הצעת פתרון לבחינת הבגרות

מדעי המחשב ב' (899205) - קיץ תשס"ג

פרק ראשון – עיצוב תכנה

הקדמה

להלן פתרון אפשרי לבחינת הבגרות ביח"ל 5-4 (מספר טופס 899205) שנערכה ביום 10.6.2003 (י' בסיון תשס"ג). הפתרון נכתב ע"י אתי מנשה ואיתן ראט. אין הכותבים אחראים לפתרון בכל דרך שהיא, והוא מוגש כעזר למורה ולתלמיד.

הפתרון נכתב בשתי סביבות העבודה המותרות בתכנית הלימודים – פסקל ו C. לפיכך, כל פתרון בסביבת העבודה מוגש בשתי הגרסאות. לעיתים, מובאות מספר אפשרויות לפתרון, וכל אחת מהן מובאת בשתי הגרסאות – פסקל ו C.

חשוב מאד לשים לב להלך החשיבה שמאחורי הפתרון. לא תמיד השאלה מנוסחת בצורה מושלמת, ולתלמיד יש קושי לעיתים להחליט מה צריך לעשות. לכן, הקפדנו להסביר את צורת החשיבה הנכונה בזמן פתרון התרגיל. בדרך כלל, גם אם השאלה לא מנוסחת בצורה מושלמת, ניתן להגיע להבנה טובה למדי ולפתרון מלא, אם קוראים כראוי, מבינים את הנקרא, ומנתחים אותו בצורה נכונה. הבנת הנקרא וניתוח הבעיה הם חלק בלתי נפרד מפתרון בעיות במדעי המחשב, ולכן על כל מורה לשים על כך דגש בזמן ההוראה בכיתה.

אנו מאחלים לכל המורים והתלמידים הצלחה רבה!

אתי ואיתן.

שאלה 1

נחזיק שני מצביעים: אחד (p) עבור L ואחד (q) עבור L2 (הרשימה החדשה).
נתקדם על L עם p. בכל פעם שהאיבר הנוכחי יהיה גדול מהקודם, מחברים אותו לסכום המצטבר (Sum).
אם האיבר הנוכחי קטן או שווה לקודם, אז מכניסים את הסכום שהצטבר עד כה ל L2, ומאתחלים אותו לערכו של האיבר הנוכחי. כך עד שהאיבר הנוכחי הוא סוף רשימה.
יש לזכור להוסיף את הסכום האחרון ל L2, שכן הרצף האחרון לא מסתיים באיבר הקטן מקודמו, אלא ע"י סוף רשימה, ולכן לא הוסף עדיין ל L2.

להלן הפתרון:

כתבו וערכו: איתן ראט ואתי מנשה. כל הזכויות שמורות ©. כל שימוש/העתקה/צילום/שכפול /ציטוט של עמוד זה או של חלקו למטרות מסחריות אסורים בהחלט ללא רשות בכתב מאיתן ראט ואתי מנשה.

Pascal

```
Procedure ascending_Lengths (L : List_Type; Var L2: List_Type);
{ Creates another list of values, that: for every sub-list
  of ascending values in L there will be one number in the new
  list, which is the sum of all those ascending values. The order
  of the values (sums) in the new list will match the order of the
  ascending sublists in L.
  Assumption: L is not empty. L2 is initialized.}
Var
    p      : Pos_Type;                  { pointer on current item in L          }
    X_Prev : List_Info_Type;            { Value of previous item                }
    X_Curr : List_Info_Type;            { Value of current item                 }
    Sum    : List_Info_Type;            { Current sum of ascending sublist      }
    q      : Pos_Type;                  { Points to the last item in L2         }
Begin
    p := List_Next(L, List_Anchor(L));
    List_Retrieve(L, p, X_Prev);        { L is NOT empty by assumption          }
    Sum := X_Prev;                      { Current sum is the first item         }
    p := List_Next(L, p);               { Second item in L                      }
    q := List_Anchor(L2);               { insertion into L2 is after r          }

    While (p <> List_End(L)) Do
    Begin
        List_Retrieve (L, p, X_Curr);
        If (X_Curr > X_Prev) Then       { Still ascending                       }
            Sum := Sum + X_Curr
        Else                            { First item with smaller value         }
            Begin
                List_Insert(L2, q, Sum);
                r := List_Next(L2, q);
                Sum := X_Curr;
            End;
        X_Prev := X_Curr;
        p  := List_Next(L, p);
    End;

    List_Insert (L2, q, Sum);           { The last sum has also to be in L2     }
End;
```

כתבו וערכו: איתן ראט ואתי מנשה. כל הזכויות שמורות ©. כל שימוש/העתקה/צילום/שכפול /ציטוט של עמוד זה או של חלקו למטרות מסחריות אסורים בהחלט ללא רשות בכתב מאיתן ראט ואתי מנשה.

