נושאים מתקדמים במבני נתונים – תרגיל 1

תרגיל 1

סעיף א

צ"ל: זמן החיפוש הממוצע באמצעות T הוא לפחות $H = \sum_{i=0}^{n} q_i \cdot \log_2 \frac{1}{q_i}$.

הוכחה:

נשתמש בהגדרות/משפטים שהוכחנו בכיתה:

1) אנטרופיה של התפלגות $p_1, \dots, p_n$ היא $\sum_{i=1}^{n} p_i \cdot \log \frac{1}{p_i}$ ולכל התפלגות אחרת $p'_1, \dots, p'_n$ מתקיים $\sum_{i=1}^{n} p_i \cdot \log_2 \frac{1}{p_i} \le \sum_{i=1}^{n} p_i \cdot \log_2 \frac{1}{p'_i}$

2) לכל עץ חיפוש בינארי T עם n צמתים ו-$n+1$ עלים כאשר הצמתים בעומקים b_i והעלים בעומקים a_i מתקיים: $\sum_{i=1}^{n} \frac{1}{3^{b_i+1}} + \sum_{i=0}^{n} \frac{1}{3^{a_i}} = 1$

$n+1$ ערכי ה-q_i כפי שהם מוגדרים כבר מגדירים לנו התפלגות. אם נוסיף להתפלגות זו עוד הסתברויות שערכן אפס זו עדיין תהיה התפלגות תקינה. על כן נוסיף עוד n ערכים $\{p_i = 0 | 1 \le i \le n\}$.

האנטרופיה של ההתפלגות החדשה היא: $\sum_{i=0}^{n} q_i \cdot \log_2 \frac{1}{q_i} + \sum_{i=1}^{n} p_i \cdot \log_2 \frac{1}{p_i}$.

כעת נגדיר התפלגות חדשה (בהתבסס על a_i, b_i מההגדרה לעיל) כך ש- $p'_i = \frac{1}{3^{b_i+1}}$ ו- $q'_i = \frac{1}{3^{a_i}}$ (זו התפלגות לפי משפט 2). לפי משפט 1 נוכל הגיד את הדבר הבא:

$$\sum_{i=0}^{n} q_i \cdot \log_2 \frac{1}{q_i} + \sum_{i=1}^{n} p_i \cdot \log_2 \frac{1}{p_i} \le \sum_{i=0}^{n} q_i \cdot \log_2 \frac{1}{q'_i} + \sum_{i=1}^{n} p_i \cdot \log_2 \frac{1}{p'_i}$$

מאחר והגדרנו את $p_i = 0$ נקבל:

$$H = \sum_{i=0}^{n} q_i \cdot \log_2 \frac{1}{q_i} \le \sum_{i=0}^{n} q_i \cdot \log_2 \frac{1}{q'_i} = \sum_{i=0}^{n} q_i \cdot \log_2 3^{a_i} = \log_2 3 \sum_{i=0}^{n} q_i a_i$$

זמן החיפוש הממוצע ב-T הוא $\sum_{i=0}^{n} q_i a_i$ וקיבלנו ש:

$$\frac{H}{\log_2 3} \le \sum_{i=0}^{n} q_i a_i$$

שזה מה שרצינו.

סעיף ב

צ"ל: זמן החיפוש הממוצע הוא $O\big(m(1+\mathrm{H})\big)$

הוכחה:

בשביל ההוכחה נשתמש בהגדרות של משקל (weight), גודל (size - סכום כל המשקלים של הבנים + הצומת), הדרגה (rank – הלוג של ה-size) וב-Access Lemma כמו שהוגדרו בכיתה ופרט:

$$\mathrm{Amort}(Splay-of-x-unter-root-t) \le 3\big(r(t)-r(x)\big)+1 = 3\log\left(\frac{s(t)}{s(x)}\right)+1$$

השינוי בשאלה הוא שה-splay מתבצע על האבא (נקרא לו p) של העלה (x) בעוד שמחפשים את העלה אבל זה לא משפיע במידה רבה על ההוכחה של ה-Access Lemma.

