ניתוח לשיעורין

הגדלת והקטנת מערך דינאמי – ArrayList:

כשהמערך מלא, נגדיל פי 2. זה דורש O(k) פעולות עבור מערך באורך k. כדי לגרום למערך באורך k להתארך צריך להכניס k+1 איברים. ובדרך, ביצענו רק logk פעולות שדרשו הגדלה. כל השאר היו רק הכנסה, O(1). כלומר, עבור n הכנסות:

$$\underbrace{(n-\log n)}_{\text{ops that were } O(1)} + \underbrace{(2^0+1)+(2^1+1)+\cdots+(2^{\log n}+1)}_{\text{ops that were } O(k)} = n-\log n+2n = O(n)$$

הקטנה: כשהמערך רבע-מלא, נקטין את הגודל לחצי. ככה כדי להגיע ממערך בגודל 2k לגודל k צריך 3k/2 הוצאות. ויוצא שהזמן לפעולה ממוצעת הוא O(1).

Union – find

מבנה נתונים יעיל לייצוג קבוצות. כל איבר הוא קודקוד, וכל קבוצה מיוצגת ע"י נציג – שורש. פעולות:

- ייצור קודקוד חדש - create.
- איחוד שתי קבוצות – מקבל שני קודקודים ומכניס את השורש של אחד מהם בתור בן של השני - union.
- מציאה – מחזיר את השורש של העץ שקודקוד שייך אליו - find.

כל קודקוד מחזיק מצביע $\pi(v)$ – אבא שלו. אצל השורש זה null.

create זה O(1). union הוא O(1) אחרי שמוצאים את השורשים – משתמש בfind. כלומר הסיבוכיות של find תקבע את הסיבוכיות של כל המבנה.

נגדיר לכל קודקוד שדה **rank - דרגה**, שהוא חסם עליון לגובה של הקודקוד. באיחוד, הקודקוד עם הדרגה הנמוכה יהפוך להיות בן של הגבוה. טענה: בעץ עם דרגה k יש לפחות 2^k קודקודים. הוכחה באינדוקציה על k: עבור k=0, מתקיים. נניח שמתקיים עבור n-1, ונוכיח עבור n: כדי להגיע לדרגה n צריך לאחד שני עצים מדרגה n-1. מהנ"א, מספר הקודקודים יהיה לפחות 2^n.

דחיסת מסלול: כשמחפשים שורש של קודקוד, כל קודקוד שעוברים בדרך יהפוך להיות בן של השורש, כשנמצא אותו. מבחינה אסימפטוטית זה אותו זמן ריצה, וזה גורם לכך שאם נעשה: n פעולות create, n-1 פעולות union, ו-n פעולות find, זמן הריצה הכולל יהיה $O(n\cdot\alpha(n))$, כאשר α היא הפונקציה ההפוכה לאקרמן. כלומר לכל n גם מאד גדול, זה כמעט מספר קבוע. אז בסה"כ הסיבוכיות שואפת ל O(n) עבור n פעולות, או O(1) לפעולה אחת בניתוח לשיעורין.

ההוכחה לזה מתבססת על הגדרת הירכיית אקרמן, לא בחומר של הקורס. (רק למסלול מצטיינים):

פונקציית אקרמן היא פונקציה רקורסיבית, המוגדרת:

$$A_1(n) = 2n$$
$$A_k(n) = A_{k-1}^n(1) = \underbrace{A_{k-1}\big(A_{k-1}\big(\cdots A_{k-1}(1)\big)\cdots\big)}_{\text{n times}}$$

אז לדוגמה: $A_2(n) = A_1^n(1) = 2(2(\cdots 2\cdot 1)\cdots) = 2^n$

$$A_3^n = A_2\big(A_2\big(\cdots A_2(1)\big)\cdots\big) = \underbrace{2^{2^{2^{\cdot^{\cdot^{2}}}}}}_{\text{n times}}$$

נגדיר לשם נוחות: $A(n) \coloneqq A_n(3)$

פונקציה הפוכה מוגדרת: $f(n) \coloneqq \min\{k|F(k)\geq n\}$ כלומר, המספר הכי קטן שאם נפעיל עליו את F, נקבל לפחות את n.

