נושאים מתקדמים במבנ"ת ואלגוריתמים
תרגיל 1

שאלה 1

א.

T עץ בינארי כלשהו על המפתחות $B_1, \dots, B_n$, ומגדירים $B_0 = -\infty, B_{n+1} = \infty$. חיפושים בעץ הם תמיד עבור מפתחות $B \neq B_i$, ונסמן בq_i את ההסתברות לחיפוש איבר בתוך האינטרוול (B_i, B_{i+1}) $(0 \leq i \leq n)$. תוחלת זמן החיפוש באמצעות העץ T היא $\Sigma = \sum_{i=0}^{n} q_i a_i$, כאשר a_i עומק העלה A_i, שמייצג את האינטרוול (B_i, B_{i+1}) בעץ. נראה להראות ש $\Sigma \geq H$, כאשר $H = \sum_{i=0}^{n} q_i \log_2 \frac{1}{q_i}$.

<u>למה:</u> יהי עץ T על המפתחות הנ"ל, וa_i עומק העלה שמייצג את האינטרוול (B_i, B_{i+1}) בעץ. אז

$$\sum_{i=0}^{n} \frac{1}{2^{a_i}} = 1$$

<u>הוכחה:</u> באינדוקציה על n.

$n = 1$: העלה היחיד בעץ הוא השורש B_1, ויש לו שני עלים $A_0 = (B_{-\infty}, B_1),\ A_1 = (B_1, B_{\infty})$.
עומק כ"א מהעלים הללו הוא 1, ומקבלים: $\sum_{i=0}^{n} \frac{1}{2^{a_i}} = \frac{1}{2} + \frac{1}{2} = 1$

נניח נכונות עבור $n-1$, ונראה עבור n. נניח T עץ עם n צמתים פנימיים. נבחר צומת פנימי O ששני ילדיו הם עלים, נסמן אותם בA_j, A_{j+1}, ונניח עומק O הוא o.
נחליף את O בעלה בעומק o. נקבל עץ T' עם $n-1$ צמתים, ונפעיל עליו את הנחת האינדוקציה.
נסמן X = סכום $\frac{1}{2^{a_i}}$ עבור כל העלים בT פרט לA_j, A_{j+1}. לפי הנחת האינדוקציה בעץ T', מקבלים $X + \frac{1}{2^o} = 1$.

חוזרים לעץ T. עומק העלים A_j, A_{j+1} הוא $o+1$, ולכן $\frac{1}{2^o} = \frac{1}{2^{o+1}} \cdot 2 = \frac{1}{2^{a_j}} + \frac{1}{2^{a_{j+1}}}$, ולכן $\sum_{i=0}^{n} \frac{1}{2^{a_i}} = X + \frac{1}{2^{a_j}} + \frac{1}{2^{a_{j+1}}} = X + \frac{1}{2^o} = 1$ כרצוי.

<u>מסקנה:</u> נגדיר $q_i' = \frac{1}{2^{a_i}}, i = 0, \dots, n$. אז $\sum_{i=0}^{n} q_i = 1$, ולכן $\{q_i'\}$ מגדירים התפלגות.

$$H = \sum_{i=1}^{n} q_i \log_2 \frac{1}{q_i} \underset{(*)}{\leq} \sum_{i=1}^{n} q_i \log_2 \frac{1}{q_i'} = \sum_{i=1}^{n} q_i \log_2 2^{a_i} = \sum_{i=1}^{n} q_i a_i = \Sigma$$

$(*)$ נובע ממינימליות של האנטרופיה, כפי שראינו בשיעור.
ולכן $\Sigma \geq H$.

ב.

חיפוש עבור איבר $B \in (B_i, B_{i+1})$ מתבצע ע"י חיפוש העלה A_i בעץ, ואז ביצוע splay בצומת ההורה של A_i. נראה שתוחלת הזמן של m חיפושים כאלה היא $O\big(m(1+H)\big)$.