אם נשתכנע ש:

$$\mathrm{Amort}(zig-zag) = \mathrm{Amort}(zig-zig) = 2 + \Delta\Phi \le 3\big(r'(p)-r(p)\big)+1$$
$$\mathrm{Amort}(zig) = 1 + \Delta\Phi \le 3\big(r'(p)-r(p)\big)$$

אז באופן מאוד דומה לאיך שהוכחנו בכיתה, אם נסכום את עלות כל צעדי הזיג-זג/זיג-זיג וזיג אחד אחרון (אם צריך):

$$\begin{aligned}\mathrm{Amort}(Splay) &\le 3\big(r_1(p)-r_0(p)\big)+\cdots+3\big(r_k(p)-r_{k-1}(p)\big)+3\big(r_k(p)-r_{k-1}(p)\big)+1\\ &= 3\big(r_{k+1}(p)-r_0(p)\big)+1 =^{1} 3\big(r(t)-r(p)\big)+1 \le^{2} 3\big(r(t)-r(x)\big)+1\end{aligned}$$

1: אחרי הצעד ה- $k+1$ הצומת p הוא כבר השורש ולכן הדרגה שלו שווה לדרגה של השורש המקורי

2: מאחר ו-p הוא האבא של x אז הדרגה שלו גדולה יותר (ויש חיסור)

ההוכחה של הסיבוכיות של הזיג-זג/זיג-זיג והזיג נשארת בדיוק כפי שהוכחנו בכיתה.

נשים לב שהשינוי שה-*splay* מתבצע על האבא אבל מחפשים את הבן מוסיף מספר קבוע של פעולות בכל חיפוש כך שהזמן הדומיננטי הוא של ה-*splay* כמו מקודם.

כעת כל מה שנותר הוא להגדיר את פונקציית המשקל החדשה כך שתתן משקל לעלים (הלא קיימים).

בהנתן שמחפשים כל עלה $q(i)$ פעמים ויש סה"כ m חיפושים אז המשקל $w_i = \frac{q(i)}{m} = q_i$ ואם נצא מנקודת הנחה שמחפשים את כל העלים (אותה הנחה שעשינו בכיתה) אז הגודל של השורש הוא 1.

כעת נוכיח את ה-*Static Optimality Theorem* (מה שהתבקשנו להוכיח מלכתחילה).

$$\sum_{i=1}^{m} \mathrm{T}(Lookup_i) = \sum_{i=1}^{m} \mathrm{Amort}(O_i) + \phi_0 - \phi_m$$

נגדיר שבכל פעולה j (מתוך m) מחפשים את x_j עם הסתברות q_{x_j} ושורש העץ הוא t_j.

$$\begin{aligned}\sum_{j=1}^{m}\mathrm{Amort}(O_i) &\le \sum_{j=1}^{m}\left(3\log\left(\frac{s(t_j)}{s(x_j)}\right)+1\right) = m + 3\sum_{j=1}^{m}\log\left(\frac{1}{q_{x_j}}\right)\\ &= m + 3\sum_{j=1}^{m}\log\left(\frac{1}{q_{x_j}}\right) =^{1} m + 3\sum_{i=1}^{n} q(i)\log\left(\frac{1}{q_i}\right) =^{2} m\left(1+3\sum_{i=1}^{n} q_i\log\left(\frac{1}{q_i}\right)\right)\\ &= m(1+3H) = O\big(m(1+H)\big)\end{aligned}$$

1: נספור לכל עלה כמה פעמים נגשו אליו ולא לפי כמה פעולות היו בסה"כ

2: ההסתברות בפועל של גישה לכל עלה

$$\phi_0-\phi_m=\sum_{i=1}^{n}\log\big(s_0(x_i)\big)-\sum_{i=1}^{n}\log\big(s_m(x_i)\big)\le\sum_{i=1}^{n}\left(\log(1)-\log\left(\frac{q_i}{m}\right)\right)=\sum_{i=1}^{n}\log\left(\frac{m}{q_i}\right)$$
$$\le\sum_{i=1}^{n}\log m=n\cdot\log m=O(m)$$

זאת בהנחה שיש (משמעותית) יותר חיפושים מאשר עלים בעץ $m \gg n$.

