אוניברסיטת ת"א
סמסטר א', תשפ"ה
הגשה עד ה- 15.12 ב-moodle

תרגיל בית 2 במבני נתונים

על כל התשובות להיות מנומקות. בכל שאלה יש לבחור במימוש היעיל ביותר האפשרי מבחינת סיבוכיות זמן. יש לענות על השאלות במקומות המוגדרים לכך.

<u>שאלה 1</u>
בהרצאה ראינו מימוש של מחסנית על-ידי מערך עם הכפלות, שמאפשר זמן amortized קבוע לפעולה.

א. נשנה את המימוש, כך שכשהמערך מתמלא נכפיל את גודלו פי $(1+\alpha)$ עבור $\alpha > 0$ במקום להכפיל פי 2. הראו שזמן הריצה amortized לפעולה הוא כעת $O(\frac{1+\alpha}{\alpha} + 1)$.
<u>חשוב</u>: *חובה להוכיח סעיף זה* ***<u>בשיטת הפוטנציאל</u>***. *(רמז: השתמשו בפונקציית פוטנציאל דומה לזו שהייתה בשיעור)*

ב. נשנה את המימוש, כך שכשהמערך שגודלו k מתמלא, נקצה מערך גדול ב- $\sqrt{k}$ תאים, ונעתיק אליו את תוכן המערך. כלומר, במקום להגדיל כפלית פי 2, אנחנו מגדילים חיבורית על-ידי יצירת מערך חדש בגודל $k + \sqrt{k}$ והעתקת k התאים המלאים אליו. שימו לב ש- k אינו קבוע לאורך הריצה. הראו שזמן הריצה amortized לפעולה הוא $\Theta(\sqrt{n})$. במילים אחרות, הראו שזמן הריצה הכולל הדרוש לסדרה של n פעולות הוא $\Theta(n\sqrt{n})$.
<u>חשוב</u>: *נזכיר שכדי להוכיח* $\Theta(f(n))$ *יש להוכיח* $\Omega(f(n))$ ***<u>וגם</u>*** $O(f(n))$.
עבור $\Omega(f(n))$ *צריך לתאר ולנתח סדרה "קשה" לטיפול, חישבו מה קורה (למשל) לאחר* $\frac{n}{2}$ *הכנסות. את החסם העליון ניתן להוכיח בסעיף זה בכל דרך שתרצו, מומלץ להשתמש בשיטת ה-accounting.*

א. נבנה פונקציית פוטנציאל מהצורה:

$$\text{Potential}(\mathrm{M,n}) = \frac{n(1+\alpha)-m}{\alpha}\ for\ n \geq \frac{m}{\alpha+1}\ and\ 0\ otherwise$$

עבור m גודל המערך n כמות האיברים במערך, ניתר לראות כי זו הינה פונקציה אי שלילית עבור $\alpha \geq 0$.

עבור הוספה פשוטה, לפני שהמערך מתמלא $n < \frac{m}{\alpha+1}$, $n+1 < \frac{m}{\alpha+1}$ כלומר נקבל:

$$amort(insert) = time(insert\ last) + potential(m,n+1) - potential(m,n) = 1 + 0 \leq 1 + \frac{1+\alpha}{\alpha}$$

עבור $n < \frac{m}{\alpha+1}$, $n+1 = \frac{m}{\alpha+1}$:

$$amort(insert) = time(insert\ last) + potential(m,n+1) - potential(m,n) \leq 1 + \frac{(n+1)(1+\alpha)-m}{\alpha} \leq 1 + \frac{1+\alpha}{\alpha}$$

עבור הוספה כשהמערך מלא נקבל:

$$amort(insert) = time(insert\ last) + potential(m(1+\alpha),n+1) - potential(m,m)$$

הכנסה במקרה הזה תכלול העתקה של המערך והוספת איבר ולכן נטען ל - $m+1$ פעולות בנוסף.
נחשב:

$$potential(m(1+\alpha),m+1) = \frac{(m+1)(1+\alpha)-m(1+\alpha)}{\alpha} = \frac{(1+\alpha)}{\alpha}$$

$$potential(m,m) = \frac{m(1+\alpha)-m}{\alpha} = m$$

לכן:

$$amort(insert) = m+1 + \frac{(1+\alpha)}{\alpha} - m = 1 + \frac{1+\alpha}{\alpha}$$

סה"כ גם במקרה הגרוע וגם במקרה הפשוט הזמן לפעולה תיקח לכל היותר $O\left(1+\frac{1+\alpha}{\alpha}\right)$ כנדרש.

