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

סמסטר חורף תשע"ו

מבחן מסכם מועד א', 16 פברואר 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.
- 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"מ איתן יעקובי (מרצה אחראי), ד"ר אנסטסיה דוברובינה, פרופ"ח תומר שלומי.

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). אם כתבתם פונקציה שאורכה יותר מ 22 שורות, זוהי אינדיקציה ברורה לכך שיש לפרק את הפתרון לפונקציות עזר. אורך הפונקציה נמדד בהתאם להנחיות שניתנו בשיעורי בית. **אין חובה** להקפיד על פונקציות קצרות בשאלות backtracking (אם כי, זה עדיין עשוי לעזור לכם).
- 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 *desc, void *src, unsigned size). שימו לב ש size הוא מספר **הבתים** שצריך להעתיק.
- אין להשתמש במשתנים סטטיים וגלובאליים אלא אם נדרשתם לכך מפורשות.
- כאשר נדרש לבצע מיון בסיבוכיות O(nlogn), ניתן להשתמש במיון מהיר (quicksort).

בהצלחה!

שאלה 1 (25 נקודות):

א. (8 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 i=n*n;
    while(n){
        n/=2;
        i+=5;
    }
}
```

סיבוכיות זמן: Θ ($logn$) סיבוכיות מקום: Θ (1)

ב. (9 נקודות): חשבו את סיבוכיות הזמן והמקום של הפונקציה f.

```
void f(int arr[], int n){
      int m=0;
      for(int i=1; i<n*n; ++i)
            for(int k=0; k<n*n; k+=i){
                  m=(m>k)?m:k;
                  printf("1");
            }
      free(malloc(m));
}
```

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

ג. (8 נקודות):חשבו את סיבוכיות הזמן והמקום של הפונקציה f.

```
int f(int n){
    if(n<=0) return 1;
    return f(n-2)+f(n-2)+f(n-2);
}
```

סיבוכיות זמן: Θ ($3^{\frac{n}{2}}$) סיבוכיות מקום: Θ (n)

שאלה 2 (25 נקודות) :

ממשו את הפונקציה הבאה, שחתימתה

```
int findShifted(int a[], int n, int x);
```

הפונקציה מקבלת כקלט מערך a של מספרים שונים זה מזה באורך n ומספר נוסף x.
על הפונקציה להחזיר את האינדקס של המספר x במערך a או 1- במידה ולא נמצא.

ידוע לנו שהמערך a הוא מערך ממוין-מוסט , כלומר מערך ממוין שכל איבר בו הוסט i איברים אחורה (כש i האיברים האחרונים מועברים לסוף המערך בהתאמה).
כמו כן , ידוע כי i תחום בין 1 ל-n1 אך פרט לכך אינכם יודעים מהו.

לדוגמה נסתכל על המערך הממוין הבא:

1	2	3	4	5

מערך ממוין-מוסט-2 שלו יראה כך:

3	4	5	1	2

דרישות: סיבוכיות זמן O(log(n)) וסיבוכיות מקום נוסף O(1).
הערות:

- ניתן להניח שכל האיברים במערך שונים זה מזה.
- ניתן להשתמש בפונקציה חיפוש בינארי שנלמדה בכיתה. הפונקציה פועלת רק על מערכים ממויינים באופן רגיל (בפרט, לא על מערך ממויין מוסט). הפונקציה מחזירה את האינדקס של המספר x במערך או -1 במידה ולא נמצא. חתימת הפונקציה:

int binarySearch(int a[], int n, int x);

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

```
int findShifted (int arr[], int n, int x)
{
    int low = 0, high = n-1, mid=0;
    while (low <= high) {
        mid = (low+high)/2;
        if (arr[n-1] <= arr[mid]) {
            low = mid+1;
```

```
        }
        else if (arr[0] >= arr[mid]) {
            high = mid;
        }
        If(arr[mid]<arr[mid-1]) {
            break;
        }
    }
    int index = binarySearch(arr, mid, x);
    if (index != -1) {return index;}
    index = binarySearch(arr+mid, n-mid, x);
    return (index==-1)?-1:index+mid;
}
```

שאלה 3 (25 נקודות) : דחיסת מחרוזות

בשאלה זו עליכם לכתוב פונקציה אשר מקבלת מחרוזת המכילה אותיות אנגליות קטנות בלבד, ודוחסת אותה. הפונקציה תמצא את זוג האותיות השונות שמופיעות ביחד (אחת אחרי השניה) הכי הרבה פעמים, ותחליף אותם בתו המיוחד '!'. יש לכתוב את הזוג שנדחס למחרוזת pair שמועברת כפרמטר.

אם יש כמה זוגות שמופיעים אותה כמות של פעמים, יש לדחוס את אחד הזוגות (לא חשוב איזה זוג).

חתימת הפונקציה:

```
void CompressString(char *str, char *pair);
```

לדוגמה: במחרוזת "abababcd" הזוג "ab" הופיע 3 פעמים, הזוג "ba" הופיע פעמיים, ושאר הזוגות הופיעו פעם אחת. לכן יש לדחוס את הזוג "ab". בסיום הפונקציה, המחרוזת str תכיל: "!!!cd", והמחרוזת pair תכיל "ab".

במחרוזת "ababa" הזוג "ab" והזוג "ba" מופיעים אותה כמות של פעמים (2). לכן ניתן לדחוס איזה שרוצים. בסיום הפונקציה או ש "str="!!a", pair="ab", או ש "str="a!!", pair="ba".

דרישות סיבוכיות: עבור מחרוזת באורך n, סיבוכיות זמן O(n), סיבוכיות מקום נוסף O(1)

הערות:

- אין לדחוס זוגות שמכילים אותה אות פעמיים. המחרוזת "aaaaaaa" לא תדחס כלל.
- יש חשיבות לסדר: הזוג "ab" שונה מהזוג "ba".
- הניחו שניתן לשנות את המחרוזת str ו pair, וכן ש pair מספיק ארוכה כדי להכניס מחרוזת בעלת 2 תווים.
- הניחו שהמחרוזת מכילה אותיות באנגלית קטנות בלבד.

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

```
#define LETTERS ('z'-'a'+1)
int tonum(char c){return c-'a';}
bool frequentPair(char *str, char *pair){
    int hist[LETTERS][LETTERS]={{0}};
    while(str[0]!='\0' && str[1]!='\0'){//fill hist
        if(str[0]!=str[1])
            hist[tonum(str[0])][tonum(str[1])]++;
        str++;
    }
    int max=0, max_i=0, max_j=0;
    for(int i=0; i<LETTERS; ++i)//find max pair in hist
        for(int j=0; j<LETTERS; ++j)
```

```
            if(max<hist[i][j]){
                max=hist[i][j];
                max_i=i;
                max_j=j;
            }
    if(max==0) return false;//no pair
    pair[0]=max_i+'a';
    pair[1]=max_j+'a';
    pair[2]='\0';
    return true;
}
void compressPair(char *str, char *pair){
    char *out = str;
    while(str[0]!='\0' && str[1]!='\0'){
        if(str[0]==pair[0] && str[1]==pair[1]){
            *out='!';
            str+=2;
        }
        else{
            out[0]=str[0];
            str++;
        }
        out++;
    }
    out[0]=str[0];
    if(out[0]!='\0')
        out[1]='\0';
}
void compressString(char *str, char *pair){
    if(frequentPair(str, pair))
       compressPair(str, pair);
```

}

שאלה 4 (25 נקודות) :

בגרסה מעט שונה של תפזורת הנכם נדרשים לחפש מילה בודדת בלוח המורכב מאותיות, כך שהמילה יכולה להופיע לא בהכרח בצורה מאוזנת/מאונכת לאורך כל המילה, אלא בצורה מעורבת, ובתנאי שהכיוונים יהיו משמאל לימין, מימין לשמאל, מלמטה למעלה או מלמעלה למטה. **אסור להשתמש פעמיים באותה אות.**

a	q	e	t	o
s	o	t	a	m
n	o	**w**	**k**	e
c	v	**a**	**h**	h
t	u	i	z	b

לדוגמא:

המילה snow היא מילה בלוח המתחילה בכיוון מאונך ובאמצע עוברת למאוזן. המילה hawk מתחילה בכיוון מאוזן, עוברת למאונך, ומסיימת במאוזן.

עליכם לכתוב פונקציה

```
bool searchWord(char board[M][M], char* word, int n);
```

המקבלת את לוח המשחק board, ומחרוזת word ומחזירה true אם המחרוזת מופיעה בלוח. אחרת, הפונקציה מחזירה false.

לדוגמא: עבור הלוח הנ"ל ועבור המחרוזת "snow" הפונקציה תחזיר true.
עבור הלוח הנ"ל ועבור המחרוזת "hawk" הפונקציה תחזיר true.
עבור הלוח הנ"ל ועבור המחרוזת "carrot" הפונקציה תחזיר false.

הערות:

- `M` מוגדר ע"י `#define`.
- יש להשתמש בשיטת backtracking כפי שנלמדה בכיתה.
- **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 VISITED '+'
bool searchWord(char board[M][M], char* word)
{
    for (int i=0; i<M; i++) {
        for (int j=0; j<M; j++) {
            if (searchWordAux(board, word, i, j) {
                return true;
```

```
            }
        }
    }
    return false;
}

bool inboard(int i, int j) {
    return (i<M && j<M && i>=0 &&j>=0);
}

bool searchWordAux(char board[M][M], char* word, int i, int j)
    if(!*word) {
        Return true;
    }
    if(!inBoard(i,j) || board[i][j] != *word) {
        return false;
    }
    char currLetter = board[i][j];
    board[i][j] = VISITED;
    if (searchWordAux(board, word+1, i+1, j) ||
        searchWordAux(board, word+1, i-1, j) ||
        searchWordAux(board, word+1, i, j+1) ||
        searchWordAux(board, word+1, i, j-1)) {
        board[i][j] = currLetter;
        return true;
    }
    board[i][j] = currLetter;
    return false;
}
```
