מבנ"ת- תרגיל מעשי 2.0

מגישים:

בר פקולה (ID : 316294297, user name : barpakula)

בר שגב (ID : 312456262, user name : barsegev)

# חלק א- תיעוד פונקציות

### דגשים:

* עקרון מימוש כללי:
  + הערמה ממומשת על ידי רשימה מקושרת דו כיוונית (לא מעגלית) שמקשרת בין שורשי העצים בה:
    - מצביע ראש head מצביע לשורש השמאלי ביותר ברשימה.
    - מצביע זנב tail מצביע לשורש הימני ביותר ברשימה.
  + כל עץ בערמה ממומש באופן הבא:
    - לכל צומת עם ילדים יש מצביע לילד השמאלי ביותר בלבד.
    - לכל צומת שהיא לא שורש של עץ יש מצביע להורה שלה. לשורשים המצביע להורה הוא null.
    - צמתים אחים (שבאותה רמה) מקושרים זה לזה ברשימה מקושרת דו כיוונית (לא מעגלית). הבן השמאלי ביותר הוא ראש הרשימה, המצביע שלו לאח שמאלי הוא null. באופן דומה הבן הימני ביותר הוא זנב הרשימה והמצביע שלו לאח ימני הוא null.
* על מנת שהתיעוד יהיה יותר קריא לא נציין כאשר מתבצעת קריאה בפונקציה ל- setters ו-getters. הסיבוכיות של כל ה-setters וה-getters שמומשו בקוד היא .

## מחלקת FibonacciHeap

### שדות המחלקה

המחלקה כוללת את השדות הבאים:

* head- מצביע לראש הרשימה המקושרת של שורשי העצים בערמה.
* tail- מצביע לזנב הרשימה המקושרת של שורשי העצים בערמה.
* min- מצביע לצומת שהוא האיבר המינימלי בערמה.
* size- שומר את גודל הרשימה.
* treeCount- שומר את כמות העצים בערמה.
* markedCount- שומר את כמות הצמתים המסומנים הנוכחית.
* linkedCount- שומר את כמות פעולות החיבור שבוצעו מתחילת ריצת התוכנית.
* cutsCount- שומר את כמות פעולות החיתוך שבוצעו מתחילת ריצת התוכנית.

### isEmpty()

**תיאור**- הפונקציה מחזירה ערך TRUE אם ורק אם הערימה ריקה.

**מימוש**- אם המצביע head לראש רשימת השורשים בערמה הוא null מחזירה TRUE, אחרת FALSE.

**סיבוכיות**-

### insert(int key)

**תיאור**- הפונקציה יוצרת צומת מסוג HeapNode שמכיל את המפתח 𝑖 ומכניסה אותו לערימה. פעולה זו מחזירה את הצומת שנוצר שמכיל את המפתח i.

**מימוש**- הפונקציה:

* יוצרת צומת מסוג HeapNode עם המפתח key.
* אם הערמה ריקה- מעדכנת את זנב רשימת שורשי העצים בערמה להצביע לצומת זו.
* **מכניסה את הצומת לראש הרשימה המקושרת של שורשי עצי הערמה:**
  + מגדירה את ראש הרשימה הנוכחי להיות next של הצומת החדשה.
  + אם ראש הרשימה לא Null מגדירה את prev של head להצביע לצומת החדשה.
  + מגדירה את head להצביע לצומת החדשה.
* **עדכון שדה שורש-** מעדכנת את שדה isSource של הצומת להיות TRUE
* **עדכון גודל הערמה-** מעדכנת את גודל הערמה ב + 1
* **עדכון כמות העצים בערמה-** מעדכנת את כמות העצים ב + 1
* **עדכון מינימום-** משווה בין i לבין המפתח ששמור בשדה האיבר המינמלי ומעדכנת את מצביע המינימום לאיבר החדש במידה והוא קטן מהנוכחי.
* מחזירה את האיבר החדש שיצרה.

**סיבוכיות-** כל הפעולות המתבצעות לוקחות זמן קבוע.

### deleteMin()

**תיאור**- מחיקת הצומת שהמפתח שלו מינימלי מבין המפתחות שבערימה (אין צורך להחזיר אותה).

**מימוש**- במידה וקיים איבר מינימלי בערמה, הפונקציה:

* שומרת מצביע לבן השמאלי ביותר של צומת המינימום בשם currChild.
* **אם קיים לפחות בן אחד לצומת המינימום** – מחברת את הרשימה המקושרת של הילדים לרשימת השורשים. בפועל, הופכת את הילדים של הצומת המינימלית להיות שורשי עצים בערמה:
  + מעדכנת את ה-prev של הבן השמאלי להיות האח השמאלי של הצומת המינימלית.
  + אם ה-prev הוא צומת – מעדכנת את ה-next שלה להצביע לבן השמאלי של הצומת המינימלית.
  + אם הצומת המינימלית היא גם ראש הרשימה המקושרת של השורשים – מעדכנת את הראש להצביע לבן השמאלי שלה.
  + מעדכנת את שדות הילדים של הצומת המינימלית כך שיגדירו אותם כשורשים:
    - את שדה ההורה משנה ל-null.
    - מדליקה את דגל השורש isRoot.
    - אם הבן הנוכחי היה מסומן – מוחקת את הסימון ומעדכנת את ספירת הצמתים המסומנים בעץ.
  + אם הצומת המינימלית היא גם זנב הרשימה המקושרת של השורשים – מעדכנת את הזנב להצביע לבן הימני שלה.
  + מעדכנת את ה-next של הבן הימני להצביע לאח השמאלי של הצומת המינימלית.
  + אם ה-next הוא צומת – מעדכנת את ה-prev שלה להצביע לבן הימני של הצומת המינימלית.
* **אם אין לצומת המינימום ילדים**:
  + אם יש לצומת המינימלית אח שמאלי (בפועל זה אומר שהיא לא ראש הרשימה המקושרת של שורשי העצים בערמה)- מעדכנת את next של האח השמאלי להצביע לאח הימני של הצומת המינימלית.

אחרת – מעדכנת את ראש הרשימה להצביע לאח הימני.

* + אם יש לצומת המינימלית אח ימני (בפועל זה אומר שהיא לא זנב הרשימה המקושרת של שורשי העצים בערמה)- מעדכנת את prev של האח הימני להצביע לאח השמאלי.

אחרת – מעדכנת את זנב הרשימה להצביע לאח השמאלי.

* **מנתקת את צומת המינימום מהערמה-** מאפסת את שדות המצביעים של הצומת המינימלית ל-null.
* קוראת ל- **consolidate**() שמבצעת את תהליך הקונסולידציה על הערמה. הפעולה לוקחת במקרה הגרוע .
* **עדכון גודל הערמה-** מעדכנת את גודל הערמה במינוס 1.
* **מעדכנת את המינימום החדש בערמה ואת כמות העצים החדשה:**
  + מאפסת את כמות העצים בערמה.
  + יוצרת מצביע למינימום הנוכחי שבהתחלה מצביע לראש הרשימה המקושרת של שורשי העצים.
  + יוצרת מצביע לראש ועוברת על שורשי העצים אחד אחד:
    - **עדכון מינימום**- משווה את המפתח במצביע המינימום הנוכחי למפתח הצומת הנוכחי ואם הוא גדול יותר מעדכנת את מצביע המינימום הנוכחי להצביע לצומת הנוכחית.
    - **עדכון כמות העצים בערמה-** מעדכנת את כמות העצים ב + 1 בכל עץ שעוברים אליו.
    - מקדמת את המצביע לשורש הנוכחי לאח הימני שלו.

