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

סמסטר אביב תשע"ג

מבחן מסכם מועד א' - פתרון, 11 יולי 2013

מספר סטודנט: | | | | | | | | | *רשום/ה לקורס:* | | | | | | 2 | 3 | 4 | 1 | 1 |

משך המבחן: 3 שעות.
חומר עזר: אין להשתמש בכל חומר עזר.

הנחיות כלליות:

- מלאו את הפרטים בראש דף זה ובדף השער המצורף, בעט בלבד.
- בדקו שיש 12 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. <u>ניתן בהחלט להשתמש בעיפרון ומחק</u>, 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.
- אלא אם כן נאמר אחרת בשאלות, **אין להשתמש בפונקציות ספריה או בפונקציות שמומשו בכיתה**, למעט פונקציות קלט/פלט והקצאת זיכרון (`malloc, free`). ניתן להשתמש בטיפוס `bool` המוגדר ב-`stdbool.h`
- אין להשתמש במשתנים סטטיים וגלובאליים אלא אם נדרשתם לכך מפורשות.
- ניתן להשתמש בהקצאות זכרון בסגנון C99 (מערכים בגודל משתנה), בכפוף לדרישות סיבוכיות זכרון.
- כשאתם נדרשים לכתוב קוד באילוצי סיבוכיות זמן/מקום נתונים, אם לא תעמדו באילוצים אלה תוכלו לקבל בחזרה מקצת הנקודות אם תחשבו נכון ותציינו את הסיבוכיות שהצלחתם להשיג.
- נוסחאות שימושיות:

$$1+\frac{1}{2}+\frac{1}{3}+\ldots+\frac{1}{n}=\Theta(\log n) \qquad 1+\frac{1}{4}+\frac{1}{9}+\frac{1}{16}+\frac{1}{25}+\ldots=\Theta(1)$$

$$1+2+\ldots+n=\Theta(n^2) \qquad 1+4+9+\ldots+n^2=\Theta(n^3) \qquad 1+8+27+\ldots+n^3=\Theta(n^4)$$

צוות הקורס 234114/7
מרצים: פרופ' ניר אילון (מרצה אחראי), ד"ר תמיר לוי, חביאר טורק.
מתרגלים: נחשון כהן, נדיה לבאי, עלי חמוד, אביב סגל, ירון קסנר, ורד כהן, אנסטסיה דוברובינה (מתרגלת אחראית).

בהצלחה!

שאלה 1 (25 נקודות):

א. (12 נקודות) חשבו את סיבוכיות הזמן והמקום של הפונקציה f המוגדרת בקטע הקוד הבא, כפונקציה של n. אין צורך לפרט שיקוליכם.

```
int f(int n)
{
    if (n == 1)
        return 1;

    return f(f(n-1));
}
```

סיבוכיות זמן: Θ (n) סיבוכיות מקום: Θ (n)

ב. (13 נקודות) נשנה עתה את הגדרת הפונקציה מהסעיף הקודם:

```
int f(int n)
{
    if (n == 1)
        return 1;

    return 1 + f(f(n-1));
}
```

מה ערך החזרה של הפונקציה f עבור הקלט n? n

סיבוכיות זמן: Θ (2^n) סיבוכיות מקום: Θ (n)

שאלה 2 (25 נקודות) :

<u>סעיף א (12 נקודות)</u>

ממשו את הפונקציה הבאה, שחתימתה

```
int limit_and_sort(int a[], int n, int m, int p[]);
```

הפונקציה מקבלת כקלט מערך a של מספרים **שלמים וחיוביים** באורך n, עוד מספר m>1 וכן מצביע p למערך של לפחות m תאים (**המערך כבר מוקצה**). על הפונקציה למצוא את כל המספרים ב- a שערכם לכל היותר m ולכתוב אותם בצורה ממוינת ובלי חזרות למערך p.
על הפונקציה להחזיר את מספר המספרים שנכתבו ל-p. לדוגמא, אם a הוא המערך הבא:

```
int a[7] = {4, 7, 5, 4, 3, 30, 201};
```

וכן n=7, m=200, אז על הפונקציה לכתוב לתוך 5 התאים הראשונים של המערך p את התוכן הבא (משמאל לימין):

3	4	5	7	30

וכן להחזיר את המספר 5.

