מבוא למדעי מחשב מ' / ח' (234114 / 234117)

סמסטר חורף תשע"ב

מבחן מסכם מועד ב', 14 מרץ 2012

מספר סטודנט	*שם משפחה*	*שם פרטי*

משך המבחן: 2.5 שעות.
חומר עזר: אין להשתמש בכל חומר עזר.

הנחיות והוראות:

- מלאו את הפרטים בראש דף זה ובדף השער המצורף, בעט בלבד.
- בדקו שיש 14 עמודים (4 שאלות) במבחן, כולל עמוד זה.
- כתבו את התשובות על טופס המבחן בלבד, במקומות המיועדים לכך. שימו לב שהמקום המיועד לתשובה אינו מעיד בהכרח על אורך התשובה הנכונה.
- העמודים הזוגיים בבחינה ריקים. ניתן להשתמש בהם כדפי טיוטה וכן לכתיבת תשובותיכם. סמנו טיוטות באופן ברור על מנת שהן לא תבדקנה.
- יש לכתוב באופן ברור, נקי ומסודר. **ניתן בהחלט להשתמש בעיפרון ומחק**, פרט לדף השער אותו יש למלא בעט.
- בכל השאלות, הינכם רשאים להגדיר ולממש פונקציות עזר כרצונכם. לנוחיותכם, אין חשיבות לסדר מימוש הפונקציות בשאלה, ובפרט ניתן לממש פונקציה לאחר השימוש בה.
- **אין להשתמש בפונקציות ספריה או בפונקציות שמומשו בכיתה אלא אם צויין אחרת בשאלה, למעט פונקציות קלט/פלט והקצאת זיכרון.**

צוות הקורס 234114/7
מרצים: דן רביב, רן רובינשטיין, רון קימל (מרצה אחראי).
מתרגלים: אלכסנדר נוס, אריאלה וולושין, שי גרץ, אנסטסיה דוברובינה, עמרי אייזנקוט, דן גרבר, חסן עבסי, שחר תמנת, חביאר טורק (מתרגל אחראי).

בהצלחה!

שאלה 1 (25 נקודות)

ממשו את הפונקציה הבאה, המקבלת כקלט מספר שלם וחיובי x, וכותבת את הייצוג הבינארי שלו לתוך המערך []a. יש לכתוב את ספרת האחדות לתוך a[0], ויתר הספרות על פי סדרן ב-a[1], a[2] וכן הלאה. על הפונקציה להחזיר את מספר הספרות הבינאריות שכתבה לתוך המערך.

לדוגמה, עבור הקלט x==13, שייצוגו הבינארי 1101, הפונקציה תכתוב למערך את התוכן הבא, ותחזיר 4:

1	1	0	1
a[3]	a[2]	a[1]	a[0]

הערות:

- ניתן להניח שהמערך []a ארוך מספיק על מנת להכיל את הייצוג הבינארי.
- השלימו את סיבוכיות הזמן של האלגוריתם שכתבתם, כתלות ב-x, במקום הבא:

סיבוכיות זמן: Θ(_____ **log x**)

```
int num_to_binary(unsigned int x, int a[]) {
    int idx = 0;
    while (x > 0) {
        a[idx] = x % 2;
        x /= 2;
        idx++;
    }
    return idx;
}
```

שאלה 2 (25 נקודות)

מחרוזת מדרגה הינה מחרוזת המכילה בחצי הראשון שלה תווי 'a' בלבד, ובחצי השני שלה תווי 'b' בלבד. דהיינו, המחרוזת הינה מהצורה "aaa…abbb…b", כאשר ידוע שמספר תווי 'a' שווה למספר תווי 'b'. שימו לב שהאורך הכולל של המחרוזת אינו ידוע מראש, אבל הוא בהכרח זוגי.

