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

סמסטר אביב תשע"ו

מבחן מסכם מועד א', 3 יולי 2016

2	3	4	1	1	

רשום/ה לקורס:

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 18 עמודים (4 שאלות) במבחן, כולל עמוד זה.
- וודאו שאתם נבחנים בקורס המתאים.
- 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.
- חובה לקרוא הוראות לכתיבת קוד המופיעות בעמוד הבא לפני פתרון המבחן.
- 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.
- נוהל "**לא יודע**": אם תכתבו **בצורה ברורה** "לא יודע/ת" על שאלה (או סעיף) שבה אתם נדרשים לקודד, תקבלו 20% מהניקוד. דבר זה מומלץ אם אתם יודעים שאתם לא יודעים את התשובה.
- שנוסחאות שימושיות:

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

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

צוות הקורס 234114/7

מרצים: פרופ"מ מירלה בן חן (מרצה אחראית), יעל ארז, איהאב ואתד

הנחיות לכתיבת קוד במבחן

- בכל השאלות, הנכם רשאים להגדיר ולממש פונקציות עזר כרצונכם. לנוחיותכם, אין חשיבות לסדר מימוש הפונקציות בשאלה, ובפרט ניתן לממש פונקציה לאחר השימוש בה (בלי צורך להצהיר על הפונקציה לפני). מותר להשתמש בפונקציה שנכתבה בסעיף אחר, בתנאי שתציינו באופן ברור איפה הפונקציה ממומשת.
- **חובה** להקפיד על תכנות מבני (כלומר, חלוקה נכונה לפונקציות).
- אלא אם כן נאמר אחרת בשאלות, **אין להשתמש בפונקציות ספריה או בפונקציות שמומשו בכיתה**, למעט פונקציות קלט/פלט והקצאת זיכרון (`malloc, free`) והפונקציה memcpy. ניתן להשתמש בטיפוס `bool` המוגדר ב-`stdbool.h`.
- חתימת הפונקציה memcpy: void memcpy(void *dest, void *src, unsigned size). שימו לב ש size הוא מספר **הבתים** שצריך להעתיק.
- אין להשתמש במשתנים סטטיים וגלובאליים אלא אם נדרשתם לכך מפורשות.

בהצלחה!

שאלה 1 (25 נקודות):

א. (5 נקודות) חשבו את סיבוכיות הזמן והמקום של הפונקציה f המוגדרת בקטע הקוד הבא, כפונקציה של n. אין צורך לפרט שיקוליכם. חובה לפשט את הביטוי ככל שניתן.

```
void f(int n, int arr[]){
    int i=0, j=0;
    for(; i < n; ++i)
       while(j < n && arr[i] < arr[j])
           j++;
}
```

סיבוכיות זמן: Θ (n) סיבוכיות מקום: Θ (1)

ב. (10 נקודות): חשבו את סיבוכיות הזמן והמקום של הפונקציות f1 ו f2 כפונקציה של n.

```
int f1(int n) {
      if (n <= 1) return n;
      return 2*f1(n-1);
}

int f2(int n) {
      if (n <= 1) return n;
      return f2(n-1) + f2(n-1);
}
```

סיבוכיות זמן f1 : Θ (n) סיבוכיות מקום f1 : Θ (n)

סיבוכיות זמן f2 : Θ (2^n) סיבוכיות מקום f2 : Θ (n)

ג. (10 נקודות):חשבו את סיבוכיות הזמן והמקום של הפונקציה f(), כפונקציה של m ו n.
ניתן להניח שסיבוכיות הזמן של malloc היא Θ(1).

```
void f(int n, int m){
    if (n <= 1) {
          int *arr=malloc(m*sizeof(int));
          for (int i=0; i<m; i++) arr[i] = 0;
          free(arr);
          return;
    }
    f(n-1, m+1);
    f(n%2, m+1);
}
```

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

שאלה 2 (25 נקודות) :

עליכם לממש את הפונקציה:

```
void tag_place(int arr[], int n, int place[]);
```

הפונקציה מקבלת מערך arr באורך n שכל האיברים בו שונים זה מזה. הפונקציה רושמת במערך place לכל איבר את המיקום שלו במערך הממוין (מ-0 עד n-1).

למשל עבור המערך arr:

10	0	5	-3	20

המערך place צריך להיות:

3	1	2	0	4

דרישות: סיבוכיות זמן O(nlog(n)) וסיבוכיות מקום נוסף O(n).

