15 מבני הנתונים

Union - Find

n אלמנטים

A

make $O(1)$

union $O(\alpha(n))$ Amortized

find $O(\alpha(n))$

<u>שימושים</u>

ניצור ומה אלגוריתם קרוסקל

למשל, אם נגלה, אז

לפי ה- הנחה פרמטרים לשמור, צריך לפי

מינימלי מתוך את כולם.

בגרפים מציאת עץ פורש מינימלי

(עיין לבחינה)

נייצג קבוצה כעץ

~~שורש~~ שורש union ~~ייצוג~~ ייצוג find

Union by rank   Path compression

גובה פרמטרים, (הפירוק שמורים כמערך)

ולמערך של שמות, ששומר מי מצביע לאיזו.

union פשוט ע"י מיזוג שני עצים $O(1)$

find עולים הורה ב- W.C. $O(n)$

~~ודואגת הורה [illegible] על הורה עד הצומת~~ ~~$O(\log n)$~~

מחזיק עומק בכל צומת אז union יקר $O(\log)$

אבל find עלה $O(1)$

Union by Rank

לכל צומת יש דרגה. כל צומת "בודד" עם דרגה 0.

כאשר עושים union מחברים קטן דרגה לגדול

והגדולה לא משתנה. (לגדול וקטן מגדילים דרגה).

אם לשני השורשים של העצים יש אותה דרגה -

אז מקשרים את השורשים שרירותית ומגדילים את

הקטן אל הגדול (הקטן והגדול מגדילים דרגה)

ניתן להוכיח שלכל צומת שהוא שורש ה דרגה

שלו גבוהה מדרגת ילדיו.

ולכן גם הקטן מדרגת השורש קטן שלו.

ניתן להוכיח שעץ מדרגה r יש לו לפחות 2^r צמתים

הוכחה באינדוקציה על r

אם עץ מדרגה r+1 יש לפחות $2^r+2^r=2^{r+1}$ ✓.

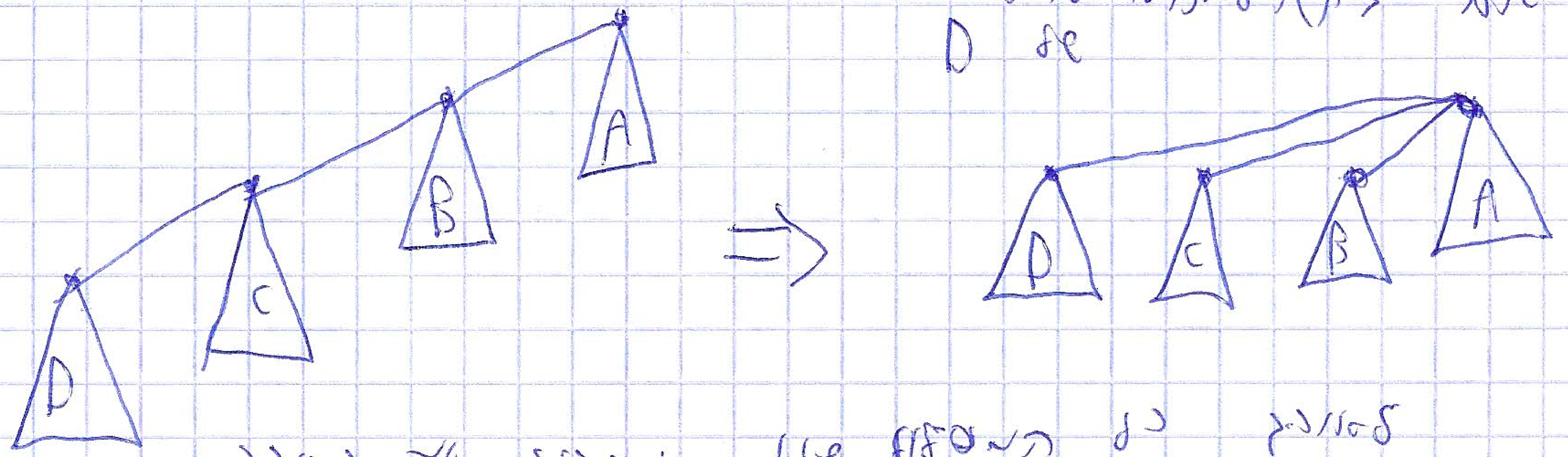
עומק עד הדרגה, עומק לוגריתמי

לכן זמן find הוא $O(\log n)$

נוסיף עוד טריק

Path compression

אחרי ביצוע find מחברים את כל הצמתים של D



לשורש כל הצמתים שבדרך במהלך ה-find.

