סמסטר ב' 2023/24

מבני נתונים ואלגוריתמים

תרגיל בית מספר 1

שאלה 1

נתון מערך A בגודל $n > 1$ המכיל מספרים שאינם בהכרח שונים זה מזה, והוא ממוין מקטן לגדול. יהי n_{min} מספר המופעים של הערך הקטן ביותר במערך, ו n_{max} מספר המופעים של הערך הגדול ביותר. לדוגמה, אם A=[2,2,2,2,5,5,7,10,10,10], אז $n_{min}=4$ ו $n_{max}=3$.

א. כתבו פסאודו-קוד עבור פרוצדורה המקבלת כקלט את A ואת גודלו, n, והמחזירה את n_{min}. זמן הריצה של הפרוצדורה צריך להיות $O(\log(n))$. אין צורך להוכיח את נכונות הפרוצדורה וזמן הריצה שלה, אך יש להסביר זאת בקיצור.

ב. נתבונן באלגוריתם הבא (כאשר ניתן להניח שהבדיקה של A[n-k]=A[n] בתוך לולאת ה while נעשית רק אם k<n).

```
Alg(A,n) {
  k := 1;
  while (k<n   and  A[n-k] = A[n] )
        k := 2k
   if (k=1)
         return(1)
  max:= n-(k/2)          /* if k>1, then k is an even integer */
  if (k<n)
         min:= n-k+1
  else
         min := 1
  while(min<max) {
         mid :=⌊(min+max)/2⌋
         if (A[mid] = A[n])
                max := mid
         else
                 min := mid+1
    }
 return(n-min+1)
}
```

(i) מה מחזיר האלגוריתם ? (השתמשו באחד הסימונים שהוצגו בתחילת השאלה.) הסבירו את תשובתכם בקיצור אך בבירור.

(ii) מה זמן הריצה של האלגוריתם כפונקציה של n עבור קלטים בהם $n_{min}=O(\log n)$ ו $n_{max}=O(\log n)$? עליכם לתת חסם עליון טוב ככל האפשר ולהסביר אותו.

שאלה 2

בשאלה זו נוכיח כי חיפוש במערך **לא ממוין** באורך n דורש לפחות n צעדים ב-worst case. הניחו בשלילה שקיים אלגוריתם לחיפוש במערך לא ממוין אשר דורש לכל היותר $n-1$ צעדים ב-worst case. <u>לצורך פשטות, אנו נתמקד במערכים בוליאניים (שמכילים אפסים ואחדות בלבד).</u> יהא A מערך של n 0ים.

א. חישבו על חיפוש של 1 במערך A על ידי האלגוריתם.
  i. מה תהיה תשובת האלגוריתם?
  ii. נמקו מדוע קיים אינדקס $1 \le i \le n$ כך שהאלגוריתם לעולם לא פונה אל $A[i]$.

ב. יהא B מערך המתקבל ממערך A על ידי שינוי התא ה-i מ-0 ל-1 (כאשר i הוא האינדקס שמצאנו בסעיף הקודם). כלומר המערך B מכיל 1 בתא ה-i ובשאר התאים מכיל 0ים. חישבו על חיפוש של 1 במערך B על ידי האלגוריתם.
  i. השוו את הרצת האלגוריתם על A להרצת האלגוריתם על B. נמקו מדוע פעולות האלגוריתם בהרצה הראשונה (עם מערך A) זהות לפעולות האלגוריתם בהרצה השנייה (עם מערך B).
  ii. אם כך, מה תהיה תשובת האלגוריתם בהרצה על B? הגיעו לסתירה.

ג. נמקו מדוע שיטת ההוכחה הזו נכשלת כאשר מדובר באלגוריתם לחיפוש **במערך ממוין**.

שאלה 3

בכיתה דנו בכך שזמן הריצה של InsertionSort על מערכי קלט שונים הוא שונה. בפרט, כאשר המערך ממוין הפוך (מגדול לקטן עם מספרים השונים זה מזה) זמן הריצה גדל ריבועית עם גודל המערך, בעוד שאם המערך ממוין (מקטן לגדול) אז זמן הריצה גדל ליניארית עם גודל המערך. בשאלה זו ננסה להבין באופן כללי יותר כיצד זמן הריצה על קלטים שונים תלוי ב"עד כמה הם אינם ממוינים". עבור מערך (בגודל n) נסמן את זמן הריצה של Insertion-Sort על המערך A ב- $T_{IS}(A)$.