<u>התאמת Access Lemma</u>

נניח $w(A)$ פונקציית משקל על <u>העלים</u> בעץ.
$s(x)$ = סכום המשקלות $w(A)$ בעלי תת העץ ששורשו בx.
$r(x) = \log_2 s(x)$.
פונקציית הפוטנציאל: $\Phi(T) = \sum r(x)$ סכום הrankים בעץ.

מחפשים איבר בעלה a, ומבצעים $splay(x)$ על צומת x שהוא ההורה הישיר של a בעץ, ברור שעלות החיפוש פרופורציונלית לעלות $splay$.

בשיעור ראינו שAccess Lemma נכונה לכל פונקציית משקל על צמתי העץ. במקרה שלנו פונקציית המשקל מוגדרת על העלים במקום על הצמתים, אבל ההוכחה תקפה באותה מידה (אפשר להתייחס לעלים בתור צמתים פנימיים לצורך ההוכחה). מקבלים לפי הAccess Lemma:

$$amortized\big(splay(x)\big) \le 3\big(r(t) - r(x)\big) + 1 = 3\log\frac{s(t)}{s(x)} + 1$$

<u>התאמת הStatic Optimality Theorem</u>

לאינטרוול $A_i = (B_i, B_{i+1}), i = 0, \dots, n$, נגדיר Q_i = מספר הגישות לאינטרוול זה (מתוך m הגישות), $q_i = \frac{Q_i}{m}$. נגדיר פונקציית משקל $w(A_i) = \frac{Q_i}{m} = q_i$ על עלי העץ.

t שורש העץ, אז $s(t) = \sum q_i = 1$.

לכל $x = A_i$ עלה בעץ, $s(x) = s(A_i) = q_i$. במקרה שלנו כל הsplay-ים מתבצעים על עלים בעץ (ולא על צמתים פנימיים), ולכן $s(x)$ לא רלוונטי עבור צמתים פנימיים.

נציב בAccess Lemma:

$$amortized\big(splay(x)\big) \le 3\log\frac{1}{q_i} + 1$$

נסמן o_j = פעולת הsplay (לצורך החיפוש) הj $(1 \le j \le m)$. אז:

$$\sum_{j=1}^{m} amortized\big(o_j\big) = \sum_{i=0}^{n} Q_i \cdot amortized\big(splay(A_i)\big)$$
$$= \sum_{i=0}^{n} Q_i \cdot \left(3\log\frac{1}{q_i} + 1\right)$$

Q_i משתנה מקרי המתאר את מספר האיברים באינטרוול $A_i = (B_i, B_{i+1})$ מתוך m משיכות, כאשר ההסתברות למשוך איבר באינטרוול A_i היא q_i. אז תוחלת Q_i היא $E(Q_i) = q_i m$. לכן:

$$E\left(\sum_{j=1}^{m} amortized\big(o_j\big)\right) = E\left(\sum_{i=0}^{n} Q_i \cdot \left(3\log\frac{1}{q_i} + 1\right)\right)$$
$$= \sum_{i=0}^{n} E(Q_i) \cdot \left(3\log\frac{1}{q_i} + 1\right)$$
$$= \sum_{i=0}^{n} mq_i \cdot \left(3\log\frac{1}{q_i} + 1\right)$$
$$= m\sum_{i=0}^{n} q_i \cdot \left(3\log\frac{1}{q_i} + 1\right)$$
$$= m\left(3\underbrace{\sum_{i=0}^{n} q_i \log\frac{1}{q_i}}_{H} + \underbrace{\sum_{i=0}^{n} q_i}_{1}\right)$$
$$= O\big(m(1+H)\big)$$

$$\sum time(o_j) = \sum amortized(o_j) + \Phi_0 - \Phi_m$$

נחסום את $\Phi_0 - \Phi_m$:

$s(x) \le 1$, ולכן החסם הבא ברור (בעלים):

$$0 \ge r(x) = \log s(x) \ge \log w(x)$$
$$0 \le -r(x) \le \log\frac{1}{w(x)} = \log\frac{m}{Q_i}$$

אז הפוטנציאל של עלה ספציפי יכול להשתנות בלכל היותר $\log\frac{m}{Q_i}$. בסה"כ כל העלים יכולים להשתנות בלכל היותר $\sum_i \log\frac{m}{Q_i}$, וזה זניח אם מניחים $q_i > 0$.