מתבצעות לכל היותר איטרציות כאלה ככמות העצים בערמה לאחר קונסולידציה.

* + מעדכנת את מצביע המינימום להצביע למינימום הנוכחי. (נשים לב שאם הערמה כעת ריקה – הראש שלה null ולכן גם המינימום יתעדכן ל-null).

**סיבוכיות-** במקרה הגרוע שנובע מפעולת הקונסולידציה במקרה שכל השורשים בערמה מדרגה 0 (n שורשים).

### consolidate()

**תיאור-** הפונקציה עושה קונסולידציה, מאחדת את שורשי העץ לכמות העצים המינימלית.

**מימוש**- הפונקציה:

* יוצרת מערך buckets בגודל .
* **מבצעת את אלגוריתם successive linking**:
  + מגדירה מצביע לשורש הנוכחי בערמה שמתחיל מראש הרשימה המקושרת של השורשים.
  + עוברת על הרשימה המקושרת של שורשי העצים בערמה. כל עוד השורש הנוכחי לא null:
    - שומרת מצביע לאח הבא של הצומת הנוכחית בשם nextTree.
    - מנתקת את הצומת מהאח השמאלי שלה – שדה prev מצביע ל-null.
    - בודקת מה הדרגה של השורש הנוכחי.
    - אם הדרגה היא i היא מסתכלת על המקום ה-i במערך. כל עוד המקום ה-i במערך לא null:
      * משווה בין מפתח השורש במצביע השורש הנוכחי לבין מפתח השורש שבמקום ה-i במערך. מעדכנת את הגדול מבניהם להיות הבן השמאלי של הקטן יותר.
      * מעדכנת את שדה isRoot בהתאם.
      * מעדכנת את מצביע ההורה והילד בהתאם.
      * מעדכנת את הדרגה של השורש + 1.
      * מעדכנת את המצביע הנוכחי להורה.
      * משנה את המיקום הקודם במערך ל-null.
      * **מעדכנת את כמות החיבורים-** מוסיפה + 1 את linkedCount.
    - הלולאה מסתיימת כאשר המקום ה-i במערך ריק (null) ואז מתבצעת השמה של העץ המחובר לתא ה-i במערך. סך הכל יתבצעו לכל היותר איטרציות.
    - המצביע הנוכחי מתעדכן להצביע ל-nextTree- השורש הבא ברשימה המקושרת של שורשי הערמה.
  + חוזר חלילה עד שעברנו על כל שורשי הערמה. סך הכל יתבצעו לכל היותר איטרציות במקרה שבו כל שורשי הערמה מדרגה 0 (צמתים בודדים).
* **בונה מחדש את הרשימה המקושרת של שורשי העצים בערמה מתוך המערך שהתקבל:**
  + מגדירה מצביע לשורש הנוכחי שמתחיל מהמקום ה-0 במערך, ומצביע לצומת הבאה.
  + עוברת באיטרציה על כל המקומות במערך. בפעם הראשונה שנתקלה בתא שמכיל שורש – מעדכנת את ראש הרשימה המקושרת של שורשי הערמה להצביע אליו. לאחר מכן, בכל פעם שנתקלת בצומת נוספת שאינה null במערך –מעדכנת את הצומת הנוכחית והצומת הבאה להיות אחים ומתקדמת עד שעברה על כל המערך.
  + מעדכנת את הזנב של הרשימה המקושרת להצביע לצומת האחרונה שחוברה לרשימה המקושרת של השורשים.

לכל היותר יתבצעו בשלב זה איטרציות ככמות השורשים בעץ.

**סיבוכיות-**  במקרה הגרוע שנובע מהמקרה שבו כל השורשים בעץ הם צמתים מדרגה 0 (n שורשים).

### findMin()

**תיאור-** החזרת הצומת (מטיפוס HeapNode ) שהמפתח שלו מינימלי מבין המפתחות שבערמה. מערימה ריקה יוחזר null.

**מימוש-** החזרת מצביע המינימום. אם הערמה ריקה הערך השמור בשדה הוא null.

**סיבוכיות-**

### getHead()

**תיאור-** הפונקציה מחזירה את האיבר הראשון ברשימה המקושרת של עצי הערמה.

**מימוש-** הפונקציה מחזירה את שדה head.

**סיבוכיות-**

### getTail()

**תיאור-** הפונקציה מחזירה את האיבר האחרון ברשימה המקושרת של עצי הערמה.

**מימוש-** הפונקציה מחזירה את שדה tail.

**סיבוכיות-**

### getTreeCount()

**תיאור-** הפונקציה מחזירה את מספר העצים בערמה.

**מימוש-** הפונקציה מחזירה את שדה treeCount.

**סיבוכיות-**

### getMarkedCount()

**תיאור-** הפונקציה מחזירה את מספר האיברים המסומנים בערמה.

**מימוש-** הפונקציה מחזירה את שדה markedCount.

**סיבוכיות-**

### meld(FibonacciHeap heap2)

**תיאור-** מיזוג ערימה נוספת heap2 עם הערימה הנוכחית.

**מימוש-** הפונקציה:

* אם הערמה הנוכחית ריקה- מעדכנת את ראש הרשימה המקושרת של השורשים בערמה להצביע לראש הרשימה המקושרת של השורשים של heap2 ואת min להצביע לאיבר המינימום של heap2.
* אחרת- מחברת את הזנב של הרשימה המקושרת של שורשי הערמה הנוכחית לראש הרשימה המקושרת של שורשי הערמה השנייה.
* אם heap2 לא ריקה:
  + מעדכנת את prev של ראש הרשימה המקושרת של heap2 להצביע לזנב של הרשימה המקושרת הנוכחית.
  + **עדכון מינימום-** משווה את המינימום של הערמה השנייה לערמה הנוכחית ומעדכנת את מצביע המינימום בהתאם.
* **עדכון זנב-** מעדכנת את הזנב של הרשימה המקושרת הנוכחית להיות הזנב של heap2.
* **עדכון גודל הערמה-** מעדכנת את גודל הערמה ב + גודל heap2.
* **עדכון כמות העצים בערמה-** מעדכנת את כמות העצים של הערמה ב + כמות העצים של heap2.
* **עדכון כמות המסומנים**- מעדכנת את כמות המסומנים של הערמה ב + כמות המסומנים ב-heap2.