C

```
list_type ascending_lengths(list_type l)
/* Creates another list of values, that: for every sub-list
   of ascending values in L there will be one number in the new
   list, which is the sum of all those ascending values. The order
   of the values (sums) in the new list will match the order of the
   ascending sublists in L.
   Assumption: L is not empty. */
{
        list_type l2 = list_init();                    // The new list that will be returned
        pos_type p=list_next(l, list_anchor(l));       // pointer on current item in L
        list_info_type x_prev = list_retrieve(l, p);   // Value of previous item. L isn't empty!
        list_info_type sum = x_prev;                   // Current sum of ascending sublist
        p = list_next(l, p);
        pos_type q = list_anchor(l2);                  // Points to the last item in L2
        list_info_type x_curr;                         // Value of current item

        while (p != list_end(l))
        {
                x_curr = list_retrieve(l, p);
                if (x < y)                             // Still ascending
                        sum += y;
                else                                   // First item with smaller value
                {
                        list_insert(&l2, &q, sum);
                        q = list_next(l2, q);
                        sum = x_curr;
                }
                x_prev = x_curr;
                p = list_next(l, p);
        }
        list_insert(&l2, &q, sum);                     // The last sum has also to be in L2
        return l2;
}
```

כתבו וערכו: איתן ראט ואתי מנשה. כל הזכויות שמורות ©. כל שימוש/העתקה/צילום/שכפול /ציטוט של עמוד זה או של חלקו למטרות מסחריות אסורים בהחלט ללא רשות בכתב מאיתן ראט ואתי מנשה.

שאלה 2

יש לכתוב פונקציה בוליאנית. ניתן לפתור בדרכים רבות. בכולן יש לזמן את הפונקציה עם פרמטר נוסף אחד לפחות. מכיון שאין זה מעניינו של המשתמש שלצורך הפתרון דרושים פרמטרים נוספים, הרי שיש להשתמש בפונקציית מעטפת, שתקבל רק את המערך, ותעביר את הפרמטרים הנוספים לפונקציה הרקורסיבית הדרושה בשאלה.

אם נסמן ב I את מיקומו של איבר מתחילת המערך, אז מיקומו של האיבר המתאים לו מחציו השני של המערך בפסקל הוא N+1-I, וב C הוא N-I, מכיון שב C האינדקס מתחיל מאפס ולא מ 1.
למשל, אם N = 10, אז האיבר המתאים לאיבר השלישי הוא:
בפסקל: האיבר ה 10+1-3=8, כלומר, A[8] מתאים ל A[3].
ב C: האיבר ה 10-3 = 7, כלומר, a[7] מתאים ל a[2], שהוא האיבר השלישי במערך.

לזוג איברים מתאימים שכאלה נקרא "הזוג הנוכחי". הזוג הראשון במערך, בפסקל, הוא, A[1] ו- A[N], ובשפת C הוא a[0] ו- a[N-1].

<u>אפשרות א'</u>: הסכום של כל זוג נוכחי צריך להיות קבוע, ושווה לזה של (בין היתר) הזוג הראשון. נחשב את הסכום של הזוג הראשון, ונעביר אותו כפרמטר לשגרה הרקורסיבית. אפשר להתחיל מבדיקת הזוג השני, ולא הראשון (כי סכום הזוג הראשון בודאי שווה לעצמו **J**).

<u>אפשרות ב'</u>: כמו באפשרות א', אלא שכאן נשתמש בסדר הביצוע של תנאים, על מנת ליצור קוד יעיל יותר וגם קצר יותר. המקרה הפשוט (כאשר i עבר את האמצע) ישולב בתנאי המורכב כתנאי הראשון. אם הוא יתקיים, אז יוחזר ערך 'אמת' ללא בדיקת המשך התנאי. רק אם i עדיין לא עבר את האמצע, המשך התנאי ייבדק, וסכום הזוג הנוכחי יושווה לסכום הקבוע. רק אם הסכומים יהיו שווים, תופעל הקריאה הרקורסיבית להמשך הבדיקה.

