של Disjoint-Set

1. ניצג כל קבוצה רשימה מקוורת L_i , כאשר ראש הרשימה L_i יהיה הנציג שלה. לכל רשימה יש שני שדות: מצביע לראש הרשימה ומצביע לסוף הרשימה. לכל איבר ברשימה יהיו שני שדות: האיבר הבא שלו וראש הרשימה בה הוא נמצא (ראו איור).



איור 51 : רשימה מקוורת יחד עם השדות הנוספים שהגדכנו

ידרוש $\mathcal{O}(1)$ בנסיבות המימוש הנאיבי.

פישוט יחזיר את $x.head$ וזה ידרוש $\mathcal{O}(1)$ בלבד הרשימות הקשורות, נוכל לתמוך במערך של איברי X שלכל אחד נשמר מצביע למקום שלו בזיכרון).

Union יחבר בין שתי רשימות הקשורות ב-(1) \mathcal{O} (לחבר את הראש של L_2 לתחילת L_1), אז יעדכו לכל האיברים ב- L_2 את שדה הראש להיות הראש של L_1 , שזה יזרוש סה"כ (n) \mathcal{O} במקרה הגרוע. סדרה של n פעולות MakeSet שלאחרת נבע $n - \sum i = \Theta(n^2)$ פעולות UnionSet תזרוש סה"כ ($n + 1$ כאשר $i = 2n - 1$) ואיחוד הרשימה יקרה במקרה הגרוע, כלומר $L_1 = \{x_i\}, L_2 = \{x_{i+1}, \dots, x_n\}$ כך שזרוש (i) Θ על כל איחוד, כלומר קיבילנו $m \cdot n = m \cdot f(m, n)$ שזה לא טוב. כדי לשפר את זה, נוכל לבחור את L_1 להיות הרשימה הארוכה יותר באיחוד, וכך נבע פחות עדכוני מצביעים.

תחת שיפור זה, הוספה ותמייה בשדה לכל רשימה תזרוש (1) \mathcal{O} . אם מספר האיברים בשתי הרשימות הוא (n) אז הסיבוכיות היא עדין (n). עם זאת, בסדרה של m פעולות, נוכל להוכיח כי זה יזרוש רק $\mathcal{O}(m + n \log n)$ פעמים, כי בכל פעם הוא יכנס לרשימה שגדולה לפחות פי 2 מוקדמתה.

משפט בהצגת Set ובහינתו היוריסטיקת האיחוד הממושקל, m פעולות של Union יקחו

$$\mathcal{O}(m + n \log n)$$

הוכחה: m פעולות FindSet, MakeSet יקחו $\mathcal{O}(m) \cdot \mathcal{O}(1) = \mathcal{O}(m)$. נותר להוכיח כי פעולות האיחוד יקחו $\mathcal{O}(n \log n)$. יש לכל היותר $1 - n$ פעולות איחוד. בכל איחוד, עדכון ה-tail לנקח (1) \mathcal{O} במקרה הגרוע. נותר להוכיח כי אנו מעדכנים את שדה הראש של המצביעים $\mathcal{O}(n \log n)$ פעמים. נוכיח כי אנו מעדכנים את $x.head$ $\mathcal{O}(\log n)$ פעמים. אחרי האיחוד ה- k שבו x מעודכן, אנו נכנסים לקבוצה שגדולה לפחות 2^k , ולכן נעדכן אותו $\mathcal{O}(\log n)$ פעמים. ■

2. כל קבוצה תיווצר על ידי עצם מסוון, כאשר השורש הוא הנציג. לכל איבר יש שדה אחד - שדה ההורה שלו בעץ בו הוא נמצא. ההורה של השורש הוא עצמו וההורה של איבר שאינו באף קבוצה הוא *null*.

Union יחבר את אחד השורשים לשורש האחר ויסיר את הצלע העצמית שלו. זה יזרוש (1) \mathcal{O} . MakeSet ייצור עצם שבו רק השורש. זה יזרוש (1) \mathcal{O} . FindSet יקופב אחר ההורה של האיבר עד שיגיע לשורש. זה יזרוש (h) \mathcal{O} כאשר h הוא עומק העץ. במקרה הגרוע, $m \cdot n$ פעולות יקחו ($n \cdot m$) \mathcal{O} עבור n פעולות FindSet כשות- h עולה כל פעם ב-1 כי איחדנו עצם עם יחידון כל פעם. נקבע בכמה פתרונות לשיפור הסיבוכיות.

(א) היוריסטיקת איחוד לפי דרגה. לפיה, באיחוד, נחבר את השורש של העץ הנמוך לשורש של העץ הגבוה. כדי לעשות זאת, נוסיף לכל איבר שדה גובה, שישמר את הגובה של תת העץ של אותו איבר. בכל פעולה איחוד, נעדכן את שדה הגובה של השורש. כך, אם ישיחס חזק בין גבהי השורשים אז גובה השורש הגבוה יותר לא ישתנה. אם יששוון בין גבהי השורשים נקבל עלייה ב-1 של הגובה.

משפט גובה העץ המskineli כמשמעותם באיחוד לפי דרגה הוא $\mathcal{O}(\log n)$

הוכחה: באינדוקציה על מספר הפעולות שנדרשו כדי לייצר את העץ.

בסיס : לפני האיחוד הראשון, $h = 0$ ולעתו אין יש קודקוד אחד.

צעד : נחלק לשני מקרים :

אם גובה העץ לא גדול, אחד העצים היה קצר מהآخر, וכך יש לפחות 2^h קודקודים מה"א בעץ הישן ולאחר מכן גם בחדש.

אם גובה העץ גדול, אז שני העצים הם באותו גובה ואותם הגובה החדש הוא $1 + h$. מספר הקודקודים מה"א בעץ החדש הוא ■

$$\text{פחות } 2 \cdot 2^h = 2^{h+1} \text{ כרצוי.}$$

מסקנה הפעולות הרכולות של m פעולות היא $\mathcal{O}(m \log n)$.

(ב) היוריסטיקת דחיסת המסלולים. לפיה, בכל פעולה FindSet נקשר כל קודקוד על מסלול החיפוש לשורש יישירות לשורש.

```
1 FindSet(x):
2     if x ≠ x.parent:
3         x.parent = FindSet(x.parent)
4     return x.parent
```

FindSet: אלגוריתם דחיסת מסלולים ב-Algorithm

משפט עם היוריסטיקת דחיסת המסלולים, המקורה הגרוע על m פעולות הוא $\mathcal{O}(m \log n)$.

(ג) גם היוריסטיקת דחיסת המסלולים וגם איחוד לפי דרגה.

משפט כמשמעותם בשתי היוריסטיקות עבור Set Disjoint Bi-coding של עיר, m פעולות ידרשו $\mathcal{O}(m\alpha(n))$ כאשר $\alpha(n)$ היא

פ' אקרמן ההופוכה, שמקיימת $3 \leq 2^{2^{65536}} - 4, \forall n \leq 2^{65536}$.

ונכל לסכם ולקבע שנוכל למש Disjoint Sets בזמן ליניארי Amortized על מספר הפעולות, כלומר במציאות רכיבי הקישורות מוזכר ב-

$\mathcal{O}(|V| + |E| \log |V|)$ במקרה הגרוע ובקריטריאון הסיבוכיות היא