שאלה 1

א'

ע"פ מדריך הלמידה עמ' 34 לכל , :   
בפרט לכן קיימים , כך שלכל ,  
  
ע"פ אותו משפט, , לכן קיימים , כך שלכל ,  
  
מסקנה:   
נסמן   
, ,   
  
ע"פ אפשרות 1 של משפט שיטת האב,

ב'

לכן

נתבונן ב־, , ,

כלומר   
על פי שיטת האב המורחבת/מקרה 2 המורחב (עמ' 44 במדריך הלמידה), כלומר

ג'

נתבונן ב , , , , .

קיים כך שלכל , לכן

לכן

ע"פ אפשרות 3 של משפט שיטת האב, ולכן .

ד'

(האיטרציה חוזרת n פעמים)  
לכל , לכן לכן   
  
(נוסחה א.3 בנספח) לכן

ה'

נסמן   
  
נציב . ()  
  
נסמן .  
()  
ע"פ מקרה 3 של משפט שיטת האב עבור הערכים , , , ,  
  
(כי ועבור מתקיים  
 )

שאלה 2

א.  
נסמן - מספר השבבים התקינים, - מספר השבבים הפגומים.  
,   
נחלק מספרים סידוריים לכל השבבים, קודם התקינים ואז הפגומים. הפגומים יתנהגו כך:

|  |  |  |
| --- | --- | --- |
| שבב שבודק | שבב שנבדק | תשובה שתוחזר |
| אחד מ־ הפגומים הראשונים | אחד מ־ הפגומים הראשונים | תקין |
| אחד מ־ הפגומים הראשונים | אחד מ־ הפגומים האחרים | לא תקין |
| אחד מ־ הפגומים הראשונים | אחד מהתקינים | לא תקין |
| אחד מ־ הפגומים האחרים | אחד מ־ הפגומים הראשונים | לא תקין |
| אחד מ־ הפגומים האחרים | אחד מ־ הפגומים האחרים | לא תקין |
| אחד מ־ הפגומים האחרים | אחד מהתקינים | לא תקין |

אם הפרופסור ישקול את שתי הטענות "שבבים 1..t תקינים, השאר פגומים" ו־"שבבים p+1..+p+t", הוא לא יוכל לקבוע איזו נכונה, כי בשני המקרים יש קבוצה (בגודל ) של שבבים שאומרים אחד על השני "תקין" ועל כל האחרים "לא תקין" (בפועל הטענה הראשונה שגויה והשניה נכונה).

ב.  
נסדר את כל השבבים בשורה, ונבדוק אותם בזוגות ((1,2), (3,4) וכו'), אם יש מספר אי זוגי, לא נבדוק את האחרון.

בכל זוג שתוצאת הבדיקה עליו היתה "תקין - תקין", שני השבבים תקינים או שני השבבים פגומים.

טענה: מבין הזוגות שתוצאת הבדיקה עליהם היתה "תקין - תקין", מספר הזוגות שבהם שני השבבים תקינים גדול או שווה למספר הזוגות שבהם שני השבבים פגומים. אם המספרים שווים, אז מספר השבבים אי זוגי והשבב האחרון תקין.  
הוכחה: נניח בשלילה כי מבין הזוגות שתוצאת הבדיקה עליהם היתה "תקין - תקין" יש יותר זוגות של שבבים פגומים מזוגות של שבבים תקינים. מספר השבבים הפגומים בקבוצה זו גדול ב־2 לפחות ממספר התקינים. נוסיף את כל השבבים מהזוגות האחרים ("תקין - פגום", "פגום - פגום"). בכל אחד מהזוגות האלה, מספר הפגומים גדול או שווה למספר התקינים. עדיין מספר הפגומים גדול ב־2 לפחות ממספר התקינים. נוסיף את השבב האחרון אם אי זוגי. מספר הפגומים בכל ה־ גדול ממספר התקינים, בסתירה לנתון. מכאן נובע החלק הראשון של הטענה.  
נניח בשלילה כי מבין הזוגות שתוצאת הבדיקה עליהם היתה "תקין - תקין" יש אותו מספר של זוגות של שבבים פגומים וזוגות של שבבים תקינים, ובנוסף זוגי או אי זוגי והשבב האחרון פגום. מספר השבבים הפגומים בקבוצה זו שווה למספר השבבים התקינים. נוסיף את כל השבבים מהזוגות האחרים ("תקין - פגום", "פגום - פגום"). בכל אחד מהזוגות האלה, מספר הפגומים גדול או שווה למספר התקינים. כעת מספר הפגומים גדול או שווה למספר התקינים, אם אי זוגי, יש שבב פגום נוסף, נוסיף אותו. מספר הפגומים בכל ה־ גדול או שווה למספר התקינים, בסתירה לנתון. מכאן נובע החלק השני של הטענה.