דרישות: סיבוכיות זמן O(m+n) וסיבוכיות מקום נוסף O(m)

אם לפי חישוביכם לא עמדתם בדרישות הסיבוכיות, אנא ציינו כאן את הסיבוכיות שהגעתם אליה: זמן _____ מקום נוסף _____

```
int limit_and_sort(int a[], int n, int m, int p[])
{
    int exists[m+1];

    for (int i=1; i <= m; i++)
        exists[i] = 0;

    // Build indicator array
    for (int i=0; i<n; i++)
        if (a[i] <= m)
            exists[a[i]] = 1;
```

```
    // Fill p[]
    int num=0;
    for (int i=1; i <= m; i++)
        if (exists[i])
            p[num++]=i;

    // Return
    return num;
}
```

הסבר: נקצה מערך עזר בגודל m+1 ונסמן בו איזה מספרים בטווח [1,m] מופיעים במערך a, על ידי הצבת 1 באינדקסים המתאימים (הערה: ניתן גם להשתמש במערך בגודל m). נבצע זאת בסיבוכיות זמן O(n).
לאחר מכן, נמלא את המערך p במספרים אשר סומנו במערך העזר, בסיבוכיות זמן O(m). סיבוכיות מקום – O(m).

<u>סעיף ב (13 נקודות)</u>

בהינתן מערך a של מספרים חיוביים, נאמר שמספר z הוא ***מיוחד ביחס ל- a*** *אם* קיימים שני מספרים x<y שנמצאים שניהם ב- a כך ש z מתחלק ב- x·y. לדוגמא אם המערך a הוא

```
int a[7] = {4, 7, 5, 4, 3, 30, 201}
```

אז המספר 24 הוא מיוחד ביחס ל- a כי הוא מתחלק ב- 3·4=12 והמספרים 3,4 במערך.
לעומת זאת, 25 אינו מיוחד ביחס ל- a כי הוא אינו מתחלק במכפלה של שני מספרים שונים ב- a.

ממשו פוקנציה המקבלת מערך a, את גודלו n, ומספר נוסף m>1. על הפונקציה להדפיס את **כל המספרים** בתחום 1,2,...,m, המיוחדים ביחס ל- a. יש להדפיס את המספרים בסדר עולה ואין להדפיס אף מספר פעמיים.

דרישות: סיבוכיות זמן $O(n+m\cdot\log^2 m)$ וסיבוכיות מקום נוסף $O(m)$

<u>ניתן ומומלץ</u> להשתמש בפונקציה מהסעיף הקודם, גם אם לא מימשתם אותה.

אם לפי חישוביכם לא עמדתם בדרישות הסיבוכיות, אנא ציינו כאן את הסיבוכיות שהגעתם אליה: זמן _____ מקום נוסף _____