לדוגמה אם: F(8)=999,999,999 ,F(7)=100,000 אז f(30,000,000)=8 כי אם ניקח את 7 נקבל פחות. רק עבור מספר גדול מF(8) נקבל f(n)=9.

הפונקציה ההפוכה ל $A_2(n)$ היא $\log_2(n)$.
ההפוכה של $A_3(n)$ היא $\log^*(n)$, ההפוכה של $A_4(n)$ היא $\log^{**}(n)$...

(תזכורת: $\log^* n$ זה מספר הפעמים שאפשר להפעיל log עד שנגיע ל1: $\log^* n = \begin{cases} 0, & \text{if } n\leq 1 \\ 1+\log^*(\log n), & \text{if } n>1 \end{cases}$)

עוד תזכורת:

$$f(n) = o\big(g(n)\big) \iff \lim_{n\to\infty}\left(\frac{f(n)}{g(n)}\right) = 0$$

כלומר אם f(n) גדל הרבה פחות לאט מ- g(n).

מתקיים: $\log^* n = o(\log\log\log n) \cdots$ $\log^* n = o(\log\log n)$, גדל יותר לאט מכל הרכבה של כמות סופית של log.
$\log^{**} n = o(\log^* \log^* \log^* \log^* n) \cdots$, גם גדל לאט יותר כל הרכבה כזאת.

הפונקציה ההפוכה ל $A(n)$ מוגדרת $\alpha(n)$, והיא גדלה יותר לאט מכל ה- $\log^{**\cdots*} n$, לכל מספר סופי של כוכביות.

תכונות של דרגה:

1) מרגע שקודקוד מפסיק להיות שורש, הוא לא יחזור להיות שורש.
2) כל קודקוד מתחיל עם דרגה 0, ועולה רק כשהוא שורש.
3) אם v לא שורש, אז $\mathrm{rank}(v) < \mathrm{rank}(\pi(v))$

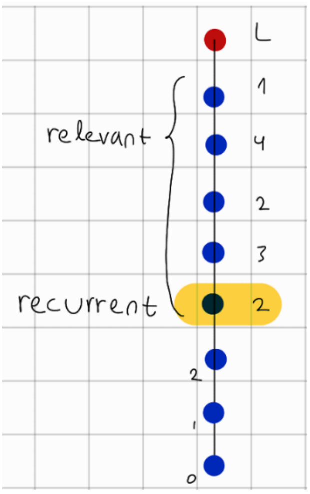
הגדרה: **קודקוד רלוונטי.** קודקוד שהוא לא שורש, עם דרגה $x \geq 3$. נגדיר $y \coloneqq \mathrm{rank}(\pi(v))$ מה הגודל של y, ביחס ל- x?
נגדיר שני פרמטרים:
פרמטר ראשי: מה ה- A_k הכי גדול שאפשר להפעיל על x, בלי לעבור את y?
$$L(v) \coloneqq \max\{k | A_k(x) \leq y\}$$
פרמטר משני: יהי $k = L(v)$. כמה פעמים אפשר להפעיל A_k על x, בלי לעבור את y?
$$l(v) \coloneqq \max\{m | A_k^m(x) \leq y\}$$
טענה: $l(v) < x$.
הוכחה: אם נפעיל את A_k x פעמים על x, נקבל לפחות $A_{k+1}(x)$.
כי אם $k = 0$ אז $A_0^x(x) = 2x$, ואם $k \geq 1$ אז $A_k^x(x) > A_k^x(1) = A_{k+1}(x)$.

טענה: אם יש n קודקודים, סכום כל הדרגות קטן מ- n.
הוכחה: כל הדרגות מתחילות ב0, כל union מגדיל דרגה ב1 לכל היותר. ואפשר לעשות לכל היותר n-1 פעמים union.