**סיבוכיות**- כל הפעולות מבצעות בזמן קבוע.

### size()

**תיאור-** הפונקציה מחזירה את מספר האיברים בערמה.

**מימוש-** הפונקציה מחזירה את הערך השמור בשדה size של הערימה.

**סיבוכיות-**

### countersRep()

**תיאור-** הפונקציה מחזירה מערך מונים כך שבאינדקס 𝑖 מופיע מספר העצים בערימה שהסדר (rank) שלהם הוא i. כלומר, היא מחזירה מערך של integers כך שלכל אינדקס 𝑖 בין 0 עד הדרגה המקסימלית של עץ שקיימת בערימה, הערך שמוחזר במערך הוא מספר העצים שקיימים בערימה מסדר i. עבור ערימה ריקה יוחזר מערך ריק.

**מימוש-** הפונקציה:

* בודקת אם הערמה ריקה- אם כן מחזירה מערך ריק.
* בודקת מה הדרגה הכי גדולה במערך- מתחילה מהערך 0. עוברת על שורשי העצים ומעדכנת את הערך בהתאם לאם הוא גדול יותר מהדרגה המקסימלית הנוכחית.
* יוצרת מערך בגודל הדרגה המקסימלית+1 של int שמאותחלים כולם ב-0. כל אחד מהם יהיה counter לכמות העצים בדרגה ה-i.
* עוברת על שורשי העצים ומעדכנת +1 במקום ה-i כאשר i זו הדרגה של העץ הנוכחי. האיטרציה מתבצעת ככמות העצים בערמה.

**סיבוכיות**- במקרה הגרוע שבו בערמה יש n שורשים.

### delete(HeapNode x)

**תיאור-** מחיקת הצומת x מהערימה.

**מימוש-** אם x זו צומת אמיתית, הפונקציה:

* קוראת ל-decreaseKey על x עם d ששווה למפתח של המינימום + המפתח הנוכחי + 10 על מנת ש-x יהפוך לאיבר המינימלי בערמה. הפעולה מתבצעת ב-.
* קוראת ל-deleteMin – הפונקציה תמחק את x שכרגע הוא המפתח המינימלי, ותבצע קונסולידציה לעץ. הפעולה מתבצעת ב- במקרה הגרוע.

**סיבוכיות-**  שנובעת מהמקרה הגרוע של deleteMin.

### decreaseKey(HeapNode x, int d)

**תיאור-** ערכו של המפתח של הצומת x יופחת בערך 𝑑≥0

**מימוש-** אם x זו צומת אמיתית, הפונקציה:

* מפחיתה את המפתח של x ב-d.
* אם x הוא לא שורש - משווה את המפתח של x למפתח של ההורה שלו. אם המפתח של x קטן מהמפתח של ההורה שלו- קוראת לפונקציה הרקורסיבית cascadingCut על x וההורה שלו.
* אם x הוא שורש והמפתח שלו כעת קטן מהמפתח של הצומת המינימלית- מעדכנת את מצביע המינימום להצביע ל-x ומסיימת.

**סיבוכיות-**

### cascadingCut(HeapNode x, HeapNode y)

**תיאור-** הפונקציה מבצעת את פעולת cascading-cut.

**מימוש-** פונקציה רקורסיבית. הפונקציה:

* קוראת ל- cut(x,y) שרצה ב- על x וההורה שלו y.
* אם y הוא לא שורש
  + אם הסימון של y הוא false - מעדכנת את הסימון ל-true ומוסיפה 1 ל- markedCount
  + אחרת- קוראת רקורסיבית לעצמה על y וההורה של y

**סיבוכיות-** נובע ממספר הקריאות הרקורסיביות המקסימלי שמתרחש כאשר x הוא העלה העמוק ביותר בעץ וכל שושלת ההורים שלו מסומנים. מספר הקריאות לפונקציה יהיה לכן .

### cut(HeapNode x, HeapNode y)

**תיאור**- הפונקציה חותכת את הצומת x מההורה שלה y.

**מימוש-** הפונקציה:

* מוסיפה 1 ל- cutsCount.
* **מעדכנת את x להיות שורש**:
  + מוסיפה 1 ל- treeCount.
  + מאפסת את הסימון של x ומחסירה 1 מ- markedCount.
  + מדליקה את דגל השורש של x.
  + ומעדכנת את מצביע האב של x ל-null.
  + משווה את המפתח של x למפתח של איבר המינימום ואם הוא יותר קטן מעדכנת את המינימום להצביע ל-x.
* **מעדכנים את הילדים של y**:
  + אם היה x הבן היחיד של y – מעדכנת את מצביע הילד להצביע ל-null.
  + אחרת-
    - אם מצביע child של y הצביע על x- מעדכנת אותו להצביע לאח הימני של x.
    - אחרת- מחברת בין האח השמאלי לאח הימני של x ולהפך אם קיים אח ימני.
* מעדכנת את הדרגה של y להיות פחות 1 אחרי ש-x כבר לא הבן שלו
* מוסיפה את x לראש רשימת שורשי העצים בערמה:
  + מעדכנת את next של x להיות ראש הרשימה המקושרת של שורשי העצים בערמה, ואת prev של ראש הרשימה הנוכחי להצביע ל-x.
  + מעדכנת את האח השמאלי prev של x להיות null.
* מעדכנת את מצביע ראש הרשימה המקושרת של שורשי הערמה להצביע ל-x.

**סיבוכיות-**

### potential()

**תיאור-** הפונקציה מחזירה את ערך הפוטנציאל הנוכחי של הערימה. הפוטנציאל, הינו Potential = #trees + 2\*#marked

**מימוש-** הפונקציה מחשבת את הפוטנציאל באמצעות שדה כמות העצים ושדה מספר הצמתים המסומנים ומחזירה את הערך.

**סיבוכיות-**

### totalLinks()

**תיאור-** פונקציה סטטית זו מחזירה את מספר כל פעולות החיבור שבוצעו מתחילת ריצת התוכנית. פעולת חיבור היא הפעולה שמקבלת שני עצים מאותו סדר ומחברת אותם.

**מימוש-** הפונקציה מחזירה את הערך השמור בשדה הסטטי linkedCount.

**סיבוכיות-**

### totalCuts()

**תיאור-** פונקציה סטטית זו מחזירה את מספר כל פעולות החיתוך שבוצעו מתחילת ריצת התוכנית. פעולת חיתוך מתרחשת עקב decreaseKey כאשר מנתקים תת -עץ מאביו.

**מימוש**- הפונקציה מחזירה את הערך השמור בשדה הסטטי cutsCount.

**סיבוכיות**-

### kMin(FibonacciHeap H, int k)

**תיאור-** פונקציה סטטית זו מקבלת ערימה H שהיא עץ (יער של עץ יחיד) שדרגתו deg(𝐻), ומספר חיובי 𝑘<𝑠𝑖𝑧𝑒(𝐻) . הפונקציה מחזירה מערך ממוין של 𝑘 הצמתים הקטנים ב - 𝐻.