זה משמר לאורך כל הדרך.

	make	union	find
W.C.	$O(1)$	$O(1)$	$O(\log n)$
Ammortized	$O(1)$	$O(\alpha(n))$	$O(\alpha(n))$

הפונקציה ההפכית של אקרמן

אלגוריתם לו מוכח ~~[illegible]~~ הפוך $\alpha(n)$

מוכח $\log^*(n)$ קטן כן מתנהג (כמעט) כמו קבוע

הגדרה - הרכבה של פונקציות

$$f^{(i)}(n) = \underbrace{f(f(f(\ldots f}_{i \text{ פעמים.}}(n)) \ldots))$$

$f(n) = n+1 \qquad f^{(5)}(n) = n+5$

$f(n) = 2n \qquad f^{(7)}(n) = 2^7 n$

$f(n) = 2^n \qquad f^{(3)}(n) = 2^{2^{2^n}}$

$f(n) = \log n \qquad f^{(2)}(n) = \log\log n$

פונקציית אקרמן Ackermann

$$A_k(n) = \begin{cases} n+1 & k=1 \\ A_{k-1}^{(n+1)}(n) & k>1 \end{cases}$$

$A_1(n) = n+1$

$A_2(n) = 2n+1$

$A_3(n) = 2^n(n+1)-1$

$A_4(n) = ?$

לא ברור איך
[illegible]

אחרון חביב

הפונקציה ↙ Ackermann ~ פונקציית

$$\bar{A}_k(n) = \begin{cases} 2n & k = 2 \\ \bar{A}_{k-1}^{(n)}(1) & k > 2 \end{cases}$$

((n) — הרכבה n פעמים; (1) ↓ על 1)

$\bar{A}_2(n) = 2n$

$\bar{A}_3(n) = 2^n$

$\bar{A}_4(n) = \text{tower}(n) = 2^{2^{2^{\cdot^{\cdot^{2}}}}}\Big\}\, n$

tower n	2	4	16	65536	2^{65536}
n	1	2	3	4	5

מספר ~~האטומים~~ החלקיקים ביקום

$10^{82} < 2^{400} <<<<<<< << 2^{65536}$

אם נקח ערך ההפכי של tower

נקבל פחות מ 5 -

הפונקציה ההפכית של tower

נקראת $\log^*(n)$

"כמה פעמים צריך לעשות log כדי להגיע מ n ל 2"

$\alpha(n)$ הפונקציה ההפכית של Ackerman הגדולה

שלבי-ביניים ~~...~~ ההפכית של $\bar{A}_k$ שלוקח

~~$\log^*(n)$~~ ההפכית

Tower(n) (שלה $\log^*(n)$

W.C. union $O(1)$

find $O(n)$

amortized $O(\log^*(n))$ בהמשך.

תזכורת.

$\log^*(r) = g \iff \exp^*(g-1) < r \leq \exp^*(g)$

$\exp^* \equiv$ tower בערך

<u>טענה</u> אם לא היינו עושים compressions

הדרגה של כל צ׳ היא הגובה שלו.

כשעושים compressions לא משנים את הדרגה (של שורש) אלא מקצרים את הגובה

אולם צמתים רבים, ~~[illegible]~~ שלהם יורד הגובה.

שימו לב, הדרגה שלהם כאילו לא היה compression

rank(x) = height of node x in <u>uncompressed</u> forest

דרגת x היא הגובה של x בגרף המקורי

ללא ביצוע compressions

<u>למה</u> בכל s פעולות (find + union)

ומס׳ המקסימלי של צמתים בדרגה r הוא לכל

היותר: $\frac{n}{2^r}$

<u>הוכחה</u>: לצומת בדרגה r יש

בגרף שלו (בעצמו) 2^r צמתים, ולצומת לא יכול להיות

לכן ~~[illegible]~~ תת׳ עצים מדרגה r שונים,

לכן לכל היותר $\frac{n}{2^r}$ צמתים כאלה.

