מבנ"ת- תרגיל מעשי 2.0

# חלק א- תיעוד פונקציות

## מחלקת FibonacciHeap

### שדות המחלקה

המחלקה כוללת את השדות הבאים:

* head- מצביע לשורש השמאלי ביותר ברשימת העצים
* tail- מצביע לשורש האחרון ברשימת העצים
* min- מצביע לאיבר המינימלי
* 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 התכנית
* cutsCount- שומר את כמות פעולות החיתוך שבוצעו מתחילת ריצת התכנית

### isEmpty()

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

**מימוש**- הרשימה שומרת מצביע לראש הרשימה. במידה והוא null מחזירה TRUE, אחרת FALSE.

**סיבוכיות**-

### insert(int i)

**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
* מוסיפה את הצמת לראש הערימה
* **עדכון שדה שורש-** מעדכנת את שדה isSource 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 הראשון של הרשימה.

**סיבוכיות-**

### deleteMin()

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

**מימוש**- הפונקציה:

* מוחקת את האיבר המינימלי ומשאירה את הבנים שלו יתומים
* קוראת ל- consolidate()
* **עדכון גודל הערמה-** מעדכנת את גודל הערמה במינוס 1
* מאפסת את כמות העצים בערמה.
* עוברת על שורשי העצים אחד אחד:
  + מחפשת את האיבר המינימלי ומעדכנת את מצביע המינימום להצביע אליו
  + **עדכון כמות העצים בערמה-** מעדכנת את כמות העצים ב + 1 בכל עץ שעוברים אליו.
  + **עדכון מינימום-** מתחילה מערך נוכחי ששווה ל-0. מעדכנת את המשתנה בכל פעם שהמפתח הנוכחי קטן יותר מהמפתח בצומת זו.

**סיבוכיות-** במקרה הגרוע

### consolidate()

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

**מימוש**- הפונקציה יוצרת מערך בגודל .

* *הפונקציה עוברת על הרשימה המקושרת של שורשי העצים בערמה. כל עוד קיים שורש:*
  + *בודקת מה הדרגה של השורש הנוכחי. אם הדרגה היא i היא מסתכלת על המקום ה-i במערך:*
    - *אם הוא null- מעדכנת את המצביע במקום ה-i במערך להצביע לשורש זה.*
    - *אם הוא לא null:*
      * *משווה בין מפתחות השורשים של שני העצים ומעדכנת את הגדול מבניהם להיות הבן השמאלי של השני (הקטן יותר)*
      * *מעדכנת את הדרגה של השורש*
      * *מעבירה את העץ החדש למקום המתאים לדרגה במערך ומשנה את המיקום הקודם ל-null*
      * *מעדכנת את המצביעים את הבנים להיות אחים*
      * ***מעדכנת את כמות החיבורים-*** *מוסיפה + 1 את* linkedCount
* *מעדכנת את המצביע של ה-head להצביע לאיבר הראשון במערך*
* *מעדכנת את המצביע של ה-tail להצביע לאיבר האחרון במערך*
* *עוברת באיטרציה על האיברים במערך ומעדכנת את המצביעים של השורשים*

***סיבוכיות-***  *במקרה הגרוע*

### findMin()

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

**מימוש-** החזרת מצביע המינימום

**סיבוכיות-**

### meld(FibonacciHeap heap2)

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

**מימוש-** הפונקציה:

* מחברת את האיבר האחרון של הערמה הנוכחית לראש הערמה השנייה
* **עדכון מינימום-** משווה את המינימום של הערמה השנייה לערמה הנוכחית ומעדכנת את מצביע המינימום בהתאם
* **עדכון גודל הערמה-** מעדכנת את גודל הערמה ב + גודל heap2
* **עדכון כמות העצים בערמה-** מעדכנת את כמות העצים ב + כמות העצים של heap2

**סיבוכיות**-

### size()

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

**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 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.

**סיבוכיות**- במקרה הגרוע

### delete(HeapNode x)

**תיאור-** מחיקת הצומת x מהערימה.

**מימוש-** הפונקציה:

* קוראת ל-decreaseKey על x עם d ששווה למפתח של המינימום פחות 1.
* קוראת ל-deleteMin – הפונקציה תמחק את x שכרגע הוא המפתח המינימלי, ותבצע קונסולידציה לעץ.

**סיבוכיות-**

### decreaseKey(HeapNode x, int d)

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

**מימוש-** הפונקציה:

* מפחיתה את המפתח של x ב-d.
* משווה את המפתח של x למפתח של ההורה שלו:
  + אם המפתח של x גדול מהמפתח של ההורה שלו- מסיימת
  + אחרת- קוראת ל-cascadingCut על x וההורה שלו

**סיבוכיות-**

### cascadingCut(HeapNode x, HeapNode y)

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

**מימוש-** פונקציה רקורסיבית. הפונקציה:

* קוראת ל- cut(x,y)
* אם y הוא לא שורש
  + אם הסימון של y הוא 0 - מעדכנת את הסימון ל-1 ומוסיפה 1 ל- markedCount
  + אחרת- קוראת רקורסיבית לעצמה על y וההורה של y

**סיבוכיות-**

### cut(HeapNode x, HeapNode y)

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

**מימוש-** הפונקציה:

* מוסיפה 1 ל- cutsCount
* מוסיפה 1 ל- treeCount
* מעדכנת את מצביע האב של x ל-null
* מאפסת את הסימון של x ומחסירה 1 מ- markedCount
* מדליקה את דגל השורש של x
* משווה את המפתח של x למפתח של איבר המינימום ואם הוא יותר קטן מעדכנת את המינימום להצביע ל-x.
* מעדכנת את הדרגה של y להיות פחות 1 אחרי ש-x כבר לא הבן שלו
* מעדכנים את הילדים של y:
  + אם היה x הבן היחיד של y – מעדכנים את מצביע הילד להצביע ל-null
  + אחרת- מעדכנים את מצביע הילד להצביע לאח של x

**סיבוכיות-**

### potential()

**תיאור-** הפונקציה מחזירה את ערך הפוטנציאל הנוכחי של הערימה. הפונטנציאל, 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:

* יוצרת ערמה חדשה Hk
* יוצרת מערך בגודל k
* מוסיפה את השורש של H למערך במקום ה-0 (זהו האיבר המינימלי ב-H)
* מבצעת באיטרציה את הפעולות הבאות עד שהמערך מלא:
  + מוסיפה את הילדים של האיבר שכרגע נוסף למערך ל-Hk (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)
  + מוסיפה את האיבר ממצביע המינימום למקום הבא במערך
  + מפעילה על Hk את deleteMin - הערמה תהפוך לערמה בינומית תקינה באמצעות קונסולידציה. נשים לב שבכל שלב יש בערמה לכל היותר deg(H)+1 עצים מהתכונה שכמות הילדים חסומה בדרגה של הצומת, והדרגה של הצומת היא לכל היותר הדרגה של הערמה. לכן גם הקונסולידציה חסומה ב- .

האיטרציה מתבצעת פעמים עד שהמערך מלא. כל איטרציה מבצעת פעולות.

* מחזירה את המערך.

**סיבוכיות**-

## 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

### 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**-

### getMark()

**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**-

### 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**-

# חלק ב- ניסויי תיאורטי