הפתרון מופיע בעמוד הבא.
הסבר: נשתמש בעיקרון הנפה של ארטוסטנס, שנלמד בכיתה. נקצה מערך עזר בו נסמן על ידי 1-ים את כל המספרים שמיוחדים ביחס ל-a – to_print[m+1]. נשתמש בפונקציה מסעיף א' על מנת למצוא את כל המספרים ב-a שערכם לכל היותר m ונשמור אותם במערך p. לאחר מכן, נעבור על כל זוגות המספרים השונים ב-p שמכפלתם קטנה או שווה ל-m, ולכל זוג כזה נסמן את כל המכפלות שלו כמספרים מיוחדים, במערך to_print. לבסוף, נעבור על המערך to_print ונדפיס את כל המספרים המיוחדים שסימנו בו. <u>חישוב הסיבוכיות</u>: הסיבוכיות של כל מה שקורה אחרי הקריאה ל limit_and_sort היא לכל היותר Θ של: $\sum_{i<j}(m/p[i]p[j]) = m\sum_{i<j}(1/p[i]p[j]) \le m\sum_{i,j}(1/p[i]p[j]) \le m\sum_{a,b}(1/ab)=m(\sum_a(1/a))(\sum_b(1/b)) =\Theta(m\log^2 m)$ (1) (2) (3) (4) הסבר: כל הסכומים על i או j הם בתחום 0..m-1 וכל הסכומים על a או b הם בתחום 1..m. אי השוויון (1) הוא בגלל שבצד ימין סוכמים יותר אברים. אי השוויון (2) הוא בגלל שהמערך p ממויין וללא חזרות, לכן $p[i] \le i+1$ לכל i. השוויון (3) הוא זהות אלגברית. המעבר (4) משתמש ב $1+1/2+\ldots+1/m = \Theta(\log m)$.

```
void print_specials(int a[], int n, int m)
{
    int p[m+1], num;
    num=limit_and_sort(a,n,m,p);

    int to_print[m+1];

    for (int i=0; i<=m; i++)
        to_print[i]=0;

    for (int i=0; i<num; i++)
    {
        for (int j=i+1; j<num && p[i]*p[j]<=m; j++)
        {
            int jump=p[i]*p[j];
            while (jump <= m) {
                to_print[jump]=1; jump += p[i]*p[j];
            }
        }
    }

    for (int i=1; i<=m; i++)
        if (to_print[i])
            printf("%d ",i);
    printf("\n");
}
```

שאלה 3 (25 נקודות):

עליכם לממש את הפונקציה:

```
void swap_first_last(char arr[])
```

הפונקציה מקבלת כקלט מערך של תוים שמכיל **שלוש** מחרוזות אחת אחרי השניה. לדוגמא:

H	E	L	L	O	\0	M	Y	\0	W	O	R	L	D	!	\0

על הפונקציה להחליף בין המחרוזת הראשונה לשלישית. למשל, בדוגמא לעיל, לאחר הפעלת הפונקציה המערך יכיל את התוכן הבא:

W	O	R	L	D	!	\0	M	Y	\0	H	E	L	L	O	\0

דרישות: סיבוכיות זמן O(n) וסיבוכיות מקום נוסף O(1), כאשר `n` הוא מספר התוים הכולל במערך `arr`. שימו לב: תניחו שהמערך `arr` מכיל בדיוק **שלושה** תווי סיום מחרוזת `'\0'` והם ניספרים בהגדרה של n (בדוגמא לעיל, `n=16`).

מותר לכם להשתמש בפונקצית עזר `void reverse(char v[], int k);` שהופכת את סדר התווים במערך נתון `v` של `k` תווים. דוגמא: אם נגדיר

```
char v[4]={'a','b','\0','5'}
```

אז הפעלת `reverse(v, 4);` תשנה את תוכן המערך `v` ל: `{'5','\0','b','a'}`. הפונקציה reverse פועלת בסיבוכיות זמן `O(k)` ומקום נוסף `O(1)`. **אין צורך לממש אותה.**

אם לפי חישוביכם לא עמדתם בדרישות הסיבוכיות, אנא ציינו כאן את הסיבוכיות שהגעתם אליה: זמן _____ מקום נוסף _____

```
#define N 3

void swap_first_last(char arr[])
{
    int str_len[N], arr_len=0;
    int i=0, str_count=0, len_count=0;

    // Calculate string lengths
    while(str_count < N) {
        if (arr[i++]) {
            len_count++; continue;
        }
        str_len[str_count]=len_count+1;
        arr_len+=str_len[str_count];
        len_count=0;
        str_count++;
    }

    // Reverse the whole string
    reverse(arr, arr_len);

    // Reverse each one of the strings
    reverse(arr, str_len[2]);
    reverse(arr+str_len[2], str_len[1]);
    reverse(arr+str_len[2]+str_len[1], str_len[0]);
}
```

הסבר: ראשית, נחשב את גדלי שלושת המחרוזות, ואת אורך המערך כולו.