טענה: ה-L המקסימלי של קודקוד רלוונטי הוא פחות מ- $\alpha(n)$.
הוכחה: מתקיים $A_{\alpha(n)}(3) = A\big(\alpha(n)\big) > n$. L הוא הk הכי גדול שמקיים $A_k(x) \leq y$. אז אם נגדיר $L = \alpha(n)$, נעבור את y.

בסה"כ: $l(v) < x$, $L(v) \leq \alpha(n)$.

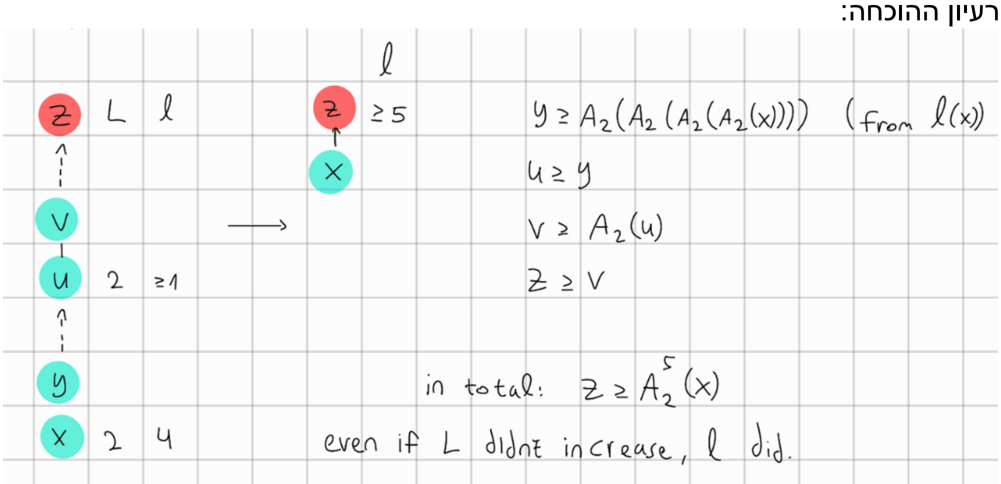
נגדיר S_i האורך של הדחיסת מסלול ה-i. אנחנו רוצים לחסום את $S_1 + \cdots + S_n$. נתבונן בדחיסת מסלול ה- i: יש לפחות $S_i - 3$ קודקודים רלוונטיים.



קודקוד רלוונטי נקרא **חוזר** אם יש עוד קודקוד רלוונטי מעליו במסלול עם אותו L.
כמה קודקודים יכולים להיות לא-חוזרים?
מתקיים $L \leq \alpha(n)$. יש לכל היותר $\alpha(n)$ קודקודים לא-חוזרים.
אז לפחות $(S_i - 3 - \alpha(n))$ הם חוזרים.

טענה: יהי v קודקוד חוזר. אחרי דחיסת מסלול, אחד מהשניים מתקיים:
1) $L(v)$ נשאר אותו דבר, ו- $l(v)$ גדל, או
2) $L(v)$ גדל.
(v "**קודק**")

רעיון ההוכחה:



נספור את מספר ה"קידומים", ב2 דרכים (ספירה כפולה כדי לקבל חסם הדוק):
מצד אחד, דחיסת המסלול ה- i גורמת ללפחות $(S_i - 3 - \alpha(n))$ קידומים. אז הסכום הוא לפחות $S - n(\alpha(n) + 3)$.
מצד שני, קודקוד רלוונטי v עם דרגה x יכול להיות מקודם לכל היותר $x \cdot \alpha(n)$ פעמים, כי $x < l(v) \le 1$ ו- $0 \le L(v) < \alpha(n)$.
וסכום כל הדרגות חסום ב n. אז סה"כ פחות מ- $n \cdot \alpha(n)$.