ניקח שבב אחד מכל זוג שתוצאת הבדיקה עליו היתה "תקין - תקין", ואם יש מספר אי זוגי ניקח גם את האחרון. זו תהיה הבעיה הקטנה, ועל פי הטענה שהוכחנו, שגם בה יותר ממחצית השבבים תקינים.

ג.  
קודם נמצא שבב תקין אחד. בסיוע השיטה שתוארה בסעיף ב: נחלק את השבבים כל פעם לקבוצה שגודלה שבבים שרובים תקינים באמצעות הבדיקות, עד שנגיע לשבב יחיד (תקין).מספר הבדיקות הנדרשות:

פתרון:  
נוכיח   
חסם עליון: נוכיח באינדוקציה . נניח   
עבור ,  
  
זה מתקיים לכל

חסם תחתון: נוכיח באינדוקציה . הנחת האינדוקציה   
עבור ,

מסקנה:

לאחר בדיקות מצאנו שבב תקין אחד. נשתמש בו כדי לבדוק את כל שאר השבבים ולמצוא את התקינים. זה יצריך בדיקות, שזה .

שאלה 3

A=[35,22,44,5,18,50,55,8,4,38]  
BUILD-MIN-HEAP(A):  
 heap-size(A)🡨10  
 i🡨5  
 MIN-HEAPIFY(A,5):  
 l🡨10  
 r🡨11  
 smallest🡨5  
 i🡨4  
 MIN-HEAPIFY(A,4):  
 l🡨8  
 r🡨9  
 smallest🡨9  
 exchange **A[4]↔A[9]**: A=[35,22,44,4,18,50,55,8,5,38]  
 MIN-HEAPIFY(A,9):  
 l🡨18  
 r🡨19  
 smallest🡨9  
 i🡨3  
 MIN-HEAPIFY(A,3):  
 l🡨6  
 r🡨7  
 smallest🡨3  
 i🡨2  
 MIN-HEAPIFY(A,2):  
 l🡨4  
 r🡨5  
 smallest🡨4  
 exchange **A[2]↔A[4]**: A=[35,4,44,22,18,50,55,8,5,38]  
 MIN-HEAPIFY(A,4):  
 l🡨8  
 r🡨9  
 smallest🡨9  
 exchange **A[4]↔A[9]**: A=[35,4,44,5,18,50,55,8,22,38]  
 MIN-HEAPIFY(A,9):  
 l🡨18  
 r🡨19  
 i🡨1  
 MIN-HEAPIFY(A,1):  
 l🡨2  
 r🡨3  
 smallest🡨2  
 exchange **A[1]↔A[2]:** A[4,35,44,5,18,50,55,8,22,38]  
 MIN-HEAPIFY(A,2):  
 l🡨4  
 r🡨5  
 smallest🡨4  
 exchange **A[2]↔A[4]**: A=[4,5,44,35,18,50,55,8,22,38]  
 MIN-HEAPIFY(A,4):  
 l🡨8  
 r🡨9  
 smallest🡨8  
 exchange **A[4]↔A[8]**: A=[4,5,44,8,18,50,55,35,22,38]  
 MIN-HEAPIFY(A,8):  
 l🡨16  
 r🡨17  
 smallest🡨8

תוצאה: A=[4,5,44,8,18,50,55,35,22,38]  
heap-size[A]=10

כעץ:

שאלה 4

א.  
כמו בערמה רגילה, נשתמש במערך A ומספר נוסף heap-size[A] ששומר את מספר האיברים בערמה d־ית, כאשר heap-size≤length[A]. נשמור את איברי העץ במערך לפי רמות, משמאל לימין.  
נגדיר שגרה שמחזירה את האינדקס של האב של צומת i:

PARENT-d(d,i):  
return

ושגרה שמחזירה את האינדקס של הבן ה־k:

CHILD(d,i,k):  
return

ב.  
בערמה d־ית בת n איברים: (הנחה - )  
כמות איברים מינימלית בערמה d־ית בגובה h:  
  
לכן  
  
  
כמות איברים מקסימלית בערמה d־ית בגובה h:  
  
 לכן  
  
הראנו כי .  
מסקנה:

ג.  
נממש שגרת עזר MAX-HEAPIFY-d(d,A,i). הנחה: העצים המושרשים בכל אחד מ־d בניו של i הם ערמות d־יות.

MAX-HEAPIFY-d(d,A,i):  
 largest🡨i  
 k🡨1  
 firstChild🡨CHILD(d,i,k)  
 while k≤d AND firstChild+k-1<=heap-size[A]  
 if A[firstChild+k-1]>A[largest]  
 then largest🡨firstChild+k-1  
 k🡨k+1  
 if largest≠i then  
 exchange A[i]**↔**A[largest]  
 MAX-HEAPIFY-d(d,A,largest)

הסבר: האלגוריתם מהווה הכללה של MAX-HEAPIFY למצב של d בנים לצומת במקום 2. אם הערך של אחד מבניו של i גדול מהערך של i, מתבצעת החלפה בין i לגדול ביותר וקריאה רקורסיבית לאותה שגרה על הבן הגדול ביותר.  
ניתוח זמן ריצה: בכל קריאה של הרקורסיה מתבצעות השוואות, האלגוריתם מספר הקריאות הרקורסיבות הוא לכן זמן הריצה של שגרת העזר במקרה הגרוע הוא .

HEAP-EXTRACT-MAX-d(d,A,i):  
 if heap-size[A] < 1  
 then error "heap underflow"  
 max🡨A[1]  
 A[1]🡨A[heap-size[A]]  
 heap-size[A]🡨heap-size[A]-1  
 MAX-HEAPIFY-d(d,A,1)  
 return max

הסבר: האלגוריתם זהה לאלגוריתם HEAP-EXTRACT-MAX(A,i) למעט החלפת הקריאה ל־MAX-HEAPIFY(a,1) בקריאה ל־MAX-HEAPIFY-d(d,A,1).  
ניתוח זמן ריצה: מתבצעות מספר סופי של פעולות ועוד קריאה ל־MAX-HEAPIFY-d, לכן זמן הריצה במקרה הגרוע הוא .

ד.  
נממש שגרת עזר HEAP-INCREASE-KEY-d(A,i,key).

HEAP-INCREASE-KEY-d(d,A,i,key):  
 if key<A[i]  
 then error "new key is smaller than current key"  
 A[i]🡨key  
 while i>1 and A[PARENT-d(d,i)]<A[i]  
 exchange A[i]↔A[PARENT-d(d,i)]  
 i🡨PARENT-d(d,i)

הסבר: השגרה זהה ל־HEAP-INCREASE-KEY למעט קריאה ל־PARENT-d במקום PARENT. הגובה של ערמה d־ית עם n איברים הוא לכן זמן הריצה הוא .

MAX-HEAP-INSERT-d(d,A,key):  
 heap-size[A]🡨heap-size[A]+1  
 A[heap-size[A]]🡨-  
 HEAP-INCREASE-KEY-d(d,A,heap-size[A],key)

הסבר: זהה ל־MAX-HEAP-INSERT למעט קריאה ל־HEAP-INCREASE-KEY-d במקום HEAP-INCREASE-KEY.  
זמן ריצה: מספר קבוע של פעולות ועוד קראיה ל־HEAP-INCREASE-KEY-d. לכן זמן הריצה הוא .

ה.  
עשיתי את זה בסעיף ד. זמן הריצה הוא .