**מימוש-** הפונקציה:

* אם k=0 או שהערמה ריקה- מחזירה מערך ריק.
* אחרת- יוצרת ערמת עזר חדשה Hk שתכיל את האיברים החשודים להיות האיבר המינימלי הבא.
* יוצרת מערך בגודל k של HeapNodes.
* מחזיקה את המצביעים הבאים:
  + currMin- מצביע לצומת עם האיבר המינימלי הבא ב-H. זה האיבר הבא שצריך להכניס למערך k המינימליים.
  + minChild- מצביע לילד של currMin.
  + temp- מצביע לאיבר האחרון שהוכנס לערמת העזר.
* מוסיפה את מפתח המינימום של H לערמת העזר, ומעדכנת את שדה kMin של הצומת החדשה שהוסיפה להצביע לצומת המקבילה (עם אותו המפתח) ב-H.
* מבצעת באיטרציה k פעמים את הפעולות הבאות עד שהמערך מלא:
  + מעדכנת את מצביע currMin להצביע לשדה kMin של צומת המינימום הנוכחי בערמת העזר.
  + מוסיפה את המפתח של currMin למקום הבא במערך.
  + מפעילה על Hk את deleteMin. הערמה תהפוך לערמה בינומית תקינה באמצעות קונסולידציה.
  + מגדירה את minChild להצביע לילד של currMin.
  + מוסיפה את הילדים של currMin שכרגע נוסף למערך ל-Hk ומעדכנת את שדה kMin שלהם להצביע לצומת המקבילה בערמה המקורית. נניח ש-H זו ערמה בינומית תקינה ואז כמות הילדים של הצומת תהיה לכל היותר הדרגה של H מהתכונה שהדרגות של הילדים של צומת x הן כל הדרגות מ-0 עד deg(x)-1.
* מחזירה את המערך.

**סיבוכיות**- .

כל הפעולות עד הלולאה לוקחת זמן קבוע. הלולאה מתבצעת פעמים עד שהמערך מלא. נשים לב שבכל איטרציה מתווספים לערמה לכל היותר deg(H)-1 שורשים מהתכונה שכמות הילדים של צומת חסומה בדרגה של הצומת, והדרגה של הצומת היא לכל היותר הדרגה של הערמה. לכן גם הקונסולידציה חסומה ב- שזו כמות העצים שיש בעץ לכל היותר בקריאה אליה. הקונסולידציה תעדכן בכל איטרציה את כמות העצים להיות המינימלית כך שבסיומה כמות העצים בערמה תהיה חסומה ב-, כך שבאיטרציה הבאה הקונסולידציה תתבצע על לכל היותר פעולות successive linking. סה"כ הפונקציה מבצעת בכל איטרציה את פעולת deleteMin שתיקח כאמור ותבצע לכל היותר הכנסות של איברים לערמת העזר ולכן כל איטרציה מבצעת פעולות. סה"כ נקבל סיבוכיות של .

## מחלקת HeapNode

### שדות המחלקה

המחלקה כוללת את השדות הבאים:

* info- הערך השמור בצומת
* key- מפתח הצומת
* rank- הדרגה של הצומת, מספר הבנים שלה
* mark- דגל לסימון הצומת ב-cascading cut
* child- מצביע לילד של הצומת
* next- מצביע לאח של הצומת
* prev- מצביע לאח הקודם של הצומת
* parent- מצביע להורה של הצומת
* isRoot- דגל לסימון אם הצומת היא שורש
* kMin- מצביע לצומת המקבילה בערמה שרוצים להחזיר את מערך k האיברים המינימליים שלה. הצומת המקבילה זו הצומת בעלת אותו המפתח. בשימוש בפונקציית kMin של הערמה.

### getKey()

**תיאור**- הפונקציה מחזירה את מפתח הצומת.

**מימוש**- הפונקציה מחזירה את שדה key של הצומת.

**סיבוכיות**-

### setKey(int k)

**תיאור**- הפונקציה מגדירה את מפתח הצומת להיות שווה ל-k.

**מימוש**- הפונקציה מעדכנת את שדה key להיות שווה ל-k.

**סיבוכיות**-

### getValue()

**תיאור**- הפונקציה מחזירה את ערך הצומת.

**מימוש**- הפונקציה מחזירה את שדה info של הצומת.

**סיבוכיות**-

### getRank()

**תיאור**- הפונקציה מחזירה את הדרגה של הצומת.

**מימוש**- הפונקציה מחזירה את שדה rank של הצומת.

**סיבוכיות**-

### setRank(int r)

**תיאור**- הפונקציה מגזירה את דרגת הצומת להיות שווה ל-r.

**מימוש**- הפונקציה מעדכנת את שדה rank של הצומת להיות שווה ל-r.

**סיבוכיות**-

### getMarked()

**תיאור**- הפונקציה מחזירה את הסימון הנוכחי של הצומת.

**מימוש**- הפונקציה מחזירה את שדה mark של הצומת.

**סיבוכיות**-

### setMark(boolean b)

**תיאור**- הפונקציה מגדירה את הסימון של הצומת להיות b.

**מימוש**- הפונקציה מעדכנת את שדה mark להיות שווה ל-b.

**סיבוכיות**-

### getChild()

**תיאור**- הפונקציה מחזירה את הילד של הצומת.

**מימוש**- הפונקציה מחזירה את הצומת השמורה במצביע child של הצומת.

**סיבוכיות**-

### setChild(HeapNode x)

**תיאור**- הפונקציה מגדירה את x להיות הילד של הצומת.

**מימוש**- הפונקציה מעדכנת את שדה child להצביע ל-x.

**סיבוכיות**-

### getNext()

**תיאור**- הפונקציה מחזירה את האח הבא של הצומת.

**מימוש**- הפונקציה מחזירה את שדה next של הצומת.

**סיבוכיות**-

### setNext(HeapNode x)

**תיאור**- הפונקציה מגדירה את x להיות אח של הצומת.

**מימוש**- הפונקציה מעדכנת את מצביע next להצביע ל-x.

**סיבוכיות**-

### hasNext()

**תיאור**- הפונקציה מחזירה true אם האיבר הבא שונה מ-null.

**מימוש**- הפונקציה מחזירה את תוצאת this.next != null.

**סיבוכיות**-

### getPrev()

**תיאור**- הפונקציה מחזירה את האח הקודם של הצומת.

**מימוש**- הפונקציה מחזירה את שדה prev של הצומת.

**סיבוכיות**-

### setPrev(HeapNode x)

**תיאור**- הפונקציה מגדירה את x להיות האח הקודם של הצומת.

**מימוש**- הפונקציה מעדכנת את שדה prev להצביע ל-x.

**סיבוכיות**-

### getParent()

**תיאור**- הפונקציה מחזירה את ההורה של הצומת.

**מימוש**- הפונקציה מחזירה את שדה parent של הצומת.

**סיבוכיות**-

### setParent(HeapNode p)

**תיאור**- הפונקציה מגדירה את ההורה של הצומת להיות p.

**מימוש**- הפונקציה מעדכנת את המצביע בשדה parent של הצומת להצביע ל- p.

**סיבוכיות**-

### getIsRoot()

**תיאור**- הפונקציה מחזירה האם הצומת היא שורש או לא.

**מימוש**- הפונקציה מחזירה את שדה isRoot של הצומת.

**סיבוכיות**-

### setIsRoot(boolean b)

**תיאור**- הפונקציה מגדירה את הצומת כשורש או שלא לפי b.

**מימוש**- הפונקציה מעדכנת את שדה isRoot להיות שווה ל-b.

**סיבוכיות**-

### getkMin()

**תיאור**- הפונקציה מחזירה את המצביע לצומת המקבילה בעלת אותו המפתח בערמה המקורית ממנה רוצים להחזיר מערך של k האברים המינימליים. בשימוש בפונקציית kMin של מחלקת הערמה.