<u>אפשרות ג'</u>: לא לחשב את הסכום מראש, אלא לבדוק אם סכום הזוג הנוכחי שווה לסכום הזוג הפנימי לו. מאחר ויש וכל זוג ייבדק פעמיים (גם כלפי החיצוני לו וגם כלפי הפנימי לו), יוצא שאם הכל יעבור בשלום (כלומר, i יגיע לאמצע), אז סימן שכל הסכומים היו זהים. במקרה זה אין צורך להמשיך עם i עד האמצע ממש, כי הזוג האמצעי כבר מחושב כאשר הזוג הקודם נבדק.

כתבו וערכו: איתן ראט ואתי מנשה. כל הזכויות שמורות ©. כל שימוש/העתקה/צילום/שכפול /ציטוט של עמוד זה או של חלקו למטרות מסחריות אסורים בהחלט ללא רשות בכתב מאיתן ראט ואתי מנשה.

פתרון בהתאם לאפשרות א':

Pascal

```
Program Q2_Bagrut_899205_2003;
Const
   N = 10;

Type
   Arr_Type = Array[1..N] of Integer;

Var
   A : Arr_Type;

   {=====================================================}
   Procedure Get_Values (Var A : Arr_Type);
   Var
      I : Integer;
   Begin
      For I := 1 To N Do
         Begin
            Write('A[',I:2,'] = ');
            Readln(A[I]);
         End;
   End;

   {=====================================================}
   Function Symmetrical_Sum (Var A : Arr_Type; I : Integer; Sum : List_Info_Type) : Boolean;
   Begin
      If (I > N div 2) Then
            Symmetrical_Sum := True
      Else
            Symmetrical_Sum := (A[I]+A[N+1-I] = Sum) And (Symmetrical_Sum(A, I+1, Sum));
   End;

   {=====================================================}
   Function Is_Symmetrical_Sum (Var A : Arr_Type) : Boolean;
   Begin
      Is_Symmetrical_Sum := Symmetrical_Sum (A, 2, A[1]+A[N]);
   End;

   {=====================================================}
Begin
   Get_Values(A);
   Writeln(Is_Symmetrical_Sum(A));
End.
```

כתבו וערכו: איתן ראט ואתי מנשה. כל הזכויות שמורות ©. כל שימוש/העתקה/צילום/שכפול /ציטוט של עמוד זה או של חלקו למטרות מסחריות אסורים בהחלט ללא רשות בכתב מאיתן ראט ואתי מנשה.

C

```c
#include <stdio.h>
#define N 10

void get_values(int a[]);
int is_symmetrical_sum (int a[]);
int symmetrical_sum (int a[],int i,int sum);

//==================================================
void main()
{
	int a[N];

	get_values(a);
	is_symmetrical_sum(a) ? printf("yes") : printf("no");
}

//==================================================
void get_values (int a[])
{
	int i;

	for (i=0; i<N; i++)
		scanf("%d", &a[i]);
}

//==================================================
int is_symmetrical_sum (int a[])
{
	return symmetrical_sum(a, 1, a[0]+a[N-1]);
}

//==================================================
int symmetrical_sum (int a[], int i, int sum)
{
	if (i > N/2-1)
		return 1;
	else
		return ((a[i]+a[N-1-i] == sum) && (symmetrical_sum(a, i+1, sum)));
}
```

כתבו וערכו: איתן ראט ואתי מנשה. כל הזכויות שמורות ©. כל שימוש/העתקה/צילום/שכפול /ציטוט של עמוד זה או של חלקו למטרות מסחריות אסורים בהחלט ללא רשות בכתב מאיתן ראט ואתי מנשה.

בשני הפתרונות הבאים לא נכתוב מחדש את התוכנית כולה, אלא רק את הפונקציה הרקורסיבית, Symmetrical_Sum, ואת פונקציית המעטפת שלה, Is_Symmetrical_Sum. כל שאר התכנית נשאר בדיוק כפי שהיה בפתרון הקודם.

