**עבודה 1 במבני נתונים**

דין אברג'יל וניב לוי

**שאלה 1:**

הפונקציות מסודרות על פי הסדר האסיפטוטי הנ"ל:

**הוכחה:**

בכל ההוכחות נראה כי כך ש עבור סימון אסימפטוטי "O".

ונראה כי כך שם  *עבור סימון אסימפטוטי "Ɵ".*

*נכון עבור לכל .*

*נכון עבור לכל .*

*נכון לכל ולכל .*

*סך הכל נקבל:*

*נכון עבור ו .*

*נכון עבור ו .*

וגם קיבלנו .

בסעיף קודם קיבלנו ,

*נכון עבור ו .*

נראה כי *:*

*נחלק ב-*

*נכון עבור ו .*

קיבלנו כי ומטרנזיטיביות נקבל כנדרש.

*נכון עבור ו .*

*נכון עבור ו .*

*נכון עבור ו .*

*נכון עבור ו .*

*נכון עבור ו .*

**שאלה 2:**

**סעיף א': הטענה נכונה**

כדי להראות שהטענה נכונה נראה שקיימת פונקציה שאכן מקיימת את התנאי.

עבור הפונקציה הטענה עבור .

**הוכחה:**

נראה שקיימים c1,c2,n0 > 0 המקיימים את אי השוויון:

נפשט את הביטוי ונקבל:

נבדוק גבול עבור הביטוי האמצעי ונקבל:

כלומר, לא נוכל למצוא c1 המקיים את אי השוויון, ולכן כנדרש.

**סעיף ב': הטענה אינה נכונה**

**הוכחה:**

נחלק משוואה שנייה ב-f(n) (f(n) > 0 לכל n ולא משפיע על סימן אי השוויון) ונקבל:

נחבר בין האי שוויוניים ונקבל:

נחלק ב-log(n) ונקבל:

אם נשאיף את הביטוי לאינסוף נקבל:

ובסה"כ:

לא נוכל למצוא c1 ו -c2 אשר מקיימים את התנאים ולכן הטענה אינה נכונה.

**סעיף ג': הטענה נכונה**

נחלק למקרים: נניח ,

מקרה 1:

מהגדרת O נובע:

*לאחר חילוק ב-f(n) מתקיים: (חלוקה ב-f(n) לא משפיעה על הסימן כי f(n)>1)*

*נכון עבור כל .*

מקרה 1: (בה"כ)

מהגדרת O נובע:

*לאחר חילוק ב-f(n) מתקיים: (חלוקה ב-f(n) לא משפיעה על הסימן כי f(n)>1 לכל n)*

אנו יודעים כי וגם לכל n, אז עבור אי השיוויון מתקיים.

*מצאנו כי עבור כל n קיים c ≥ 2 אשר עומד בתנאים. משל.*

**שאלה 3:**

**סעיף א':**

נסמן:

נגדיר פונקציית עזר S המוגדרת כך:

*נשתמש בשיטת המאסטר עבור S(m) :*

נראה כי :

נכון לכל *.*

*לכן לפי שיטת המאסטר קיבלנו כי:*

בחזרה לפונקציה המקורית נקבל:

**סעיף ב':**

נשתמש בשיטת המאסטר עבור T(n):

**נראה כי:**

1. קיים ɛ>0 כך ש: :

*נכון עבור c=1, n0=2 וכל* 0< ɛ<0.7*.*

1. קיים 0<c<1 כך ש: :

נפשט את הביטוי האמצעי:

*ובסה"כ נקבל:*

נכון עבור כל 5/8<c<1 וn0=2.

הוכחנו את 1 ו-2 כנדרש, לכן על פי שיטת המאסטר:

**סעיף ג':**

**הוכחה בשיטת ההצבה:**

**ניחוש חסם עליון:**

*ראשית, נראה כי n-d=O(n) כאשר d>0 קבוע כלשהו.*

*נכון עבור c=1 וn0=1 לדוגמה.*

*לכן אפשר להוכיח כי ומטרנזיטיביות נקבל כנדרש.*

*מקרה בסיס (n=1):*

*נכון n0=d+1 ו-a=1.*

*הנחה: נניח שהטענה נכונה עבור k (1≤k<n):*

*צעד:*

*כעת נותר להוכיח: ונקבל:*

*נכון לn0=1, ול- (a תמיד חיובי כי d>0 קבוע חיובי).*

**ניחוש חסם תחתון:**

מקרה בסיס (n=1):

נכון לכל b≤1.

*הנחה: נניח שהטענה נכונה עבור k (1≤k<n):*

*צעד:*

*כעת נותר להוכיח: ונקבל:*

*נכון לכל b>0 ולכל n0>0.*

*קיבלנו ש- וגם לכן בסך הכל .*

**סעיף ד':**

**הוכחה בשיטת ההצבה:**

**ניחוש חסם עליון:**

*הוכחה:*

*מקרה בסיס (n=2):*

*נכון לכל c>0.5.*

*הנחה: נניח שהטענה נכונה עבור k (1≤k<n):*

*צעד:*

כעת, נותר להוכיח: (עבור c>0 ו-n0>2)

ונקבל: , וזה נכון עבור c=2 וn0=2.