**מימוש**- הפונקציה מחזירה את שדה kMin של הצומת.

**סיבוכיות**-

### setkMin(HeapNode node)

**תיאור**- הפונקציה מגדירה את המצביע kMin להצביע ל-node. node זו הצומת המקבילה בעלת אותו המפתח בערמה המקורית ממנה רוצים להחזיר מערך של k האברים המינימליים. בשימוש בפונקציית kMin של מחלקת הערמה.

**מימוש**- הפונקציה מעדכנת את המצביע בשדה kMin של הצומת להצביע ל- node.

**סיבוכיות**-

# חלק ב- ניסויי תיאורטי

## שאלה 1

### סעיף א

ננתח את סיבוכיות זמן הריצה של סדרת הפעולות הנתונה:

* ראשית יש לנו פעולות הכנסה, כאשר עלות כל פעולה כזו היא . סה"כ .
* לאחר מכן מבצעים פעולת מחיקת האיבר המינימלי שהעלות שלה תלויה בעלות של פעולת הקונסולידציה. מקרה זה הוא הגרוע ביותר בו לאחר מחיקת איבר המינימום הערמה כוללת m שורשים מדרגה 0, ולכן כפי שראינו בהרצאה הפעולה תתבצע ב- .
* לאחר מכן מבצעים פעולות decreaseKey שעלות הפעולה שלה במקרה הגרוע היא (בפועל במקרה זה העלות נמוכה יותר, אבל סיבוכיות הפעולות האחרות ברצף שלושת הפעולות גרועה יותר ולכן אסימפטוטית בניתוח הכולל זה לא משנה. נרחיב על מספר הפעולות בפועל בסעיפים הבאים). סה"כ .

בסך הכל נקבל כי עלות רצף הפעולות תהיה .

### סעיף ב

הטבלה שהתקבלה:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Potential** | **totalCuts** | **totalLinks** | **Run-Time (ms)** | **m** |
| 29 | 10 | 1023 | 1 |  |
| 44 | 15 | 32767 | 11 |  |
| 59 | 20 | 1048575 | 55 |  |
| 74 | 25 | 33554431 | 8613 |  |

### סעיף ג

ננתח את כמות פעולות ה-link, פעולות ה-cut ואת הפוטנציאל של המבנה בסוף סדרת הפעולות.

**פעולות ה-link**: פעולות הlink מתרחשות בשלב ה- successive linking כאשר האלגוריתם מחבר בכל פעם שני תתי עץ באותה הדרגה. בשלב ההכנסה, כל העצים שנחבר יהיו מדרגה 0 וכמות העצים השונים תהא שהוא חזקה שלמה של 2. אם נסתכל על חתך רוחבי של האלגוריתם הוא בפועל יחבר כל זוג מהצורה כלומר יוסיף חיבורים. בנוסף, האלגוריתם יחבר בין עצים בדרגה 1, כך שהשורשים שיתחברו הם , כלומר יוסיף עוד חיבורים. כך הלאה במהלך עד שיתבצע החיבור האחרון בין שני עצים מדרגה . בסה"כ יהיו חיבורים בין העצים מדרגה , כלומר:

כלומר, זה מספר פעולות ה-link שיתבצעו בסדרה.

נשים לב שבטבלה כמות החיבורים תואמת לגודל זה בכל אחת מהריצות.

**פעולות ה-cut:** באה לידי ביטוי כחלק מפעולת decrease key במידה והמפתח שהקטנו כעת קטן יותר מהמפתח של ההורה שלו. ננתח באילו מהצמתים באיטרציה זה המצב ומתבצע בהן cut, ואם מתבצע גם cascading cuts שיכולה להוסיף חיתוכים נוספים לשקלול. לאחר שהכנסנו לעץ את כל הצמתים אנו מבצעים פעולת מחיקת מינימום אשר משאירה את הערימה עם בדיוק m צמתים, שזו כאמור חזקה של 2. הערמה שתתקבל תכיל עץ בינומי מלא יחיד מדרגה . לאחר מכן תתבצע האיטרציה שמקטינה את המפתח של הצומת ה- כאשר .

תחילה, נשים לב כי המפתח הגדול ביותר בערמה הוא המפתח ה-. לכן אם נחסיר מצומת שהיא לא הצומת המינימלית ושלא התבצעה decrease key על ההורה שלה לפניה, יתבצע בהכרח cut. אחרת, המבנה של הערמה לאחר הפעולה נשמר, ההורים גדולים מהילדים שלהם ולכן לא יתבצע cut.

נתבונן בצמתים שהמפתח שלהם הוא . נשים לב למספר מאפיינים של הערמה ומתוך כך נבין באילו צמתים מדובר:

* מכיוון שהצמתים הוכנסו לערמה בסדר יורד, נקבל כי לפני פעולת delete min הצמתים מסודרים בסדרה עולה משמאל לימין. פעולת ה- successive linking עוברת באיטרציה על הצמתים מהקטן לגדול, וראינו כי מתוך כך ההורה של כל עלה בעץ יהיה המפתח הקודם לו , כלומר המפתח של הורה של עלה תמיד קטן ממנו ב-1.
* כפי שהסברנו למעלה, מכיוון שפעולת successive linking עוברת באיטרציה על הצמתים מהקטן לגדול, תתי-העצים הנוצרים מכילים מספרים עוקבים. לכן באיחוד של שני תתי-עצים מדרגה k-1 לעץ בדרגה k, נקבל כי העץ בעל המפתח הקטן יותר מבין השניים מכיל מפתחות שקטנים ממש משורש העץ בעל המפתח הגדול יותר. מכלל הערמה הם גם קטנים ממש מכל המפתחות בעץ שלו. מהאופן שבו אנו מבצעים חיבורים, תת העץ בעל המפתח הגדול יותר יהיה הבן השמאלי ביותר של המפתח הקטן יותר. מכיוון שבתת-עץ מדרגה k-1 יש צמתים, מפתח השורש של תת-העץ השמאלי יהיה גדול ב- מהמפתח של ההורה החדש שלו. כלומר המפתחות של השורשים של תתי-העצים של הבנים השמאליים בערמה הם

. לכן הביטוי הכללי המתקבל עבור שורש של תת-עץ שמאלי בדרגה k יהיה

כלומר, הצמתים שהמפתח שלהם הם שורשי תתי עצים שהם הבן השמאלי ביותר של ההורה שלהם בכל רמה בעץ הבינומי.