ב. **חסם עליון** בשיטת חשבונאות:
נאמר שמילוי של תא עולה $2+\sqrt{n}$ מטבעות, כאשר מטבע 1 משלם על הכנסה למערך ו$1+\sqrt{n}$ נשמרים בחשבון.
גודלו ההתחלתי של המערך הינו k ומכאן שעד מילוי המערך נצברים בחשבון $(1+\sqrt{n})k$ *מטבעות. בעת הכנסה נוספת של המטבע הבא גודל המערך החדש יהי* $k+\sqrt{k}$ *ויתבצעו* k *העתקות. כלומר נדרש ל* $k+\sqrt{k}$ *מטבעות לפחות בבנק ואכן עבור* $n \geq 1$ *מתקיים* $k+\sqrt{k} \leq (1+\sqrt{n})k$ *ומכך נאמר עבור הכנסה של* n *מטבעות נקבל כי* $(2+\sqrt{n})n = O(n\sqrt{n})$

חסם תחתון:
לאחר שהוספנו $\frac{n}{2}$ איברים נותר להכניס כמות זהה. בכל הגדלה של המערך יכנסו לכל היותר $\sqrt{n}$ מקומות חדשים ולכן עבור $\frac{n}{2}$ הכנסות לכל הפחות יתבצעו $\frac{\frac{n}{2}}{\sqrt{n}} = \frac{\sqrt{n}}{2}$ הגדלות ובכל אחת מהן יעותקו לכל הפחות $\frac{n}{2}$ איברים ולכן תעשה $\Omega(\mathrm{n}\sqrt{n})$ עבודה לכל הפחות. עבור n שאינו שלם ניתן לחסום את מספר ההגדלות בעיגול למעלה של הביטוי הנ"ל וכן יתכן שההגדלה האחרונה קרתה ב $\frac{n}{2}-1$ כלומר לא נצטרך להגדיל עד שיכנסו עוד $\frac{n}{2}-1$ מטבעות נוספים אך גם במקרה הזה נדרש להגדיל ב $\sqrt{n}$ לפחות. בכל מקרה נקבל כי החסם התחתון $\Omega(\sqrt{n})$ עבור כל הכנסה ועבור n הכנסות $\Omega(\mathrm{n}\sqrt{n})$

שאלה 2

בתרגול ניתחנו את זמן הריצה amortized של מונה בינארי אינסופי עם פעולת Increment.
שלושת הסעיפים הבאים **לא** קשורים זה לזה והם בלתי תלויים זה בזה.

א. הראו שלא ניתן לממש מונה בינארי אינסופי, שתומך גם בפעולת Increment וגם בפעולת Decrement (הפחתה של 1 מערך המונה), בזמן amortized קבוע לפעולה. רמז: הראו שלכל N קיים רצף של N פעולות שזמן הריצה שלו $\omega(N)$. הניחו שלא מגיעים לערכים שליליים.

ב. כדי לשפר את יעילות המונה נשתמש בספרות $0, +1, -1$ (במקום רק ב-0 ו-1, מכונה בספרות "signed-bits representation"). ערכו של מספר המיוצג ע"י סדרת הספרות $t_{k-1}, \dots, t_0$ מוגדר להיות:

$$\sum_{i=0}^{k-1} 2^i t_i$$

למשל $1, 0, -1$ *הוא הייצוג של* $3 = 2^2 - 2^0$.

פעולת increment של מספר בייצוג כזה מתבצעת באופן דומה לביצועה במערכת המספרים הרגילה. מוסיפים 1 לספרה הימנית ביותר. אם ערכה הפך ל-2, הוא משתנה ל-0 וגוררים את העודף לספרה הבאה משמאל. Decrement מתבצע בצורה דומה: מורידים 1 מהספרה הימנית ביותר, אם ערכה הפך ל-(2-), הופכים אותו ל-0, וגוררים את החוסר (1-) לספרה שמשמאל. דוגמא: המספר $1, 0, -1$ פחות 1, נקבל $1, -1, 0$. כעת נוסיף לו 1 ונקבל $1, -1, 1$. שימו-לב שקיבלנו שתי צורות שונות לייצוג של 3: $1, 0, -1$, ו- $1, -1, 1$. נגדיר את עלות הפעולה להיות **מספר הספרות המשתנות** כאשר מבצעים את הפעולה. הוכיחו, כי בייצוג שכזה העלות של סדרה של n פעולות increment ו-decrement כאשר מתחילים ממונה שערכו 0 היא $O(n)$.

