מבני נתונים – פתרון 8 - ערימה

שאלה 1

ניתוח זמן הריצה של מיון-ערימה על מערך A בגודל n **הממוין בסדר עולה**:

השלב הראשון במיון-ערימה הוא לבנות מהמערך ערימת מקסימום. הלולאה בתוך BuildHeap תרוץ n/2 פעמים, וכפי שהוכח בכיתה, פעולה זו עולה $\Theta(n)$.

לאחר מכן HeapSort מבצעת את הפעולות הבאות:

1. מוציאים מהמערך את האיבר הראשון (השורש), שהינו האיבר המקסימלי, ומכניסים אותו למיקום הפנוי הגדול ביותר במערך.
2. מוציאים את האיבר האחרון במבנה, שמים אותו במקום השורש, ומסדרים את המבנה שנותר כערימת מקסימום ע"י הפונקציה InsertHeap.

בשלב 2 ייכנס במקומו של השורש איבר שקודם היה עלה – כלומר יש איברים גדולים ממנו במערך. לכן בפעולת InsertHeap נצטרך להיכנס ללולאה ולבצע תיקונים במבנה שנותר, ומכאן שהעלות תהיה $\Theta(logn)$ לכל איטרציה , ובסה"כ $\Theta(nlogn)$.

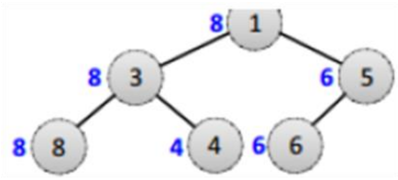
עבור מערך שכבר ממוין בסדר יורד, עדיין תתבצע פעולת בניית הערימה (למרות שהמערך כבר מקיים את התנאים של ערימת מקס'). גם כאן הלולאה בתוך BuildHeap תרוץ n/2 פעמים, וזמן הריצה יהיה $\Theta(n)$.

לאחר מכן HeapSort תבצע אותם שלבים שתוארו קודם, תוך הפעלת InsertHeap עבור כל איבר במערך, ולכן זמן הריצה יהיה $\Theta(nlogn)$.

לסיכום, בשני המקרים זמן הריצה יהיה $\Theta(nlogn)$ (ומכאן רואים שזה באמת זמן הריצה בכל מקרה, בלי תלות בסדר הנתונים).

שאלה 2 המשפטים המודגשים הם הפרטים הטכניים שתלמידים נטו לשכוח.

פתרון – אפשרות א



נשמור את כל האיברים בערימת מינימום אחת.
כל צומת יחזיק שדה נוסף Max, שישמור את הערך המקסימלי בתת-עץ שמושרש בו (כולל הצומת עצמו).

פעולות:

Init - נבצע BuildHeap, ולאחר מכן נבצע סיור postorder לחישוב שדות הMax. עלות - $\theta(n)$.

Insert(x) - נכניס כרגיל את האיבר לערימה ונעדכן את שדות ה-Max, במסלול ההכנסה מלמטה עד לשורש. עלות - $\theta(logn)$.

FindMin - נחזיר את ערך השורש (באינדקס 0). עלות - $\theta(1)$.

FindMax - נחזיר ערך ה-Max שבשורש. עלות - $\theta(1)$.

DelMin - נמחק את השורש ונסדר את הערימה כרגיל, תוך עדכון ערכי הMax לאורך שני מסלולים:

- המסלול מהעלה האחרון (זה שנמחק) עד השורש.
- המסלול שבו סידרנו את הערימה, מלמטה עד השורש.
  עלות כוללת - $\theta(logn)$.

DelMax –

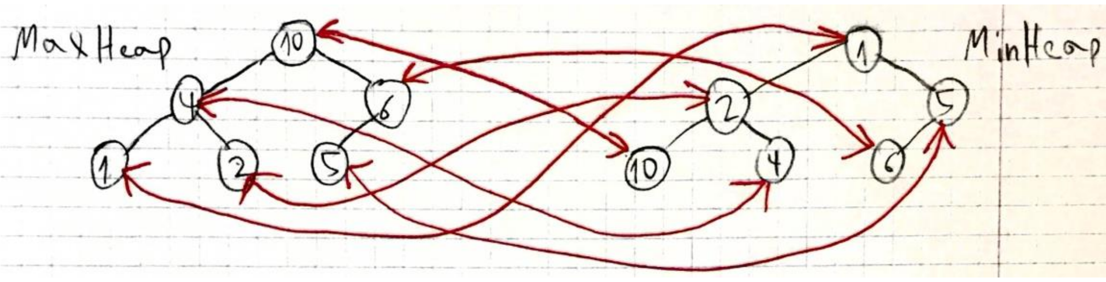
- נמצא את המקסימום באופן הבא:
  נתחיל מהשורש, נבדוק למי מהבנים ערך Max גדול יותר ונלך אליו. נמשיך רקורסיבית.
- נמחק עלה זה כפי שהראינו קודם, נסדר את הערימה ונעדכן את ערכי ה-Max לאורך שני המסלולים.
  עלות כוללת - $\theta(logn)$.