פתרון בהתאם לאפשרות ב':

כמו באפשרות א', אלא שכאן נשתמש בסדר הביצוע של תנאים, על מנת ליצור קוד יעיל יותר וגם קצר יותר. המקרה הפשוט (כאשר i עבר את האמצע) ישולב בתנאי המורכב כתנאי הראשון. אם הוא יתקיים, אז יוחזר ערך 'אמת' ללא בדיקת המשך התנאי. רק אם i עדיין לא עבר את האמצע, המשך התנאי ייבדק, וסכום הזוג הנוכחי יושווה לסכום הקבוע. רק אם הסכומים יהיו שווים, תופעל הקריאה הרקורסיבית להמשך הבדיקה.

Pascal

```
Function Symmetrical_Sum (Var A : Arr_Type; I : Integer; Sum : List_Info_Type) : Boolean;
Begin
      Symmetrical_Sum := (I > N div 2) Or
                                 (A[I]+A[N+1-I] = Sum) And (Symmetrical_Sum (A, I+1, Sum));
End;

{=================================================}
Function Is_Symmetrical_Sum (Var A : Arr_Type) : Boolean;
Begin
       Is_Symmetrical_Sum := Symmetrical_Sum (A, 2, A[1]+A[N]);
End;
```

C

```
int is_symmetrical_sum (int a[])
{
       return symmetrical_sum (a, 2, a[0]+a[N-1]);
}

//=================================================
int symmetrical_sum (int a[], int i, list_info_type sum)
{
       return ((i > N / 2 - 1) || (a[I]+a[N-I] == sum) && (symmetrical_sum (a, I+1, sum)));
}
```

כתבו וערכו: איתן ראט ואתי מנשה. כל הזכויות שמורות ©. כל שימוש/העתקה/צילום/שכפול /ציטוט של עמוד זה או של חלקו למטרות מסחריות אסורים בהחלט ללא רשות בכתב מאיתן ראט ואתי מנשה.

פתרון בהתאם לאפשרות ג':

לא לחשב את הסכום מראש, אלא לבדוק אם סכום הזוג הנוכחי שווה לסכום הזוג הפנימי לו. מאחר ויש וכל זוג ייבדק פעמיים (גם כלפי החיצוני לו וגם כלפי הפנימי לו), יוצא שאם הכל יעבור בשלום (כלומר, i יגיע לאמצע), אז סימן שכל הסכומים היו זהים. במקרה זה אין צורך להמשיך עם i עד האמצע ממש, כי הזוג האמצעי כבר מחושב כאשר הזוג הקודם נבדק.

Pascal

```
Function Symmetrical_Sum (Var A : Arr_Type; I : Integer) : Boolean;
Begin
    Symmetrical_Sum := (I > N div 2-1) Or
                        (A[I]+A[N+1-I] = A[I+1]+A[N-I]) And (Symmetrical_Sum (A, I+1));
End;


{=================================================}
Function Is_Symmetrical_Sum (Var A : Arr_Type) : Boolean;
Begin
    Is_Symmetrical_Sum := Symmetrical_Sum (A, 1);
End;
```

C

```
int is_symmetrical_sum (int a[])
{
    return symmetrical_sum (a, 0);
}


//=================================================
int symmetrical_sum (int a[], int i, list_info_type sum)
{
    return ((i > N / 2 - 2) || (a[i]+a[N-i] == a[i+1]+a[N-i-1]) && (symmetrical_sum (a, i+1)));
}
```

כתבו וערכו: איתן ראט ואתי מנשה. כל הזכויות שמורות ©. כל שימוש/העתקה/צילום/שכפול /ציטוט של עמוד זה או של חלקו למטרות מסחריות אסורים בהחלט ללא רשות בכתב מאיתן ראט ואתי מנשה.

שאלה 3

א. האלגוריתם סוד3 הוא פעולה בוליאנית. יש רק אפשרות אחת מפורשת להחזרת 'אמת' באלגוריתם זה, והיא כאשר האלגוריתם הגיע לעץ ריק. על מנת להגיע לעץ ריק, האלגוריתם צריך להגיע כל פעם להוראה (4), כדי שיוכל לרדת לרמה הבאה באמצעות הקריאה הרקורסיבית. אבל על מנת להגיע להוראה (4), ערכו של הצומת הנוכחי (T) צריך להיות שונה מ X. בנוסף, מאחר והוראה (4) היא תנאי "וגם", הרי שכל הזימונים הרקורסיביים חייבים להסתיים בתוצאה 'אמת' על מנת שהאלגוריתם סוד3 יחזיר 'אמת'. משמעות הדבר היא, שלכל אורך ריצתו, על האלגוריתם להגיע תמיד לעץ ריק, ומכאן, שערכו של כל צומת שנסרק בעץ חייב להיות שונה מ- X. האלגוריתם סורק את כל צמתי העץ (בסדר תחילי). לכן:

האלגוריתם סוד3 מחזיר 'אמת' אם הערך X לא נמצא בעץ T, ו- 'שקר' אחרת.