ג. בסעיף זה נרחיב את מבנה הנתונים. נוסיף פעולת RESET, אשר מאפסת את כל הביטים שמייצגים את המספר שמראה המונה.
המונה כעת שומר את המיקום של הביט הכי שמאלי במונה במשתנה עזר(מחוץ למונה), הפעולה RESET מאפסת את כל הביטים עד הביט הכי שמאלי (כולל ביטים עם ערך 0). שימו לב שלאחר פעולת RESET הערך שמראה המונה הוא 0 והביט הכי שמאלי שבשימוש הוא 1. הראו שזמן הריצה AMORTIZED לפעולה נשאר O(1). שימו לב –סדרת פעולות על מבנה הנתונים כוללת כעת גם פעולות increment וגם פעולות RESET, ואין שום אילוץ על סדר הפעלתן. בנוסף שימו לב שגם מעבר על המערך (אפילו בלי לשנות ביטים) צריך להילקח בחשבון בניתוח זמן הריצה.

א. פעולת increment היא הגדלה של מספר בינארי ב-1 במקרה הגרוע עלות של פעולה כזו תהיה עבור הגדלה של 111..11 ל 100..000 ותדרוש $\Omega(log(n))$ פעולות

באופן דומה decrement היא הקטנה של מספר בינארי ב-1. במקרה הגרוע עלות של פעולה כזו תהיה עבור הגדלה של 100..000ל 01..111 ותדרוש $\Omega(log(n))$ פעולות גם כן. עבור סדרת פעולות באורך n של העלה והורדה מסוג זה ידרשו $\Omega(log(n)n)$ פעולות. וכן מתקיים $log(n)n = \omega(N)$. אם הטענה הייתה מתקיימת היינו אומרים כי סך העבודה תהיה מגודל $O(n)$ מה שבוודאי לא מתקיים.

*כדי לייצג את המספר n בבסיס בינארי נדרש ל $O(log(n))$ תווים.

ב. נבחין כי הביט הk משתנה רק לאחר לפחות 2^k שינויים של שאר הביטים. על מנת ש הביט הנ"ל ישנתה שאר הביטים שלפניו צריך להיות רצף של 1 או 1- רק אז פעולת curry המתוארת תתרחש. כלומר הביט ה0 ישתנה בכל צעד הביט ה1 ישתנה בכל 2 צעדים וכך הלאה. ומכך הסכום הכולל של שינויי כל הביטים מוגבל על יד $\sum_{k=0}^{\infty}\frac{n}{2^k}$ וכן

$$\sum_{k=0}^{\infty}\frac{n}{2^k} = 2n = O(n)$$

ג. נראה זאת בעזרת שיטת הפוטנציאל ונגדיר את פונקציית הפוטנציאל להיות מספר הביטים השונה ל0. נזכור כי העלות של מעבר על המחרוזת עד לתו הk הינה $k = O(k)$

עלות **increment** ו**decrement** : הפוטנציאל ישלם בשרשרת הביטים שישתנו לאפס למעט התו האחרון בשרשרת השינויים ו"ירוויח" את התו האחרון שישתנה מאפס ולכן הפוטנציאל בסוף התהליך:

$$\text{diff}(\Phi) = 1 - (k-1) = 2-k$$

$$\text{amort(oper)} = \text{diff}(\Phi) + \text{cost of looping } through \text{ the chain} = 2-k + k = 1 \rightarrow O(1)$$

עלות reset: *העלות של פעולה זה היא מעבר על כל תווי המחרוזת נניח n וכן הפוטנציאל יורד לאפס "משלם" על כל ביט שהשתנה ל0 בעקבות הפעולה ואז שווה ל 1 עבור הביט האחרון שמשתנה ל1 עבור הביט השמאלי ביותר. עבור n פעולות* increment וdecrement תתבצע פעולת reset יחידה. וכמו שראינו בכיתה:

$$\text{זמן אמורטייזד} = \frac{\text{זמן לסדרת פעולות גרועה}}{\text{אורך הסדרה}}$$