שאלה 2

מבצעים הכנסה של איבר $B \notin T$ לT Splay tree באופן הבא:

- מחפשים את B בעץ, החיפוש מגיע לעלה A.
- מחליפים את A בצומת חדש v המכיל את B.
- מבצעים splay בv.

<u>טענה:</u> העלות האmortized של פעולת הכנסה הממומשת באופן הנ"ל היא $O(\log n)$.

<u>הוכחה:</u>

ראשית, ניזכר כיצד מתבצעת הכנסה בצורה שראינו בכיתה. נניח שמכניסים איבר $B \notin T$ לעץ, ונניח לשם הנוחות שB קטן מכל האיברים בT (או גדול מכולם; אחרת מבצעים *split* ו*concat* לצורך ההכנסה). אז ההכנסה מתבצעת באופן הבא:

- מבצעים splay באיבר הקטן ביותר של T, נסמן אותו בB'
- מוסיפים את B בתוך הבן השמאלי של B'
- מבצעים $splay$ בB.

שתי הפעולות האחרונות מתבצעות בזמן $O(1)$. הפעולה הראשונה מתבצעת בעלות splay, שהיא amortized $O(\log n)$.

הבעיה בצורת ההכנסה החדשה היא בכל ש$splay$ מבוצע בפועל על עץ בעומק גדול ב1 מהעץ המקורי, ולכן "קשה" לאמוד את התוספת לפוטנציאל.

לצורך הניתוח, ניזכר בגרסא המשופרת של הAccess Lemma: לכל קבוע $c \geq 1$, העלות האmortized של ביצוע splay לצומת x בעץ עם שורש t חסומה ע"י:

$$3c\big(r(t) - r(x)\big) + 1 - (l-1)(c-1) = 3c\log\frac{s(t)}{s(x)} + 1 - (l-1)(c-1)$$

כאשר l אורך מסלול הsplay.

אז ההבדל בין שתי הפעולות נעוץ באורך המסלול עליו מבצעים *splay* – במימוש המקורי, מבצעים *splay* על הצומת הקרוב ביותר לB בעץ, נסמן אותו בB', ובמקרה הנוכחי מבצעים *splay* על B לאחר שמשרשרים אותו כבן של B'. פירוש הדבר שאורך מסלול ה*splay* גדל ב1. בנוסף, $s(x)$ זהה בשני המקרים. לכן, ההפרש בין העלות ה*amortized* בשני המקרים נעוץ בהפרש הבא:

$$(l-1)(c-1) - \big((l+1)-1\big)(c-1) = (c-1)$$

במילים אחרות, מדובר בהפרש קבוע ולכן העלות האmortized גם כאן היא $O(\log n)$.

שאלה 3

ראשית, נציג פעולת reverse על עצים בכלל, ועל splay trees בפרט. פעולת $reverse(x)$ (צומת x בעץ T), לוקחת את כל תת העץ ששורשו בx, והופכת את סדר העלים בו. אפשר להסתכל על הפעולה כהיפוך יחס הסדר בתוך תת העץ הזה. מימוש פשוט של reverse על כל עץ ניתן לבצע בזמן לינארי במספר הצמתים בעץ: עוברים על כל צומת בעץ, והופכים בין המצביעים שלו לבן הימני והשמאלי.

לצורך מימוש יעיל יותר, נוסיף מידע לעץ. בכל צומת x נחזיק ביט יחיד בשם $reverse(x)$, ועוד שני מצביעים ל$major(x), minor(x)$. נשמור על הכלל הבא:

- $major(x) = right(x), minor(x) = left(x)$ אם במסלול $x \rightsquigarrow t$ (t שורש העץ), יש בדיוק מספר זוגי של אבות y כך ש$reverse(y) = 1$ (כולל את x עצמו).
- אחרת – $major(x) = left(x), minor(x) = right(x)$.