משני מאפיינים אלו נסיק כי הצומת בעלת המפתח ה- תהיה הבן מדרגה 0 של הצומת ה- שהיא הבן השמאלי ביותר של ההורה שלו.

כלומר, פעולת decrease key מתבצעת על עלים שההורים שלהם שונים מכיוון שכל אחד בעל דרגה שונה. מכיוון שתמיד מדובר בעלים, בשום שלב לא הפחתנו להורים שלהם את המפתח ולכן נקבל כי המפתח של אותו עלה קטן מההורה שלו. לכן, בכל קריאה ל-decrease key תתבצע פעולת cut אחת בלבד לאותו עלה. בנוסף, לכל הורה נחתך בן יחיד ולכן לא תתבצע פעולת cascading cuts, וכל הורה מלבד שורש העץ יסומנו. מניתוח זה נקבל כי סיבוכיות decrease key במקרה זה היא בעצם מכיוון שלא התבצע בשום שלב cascading cuts שעלותה במקרה הגרוע היא והבצעה רק פעולת עדכון המפתח ו-cut שעלותן קבועה.

הפעולה מתבצעת באיטרציה פעמים ולכן יתבצעו בסדרת הפעולות פעולות cut.

נשים לב שגם בטבלה כמות החיתוכים תואמת לחזקה של m.

**פוטנציאל:** ראינו בהרצאה כי פוטנציאל הערמה שווה ל -

ננתח מהי כמות העצים וכמות הצמתים המסומנים בערמה.

כמות העצים- ראינו כי לאחר פעולת delete min אנו מקבלים ערמה שמכילה עץ בינומי מלא יחיד מדרגה . לאחר מכן ראינו כי מתבצעים חיתוכים בפעולת decrease key . כל חיתוך הופך לעץ נוסף בערמה ולכן לאחר כל פעולות decrease key נקבל שנוספו לערמה עצים. יחד עם השורש נקבל כי יש בערמה עצים בערמה לאחר כל שרשרת הפעולות.

כמות הצמתים המסומנים- כאמור, כל צומת נחתכת מהורה אחר שלאחר החיתוך מסומן, מלבד השורש. נשים לב שמכך שהצמתים שנחתכים הם צמתים שהם הבן הימני ביותר של ההורה שלו, גם הבן הימני ביותר של שורש העץ יחתך. כפי שראינו בניתוח לעיל, מדובר בצומת שהמפתח שלה הוא 1. לכן, כמות הצמתים המסומנים תהיה ככמות החיתוכים שבוצעו פחות החיתוך של הצומת 1 שההורה שלה לא יסומן. סך הכל נקבל כי יש בערמה צמתים מסומנים לאחר כל שרשרת הפעולות.

אם נציב בנוסחת הפוטנציאל נקבל כי הוא שווה ל –

ונשים לב שגם במקרה הזה התוצאות שקיבלנו בטבלה תואמות לנוסחה זו.

### סעיף ד

כל הפעולות על decrease key מתבצעות בדומה לסעיף ג ולכן נקבל בסוף פעולת delete min עץ בינומי מלא מדרגה כפי שראינו.

**פעולות link**: ראינו כי פעולת link מתבצעת רק כחלק מפעולת insert שבמקרה הזה לא השתנתה, ולכן נקבל שיתבצעו פעולות link.

**פעולות cut:** כפי שראינו בניתוח בסעיף ג, האיברים ה- עבור הם הבן השמאלי ביותר מבין הבנים ברמה ה-i – השורש הוא הרמה ה- ולאחר מכן יורדים בעץ בכל פעם לבן השמאלי ביותר שהוא גם מדרגה i עד שמגיעים לצומת מדרגה 1. כלומר, במקרה בו decrease key תתבצע על המפתחות היא תתבצע בפועל על שורש העץ, לאחר מכן על הבן השמאלי של שורש העץ, לאחר מכן על הבן השמאלי שלו וכן הלאה עד הבן השמאלי האחרון שיהיה מדרגה 1.

ראינו בסעיף הקודם שאם מתבצע decrease key על מפתח שהוא המפתח המינימלי בערמה (במקרה הזה הוא שורש העץ) כללי הערמה ישמרו כי הוא יישאר המינימלי. בנוסף, אם תתבצע decrease key על מפתח שההורה שלו הופחת לפניו ב- אז כללי הערמה גם ישמרו ובעצם לאחר ההפחתה של מפתח הצומת הנוכחית ההפרש בין מפתח ההורה שלו והמפתח שלו יחזור להיות כפי שהיה לפני שהתחילו פעולות decrease key כי שניהם הופחתו באותו ערך. מכיוון שבמקרה הזה אנחנו בכל פעם מבצעים decrease key על השורש, ולאחר מכן על הבן השמאלי ולאחר מכן על הבן השמאלי שלו וכן הלאה, נקבל כי תמיד תתבצע הפחתה של מפתח ההורה לפני הילדים שלו וכללי הערמה ישמרו. כלומר, לא יתבצעו פעולות cut בכלל כי לא הופר המבנה של הערמה.

**פוטנציאל**: כאמור, הפוטנציאל תלוי במספר העצים ובכמות הצמתים המסומנים בערמה. נשים לב שמכיוון שלא התבצעו פעולות cut אז מספר העצים לא גדל וכן גם לא סומנו צמתים כלל. מכיוון שיש עץ יחיד בערמה נקבל כי

כלומר, במקרה הזה הוא קבוע.

### סעיף ה

במקרה זה לא מתבצעת פעולת delete min ולכן:

* כמות הצמתים בערמה נשארת
* הערמה לא עוברת קונסולידציה ולכן המבנה שלה נשאר שרשרת צמתים מדרגה 0 שמסודרים משמאל לימין מהקטן לגדול.

**פעולות link:** במקרה זה לא התבצעה קונסולידציה שבה מתבצעת פעולת ה-link. לכן כמות החיבורים תהיה 0.

**פעולות cut:** במקרה זה לא התבצעו חיבורים, כל הצמתים בערמה מדרגה 0 ובפרט שורשים. לכן לא יתבצעו פעולות cut, כל מפתח יופחת ב- ומיקומו בערמה לא ישתנה ולא יתבצע שום סימון.

