2014 א' – מועד 86.

1. נפתור בעזרת הצבה ושיטת האב:

נציב:

נציב פעם שניה:

נשתמש בשיטת האב:

ולכן לפי מקרה 2 המורחב:

נציב חזרה ונקבל:

* 1. ניצור מערך בגודל 2k, ונכניס אליו כל פעם K איברים מתוך הרשימה.  
     בפעם הראשונה יש K מספרים, אז אין צורך לעשות כלום. מהפעם השניה יש 2K איברים ובכל הכנסה, נעשה SELECT על מנת למצוא את ערך המיקום K ונעשה מסביבו PARTITION. מכיוון שיש 2k איברים במערך, כל פעולה כזאת תקח O(k) זמן ריצה.  
     אחרי כל חלוקה, יהיו לנו בסוף הרשימה את K המספרים הגדולים עד עתה, ואת ה-K הבאים נכניס במקום הקטנים ונחזור כך עד סוף הרשימה.  
     בסה"כ נבצע את הפעולה הזאת n/k פעמים, ולכן כל פעולה ההכנסה ומציאת ה- K הגדולים יקח לנו O(n) זמן.  
     לאחר מכן, נמיין את K הגדולים ב- klgk ובסה"כ: O(n + klgk)
  2. קוראים את כל הרשימה למערך בגודל n, עושים SELECT על הערך מיקום n-k, ומבצעים סביבו PARTITION. סה"כ O(n).

1. נשתמש בעץ אדום שחור כשהצמתים הם לפי המפתחות, בכל צומת יש מחסנית כדי להכניס איברים עם אותו מפתח ושדה count שייצג את מספר האיברים באותו צומת. ובערימת מקסימום לייצוג השכיחויות. בכל צומת יוחזק מצביע לאיבר המתאים בערימה.  
   **BUILD** – נבנה עץ אדום שחור מרשימה ממוינית לוקח, זמן ריצה - O(n), כל איבר שהמפתח שלו כבר קיים, יוכנס לתוך המחסנית ויעדכן את ה- count. לאחר בניית העץ,נסרוק את m הצמתים ונכניס אותם למערך עזר עם מצביעים דו כיווניים בין הצומת לאיבר במערך כאשר הערך במערך הוא ה- count (השכיחות), ניצור ערימת מקסימום ממערך זה.  
   זמן ריצה O(n+m) אבל מכיוון ש- n>m, סה"כ נקבל O(n).  
   **INSERT** – מכיוון שיש לנו m צמתים, יקח לנו lgm זמן למצוא את הצומת המתאים. עדכון המחסנית, השדה count והאיבר המתאים בערימה לוקח O(1). תיקון הערימה ע"י "גלגול כלפי מעלה" במידת הצורך – lgm.  
   אם המפתח לא קיים, נוסיף צומת חדש לעץ עם count = 1, נוסיף אותו גם כאיבר חדש לעריצה ומצביע דו כיווני בין הצומת לערימה.  
   סה"כ זמן ריצה O(lgm) – מציאת איבר או הוספה ותיקון הערימה או הוספת איבר לערימה.  
   **DELETE-LAST** – מציאת הצומת עם המפתח (lgm), POP למחסנית ועדכון הcount (O1), ע"י המצביע נוריד גם 1 מהאיבר המתאים בערימה ונבצע heapify.  
   אם ע"י המחיקה, הגענו ל count = 0 נמחוק את האיבר מהעץ ומהערימה ונתקן - ובסה"כ O(lgm).  
   **MAX-FREQ** – החזרת השורש של ערימת המקסימום O(1).  
   1. גובה העץ T הוא h, גובה התת עץ הימני הוא h-1 ומהנתון אנו יודעים שגובה התת עץ השמאלי הוא h-3.  
      ידוע לנו גם שהתת עצים הימני והשמאלי הם מאוזנים וכך גם כל הבנים שלהם ולכן הם עצים שלמים. מספר הצמתים כפונקציה של גובה בעץ שלם הוא:   
      ולכן מספר הצמתים בעץ הימני הוא: ומספר הצמתים בתת עץ השמאלי הוא , כמובן שצריך להוסיף את הצומת של השורש ובסה"כ מספר הצמתים בעץ T כפונקציה של h הוא: .
2. מבנה הנתונים יהיה עץ אדום שחור עם 2 שדות נוספים D ו- I. ה-D יסמן את הערך שצריך להוריד מכל הצמתים שקטנים ממנו, ו- I יסמן את הערך שצריך להוסיף לכל הצמתים הגדולים ממנו.  
   DECREASE – סורקים את העץ כלפי מטה מהשורש, אם מפתח הצומת קטן מ- K, נעדכן את הערך D הנוכחי שלו ל- D=D-d, ונמשיך לסרוק את התת עץ הימני. אם הוא גדול או שווה ל-K, נמשיך לסרוק את התת עץ השמאלי. סה"כ אפשר להגיע לגובה העץ – O(lgn).  
   INCREAE – שוב נסרוק מהשורש, הפעם אם המפתח קטן מ- K נמשיך לסרוק את התת עץ הימני ואם הוא גדול או שווה ל – K, נעדכן את הערך I של הצומת ל- I=I+d.  
   זמן ריצה – O(lgn).  
   SEARCH – נשתמש ב-2 משתני עזר d ו – i שיואתחלו ב-0. בכל צומת שמגיעים, מוסיפים את ערכי D, I של הצומת ל- d ו- i העזרים. נבצע השוואה: key[x]+d+i מול K שקיבלנו בשגרה כאשר X הוא הצומת הנוכחי. אם K קטן מהתוצאה נמשיך ללכת על התת עץ השמאלי לא לפני שנבטל את עדכון ה- i שלנו ע"י = i=i-I , אם גדול אז נמשיך לסרוק את התת עץ השמאלי ונשווה ונבטל את עדכון d ע"י d = d – D. אם שווה, מצאנו. סה"כ זמן ריצה הוא שוב גובה העץ – logn.  
   INSERT – הכנסה רגילה לעץ א"ש, רק שבכל צומת בדרך אם פונים ימינה נוסיף את הערך D ל-D שבאיבר שלנו, ואם פונים שמאלה נוסיף את הערך i.