ממשו את הפונקציה הבאה, המקבלת מחרוזת מדרגה s, ומחזירה את האינדקס של תו 'b' כלשהו במחרוזת. ניתן להניח ש-s הינה מחרוזת מדרגה חוקית ומכילה לפחות 2 תווים.

דרישות סיבוכיות: על הפונקציה לעמוד בסיבוכיות זמן O(log n) כאשר n הוא אורך המחרוזת, וכן בסיבוכיות מקום O(1).

```
int find_b(char* s) {
    int idx=1;
    while (s[idx] == 'a')
        idx *= 2;
    return idx;
}
```

שאלה 3 (25 נקודות)

סעיף א (15 נקודות)

בהינתן מערך []a באורך n הממוין בסדר עולה, ידוע כי הסיבוכיות של חיפוש ערך x כלשהו במערך הינה O(log n), באמצעות חיפוש בינארי.

לנוחותכם, מצורף קוד הממש חיפוש בינארי:

```
int binsearch(int a[], int n, int x)
{
    int low = 0, high = n-1, mid;
    int val;

    while (high>=low) {
        mid = (high+low)/2;
        val = a[mid];
        if (x==val) return mid;
        else if (x<val) high=mid-1;
        else low=mid+1;
    }

    return -1;
}
```

בשאלה זו נניח כי המערך []a הינו מערך מיוחד, בו סיבוכיות הגישה לאיבר a[k] תלויה באינדקס אליו ניגשים. ספציפית, קריאת האיבר a[k] דורשת k פעולות (במקום פעולה אחת כפי שהנחנו עד כה).

חשבו את סיבוכיות הזמן של פונקצית החיפוש הבינארי במקרה זה, כתלות באורך המערך n, וכתבו את התוצאה במקום המתאים למטה. יש לחשב סיבוכיות עבור המקרה הגרוע ביותר, כפי שנלמד בכיתה. כמו כן הסבירו במקום המתאים כיצד הגעתם לפתרון.

סיבוכיות זמן (מקרה גרוע ביותר): Θ(**n log(n)**)

הסבר: המקרה הגרוע ביותר קורה כאשר $x \geq a[n-1]$. לכן, נחשב את הסיבוכיות לפי מקרה זה.
ראשית נכתוב טבלה קצרה המתארת את מספר הפעולות בכל איטרציה (ניתן להזניח פעולות בעלות זמן קבוע כיוון שהפעולה הדומיננטית היא הגישה למערך):

איטרציה	פעולות
1	$\frac{n}{2} = n - \frac{n}{2}$
2	$\frac{3n}{4} = n - \frac{n}{4}$
3	$\frac{7n}{8} = n - \frac{n}{8}$
k	$n - \frac{n}{2^k}$

כעת, מספר האיטרציות הוא $\log n$, כמו בחיפוש בינארי רגיל. לכן, מספר הפעולות הכולל הוא:

$$\mathrm{T}(n) = \sum_{k=1}^{\log n} n - \frac{n}{2^k} = n\log n - n\left(\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \cdots\right)$$

נחסום את איבר אחרון: $\frac{1}{2}n \leq n\left(\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \cdots\right) \leq 2n$

$$c_1 n\log n \leq n\log n - 2n \leq T(n) \leq n\log n - \frac{1}{2}n \leq c_2 n\log n$$

לכן עבור $c_1 = \frac{1}{2}, c_2 = 2$, מקבלים כי $T(n) = \Theta(n\log n)$

<u>סעיף ב (10 נקודות)</u>

בהינתן מערך []a באורך n (לא בהכרח ממוין), ידוע כי סיבוכיות הזמן של מיון מערך זה באמצעות max-sort או bubble-sort הינה $O(n^2)$ לכל אחד מהאלגוריתמים.