פתרון – אפשרות ב

נגדיר את המבנה הבא: ערימת מקסימום ומינימום במקביל, כלומר המבנה יכיל שני מערכים, האחד לייצוג ערימת מקסימום והשני לייצוג ערימת מינימום.

כל איבר בערימה יכיל את השדות הבאים:

- data – הערך של האיבר
- max_index – המיקום של האיבר במערך שמייצג את ערימת המקסימום
- min_index – המיקום של האיבר במערך שמייצג את ערימת המינימום.



מבנה נתונים זה מאפשר את הפעולות הבאות בסיבוכיות שנדרשה:

1. init() – כפי שראינו בכיתה, בהינתן n איברי קלט העלות של יצירת ערימה בינארית, לא משנה אם מדובר בערימת מקס' או מינ', הינה $O(n)$, ולכן במבנה הנתונים שהוגדר הסיבוכיות תהיה $2n = O(n)$.
   נשים לב כי בעת אתחול המבנה, השדות הפנימיים של כל איבר צריכים להתעדכן עם המיקומים שהאיבר קיבל בערימת המקסימום (max_index) ובערימת המינימום (min_index).
2. insert(x) – ידוע כי העלות של הוספת איבר לערימה בינארית היא $O(logn)$, ולכן גם העלות של הכנסת איבר למבנה הנתונים שהגדרנו תהיה $2logn = O(logn)$, כי מכניסים את האיבר לשתי ערימות בינאריות.
   כמו כן, לכל איבר שהמיקום שלו השתנה (האיברים לאורך מסלולי ההכנסה), עלינו לעדכן את השדות max_index ו- min_index, עם המיקומים שהוא קיבל בשתי הערימות. (למשל, אם משנים מיקום של איבר בערימת המקסימום, נשתמש בשדה min_index כדי לדעת איפה האיבר ממוקם בערימת המינימום. נלך למיקום זה ונעדכן בו את השדה max_index.) עלות העדכון לכל איבר תהיה $O(1)$, ובסה"כ לכל האיברים במסלולי ההכנסה $O(logn)$.
3. FindMax() – מחזיר את האיבר הראשון בערימת המקסימום, סיבוכיות של $O(1)$.
4. FindMin() – מחזיר את האיבר הראשון בערימת המינימום, סיבוכיות של $O(1)$.
5. DelMax() – נוציא את השורש מערימת המקסימום כרגיל, תוך עידכון שדות האינדקס לאורך המסלול כמו בפעולת הכנסה.
   בנוסף לכך, בעזרת השדה min_index נוציא את האיבר המקסימלי גם מערימת המינימום (נשים במקומו את העלה האחרון). נתקן את הערימה ונעדכן את שדות האינדקס לאורך שני מסלולים:
   - המסלול מהעלה האחרון עד השורש.
   - המסלול מהמקום בו היה האיבר המקסימלי עד השורש.

   עלות כוללת - $\theta(logn)$.
6. DelMin() – מימוש סימטרי לפונקציית DelMax(), ולכן העלות גם תהיה $\theta(logn)$.

לסיכום – מבחינת זמני הריצה – שני הפתרונות עונים על הדרישות ולכן שניהם טובים.

אמנם, מבחינת דרישות זיכרון, הפתרון השני דורש שלכל איבר יהיו עוד שני שדות של אינדקסים (max_index ו- min_index), וגם מערך נוסף לצורך ערימה נוספת, בעוד הפתרון הראשון דורש לכל איבר תוספת של שדה אחד בלבד (Max).

<u>שאלה 3</u>

כדי לממש מחסנית באמצעות תור עדיפויות, כל איבר חדש שייכנס לתור יקבל עדיפות גבוהה יותר מזו של האיבר שנכנס לפניו.

המימוש יתבצע על ידי ערימת מקסימום. העדיפות תיקבע על ידי שדה counter שנוסיף לכל איבר. האיבר הראשון שייכנס יקבל לתוך ה-counter שלו את הערך 0. כל איבר נוסף שייכנס יקבל לתוך ה-counter שלו את ערך הcounter + 1 של האיבר שנמצא באינדקס ה-0 (האיבר בעל העדיפות הגבוהה ביותר).

בשיטה זו האיבר האחרון שייכנס יקבל את העדיפות הגבוהה ביותר, ולכן תור העדיפות יתנהג בצורה של LIFO כנדרש במחסנית:

- בהכנסת איבר חדש הוא יקבל את העדיפות הגבוהה ביותר מכל האיברים שנכנסו לפניו – בדיוק כמו בפעולת push.
- בהוצאה מתור עדיפות ייצא האיבר בעל העדיפות הגבוהה ביותר, ובמקרה הזה מדובר באיבר האחרון שנכנס – בדיוק כמו בפעולת pop.
- הצצה לאיבר הראשון בתור תציג את האיבר בעל העדיפות הגבוהה ביותר, ובמקרה הזה מדובר באיבר האחרון שנכנס – בדיוק כמו בפעולת top.