**פוטנציאל:** במקרה הזה כמות העצים בסוף סדרת הפעולות תהיה ככמות הצמתים שהכנסנו לערמה. כאמור לא בוצעו פעולות חיתוך ולכן לא סומנו צמתים כלל. סך הכל נקבל כי

### סעיף ו

במקרה זה שלושת השורות הראשונות מבצעות כפי שמתואר בסעיף ג. לכן לפני שמבוצעת פעולת ה-decrease key הנוספת:

* כמות החיבורית שבוצעו היא
* כמות החיתוכים שבוצעו היא
* כמות העצים בערמה היא העץ המקורי והעצים שנחתכו ב-decrease key ולכן היא
* כמות הצמתים המסומנים היא כל צמתי מלבד השורש ולכן היא

**פעולות link**: לא קראנו לפעולת delete min ולכן לא התבצעה עוד קונסולידציה שמבצעת חיבורים נוספים. כמות החיבורים נותרה .

**פעולות cut**: בלולאת decrease key חתכנו את הבנים מדרגה 0 של מפתחות שהם כאמור השורש, הבן השמאלי שלו, הבן השמאלי שלו וכן הלאה עד שהגענו לבן שמאלי שהדרגה שלו 1. כלומר, כעת קיבלנו שושלת שמתחילה בבן השמאלי של השורש ועד הבן השמאלי ביותר מדרגה 1 של צמתים מסומנים מכיוון שלכל אחד מהם נחתך הבן מדרגה 0. כמו כן, לא התבצע decrease key על אף צומת בשושלת הזו.

כעת נקרא ל-decrease key הנוסף. נשים לב שהוא מתבצע על המפתח ה- שהוא מהצורה כאשר , כלומר הוא הבן השמאלי האחרון ממנו מתבצע חיתוך, ולכן הוא חלק משושלת הצמתים המסומנים. נשים לב שמפתח ההורה שלו לא שונה במהלך הפעולות הקודמות ולכן הפחתת המפתח שלו ב- תגרום לכך שהמפתח שלו יהיה קטן מההורה שלו ותתבצע פעולת cut. מכיוון שכל שושלת ההורים שלו עד השורש מסומנים תקרא גם פעולת cascading cuts שתחתוך את כל ההורים בשושלת חוץ מאת שורש העץ. סך הכל יחתכו מהעץ כל הצמתים שהיו מסומנים ולכן יתבצעו בפעולה זו חיתוכים.

סך הכל נקבל כי בוצעו חיתוכים.

**פוטנציאל**: במקרה זה כמות העצים בסוף סדרת הפעולות תהיה כמות העצים שהייתה לפני שורה 4 כפי שכתוב לעיל וכמות העצים שנוספו לאחר הפעולה שכאמור אלו כל הצמתים שהיו מסומנים בעץ. סך הכל יהיו בערמה .

כמות הצמתים המסומנים כאמור תתאפס מכיוון שכולם נחתכו מהעץ המקורי בפעולת cascading cuts.

לכן נקבל כי

**עלות פעולת decreaseKey היקרה ביותר**: כפי שראינו בניתוחים הנ"ל, המקרה הגרוע של פעולת decrease key מתרחש בשורה #4. במקרה זה קיימת שושלת באורך הורים מסומנים וכאשר הופחת המפתח של הצומת שבקצה השושלת הזו זה גרר שרשרת של פעולות cut לכל השושלת עד השורש (לא כולל). כלומר, במקרה זה פעולת decrease key אחת גררה פעולות cut, כל פעולת cut כאמור מתבצעת בזמן קבוע ולכן עלות הפעולה תהיה .

### טבלה מסכמת לסעיפים ג-ו

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **decreaseKey**  **max cost** | **Potential** | **totalCuts** | **totalLinks** | **Case** |
| (skip) |  |  |  | (c) original |
| (skip) |  |  |  | (d) decKey( |
| (skip) |  |  |  | (e) remove line #2 |
|  |  |  |  | (f) added line #4 |

## שאלה 2

### סעיף א

הטבלה שהתקבלה:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Potential** | **totalCuts** | **totalLinks** | **Run-Time (ms)** | **m** |
| 6 | 0 | 723 | 4 | 728 |
| 6 | 0 | 6555 | 8 | 6560 |
| 9 | 0 | 59040 | 36 | 59048 |
| 10 | 0 | 531431 | 125 | 531440 |
| 14 | 0 | 4782955 | 1956 | 4782968 |

### סעיף ב

ננתח את זמן הריצה האסימפטוטי של סדרת הפעולות כפונקציה של m.

תחילה מתבצעות פעולות insert. כפי שראינו סיבוכיות כל פעולה היא . לכן סיבוכיות סדרת ההכנסות תהיה סך הכל .

לאחר מכן מתבצעות פעולות delete min. נשים לב כי בפעולת המחיקה הראשונה, כל האיברים בערמה הם עצים בדרגה 0. זהו המקרה הגרוע של delete-min ולכן סיבוכיות זמן הריצה של פעולה זו כפי שראינו בהרצאה היא . לאחר מכן מתקבל ערמה בינומית תקינה. מרגע זה הערמה תשאר ערמה בינומית רגילה (ולא עצלה או פיבונאצ'י) מכיוון שמתבצעות רק פעולות delete min שבכל קריאה אליהן מתבצעת קונסולידציה לערמה שבסופה מתקבל עץ בינומי תקין, וכן לא מתבצעות פעולות decrease key שעלולות לעוות את המבנה של העצים הבינומיים במידה ומתבצע בהן cuts.

בנוסף, נשים לב למבנה של הערמה שהתקבלה שמשפיע על סיבוכיות פעולות delete min המתבצעות לאחר הפעולה הראשונה:

* נשים לב כי סדר הכנסת האיברים לערמה הוא מהקטן לגדול ומכיוון שכל צומת מתווספת לערמה משמאל נקבל כי האיברים מסודרים ברשימה יורדת כך שהשורש השמאלי ביותר הוא m והימני ביותר הוא 0.
* נשים לב בנוסף כי במקרה הזה לאחר מחיקת האיבר המינימלי הראשון כמות הצמתים בערמה היא לא בהכרח חזקה שלמה של 2. נזכר כי הערמה תכיל עצים בדרגות ה-i שבייצוג הבינארי של m מכילים אחדות במקומות ה-i. בנוסף, המקום ה- בייצוג הבינארי יהיה בוודאות 1, כלומר בהכרח יהיה עץ מדרגה לאחר פעולת הקונסולידציה הראשונה. העצים האחרים שתכיל הערמה יקבעו כאמור בהתאם למקומות הדלוקים בייצוג הבינארי של m.
* בפעולת delete min הראשונה שמתבצעת, פעולת הקונסולידציה רצה מהשורש ה-m לשורש ה-1. נשים לב כי בדומה למה שראינו בשאלה 1, נקבל כי כל פעולת link שתתבצע תתבצע בין שני עצים שהמפתחות באחד מהם גדולים ממש מהמפתחות בעץ השני שגם יהיה השורש החדש לאחר ה-link (למרות שבמקרה הזה אנו רצים מהגדול לקטן בניגוד לשאלה 1, פעולה ה-link דואגת שהמפתח הקטן יותר יהיה השורש ולכן זה מתקיים גם במקרה הזה).
* בנוסף נשים לב כי פעולת הקונסולידציה היא טורית כך שקודם מורכב העץ בדרגה הכי גדולה, לאחר מכן העץ בדרגה שבאה אחריו וכן הלאה. זאת מכיוון שבכל הוספה של צומת ל"דליים" זה יכול לגרום לשרשרת חיבורים שיכולה רק ליצור עץ גדול יותר. לכן בפעולת הקונסולידציה הראשונה שמתבצעת קודם יתמלא העץ הבינומי מדרגה , ולאחר מכן המקום ה-i הבא בגודלו שדולק בייצוג הבינארי של m, וכן הלאה.