לנוחותכם, מצורף קוד הממשש את שני אלגוריתמי המיון (שימו לב שהפונקציה ()swap מקבלת שני מצביעים ומחליפה את תוכנם):

```
void maxsort(int a[], int n)
{
    int i, j, max_id;

    for (i=n-1; i>0; i--) {
        max_id=0;
        for (j=1; j<=i; j++) {
           max_id = a[j]>a[max_id] ? j : max_id;
        }
        swap(&a[i], &a[max_id]);
    }
}
```

```
void bubblesort(int a[], int n)
{
    int i, j;

    for (i=n-1; i>0; i--) {
        for (j=0; j<i; j++) {
             if (a[j]>a[j+1]) {
                 swap(&a[j],&a[j+1]);
             }
        }
    }
}
```

בשאלה זו נניח כי המערך []a הוא כזה שבו <u>החלפה</u> של שני איברים (באמצעות הפונקציה swap) <u>תלויה במרחק ביניהם</u>. ספציפית, החלפת האיבר a[i] והאיבר a[j] לוקחת **$(i-j)^2$** <u>פעולות</u>, במקום פעולה אחת כפי שהנחנו עד כה. שימו לב שיתר הפעולות על המערך <u>נותרות ללא שינוי</u>, ובפרט, השוואה של שני איברים לוקחת פעולה אחת בלבד (כרגיל).

עבור מערך מהסוג המתואר, מה תהיה סיבוכיות הזמן של כל אחד מאלגוריתמי המיון הנ"ל? ולאור זאת, איזה מבין שני האלגוריתמים עדיף במצב זה (מבחינת סיבוכיות זמן)?

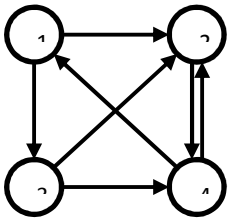
סיבוכיות זמן **max-sort**: $\Theta($ **n^3** $)$ סיבוכיות זמן **bubble-sort**: $\Theta($ **n^2** $)$

האלגוריתם העדיף (max-sort, bubble-sort, או אותה עדיפות): **<u>bubble-sort</u>**

שאלה 4 (25 נקודות)

בשאלה זו נתון גרף מכוון בעל N קודקודים (N מוגדר כ-#define). הגרף מתואר על ידי המטריצה a[N][N], כאשר a[i][j] מכיל 1 אם קיימת קשת מהקודקוד i אל הקודקוד j, ו-0 אחרת (שימו לב שהמטריצה אינה סימטרית בהכרח, כלומר ייתכן שישנה קשת מ-i ל-j אבל לא קשת בכיוון ההפוך). כמו כן ניתן להניח שבגרף אין קשתות מקודקוד לעצמו, כלומר לכל i, a[i][i]=0.

עליכם לממש את הפונקציה הבאה, המקבלת את המטריצה a[N][N] וכן אינדקס של קודקוד v (בין 0 ל-N-1). הפונקציה מחזירה 1 במידה וקיים מסלול בגרף המתחיל מ-v, עובר דרך כל אחד מהקודקודים בגרף פעם אחת בדיוק, וחוזר ל-v. הפונקציה מחזירה 0 אם לא קיים מסלול שכזה.



למשל, בדוגמה משמאל קיים מסלול כזה עבור הקודקוד v=1:
1 ← 3 ← 2 ← 4 ← 1. מסלול זה מתחיל ומסתיים בקודקוד v, ועובר דרך כל יתר קודקודי הגרף פעם אחת בדיוק.

הערה: על האלגוריתם להיות רקורסיבי ולעבוד בשיטת backtracking.

```
int find_cycle(int a[N][N], int v) {
    int used[N] = {0};
    return find_cycle_aux(a,used,0,0);
}

int find_cycle_aux(int a[N][N], int used[], int index, int count)
{
    int i;
    if (count==N-1)
       return a[index][0];
    used[index] = 1;
    for (i = 0; i < N; i++) {
       if (a[index][i] && !used[i]) {
          if (find_cycle_aux(a,used,i,count+1))
             return 1;
       }
    }
    used[index] = 0;
    return 0;
}
```