בכל מקרה, נשמר את האינוריאנטה שטיול ע"פ הסדר בעץ ששורשו בx, מתבצע בתהליך רקורסיבי של $traverse(minor(x)), x, traverse(major(x))$. יחס הסדר הנ"ל משמש אותנו בחיפושים של העץ, עדכונים וכו'. גם סיבובים ופעולות splay (zig-zig vs zig-zag) תמיד יתבצעו בהשוואה לminor, major ולא בהשוואה לleft, right.

כעת, אפשר לממש פעולת $reverse(x)$ על כל צומת בעץ בזמן קבוע $O(1)$ – פשוט ע"י הפיכת הreverse bit בצומת.

צריך לשמר את המידע החדש בשינויים על העץ. מספיק לתאר איך הוא נשמר בפעולות סיבוב. אז נניח x בן של y בעץ, וכעת מבצעים סיבוב סביב y. אז הערכים החדשים יהיו:

$$reverse'(y) = reverse(x)$$
$$reverse'(x) = reverse(x) \oplus reverse(y)$$

($\oplus$ - פעולת exclusive or על ביטים).

קל להשתכנע שעבור כל ערכים מקוריים של $reverse(x), reverse(y)$ וזוגיות של reverse במסלול עד y, הערכים החדשים שומרים על אותו סדר.

בנוסף צריך לשמר את המידע בconcat וsplit של עצים – גם כאן הסיפור קל למדי ומוסיף עלות קבועה לפעולה.

נחזור לפעולת evert: נתחיל במקרה הפשוט של שרוכים. נניח שT הוא שרוך p, v זנב השרוך וt השורש שלו, V העץ המייצג את T. פעולת evert הופכת את כל הקשתות על השרוך, ולכן היא שקולה לפעולת reverse על העץ V. אז במקרה הזה, המימוש מתבצע באופן ישיר בזמן $O(1)$.

במקרה הכללי, T הוא עץ דינאמי, $v \in T$, t שורש העץ, V הייצוג הוירטואלי של העץ, ורוצים להפוך את המסלול בין v לt. נתחיל בביצוע $splay$ לv. בסיום הsplay שני הצמתים v, t נמצאים על אותו solid splay tree, נסמן אותו בT'. v בשורש V וt הוא שורש T, ולכן אין עוד צמתים בT'. נבצע inverse על T', שיוביל להיפוך הקשתות במסלול. על מנת לשמור על עלות $amortized$ של הפעולה, נבצע לבסוף $splay$ נוסף על השורש t.

פסאודו קוד:

```
evert(v):
    splay(v)
    p = path(v), T' = T(v)
    inverse(T')
    splay(t)
```

סיבוכיות: בסה"כ האלגוריתם מבצע מספר קבוע של פעולות על העץ T, כאשר את כולן כבר ראינו וניתחנו, פרט לinverse. כל הפעולות מתבצעות בזמן $O(\log n)$, וinverse מתבצעת בזמן $O(1)$. הסיבוכיות הכוללת $O(\log n)$, וכפי שראינו קודם אין פגיעה בסיבוכיות פעולות אחרות.

שאלה 4

לכל קשת e נתון משקל $w(e)$ שנקבל בעת הכנסת הקשת לעץ הדינאמי ולא משתנה. רוצים לתמוך בפעולה $sum_w(v) = \sum_{e \in P} w(e)c(e)$, כאשר P המסלול מv אל שורש העץ הדינאמי T.