א. ראשית נתבונן במקרים הבאים. נניח שבמערך הקלט יש את הערכים $1,\ldots,n^{2/3}$ כאשר $n^{1/3}$ ו $n^{2/3}$ הם מספרים שלמים. נתבונן בשני מקרים, כאשר בשניהם המערך ממוין מגדול לקטן. במקרה הראשון, שנסמנו A1, במקומות 1 עד $n^{2/3}-1$ במערך A1 יש מופע אחד של כל אחד מהערכים $2,..,n^{2/3}$ כאשר הם ממוינים מגדול לקטן, ובמקומות $n^{2/3}$ עד n יש את הערך 1. במקרה השני, שנסמנו A2, במקומות 1 עד $n^{1/3}$ יש את הערך $n^{2/3}$, במקומות $n^{1/3}+1$ ועד $2n^{1/3}$ יש את הערך $n^{2/3}-1$, ובאופן כללי, עבור כל מספר שלם חיובי $k=0,\ldots, n^{2/3}-1$, במקומות $k\cdot n^{1/3}+1$ עד $(k+1)\cdot n^{1/3}$ יש את הערך $n^{2/3}-k$ (בפרט, במקומות $n-n^{1/3}+1$ עד n יש את הערך 1).
לדוגמה, אם n=8 אז A1=[4,3,2,1,1,1,1,1] ו A2=[4,4,3,3,2,2,1,1] .
מהם $T_{IS}(A1)$ ו $T_{IS}(A2)$ כפונקציה של n?

עבור הסעיפים הבאים, נשתמש בסימון הבא. עבור כל אינדקס $1 \le j \le n$ נסמן ב- $L_A(j)$ את מספר האיברים במערך A שערכם **גדול** מ A[j] אך הנמצאים **לפניו** ב A. כלומר $L_A(j) = |\{1\le i<j : A[i] > A[j]\}|$
לדוגמא, אם A = [5,4,4,3] אזי $L_A(1) = 0, L_A(2) = 1, L_A(3) = 1, L_A(4) = 3$.

ב. נניח ש A מערך עם n מספרים שונים שממוין מהקטן לגדול. מהו $L_A(j)$ עבור $1\le j\le n$?
ג. נניח ש A מערך עם n מספרים שונים שממוין מהגדול לקטן. מהו $L_A(j)$ עבור $1\le j\le n$?
ד. מה הקשר בין $L_A(j)$ לבין מספר האיטרציות של לולאת ה-while על המערך A באיטרציה ה- j של לולאת ה-for?
ה. מהו $T_{IS}(A)$ כפונקציה של n ו $L_A(1),L_A(2),\ldots,L_A(n)$? ודאו שבשילוב עם סעיפים ב' ו ג' אתם מקבלים תשובה נכונה עבור זמן הריצה של IS על מערך ממוין מהקטן לגדול ומהגדול לקטן (של מספרים שונים).
ו. (i) האם לכל n קיים מערך A בגודל n כך ש – $cn^{5/6}\le T_{IS}(A)\le c' n^{5/6}$ עבור קבועים c,c'?
(ii) האם לכל n קיים מערך A בגודל n כך ש – $cn^{13/6} \le T_{IS}(A) \le c' n^{13/6}$ עבור קבועים c,c'?
(iii) האם לכל n קיים מערך A בגודל n כך ש $cn^{6/5}\le T_{IS}(A)\le c' n^{6/5}$ עבור קבועים c,c'?
נמקו את תשובתכם. כלומר, אם תשובתכם שלילית אז עליכם להסביר בבירור מדוע לא קיים כזה מערך, ואם תשובתכם חיובית, אז עליכם לתאר מערך בגודל n הנותן זמן ריצה שכזה (תיאור מערך בגודל n עבור n כללי צריך להתבצע בדומה לזה של המערך המתואר בסעיף **א**).

שאלה 4

נתון האלגוריתם הבא:

```
Algorithm Alg (int n)  {
   Proc1(n);
   Proc2(n);
   Proc3(n);
}
```

האלגוריתם מקבל כקלט מספר שלם n, קורא לשלוש פרוצדורות ושולה להן את n.
נסמן את זמני הריצה של שלש הפרוצדורות Proc1,Proc2,Proc3 ב- $T_1(n),T_2(n),T_3(n)$ בהתאמה, ואת זמן הריצה של כל האלגוריתם ב- $T_A(n)$.

נתון ש $T_1(n) = \Omega(n\log^3 n), T_2(n) = O(n^2), T_3(n) = \Theta(n^{\frac{1}{2}})$

אילו מבין החסמים הבאים <u>בהכרח נכונים</u>, אילו <u>בהכרח אינם נכונים</u>, ואילו <u>אינם בהכרח נכונים</u> (כלומר, ייתכן והם נכונים וייתכן ואינם נכונים)? נמקו בקצרה את תשובותיכם.