- הפונקציה אינה יכולה לשנות את המערך arr.
- מותר להשתמש בפונקציות עזר.

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

```
void mem_cpy(int dst[], int src[], int n) {
      for (int i = 0; i < n; i++)
            dst[i] = src[i];
}

void merge(int a[], int na, int b[], int nb, int c[]) {
      int ia = 0, ib = 0, ic = 0;
      while (ia < na || ib < nb) {
            if (ia >= na)
                  c[ic++] = b[ib++];
            else if (ib >= nb)
                  c[ic++] = a[ia++];
            else if (a[ia] < b[ib])
                  c[ic++] = a[ia++];
            else
                  c[ic++] = b[ib++];
      }
}
```

```
void merge_sort_aux(int arr[], int n, int* helper) {
	if (n <= 1)
		return;
	int len = n / 2;
	merge_sort_aux(arr, len, helper);
	merge_sort_aux(arr + len, n - len, helper);
	merge(arr, len, arr + len, n-len, helper);
	mem_cpy(arr, helper, n);
}

void merge_sort(int arr[], int n) {
	int* helper = (int*)(malloc(sizeof(int)*n));
	if (!helper)
		return;
	merge_sort_aux(arr, n, helper);
}

int bin_search(int arr[], int n, int x) {
	int l = 0, r = n - 1;
	while (l <= r) {
		int m = (l + r) / 2;
		if (arr[m] > x)
			r = m - 1;
		else if (arr[m] < x)
			l = m + 1;
		else
			return m;
	}
	return -1;
}

void tag_place(int arr[], int n, int place[]) {
	int* helper = (int*)malloc(sizeof(int)*n);
	if (!helper)
		return;
	mem_cpy(helper, arr, n);
	merge_sort(helper, n);
	for (int i = 0; i < n; i++)
		place[i] = bin_search(helper, n, arr[i]);
}
```

שאלה 3 (25 נקודות) : מחרוזות

עליכם לממש את הפונקציה:

```
int count_mixed_str(char *s1, char* s2)
```

הפונקציה מקבלת שתי מחרוזות המכילות **אותיות אנגליות קטנות בלבד**, וצריכה להחזיר כמה פעמים המחרוזת s1 מופיעה במחרוזת s2 **ברצף אך ללא התחשבות בסדר האותיות של s1**. אין לשנות את המחרוזות s1, s2.

לדוגמא:

- עבור s1 = "abc", s2 = "abcadb" הפונקציה תחזיר 2, שני המופעים של s1 ב – s2 הם: **abc**adb ו – a**bca**db.
- עבור s1 = "hi", s2 = "night" הפונקציה תחזיר 0, כי האותיות h ו – i לא מופיעות סמוכות.
- עבור s1 = "bba", s2 = "abba" הפונקציה תחזיר 2, שני המופעים של s1 ב – s2 הם: **abb**a ו – a**bba**. שימו לב שכמו בדוגמא הזאת, ייתכן שבמחרוזות יהיו כמה מופעים של אותה אות.

דרישות: סיבוכיות זמן O(m+n) כאשר s1 באורך m ו – s2 באורך n, וסיבוכיות מקום נוסף O(1).

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

```
#define NUM_LETTERS 26

bool cmp_hists(int* h1, int* h2)
{
	for (int i = 0; i < NUM_LETTERS; i++)
	{
		if (h1[i] != h2[i])
			return false;
	}
	return true;
}
```

```
int count_mixed_str(char *s1, char *s2) {
	int hist_1[NUM_LETTERS] = { 0 };
	int hist_2[NUM_LETTERS] = { 0 };

	int l1 = 0, l2 = 0;
	while (s1[l1])
		l1++;
	while (s2[l2])
		l2++;

	if (l1 > l2)
		return 0;

	// here we know s1 is not longer than s2
	for (int i = 0; i < l1; i++)
	{
		hist_1[s1[i] - 'a']++;
		hist_2[s2[i] - 'a']++;
	}

	int cnt = cmp_hists(hist_1, hist_2);

	for (int i = 0; i < l2 - l1; i++)
	{
		hist_2[s2[i]-'a']--;
		hist_2[s2[i + l1]-'a']++;
		if (cmp_hists(hist_1, hist_2))
			cnt++;
	}
	return cnt;
}
```

שאלה 4 (25 נקודות) :

בהנתן מטריצה ריבועית בגודל קבוע NxN אשר מכילה מספרים שלמים חיוביים בלבד, נגדיר "מסלול" כאוסף של תאים סמוכים. תאים סמוכים יכולים להיות שכנים מימין, משמאל, מלמעלה או מלמטה (לא באלכסון).