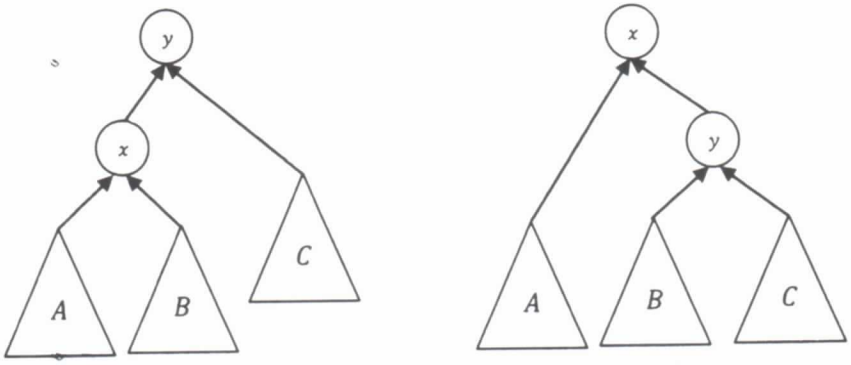
עבור $e = (v,u)$, נגדיר $w(v) = w(e)$, כלומר המשקל יישמר בצומת ההתחלה. בשורש המשקל 0.
נוסיף לכל קודקוד את המשתנים הבאים:

$\Delta sum(v) = sum_w(p(v)) - sum_w(v)$
$W(v) = \sum_{u \in T(v)} w(v)$

בשורש, $\Delta sum(v) = sum_w(v)$.

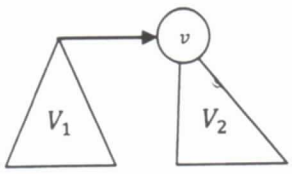
אז כדי לחשב $sum_w(v)$, נבצע *splay* בv כדי להביא אותו אל השורש, ואז נחזיר את $\Delta sum(v) = sum(v)$.
נראה שניתן לתחזק את המידע החדש בעלות קבועה לפעולה. בהתאם, פעולת חישוב הסכום פרופורציונלית לפעולת *splay*, ולכן היא מתבצעת בעלות $O(\log n)$ *amortized*. לכל שאר הפעולות נוסף פאקטור קבוע ולכן הסיבוכיות לא משתנה.

<u>סיבובים:</u> נניח x בן של y בעץ, ומסובבים סביב y:



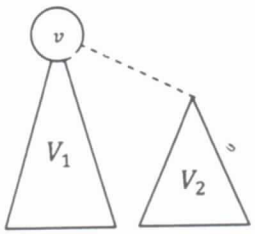
$\Delta sum'(x) = \Delta sum(y)$
$\Delta sum'(y) = \Delta sum(y) + \Delta sum(x)$
Update A, B, C appropriately.

<u>$V = concat(V_1, V_2)$:</u> נניח מחברים את שורש V_1 כבן השמאלי של הצומת השמאלי ביותר בV_2, ונסמן את הבן השמאלי ביותר הזה v. מבצעים splay לv ומביאים אותו לשורש. מחברים אליו את V_2, ואז מעדכנים:



$W'(v) = W(V_1) + W(V_2)$
$\Delta sum'(v) = sum(V_2) + sum(V_1) = \Delta sum(v) + \Delta sum(V_1)$
$\Delta sum'(V_1) = sum(V_2) = \Delta sum(V_2)$

<u>$V_1, V_2 = split(v, V)$:</u> עושים *splay* בv, וחותכים את אחד הבנים שלו, נניח הבן הימני.



$W'(V_1) = W(left(v)), W'(V_2) = W(right(v))$
$\Delta sum'(V_2) = sum(V_2) = sum(v) - \Delta sum(left(v))$
$\Delta sum'(v) = sum(V_1) = sum(v) - \Delta sum(right(v))$

<u>*delete, insert*</u> נגזרים ישירות מ$insert, split$.

<u>$addcost(c)$:</u> שומרים את העלויות בתור $\Delta cost$ כמו שראינו עד עכשיו. אין השפעה על $W(v)$ לכל הצמתים, ולכן יש להסביר רק איך לעדכן את Δsum. נניח v שורש העץ. אז:

$$\Delta sum'(v) = sum'(v) = sum(v) + c \cdot \Big(w(v) + W\big(right(v)\big)\Big)$$
$$\Delta sum'\big(left(v)\big) = \Delta sum\big(left(v)\big) + c \cdot \Big(w(v) + W\big(right(v)\big)\Big)$$
$$\Delta sum'\big(right(v)\big) = \Delta sum\big(right(v)\big) - c \cdot w(v)$$

mincost: ללא שינוי.