ב. האלגוריתם סוד1 סורק את העץ T בסדר תחילי.

עבור כל צומת, האלגוריתם בודק אם ערכו אינו מופיע בתת-העץ השמאלי שלו (ע"י הפעלת סוד3 על תת-העץ השמאלי וערכו של הצומת) וגם לא בתת-העץ הימני שלו (גם בעזרת סוד3). אם אכן ערכו של הצומת לא מופיע שנית בתת-עץ שהוא שורשו, אז ערכו מוכנס לרשימה L, מיד לאחר עוגן הרשימה. לכן:

האלגוריתם סוד2 מוסיף לרשימה L את כל ערכי העץ הבינארי T, אשר אינם מופיעים פעם נוספת בתת-עץ שהם שורשו.

האלגוריתם סוד1 מאתחל את הרשימה ומעביר אותה מאותחלת וריקה לסוד2, על מנת שזה יוכל לבצע את פעולתו כראוי. סוד1 הוא פונקציית המעטפת של סוד2. לכן:

<u>האלגוריתם סוד1 מחזיר רשימה של כל ערכי העץ הבינארי T, אשר אינם מופיעים פעם נוספת בתת-עץ שהם שורשו</u>.

אפשרות אחרת לניסוח מטרת האלגוריתם סוד1:

<u>האלגוריתם סוד1 מחזיר רשימה של כל ערכי העץ הבינארי T, אשר אינם נמצאים פעם נוספת על מסלול כלשהו שמתחיל בהם ומסתיים בעלה</u>.

ג. אם כל ערכי העץ T שונים זה מזה, אז אף ערך לא יופיע שנית בתת-העץ שהוא בשורשו. לכן הרשימה המוחזרת תכיל את כל ערכי העץ T.

אפשר לומר גם שהרשימה תהיה בעצם מסודרת לפי סדר סריקה תחילי של העץ, אבל הפוך: הערך הראשון בסריקה בסדר תחילי של העץ יהיה אחרון ברשימה, והערך האחרון בסריקה בסדר תחילי של העץ יהיה ראשון ברשימה. זאת משום שהאלגוריתם סוד2 סורק את העץ בסדר תחילי, אבל מכניס את האיברים לרשימה תמיד אחרי העוגן, ובכך הופך את סידרם.

כתבו וערכו: איתן ראט ואתי מנשה. כל הזכויות שמורות ©. כל שימוש/העתקה/צילום/שכפול /ציטוט של עמוד זה או של חלקו למטרות מסחריות אסורים בהחלט ללא רשות בכתב מאיתן ראט ואתי מנשה.

שאלה 4

לגבי האנשים המגיעים למתקן "מים גועשים": אין בשאלה קביעה כלשהי לגבי הפרטים המזהים כל אחד. מדובר רק על "קבוצות" ו "בודדים". מצד שני, אין להניח שכל האנשים זהים. כל אדם שונה מחברו. יש צורך ליצור הפרדה כלשהי בין זהותו של אדם שהגיע קודם לזהותו של אדם שהגיע אחר כך. לשם כך מספיק לתת לכל אדם מספר סידורי (כמו בתור בקופת חולים או במשרד הפנים). מה שייצג את האדם בתור הוא מספרו הסידורי. לשם הנוחות, נקבע שמספר סידורי הוא מספר ארבע ספרתי. לגבי הקבוצות: מאחר ואין כל התייחסות בשאלה לאנשים השונים המרכיבים את הקבוצה, אלא לקבוצה כגוף אחד (דבר המודגש גם בכך ש "קבוצה עולה לסירה בשלמותה"), אין טעם לייצג כל אחד מהאנשים הנמצאים בה בנפרד. מאחר והקבוצה עולה כגוף אחד לסירה, מספיק לייצג אותה על ידי מספר האנשים הנמצאים בה.