ולכן הזמן הכולל הוא:

$$\sum_{i=1}^{m}\mathrm{T}(Lookup_i)=\sum_{i=1}^{m}\mathrm{Amort}(O_i)+\phi_0-\phi_m\le O\big(m(1+H)\big)+O(m)=O\big(m(1+H)\big)$$

תרגיל 2

צ"ל: פעולת הכנסה לעץ Splay לוקחת $O(\log n)$

הוכחה:

נשתמש באותה הגדרת פוטנציאל כמו בכיתה (סכום ה-ranks) עם פונקציית משקל שנותנת 1 לכל צומת.
נשים לב שההכנסה משנה את הפוטנציאל פעמיים, פעם ראשונה של כל הצמתים במסלול מ-v (הצומת החדש) לשורש בעת ההכנסה של v (מ-Φ_1 ל-Φ_2) ופעם שנייה ע"י פעולת ה-splay (מ-Φ_2 ל-Φ_3).

לשם ההוכחה נשתמש ב-Access Lemma המשודרגת שהוכחה בכיתה שמתחשבת באורך המסלול מהשורש לצומת עליו מבצעים Splay. נזכיר את ניסוחה:

- לכל קבוע $c>1$ הזמן ה-amortized לבצע Splay על צומת v בעץ עם שורש t כאשר ℓ הוא אורך המסלול ה-Splay הוא לכל היותר:

$$3c\big(r(t)-r(v)\big)+1-(\ell-1)(c-1)=3c\cdot\log\left(\frac{s(t)}{s(v)}\right)+1-(\ell-1)(c-1)$$

אם היינו עושים Splay לאבא של v (נקרא לו p) ללא ההוספה שלו, עם פונקציית המשקל הנ"ל היינו מקבלים שהעלות היא $3c\cdot\log(n)+1-(\ell-1)(c-1)$.
הבה נראה מה השתנה ע"י זה שהוספנו את v ועשינו Splay עליו.

- הפוטנציאל שהשתנה הוא 1 עבור עבור צומת בדרך מ-v לשורש, שזה לכל היותר ℓ
- עשינו פעולת Rotate נוספת תוך כדי ה-Splay של v (כי הוא רמה אחת מתחת ל-p), סה"כ $O(1)$.

מכאן ש-

$$\begin{aligned}Cost(Insert-v)&=Cost(Splay-p)+\ell+\ O(1)\\&=3c\cdot\log(n)+1-(\ell-1)(c-1)+\ \ell+\ O(1)=O(\log n)\end{aligned}$$

תרגיל 3

צ"ל: להוסיף תמיכה ב- Evert ב- $O(\log n)$ מבלי לפגוע בזמן ריצת שאר הפעולות

הוכחה:

הפתרון הוא להוסיף לכל צומת ביט "reverse". סכום (מודולו 2) של הביטים האלו מהצומת עד השורש של ה-solid tree בו הוא נמצא אומר האם הבן הוא באמת ימני או שמאלי. לדוגמא אם הצומת הוא בן ימני והסכום הוא 1 אז הוא למעשה בן שמאלי. נקרא לביט זה b.

לצורך המימוש של כלל הפעולות נזדקק לשלושה פרימיטיבים:

- Unreverse(v): המטרה בפעולה זו היא לסדר את סדר הבנים ב-v. אם b(v)=1 אז מאפסים אותו, מחליפים הין הבן הימני לשמאלי והופכים את ביטי ה-b של הבנים.
- Rotate(v): פעולה זו היא חלק מפעולת ה-splay. נניח ש-u הוא האבא של v. הפעולה נעשת כמו מקודם רק שצריך לשים לב לשאם ביט ה-reverse של v דלוק אז צריך קודם לעשות unreverse ורק אז לעשות Rotate כרגיל.
- Splice(v,u): גם כאן צריך לשים לב לביט ה-reverse כדי לדעת האם להחליף את v עם הבן השמאלי או הימני של u. מאחר ו-u הוא שורש אז יש גישה לביט ה-reverse שלו ואפשר בלי הליכה על העץ לדעת אם הבנים שלו הפוכים או לא.