עליכם לממש את הפונקציה:

```
bool find_path_sum(int mat[N][N], int sum, int path[N][N]);
```

אשר מקבלת מטריצה ריבועית בגודל קבוע NxN (ניתן להניח ש – N מוגדר ב - #define), אשר מכילה מספרים שלמים חיוביים (גדולים ממש מ – 0), סכום חיובי sum, ומטריצה ריבועית בגודל NXN. הפונקציה צריכה להחזיר true במידה וקיים "מסלול" במטריצה mat עם הסכום sum ו – false אחרת. שימו לב שכל תא במסלול נספר פעם אחת.

המטריצה path, בגודל NxN, משמשת לסימון מסלול עם הסכום המבוקש:

אם נמצא מסלול עם הסכום המבוקש, path תכיל את הערך 1 בתאים אשר משתתפים במסלול ואפסים בשאר התאים. אם לא נמצא כזה מסלול path צריכה להכיל אפסים בלבד בסיום ריצת הפונקציה. אם קיים יותר ממסלול אחד, path תכיל את אחד המסלולים עם הסכום המבוקש.לדוגמא, בהינתן המטריצה mat הבאה:

2	41	3	15
1	2	4	6
7	8	10	54
63	22	1	4

והסכום 4, הפונקציה תחזיר true והמטריצה path יכולה להיות אחת משתי האפשרויות הבאות:

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	1

0	0	0	0
0	0	1	0
0	0	0	0
0	0	0	0

עבור הסכום 9 לעומת זאת, הפונקציה תחזיר false ו – path תכיל אפסים בלבד.

עבור הסכום 3, הפונקציה תחזיר true, והמטריצה path יכולה להיות אחת מ-3 האפשרויות הבאות:

0	0	1	0
0	0	0	0
0	0	0	0
0	0	0	0

0	0	0	0
1	1	0	0
0	0	0	0
0	0	0	0

1	0	0	0
1	0	0	0
0	0	0	0
0	0	0	0

הערות:

- יש להשתמש בשיטת backtracking כפי שנלמדה בכיתה.
- אסור לשנות את תוכן המערך mat.
- **מותר** להשתמש בלולאות.
- בשאלה זו אין דרישות סיבוכיות, אולם כמקובל ב – backtracking יש לוודא שלא מתבצעות קריאות רקורסיביות מיותרות עם פתרונות שאינם חוקיים.
- מותר להשתמש בפונקציות עזר.

```
#define USED 1
#define UNUSED 0
// N can get any value, 5 is just an example
#define N 5
bool find_path_sum(int mat[N][N],int sum,int path[N][N])
{
    int i,j;
    clear_path(path);
    //We need to check each of the possible starting points
    for(i=0;i<N;i++) {
        for(j=0;j<N;j++) {
            if(find_aux(mat,sum,path,i,j)) {
                return true;
            }
        }
    }
    return false;
}

bool find_aux(int mat[N][N],int sum,int path[N][N],int i,int j)
{
    //This condition must be the first one
    if(sum==0) return true;
    if((sum<0)||(path[i][j]=USED)||out_of_bounds(i,j))
    {
        return false;
    }
    path[i][j]=USED;
    if(find_aux(mat,sum-mat[i][j],path,i+1,j) || find_aux(mat,sum-
mat[i][j],path,i-1,j)
       find_aux(mat,sum-mat[i][j],path,i,j+1) || find_aux(mat,sum-
mat[i][j],path,i,j-1))
    {
           return true;
    }
    //We need to clear the path[i][j] for future solutions
    path[i][j]=UNUSED;
    return false;
}
```

```
bool out_of_bounds(int i,int j)
{
    return ((i<0)||(j<0)||(i>=N)||(j>=N));
}

void clear_path(int path[N][N])
{
    int i,j
    for(i=0;i<N;i++)
    {
        for(j=0;j<N;j++)
        {
            path[i][j]=0;
        }
    }
}
```