מאחר וגם האנשים הבודדים וגם הקבוצות עולים לסירות לפי סדר הגעתם, יהיו כאן שני תורים: אחד של "בודדים" ואחד של "קבוצות".

ייצוג סירה: מטרת הנהלת הלונה פארק היא "למחשב את שיבוץ האנשים לסירות". בהנחה שההנהלה גם אחראית על השיט, היא חייבת לדעת מי נמצא על כל סירה. לכן גם הגיוני שכל סירה תכלול את רשימת האנשים שנמצאים בה. כמובן שלכל סירה יש מספר משלה, המייחד אותה (בתחום 1-14). לכן סירה תיוצג ע"י רשומה, ובה שני שדות: מספר הסירה, ורשימת הנוסעים, שיכולה להיות מערך בגודל 8 (או רשימה, או תור, כאן זה לא משנה, אבל מאחר והגודל נתון, השימוש במערך הוא יותר טבעי).

מאחר ויש 14 סירות, עולה הרעיון שכל סירה תהיה מיוצגת ע"י רשומה, ו- 14 הסירות יהיו מערך של סירות. הרעיון אפשרי, אך לא כל כך מתאים לתנאי השאלה. בשאלה נתון שהסירות יצאו בזו אחר זו. מאחר וכל סירה גם צריכה לחזור, ואח"כ לצאת שוב, וכו', אין כל ביטחון שהסירות תמיד יצאו בסדר ההתחלתי. גם הניסוח, "בזו אחר זו", לא מלמד שהן יצאו בסדר עוקב של מספריהן הסידוריים, אלא רק על כך שלא תצאנה שתי סירות יחד. רעיון טוב יותר ממערך עבור הסירות הוא שימוש בתור. כל סירה שיוצאת תצא מראש התור, וכל סירה שחוזרת תחזור לסוף התור.

מתקן השיט "מים גועשים" ייוצג אם כן כרשומה, בת שלושה שדות: תור הסירות, תור הקבוצות ותור הבודדים.

הייצוג בסביבת העבודה בעמ' הבא.

כתבו וערכו: איתן ראט ואתי מנשה. כל הזכויות שמורות ©. כל שימוש/העתקה/צילום/שכפול /ציטוט של עמוד זה או של חלקו למטרות מסחריות אסורים בהחלט ללא רשות בכתב מאיתן ראט ואתי מנשה.

```
Const
        N_Boats         = 14;
        N_Seats         = 8;
Type
        One_Person_Type     = 1000..9999;
        One_Group_Type      = 2..8;
        Empty_Seats_Type    = 0..N_Seats;
        Boat_Num_Type       = 1..N_Boats;
        Aboard_Type  = Array[1..N_Seats] of Integer;
        Boat_Type      = Record
                                Num    : Boat_Num_Type;
                                People : Aboard_Type;
                           End;
```

נגדיר "**תור-סירות**" כתור, אשר שדה המידע (Info) של כל איבר בו הוא מטיפוס Boat_Type.

נגדיר "**תור-בודדים**" כתור, אשר שדה המידע (Info) של כל איבר בו הוא מטיפוס One_Person_Type.

נגדיר "**תור-קבוצות**" כתור, אשר שדה המידע (Info) של כל איבר בו הוא מטיפוס One_Group_Type.

```
        Mitkan_Shayit_Type  = Record
                                  Boats  : תור-סירות     ;
                                  Alone  : תור-בודדים    ;
                                  Group  : תור-קבוצות    ;
                              End;

Var
        MS : Mitkan_Shayit_Type;
```

כתבו וערכו: איתן ראט ואתי מנשה. כל הזכויות שמורות ©. כל שימוש/העתקה/צילום/שכפול /ציטוט של עמוד זה או של חלקו למטרות מסחריות אסורים בהחלט ללא רשות בכתב מאיתן ראט ואתי מנשה.