לאחר מכן, נבצע 4 פעולות היפוך: הראשונה תהפוך את סדר כל התוים במערך, ושלושת פעולות ההיפוך הבאות יהפכו את סדר התוים בשלושת המחרוזות. בשלושת פעולות ההיפוך האלה יש לכלול גם את תו סיום המחרוזת '\0' על מנת לשים אותם בסוף המחרוזות.

שאלה 4 (25 נקודות) :

קליקה היא קבוצה של אנשים שבה כל זוג אנשים הם חברים. נתון מערך דו מימדי של חברויות `int friends[N][N]`: עבור זוג אנשים `i` ו- `j`, הערך `friends[i][j]` הוא 1 אם `i` ו-`j` חברים, ו-0 אחרת. המערך סימטרי, כלומר מתקיים `friends[i][j]==friends[j][i]` לכל `i` ו- `j`.

עליכם לכתוב פונקציה

```
int hasClique(int friends[][N], int k)
```

המקבלת מערך חברויות `friends`, ומספר `k`. הפונקציה תחזיר 1 אם קיימת קליקה המכילה לפחות `k` אנשים, ו-0 אם לא קיימת קליקה כזו.

לדוגמא, עבור מטריצת החברויות הבאה:

	0	1	2	3
0		1	1	0
1	1		1	0
2	1	1		1
3	0	0	1	

קבוצת החברים {0,1,2} היא קליקה בגודל 3, וקבוצת החברים {2,3} היא קליקה בגודל 2, אך לא קיימת קליקה בגודל 4 (כי לדוגמא 3 ו-0 לא חברים).

הערות:

- `N` מוגדר ע"י `#define`.
- יש להשתמש בשיטת backtracking כפי שנלמדה בכיתה.
- בשאלה זו אין דרישות סיבוכיות, אולם כמקובל ב-backtracking יש לוודא שלא מתבצעות קריאות רקורסיביות מיותרות עם פתרונות שאינם חוקיים.

הפתרון מופיע בעמוד הבא.
הסבר: קיימות כמה דרכים לפתור את השאלה. בפתרון הנתון, נשתמש במערך עזר בו נסמן את החברים שנבחרו לקליקה, chosen. בכל צעד, נבדוק שני פתרונות אפשריים: הראשון הוא כאשר הבן אדם הנוכחי, curr, הוא חבר בקליקה, והשני – כאשר curr אינו חבר בקליקה. קודם נבדוק האם ניתן להרכיב את הפתרון השני, כאשר curr אינו חבר בקליקה. אם פתרון כזה קיים – נחזיר 1. אם הפתרון לא קיים, נבדוק האם ניתן להרכיב את הפתרון כאשר curr הוא חבר בקליקה. במקרה זה, נצטרך לוודא כי אכן ניתן להוסיף את curr לקליקה, על ידי בדיקה ש-curr הוא חבר של כל מי שכבר נמצא בקליקה. את הבדיקה האחרונה נעשה על ידי המעבר על השורה המתאימה של מטריצת החברויות friends. בכל קריאה רקורסיבית, k הינו מספר האנשים שיש עדיין להוסיף לקליקה. בסיס הרקורסיה: אם כבר הרכבנו את הקליקה (k==0) הפונקציה תחזיר 1. אם לא ניתן להוסיף עוד אנשים לקליקה (curr==N), הפונקציה תחזיר 0.

```
int clique_aux(int friends[N][N], int k, int chosen[], int curr)
{
    if(k==0) return 1;
    if(curr==N) return 0;

    // Without k-th person
    chosen[curr]=0;
    if (clique_aux(friends, k, chosen, curr+1))
        return 1;

    // With k-th person
    int i;
    for(i=0; i<curr; ++i){
        if (chosen[i] && friends[curr][i]==0)
            return 0;
    }
    chosen[curr]=1;
    return clique_aux(friends, k-1, chosen, curr+1);
}

int clique(int friends [N][N], int k){
    int chosen[N];
    return clique_aux(friends, k, chosen, 0);
}
```