ומכך עבור n **פעולות** increment וdecrement תתבצע פעולת איתחול אחת שהיא גרועה ולכן:

$$\text{amort(reset)} = \frac{o(n)}{n} = o(1)$$

*עבור $n = 0$ פעולת reset תהיה כמובן מ $\boldsymbol{O(1)}$

שאלה 3

הוכיחו/הפריכו את הטענות הבאות (הפרכה ע"י דוגמה נגדית, הוכחה באמצעות נימוק קצר).

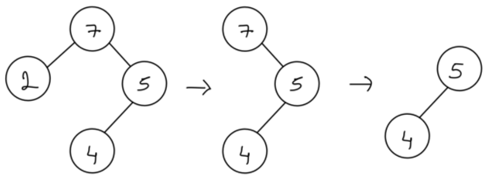
א. פעולת המחיקה מעץ חיפוש בינארי היא חלופית. כלומר, העץ המתקבל ממחיקת x ואחריו y זהה לעץ המתקבל ממחיקת y ואחריו x, לכל x, y בעץ.

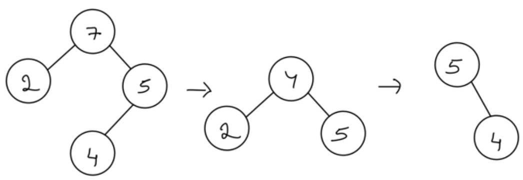
ב. יהיו u, v שני צמתים בעץ חיפוש בינארי כך ש- $u < v$, אזי הצמתים במסלול מ-u אל v בעץ (בהתעלם מכיווני הקשתות) ממוינים לפי מפתחות בסדר עולה.

ג. יהיו T_1, T_2 שני עצי חיפוש בינאריים בגודל n כך שסדרת המפתחות בשני העצים זהה. אזי, קיימת סדרה של $O(n)$ גלגולים (סיבובי קשתות) אשר הופכת את T_1 ל- T_2. (רמז: נסו להגיע משני העצים לאותו עץ T_3 ע"י סדרה של $O(n)$ גלגולים).

א

א. דוגמה נגדית: במקרה הראשון נמחק את 2 ואז את 7 במקרה השני נמחק את 7 ואז את 2. כפי שניתן לראות העץ המתקבל אינו זהה





ב. דוגמה נגדית: נשתמש בעץ המקורי בדוגמה מסעיף א. עבור המסלול בין 2 ל4 כאשר $2<4$ כמובן ניתן לראות כי המסלול אינו בסדר עולה אלא בסדר עולה ואז יורד $2 < 7 > 5 > 4$

ג. בעץ בעל n קודקודים ישנן n -1 קשתות ומכך נסיק כי ניתן לגלגל כל עץ לעץ שרוך בו האיבר המינימלי הוא השורש והאיבר יסודרו בסדר עולה בתת עץ השמאלי של השורש ב כ$O(n)$ או $n-1$ פעולות. נגלגל את $T_1,$ לעץ שרוך שמאלי ב$n-1$ פעולות ומכך שאפשר להמיר את T_2 ב $O(n)$ לעץ זהה נבצע את אותן מספר פעולות בסדר הפוך על מנת להמיר את העץ שרוך השמאלי לT_2 לשם כך נדרשות $2n-2$ פעולות. ולכן נסיק כי הדבר אפשרי ב$O(n)$

שאלה 4

עץ d-ארי הוא עץ שבו לכל צומת **יש** לכל היותר d בנים. אומרים כי העץ הוא **עץ d-ארי נחמד** אם לכל צומת שאינו עלה **יש** *בדיוק* d בנים.

א. מהו מספר העלים בעץ d-ארי נחמד בעל n צמתים? עליכם לכתוב ביטוי מדוייק (ולא "אסימפטוטי") ולהוכיח את נכונותו.

ב. בהינתן **עץ d-ארי נחמד** בגובה h עם L עלים הוכיחו שמתקיים $L \leq d^h$.