ממשק עברי לטיפוס הנתונים "מים גועשים":

שם הפעולה	תיאור הפעולה
אתחל-מתקן	פעולה המחזירה מתקן "מים גועשים" מאותחל וריק.
מלא-סירה(MS)	הפעולה ממלאת את הסירה הבאה בתור במידת האפשר, ללא שיחרורה מהמעגן. הנחה: המתקן מאותחל.
שחרר-סירה(MS)	הפעולה משחררת את הסירה הבאה בתור להפלגה. הנחה: המתקן מאותחל, ויש סירה שניתן לשחרר.
הכנס-סירה(MS, Num)	הפעולה מכניסה את הסירה שמספרה Num למאגר הסירות הממתינות להפלגה, במתקן MS. הנחות: מספר הסירה תקין (1-14) ולא קיים במתקן, המתקן מאותחל.
מקומות-פנויים(MS)	הפעולה מחזירה את מספר המקומות הפנויים בסירה שמתמלאת כרגע, במתקן MS. הנחה: המתקן מאותחל ויש בו סירה אחת לפחות.
יש-בודדים?(MS)	הפעולה מחזירה 'אמת' אם יש לפחות "בודד" אחד שממתין לעלות לסירה במתקן MS, ו 'שקר' אחרת.
הכנס-בודד (MS, Num)	הפעולה מכניסה את האדם שמספרו הסידורי Num לתור הממתינים הבודדים לשיטבמתקן MS. הנחה: Num תקין, המתקן MS מאותחל.
הכנס-קבוצה (MS, Num)	הפעולה מכניסה את הקבוצה שמונה Num אנשים לתור הקבוצות הממתינות לשיט במתקן MS. הנחה: Num תקין, המתקן MS מאותחל.

כתבו וערכו: איתן ראט ואתי מנשה. כל הזכויות שמורות ©. כל שימוש/העתקה/צילום/שכפול /ציטוט של עמוד זה או של חלקו למטרות מסחריות אסורים בהחלט ללא רשות בכתב מאיתן ראט ואתי מנשה.

פרק שני – תורת המחשב

שאלה 9

א. מעקב:

$N=3$

$i=1$

$$\begin{pmatrix} 2 & 1 & 1 & 11 \\ 1 & 2 & 1 & 12 \\ 3 & 1 & -1 & 5 \end{pmatrix} \begin{bmatrix} k=2 \\ j=1..4 \end{bmatrix} \xrightarrow{F=-\frac{1}{2}} \begin{pmatrix} 2 & 1 & 1 & 11 \\ 0 & \frac{3}{2} & \frac{1}{2} & \frac{13}{2} \\ 3 & 1 & -1 & 5 \end{pmatrix} \begin{bmatrix} k=3 \\ j=1..4 \end{bmatrix} \xrightarrow{F=-\frac{3}{2}} \begin{pmatrix} 2 & 1 & 1 & 11 \\ 0 & \frac{3}{2} & \frac{1}{2} & \frac{13}{2} \\ 0 & -\frac{1}{2} & -\frac{5}{2} & -\frac{23}{2} \end{pmatrix}$$

$i=2$

$$\begin{pmatrix} 2 & 1 & 1 & 11 \\ 0 & \frac{3}{2} & \frac{1}{2} & \frac{13}{2} \\ 0 & -\frac{1}{2} & -\frac{5}{2} & -\frac{23}{2} \end{pmatrix} \begin{bmatrix} k=3 \\ j=2..4 \end{bmatrix} \xrightarrow{F=\frac{1}{3}} \begin{pmatrix} 2 & 1 & 1 & 11 \\ 0 & \frac{3}{2} & \frac{1}{2} & \frac{13}{2} \\ 0 & 0 & -\frac{7}{3} & -\frac{28}{3} \end{pmatrix}$$

המטריצה שתודפס:

$$\begin{pmatrix} 2 & 1 & 1 & 11 \\ 0 & \frac{3}{2} & \frac{1}{2} & \frac{13}{2} \\ 0 & 0 & -\frac{7}{3} & -\frac{28}{3} \end{pmatrix}$$

הפעלת תהליך ההצבה לאחור:

$$-\frac{7}{3}z = -\frac{28}{3} \Rightarrow \boxed{z=4}$$

$$\frac{3}{2}y + \frac{1}{2}\times 4 = \frac{13}{2} \Rightarrow \boxed{y=3}$$

$$2x + 1\times 3 + 1\times 4 = 11 \Rightarrow \boxed{x=2}$$

ב. נשנה את שתי הלולאות הראשונות:

את הוראה (1) ל: עבור I מ- 2 עד N בצע:

את הוראה (1.1) ל: עבור K מ 1 עד I-1 בצע:

השינוי בלולאה הראשונה: כי כעת אין צורך לאפס מתחת לאיבר הראשון, אבל יש צורך לאפס מעל האחרון. לכן I עכשיו גדול ב- 1 מהאלגוריתם הראשון.