מכל הנקודות הנ"ל נקבל כי העץ בעל הדרגה הגדולה יכיל את המפתחות הכי גדולים, העץ בעל הדרגה הבאה בגודלה יכיל את המפתחות הבאים בסדרה החל מהאיבר ה-, וכן הלאה. כלומר, איבר המינימום יהיה בהכרח בעץ בעל הדרגה הקטנה ביותר בערמה.

לכן, כל פעולת delete min שנבצע לאחר פעולת delete min הראשונה תתבצע על העץ בערמה שהוא הקטן ביותר בדרגתו. נסמן את הדרגה של העץ ששורשו הצומת המינימלית ב-k. מחיקה של הצומת המינימלית תוסיף לרשימת שורשי העצים את הילדים שלה שראינו בהרצאה שהדרגות שלהם בעץ בינומי תקין הן כל הדרגות מ-0 עד k-1. מכיוון ש-k היא הדרגה הכי קטנה בערמה, לא קיימים בה עצים מהדרגות 0 עד k-1 ולכן לאחר המחיקה אין כפילויות של עצים מאותה הדרגה, לא יתבצעו פעולות link ובזה תסתיים פעולת delete min.

כפי שראינו בניתוח בשאלה 1, האיברים בכל עץ מסודרים באופן כזה שהבן הימני ביותר הוא הכי קטן (בפועל הוא עלה שגדול ב-1 מהשורש שלו), וככל שמתקדמים שמאלה המפתחות הולכים וגדלים עד שהמפתח הגדול ביותר הוא הבן השמאלי. מכך ומכך שבעת פעולת הקונסולידציה הראשונה כל פעולת link חיברה בין שני עצים שמפתח השורש הגדול מבניהם גדול ממש מכל המפתחות בעץ ששורשו המפתח הקטן, נקבל כי בכל פעולת delete min שנבצע ימחק השורש של העץ בעל הדרגה הקטנה ביותר בערמה – תחילה הצומת שהייתה הבן הימני, לאחר מכן האח השמאלי שלה, הבן שלו וכן הלאה עד שתגמר ריצת התוכנית או שימחקו כל הצמתים שהיו בעבר בעץ מדרגה k. אם התוכנית תמשיך למחוק איברים מינימליים היא תעבור לעץ הבא בעל הדרגה המינימלית בערמה וכל התהליך יחזור חלילה. לכן כפי שראינו, בכל מחיקות המינימום האלו לא יתבצעו פעולות link, בכל מחיקה לא מתווספים לערמה צמתים שדרגתם כבר קיימת בה.

מכל זה נסכם:

* עלות פעולת delete min הראשונה היא כפי שראינו בהרצאה
* עלות כל פעולות delete min שמתבצעות לאחר מכן תהיה מכיוון שכל מה שיתבצע זה פעולות שלוקחות זמן קבוע. סך הכל מתבצעות פעולות כאלו, ולכן העלות שלהן תהיה .

סך הכל נקבל

### סעיף ג

נראה כמה פעולות link ופעולות cut מתבצעות במהלך סדרת הפעולות, ומהו הפוטנציאל של המבנה בסוף הסדרה.

**פעולות link:** כפי במפורט בניתוח הסיבוכיות בסעיף הקודם, בפעולת delete min הראשונה על הערמה יתבצעו פעולות link שבונות את העצים הבינומיים בה. לאחר מכן כל פעולות delete min לא יבצעו פעולות link נוספות (מפורט בסעיף הקודם למה).

ננתח כמה פעולות link מתבצעות בעת בניית הערמה הבינומית התקינה בקונסולידציה הראשונה. נשים לב כי כאמור לאחר הקונסולידציה הערמה מכילה עצים בכל הדרגות ה-i שהמקום ה-i בייצוג הבינארי של m דולק בהם. בשאלה 1 ראינו שכאשר העץ הוא עץ בינומי מלא שלצורך הניתוח מכיל k צמתים, יתבצעו חיבורים בקונסולידציה הראשונה. אם בערמה הסופית שנקבל יהיו עצים בינומיים מדרגות כמות החיבורים שיתבצעו תהיה:

כלומר, נקבל כי כמות החיבורים שתעשה היא m פחות כמות הביטים הדולקים בייצוג הבינארי של m שבמקרה הזה סימנו אותה ב-n. בפועל מדובר ב-m פחות כמות העצים בערמה לאחר פעולת הקונסולידציה הראשונה.

נשים לב שבמקרה הפרטי שהרעמה מכילה עץ בודד, זה גם מתקיים כפי שראינו בשאלה 1. בנוסף, אם נבדוק את הביטוי הכללי על הקלטים שהרצנו בסעיף א נקבל שזה אכן מתקיים:

|  |  |  |
| --- | --- | --- |
| **totalLinks** | **treeCount after first deleteMin** | **m** |
| 723 | 5 | 728 |
| 6555 | 5 | 6560 |
| 59040 | 8 | 59048 |
| 531431 | 9 | 531440 |
| 4782955 | 13 | 4782968 |

**פעולות cut:** מכיוון שלא מתבצעת פעולת decrease key במהלך סדרת הפעולות, לא יתבצעו פעולות cut. לכן יתבצעו 0 פעולות cut.

**פוטנציאל:** כאמור, ראינו בהרצאה כי פוטנציאל הערמה שווה ל -

כמות הצמתים המסומנים- ראינו כי לא מבוצעות פעולות cut ולכן גם לא מתבצע סימון לשום צומת. 0 צמתים מסומנים.

כמות העצים- לאחר פעולת delete min הראשונה נותרו בערמה m צמתים שמסודרים בערמה בינומית תקינה כפי שפרטנו בסעיפים הקודמים. הערמה כאמור מכילה עצים בכל הדרגות ה-i שהמקום ה-i בייצוג הבינארי של כמות הצמתים דולק בהם. מכיוון שלא התבצעה פעולת decrease key ולא התבצעו חיתוכים אנו יודעים שהערמה מכילה רק עצים בינומיים תקינים.

בכל פעולת deleteMin גודל הערמה קטן ב-1, וכך גם משתנה המבנה שלה שיתאים לייצוג הבינארי של כמות הצמתים החדשה. בפועל ניתן להתייחס לייצוג הבינארי של כמות הצמתים בערמה כמונה שמייצג את מבנה הערמה, ובכל פעולת מחיקת מינימום הוא מחסיר 1 מהייצוג הבינארי. לכן לאחר פעולות מחיקה הוא החסיר מהייצוג הבינארי של m את הייצוג הבינארי של (לא כולל את פעולת delete min הראשונה). לכן כמות העצים שתהיה בערמה בסוף סדרת הפעולות תהיה מספר הביטים הדולקים בייצוג הבינארי של . נסמן את הביטוי של מספר הביטים הדולקים בייצוג הבינארי של באופן הבא:

ולכן נקבל כי הפוטנציאל יהיה שווה ל-