א. אראה באינדוקציה על גובה העץ h כי . $\frac{nd-n+1}{d}$ = עלים

עבור $h=0$ מספר העלים הינו 1 ואכן המשוואה מקיימת זאת.

עבור h = 1 *מספר העלים הינו* d *ואכן המשוואה מקיימת זאת.*

נניח כי הדבר מתקיים עבור עץ בגובה $h-1$ ונראה כי הדבר מתקיים עבור עץ בגובה h. נבחין כי כל הבנים של השורש הם בעצמם עצים d ארים נחמדים וכן כמובן יש d כאלו. נסמן ב – n_i את מספר הצמתים בתת העץ הi וכן כל לראות כי $n = 1 + \sum n\_i$ ומכך שעלי התתי עצים הם העלים של העץ עצמו נשתמש בהנחת האינדוקציה ונחשב:

$$\text{עלים} = \sum_{1}^{d} \frac{n\_id - n\_i + 1}{d} = \frac{(n-1)d - n - 1 + 1}{d} = \frac{nd - n + 1}{d}$$

כלומר הוכחנו את הטענה כנדרש.

ב. *שוב, אוכיח באינדוקציה:*

עבור $h=0$ מספר העלים הינו 1 ואכן המשוואה מקיימת זאת $1 \leq 1$.

עבור h = *1* מספר העלים הינו d ואכן המשוואה מקיימת זאת $d \leq d$.

אניח זאת עבור עץ בגובה $h-1$ ונראה כי הדבר מתקיים עבור עץ בגובה h. שוב נתבונן בd תתי העצים בגובה $h-1$ וכן לכל עץ i נסמן L_i i עלים כך ש $L = \sum L_i$ ומהנחת האינדוקציה $L = \sum_1^d L_i \leq \sum_1^d d^{h-1} = d^h$ כנדרש.

שאלה 5

א. כתבו פסאודו קוד לפונקציה המקבלת שורש של עץ בינארי ומחזירה האם הוא עץ **חיפוש**.
ב. כתבו פסאודו קוד **לא רקורסיבי** המקבל שורש של עץ בינארי ומדפיס את מפתחות העץ in-order.
ג. כתבו פסאודו קוד המקבל מערך בגודל n המייצג סריקת pre-order של עץ חיפוש בינארי ומחזיר שחזור של העץ (יש להחזיר את השורש).
הערה: על זמני הריצה להיות **לינאריים**. בסעיף ב' אין להניח שלצומת יש מצביע לאביו.

א. נגדיר פונקציה רקורסיבית העוברת על כל צומת בגרף ומוודאת שהיא נמצאת בטווח הערכים המתאים. נעבור על כל צומת פעם אחת ונבצע $o(1)$ פעולות כך שעבור n צמתים סיבוכיות הפונקציה היא $o(n)$

```
Is_search_helper (node, max, min):
  If node = null then return true
  Val ←node.val
  If (val < max or val > min) then return false
  Return Is_search_helper (node.left, val,min) and Is_search_helper (node.right, max ,val)

Is_search_helper (node):
  Return Is_search_helper(node,∞, −∞)
```

א. ב. נגדיר פונקציה העוברת על כל צומת בגרף. נעבור על כל צומת פעם אחת ונבצע $o(1)$ פעולות כך שעבור n צמתים סיבוכיות הפונקציה היא $o(n)$

```
In order(root):
  S ← stack()
  Curr ← root
  While(curr ≠ null  or s is not empty):
      While(curr ≠ null):
           s.push(curr)
           curr ← curr.left
      curr ← s.pop()
      print(curr)
     curr ← curr.right
```

ב. הפונקציה עושה שימוש במחסנית על מנת לבנות את העץ וכן נעבור על כל צומת בעץ פעם 1 ולכן סיבוכיות זמן הריצה היא $o(n)$

```
Pre order(list):
   If list.length is 0 return null
   root ← new treenode(list[0])
  stack    ← stack()
   stack.push(root)
   right ← []
   for i ← 1 up to list.length() - 1:
         current←  new treenode(list(i))
        if list[i]<stack.top().val then:
           stack.top().left ← current
           stack.push(current)
      else:
            parent ← null
           While stack is not empty and list[i] > stack.top().val:
               parent ← stack.pop()
    parent.right ← current
           stack.push(current)

   Return root
```