→ אם יוצרים צמתים שיש להם ~~[illegible]~~ בדרגה r, אז באמצעות union - הדרגה מונוטונית <u>ממש</u> בדרגה

למה: compression לא פוגע על כך:

אם x הוא צומת בדרגה r, כאשר הבנים שלו

הולך לשרשרת, הדרגה שלו לא משתנה, כי הוא לא שורש.

אז הוא לעולם לא יחזור להיות שורש

וגם שלו, כי הדרגה לא כל מי שהולך

הצורה שלו הדרגה נשארת שלמה.

__הערה:__ לכל צומת שני מצביעים (הורה/דרגה)!

הצבע על הורה שלו - מצביע לדרגה

הערה: ערכי דרגות לא מתעדכנים, רק מצביעי הורה.

__הערה:__ אם x הוא צאצא ממש של y (בחיסכון שגם).

אז $rk(x) < rk(y)$ כלומר $rk(x)$ לא יורד

והדרגה של y לא יורדת.

זה compression? צומת שלה מסלול compression

[illegible] לא מוגדר

שלו, כיוון שהגדלה

משמעותית __ממש__ הגבלה

__טענה:__ כמות [illegible]

נגדיר $group(x) = \log^*(rank(x))$

group	rk
1	$rk \le 2$
2	$rk \le 4$
3	$rk \le 16$
4	$rk \le 65536$
5	$rk \le 2^{65536}$

אם נריץ $find(x_k)$ המסלול $x_1, x_2, x_3, .., x_k$

כך ש x_{i+1} ההורה של x_i. אם

$group(x_{i+1}) = group(x_i)$ אז x_i נקרא

נשלם את המעבר של ההורה $\{x_i, x_{i+1}\}$ על x_{i+1}

נשלם על ה[illegible] של x_k לשורש.

כל [illegible] [illegible] הקודמות

סה"כ $O(\log^* n)$

כמה פעמים אנחנו מעלים דרגת x

קורה שהוא עולה ? compression

בדרגות לכל משתנה.

בכל compression, x מקבל הורה חדש.

ולכן דרגת ההורה של x גדלה כי ההורה החדש גבוה

ולכן הדרך הנמוכה של ההורה של x.

כמה פעמים אוכל לשפר את x?

אוכל לשפר את x רק עד ההורה

[illegible] הקודקוד כמו [illegible] של

נניח ש- x הקודקוד [illegible] של

ויש [illegible] compressions. דרך x, אז

שמעבר את x, עד ה find [illegible].

בכל compression ←
יש הורים
שהם מקבלים הורה
[illegible] של הורה
הורה לכל משתנה.
כל [illegible]
[illegible] יש [illegible]
[illegible] כבר, [illegible]
כל [illegible] של
[illegible] לפי [illegible]
find [illegible]

$O(\log^* x)+1=O(\log^* x)$

נגדיר הקודקודים g ישנם כל [illegible]

$tower(g) - tower(g-1)$

נסמן מס' הנשמות קודקודים g: $N(g)$

$$N(g) \le \sum_{r=tower(g-1)+1}^{tower(g)} \frac{n}{2^r} \le \frac{n}{2^{tower(g-1)+1}}\left[1+\frac{1}{2}+\frac{1}{4}+\frac{1}{8}\dots\right] =$$

לכל דרגה מס' הנשמות [illegible]

$$\le \frac{n}{2^{tower(g-1)+1}}\cdot 2 = \frac{n}{2^{tower(g-1)}} = \frac{n}{tower(g)}$$

מספר המצבים [illegible] קודקודים g הוא

סך כל הדרגות של קבוצה g $\le N(g)\cdot(tower(g) - tower(g-1)) = \frac{n}{tower(g)}(tower(g) - tower(g-1)) = O(n)$

מס' הנשמות בקבוצה — דרגות בקבוצה

$n\log^* n$ סך הכל דרגות [illegible] $\log^* n$ יש

$O(\log^* n)$ Amortized לכן

הנחיות

מבנה הנחיות

פתוח סגור

אמריקאית / true false

[illegible] צריך להסביר את התשובה בקצרה

אלגוריתם התכונות פתרון בעיות