פעולת ה-Splay תעבוד כמו מקודם רק עם ה-Rotate החדש שהסיבוכיות שלו לא השתנתה (התווסף מספר קבוע של פעולות). גם המעבר בין ה-solid sub trees יעשה ע"י ה-Splice החדש ובסוף יעשה Splay חדש (על ה-Solid sub tree האחרון)

ופעולת ה-Evert, אשר עבורה הכנסנו את ביט ה-Reverse תעבוד בצורה הבאה:

- מבצעים Splay ב-v כדי להביא אותו לשורש
- הופכים את הבן הימני של v, נקרא לו u, לבן השמאלי אם אין בן שמאלי או לבן מקווקוו של הבן השמאלי (אם יש כזה)
- הופכים את הביט b(u)

בצורה כזאת v הופך להיות שורש של עץ ה-solid sub tree הראשי ואין לו בנים ימניים כך שהוא גם השורש של העץ האמיתי שהעץ הוירטואלי מייצג. ביט ה-Reverse נתן לנו ב-O(1) לשנות את כיוון כל מי שהיה אחרי v בעץ. זמן ריצה: $O(\log n)$

זמן ריצת שאר הפעולות לא ניזוק, כל מה שהן צריכות לעשות זה בזמן שעוברים על העץ להסתכל על ביט ה-Reverse כדי לדעת עם הבן הימני הוא באמת הבן הימני וכנ"ל לגבי השמאלי.

תרגיל 4

צ"ל: להוסיף תמיכה במשקולות בכל צומת ובפעולת ה-Weighted Sum ב- $O(\log n)$ מבלי לפגוע בזמן ריצת שאר הפעולות

הוכחה:

כדי לתמוך בפעולה נצטרך להוסיף מספר שדות לכל צומת:

- שדה משקל – Weight: לא משתנה ונקבע בעת הוספת הצומת לעץ
- שדה סכום משקלים – WeightSum: סכום המשקלים בתת העץ כולל הצומת עצמו
- שדה הפרש עלות – $\Delta AddCost$: ישמור את כל ה-Cost-ים שהתווספו לצומת זה בפעולות AddCost.
- שדה סכום משקולות WeightedSum:
  - אם v הוא שורש ה-solid sub tree אז השדה יחזיק את הסכום $\sum_{e\in P} c(e)\cdot w(e)$ כאשר P הוא הנתיב מ-v לשורש.
  - אחרת הוא יחזיק את ההפרש בין הסכום המשוקלל של האבא לסכום שלו.

כדי לנהל את השדות הנ"ל בצורה נכונה צריך לעדכן את הפעולות הבאות:

- סיבוב:
  - אם מעבירים תת עץ מבן ימני v לאבא שלו p (או להפך) צריך לשים לב אם שדה ה- $\Delta AddCost$ גדול מאפס, אם כן יש לבצע את העדכונים הבאים:
    - $WeighedSum'(v) = WeightSum(v) + WeightSum(v)\cdot \Delta AddCost(v)$ זאת כדי להחיל את תוספת ה-Cost על הסכום המשוקלל
    - אם ל-v יש בנים: $\Delta AddCost'(v) = \Delta AddCost'\big(p(v)\big)$, אחרת $\Delta AddCost'(v) = 0$
    - $\Delta AddCost'\big(p(v)\big) = 0$
  - כמו כאשר מבצעים סיבובים צריך גם כן לעדכן את ה-WeightedSum בהתאם לבנים שהשתנו. למשל אם ל-v היה בן u שעבר לאבא שלו אז צריך להפחית מה-WeightedSum שלו את המשקל המשוקלל שלו.
- AddCost:
  - נבצע Splay ונעדכן את שדה ה-$\Delta Cost$ כמו מקודם
  - נוסיף לשדה ה-$\Delta AddCost$ את ה-Cost שהתווסף (נקרא לו ΔC)
  - נעדכן את ה-WeightedSum של הצומת עליו התבצע ה-Splay ב- $WeightSum(v)\cdot \Delta C$
- חיבור שני עצים:
  - אם v מתווסף כבן ימני של u אז:
    - $WeightedSum'(u) = WeightedSum(u) + WeightedSum(v)$
    - $WeightedSum(v) = WeightedSum'(u) - WeightedSum(v)$
    -

את פעולת ה-$WeightedSum(v)$ נבצע ע"י:

- ביצוע Splay על v
- החזרת שדה ה-WeightedSum שלו. נשיב לב ששדה זה כבר מעודכן תודות לעדכנים שנעשו תוך כדי הסיבובים בפעולות ה-Splay.

זמן ריצה: $O(\log n)$

כמו כן נשים לב שהוספנו בחלק מהפעולות מספר קבוע של פעולות נוספות, דבר אשר אינו פוגע בזמן הריצה של שאר הפעולות.