**ניחוש חסם תחתון:**

*הוכחה:*

*מקרה בסיס (n=1):*

*נכון לכל c1>*0*.*

*הנחה: נניח שהטענה נכונה עבור k (1≤k<n):*

צעד:

כעת, נותר להוכיח: (עבור c>0 ו-n0>2)

ונקבל: , וזה נכון עבור c=1 וn0=2.

*לסיכום קיבלנו: וגם .*

לכן .

**סעיף ה':**

נשתמש בשיטת האיטרציה:

הפונקציה תעצר כאשר n-i = 1 => i = n-1

**שאלה 4:**

**(a** ניתן לראות כי שורת ה- for הראשונה מתבצעת n-1 פעמים ועבור כל פעם, מתבצעת שורת ה-for השנייה n-(i+1) פעמים ובתוכה מספר סופי של פעולות קבועות (c) כלומר,

**(b** ניתן לראות כי זמן הריצה של הפונ' הוא:

הפונ' תעצר כאשר

**C)** נבחן את זמן הריצה במקרה הטוב ביותר והמקרה הגרוע ביותר. נבחן את המקרה הגרוע ביותר; ניתן לראות כי עבורpower זוגי הקטנת הבעיה היא הרבה יותר משמעותית מאשר עבור power אי זוגי, כמו כן, ניתן לראות כי עבור power אי זוגי הקטנת הבעיה היא בהכרח ל - power זוגי. מכאן שהמקרה הגרוע ביותר(זמן ריצה הארוך ביותר) הוא עבור power אי זוגי כך שבכל קריאה לפונ' הרקורסיבית, ה-power משנה את הזוגיות שלו(למשל: power = … 31, 30, 15, 14, 7, 6, 3, 2, 1 ). הקריאה הראשונה לפונ' הרקורסיבית במצב זה הוא עבור power אי זוגי, לאחר מכן מתבצעות מספר פעולות קבועות ומתבצעת הקטנת בעיה ל- power-1 שהוא כמובן זוגי. לאחר מכן מתבצעת הקטנת בעיה ל- (power-1)/2 ואז מקבלים power אי זוגי ומתבצע אותו תהליך שוב, עד אשר power=1 . לכן זמן הריצה במקרה הכי גרוע הוא:

הפונ' תעצר כאשר:

אז הגענו לכך שזמן הריצה של הפונ' הוא O(log(n)) וכעת נבחן את המקרה הטוב ביותר ונראה שהוא ולכן זמן הריצה של הפונ' הוא (log(n)) ולכן בסה"כ זמן הריצה של הפונ' הוא .

במקרה המהיר ביותר הפונ' תקבל power זוגי ובכל קריאה רקורסיבית ה- power יישאר זוגי וההקטנה תהיה פי 2 עד אשר power=1. זמן הריצה במקרה כזה יהיה:

הפונ' תעצר כאשר:

**שאלה 5**

**א)**

int theIndex(int arr[], int x)

{

if (arr.size == 0) {

return -1;

}

int curr = 1;

while (curr < arr.size && arr[curr] < x) {

curr = curr \* 2;

}

return binarySearch (arr, x, curr /2, min(curr, arr.size));

}

ניתן לראות כי בפונקציה יש מספר פעולות קבועות שכפי שלמדנו בכתה אינן משפיעות על חישוב זמן הריצה. כמו כן, ניתן לראות כי ישנה לולאת while נחשב את מספר הפעמים (i) שנכנס ללולאה לכל היותר עבור איבר הנמצא באינדקס d:

בנוסף יש בפונקציה קריאה לחיפוש בינארי שכמו שלמדנו מתבצעת ב log(n) אך הטווח אליו נעשית הקריאה לחיפוש הבינארי הוא d לכל היותר ולכן גם זמן הריצה של החיפוש הבינארי הוא O(log(d)) ובסה"כ חיבור של כל אלו משאיר אותנו עם O(log(d))

binarySearch(int arr[], int x, int low, int high) {

if (high < low)

return -1

mid = (low + high) / 2

if (arr[mid] > x)

return binarySearch(arr, x, low, mid-1)

else if (arr[mid] < x)

return binarySearch(arr, x, mid+1, high)

else

return mid

}

**ב)**

int median(int[] arr1, int[] arr2)

{

int x=0; int y=0; int size; int output=0;

if((arr1.length+arr2.length)%2 == 0)

size = (arr1.length+arr2.length)/2;

else

size = (arr1.length+arr2.length + 1)/2;

for(int i=0; i<size;i++)

{

if(x==arr1.length || y==arr2.length)

{

if(y>arr2.length)

{

output = arr1[x];

x++;

}

else

{

output = arr2[y];

y++;

}

}

else

{

if(arr1[x]>=arr2[y])

{

output = arr2[y];

y++;

}

else

{

output = arr1[x];

x++;

}

}

}

return output;

}

ניתן לראות כי בפונקציה מספר פעולות קבועות ועוד לולאת for המתבצעת לכל היותר (n1+n2)/2 כאשר n1 ו n2 הם הגדלים של המערכים ולכן זמן הריצה הוא O(n).