אז: $S - n(\alpha(n) + 3) < \text{total promotions} \le n\alpha(n) \Rightarrow S < 2n\alpha(n) + 3n = O(n\alpha(n))$
S הוא מספר הצעדים הכולל ב-n דחיסות מסלול. אז בניתוח לשיעורין, כל צעד הוא $O(\alpha(n))$,
שזה כמעט O(1) עבור כל n סופי.

סיכום: בגלל הדרכים שבהם הדרגה של קודקוד משתנה, סכום הדרגות חסום ב- n.
זה נותן לנו להגדיר יחס בין דרגה של קודקוד לדרגה של אבא שלו. היחס הזה מתואר על ידי נוסחה רקורסיבית.
בזמן דחיסת מסלול, היחס הזה משתנה. בגלל שהיחס קשור לדרגה, אפשר לחסום אותו עם n והנוסחה הרקורסיבית.
זה נותן לנו דרך לחסום את מספר הצעדים בכל דחיסות המסלול יחד, ויוצא שזה n כפול יחס הפוך לנוסחה הרקורסיבית.
הנוסחה הרקורסיבית היא מה שהגדרנו ע"י היררכיית אקרמן, ולכן הפונקציה ההפוכה היא $\alpha(n)$.

הסבר מפורט יותר במילים:
כשנאחד שורשים נכניס את השורש עם דרגה נמוכה מתחת לגדול, וזה גורם לכך שהדרך היחידה להגדיל דרגה היא לאחד שני קודקודים מאותה דרגה, ולכן בעץ עם דרגה K יש לפחות 2 בחזקת K קודקודים.

ובגלל שכדי להגדיל דרגה צריך שני קודקודים עם אותה דרגה, סכום הדרגות חסום בN.

ואם נעשה דחיסת מסלול, זה גורם לקודקודים להתקרב לשורשים.
נגיד שיש עץ שהוא רק ענף אחד באורך N.
אם נחפש קודם את האחרון – הזמן לזה יהיה N, והזמן לכל החיפושים אחרי יהיה 1.
אם נחפש את הראשון, ואז השני, וכו', הזמן לכל אחד יהיה 1.
אם נתחיל מהאמצעי, הזמן שלו יהיה חצי N, ואחרי זה לכל מה שהיה מעליו יהיה 1, ולמקסימלי ממה שהיה מתחתיו חצי N...
בסה"כ רואים שדחיסת מסלול גורמת לזמן חיפוש ממוצע להיות 1. אבל יכול להיות שהדחיסה עצמה לוקחת הרבה זמן.
אז רק צריך לחסום את מספר הצעדים שעושים בהרבה דחיסות מסלול ביחד, ולחסום את זה.
כדי לעשות את זה, נגדיר יחס בין הדרגה של קודקוד לדרגה של אבא שלו.
נראה שיש לו חסם עליון שמוגדר על ידי נוסחה רקורסיבית (כאן האקרמן נכנס – אקרמן היא פונקציה רקורסיבית).
בכל צעד בדחיסת מסלול, היחס הזה משתנה. (קוראים לזה "קידום" של הקודקוד).
בכל דחיסת מסלול יש חסם תחתון למספר הקידומים, שמגדיר לנו חסם תחתון למספר הצעדים בכל דחיסות המסלול.
אבל יש גם חסם עליון למספר הפעמים שקודקוד יכול להיות מקודם, בגלל החסם העליון ליחס של הדרגות.
החסם הזה מוגדר על ידי הפונקציה ההפוכה של אקרמן (כי זה ההפוך של מה שמגדיר את היחס של הדרגות).
אז לכל דחיסת מסלול צריך לפחות X קידומים, אבל כל קודקוד עם דרגה Y יכול להיות מקודם עד Y כפול אלפא פעמים.
וסכום כל הדרגות חסום בN. זה מגביל לנו את מספר הצעדים הכולל לבערך N כפול אלפא, עבור N קודקודים.
אז הזמן הממוצע לכל קודקוד הוא אלפא.