(1) $T_A(n) = \Omega(n^{1/2})$

(2) $T_A(n) = \Omega(n^2)$

(3) $T_A(n) = O(n)$

שאלה 5

נתון האלגוריתם הבא המקבל כקלט מערך A בגודל n המכיל מספרים וממיין אותו. האלגוריתם גם מקבל פרמטר $1 \le k \le n$. האלגוריתם עושה שימוש בשתי פרוצדורות. האחת, InsertionSort2, המקבלת כקלט מערך A ושני פרמטרים $1 \le p < r \le n$, פועלת בדיוק כמו InsertionSort שראינו בכיתה, מלבד שהיא פועלת על תת מערך A[p,…,r] במקום על מערך שלם. לנוחיותכם, הקוד שלה מופיע בעמוד הבא. הפרוצדורה Merge היא הפרוצדורה הנקראת על ידי MergeSort, כפי שלמדנו בכיתה. (להזכירכם, בהינתן פרמטרים $p\le q<r$, Merge(A,p,q,r) פועלת תחת ההנחה שתתי המערכים A[p,..,q] ו A[q+1,…,r] ממוינים, וממזגת בינהם. זמן הריצה שלה ליניארי ב $r-p+1$.) לנוחיותכם, הקוד של Merge מופיע בעמוד הבא.

<u>נתון ש n ו k שניהם חזקות שלמות של 2.</u>

```
Alg(A,n,k)  {
   1.  if (k > 1)
   2.      for (j = 0 to (n/k)-1)
   3.          InsertionSort2(A,j·k+1,(j+1)·k)
   4.  for (i= log k +1  to log n)
   5.      for (j=0 to n/2^i - 1)
   6.              Merge (A , j·2^i + 1, j·2^i + 2^(i-1), (j+1)·2^i )
}
```

<u>עבור סעיפים א-ג נתון ש k=1</u> (כלומר האלגוריתם מתחיל את פעולתו בשורה 4, ורץ מ i=1 (עד log n)).

א. נתון המערך הבא בגודל 8: A=[4,7,2,8,5,1,6,3]. כיצד ייראה המערך A בסיום האיטרציה השנייה של לולאת ה for בשורה 4 (כלומר, סיום האיטרציה בה i=2)?

ב. כעת הניחו גודל n כללי (עדיין חזקה שלמה של 2). נתעניין במערך A בסיום האיטרציה ה i של לולאת ה for שבשורה 4 (עבור $1\le i\le \log n$ כללי). תנו חסם עליון טוב ככל האפשר כפונקציה של n ו i על מספר האינדקסים $1\le x\le n-1$ כך שבשלב זה ייתכן ו $A[x]>A[x+1]$. הסבירו בקיצור את תשובתכם.

ג. כעת הניחו $n\geq8$, (ועדיין חזקה שלמה של 2). נרצה להשוות בין הקריאות לפרוצדורה Merge על ידי האלגוריתם Alg לבין הקריאות לפרוצדורה Merge על ידי MergeSort שלמדנו בכיתה, על מערך באותו גודל n. בחרו בתשובה הנכונה <u>ונמקו את תשובתכם</u>:

(1) שני האלגוריתמים מבצעים את אותן הקריאות ל Merge (כלומר, על אותם אינדקסים) ובאותו הסדר.

(2) שני האלגוריתמים מבצעים את אותן הקריאות לMerge אך בסדר שונה.

(3) שני האלגוריתמים לא מבצעים את אותן הקריאות ל Merge.

עבור סעיפים ד-ה <u>נסיר את ההנחה ש k=1, כלומר k≥1</u> (וכמו n, הוא חזקה שלמה של 2).

ד. נתחו את זמן הריצה של Alg כפונקציה של n ו k. כלומר, תנו פונקציה $g(n,k)$ כך שזמן הריצה הוא $\Theta(g(n,k))$ ונמקו את תשובתכם. שימו לב שעליכם להראות שזמן הריצה הוא גם $O(g(n,k))$ וגם $\Omega(g(n,k))$.

ה. האם הטענה הבאה נכונה: לכל $n\geq8$ קיים מערך A בגודל n עליו זמן הריצה של Alg(A,n,k) כאשר k=1 גדול אסימפטוטית מאשר זמן הריצה כאשר k=n. אם תשובתכם שלילית, עליכם להסביר מדוע, ואם תשובתכם חיובית, עליכם לתאר מערך A שכזה.

```
Merge(array A; integers p,q,r)  {
      i ← p, j ← q+1,  k ← 1
      while (i ≤ q  and  j ≤ r) {
           if  (A[i] ≤ A[j]) {
                B[k] ← A[i]
                 i  ← i+1
            }
           else  {
                B[k] ← A[j]
                 j ← j+1
           }
           k ← k+1
      }
      while (i ≤ q )  {
           B[k] ← A[i]
           i  ← i+1
           k ← k+1
      }
      while (j ≤ r)  {
           B[k] ← A[j]
           j ← j+1
           k ← k+1
      }
      for (i = p to r)
           A[i] ← B[i-p+1]
}
```

```
/* A is an array of size n, 1≤p<r≤n  .
   If p=1, r=n, then we get InsertionSort */

InsertionSort2(array A; integers p,r)  {
      for   (j = p to r)  {
           newnum ← A[j]
           i ← j-1
           while ( i > p-1  and  newnum < A[i] )   {
                A[i+1] ← A[i]
                i ← i-1
           }
           A[i+1] ← newnum
      }
}
```