השינוי בלולאה השניה: תפקידו של K כעת לרוץ על כל השורות החל בראשונה ועד זו שלפני השורה ה I, כדי שהאיברים שמעל האלכסון הראשי יתאפסו.

כתבו וערכו: איתן ראט ואתי מנשה. כל הזכויות שמורות ©. כל שימוש/העתקה/צילום/שכפול /ציטוט של עמוד זה או של חלקו למטרות מסחריות אסורים בהחלט ללא רשות בכתב מאיתן ראט ואתי מנשה.

שאלה 10

א. הפלט, בהנחה של שורת פלט מודפסת בשורה נפרדת, יהיה:

DABF
DACE

ב. האלגוריתם מדפיס את כל המסלולים באורך 3 שיוצאים מהצומת Letter, ואשר כל אחד מהצמתים המרכיבים אותם מסומן ע"י אות שאינה גדולה מ LastNode. המסלולים יודפסו בסדר לקסיקוגרפי של שמות הצמתים.

מאחר ונתון בהנחת האלגוריתם ש LastNode היא האות (המסמלת צומת) הגדולה ביותר בגרף, הרי שהאלגוריתם מדפיס את כל המסלולים באורך 3 היוצאים מהצומת Letter, בסדר לקסיקוגרפי של שמות הצמתים.

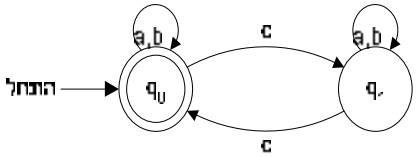
כתבו וערכו: איתן ראט ואתי מנשה. כל הזכויות שמורות ©. כל שימוש/העתקה/צילום/שכפול /ציטוט של עמוד זה או של חלקו למטרות מסחריות אסורים בהחלט ללא רשות בכתב מאיתן ראט ואתי מנשה.

שאלה 11

ניתן לבנות אוטומט חיתוך. קודם נבנה את שני האוטומטים המרכיבים את השפות השונות.

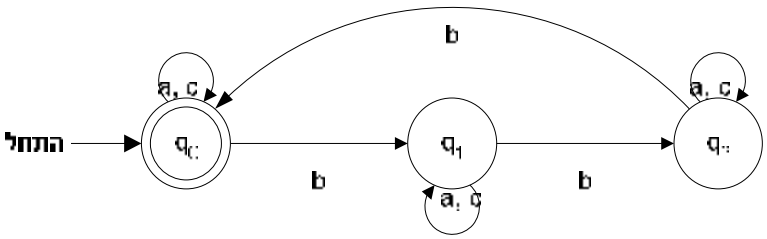
נגדיר את L_1 כאוסף המילים מעל {a, b, c} אשר מספר ה c-ים בהן זוגי.

להלן אוטומט סופי דטרמיניסטי A_1 המקבל את L_1:

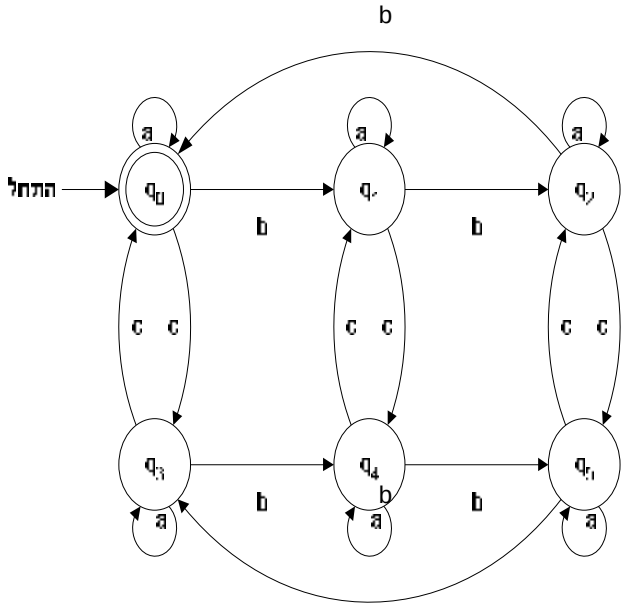


נגדיר את L_2 כאוסף המילים מעל {a, b, c} אשר מספר ה b-ים בהן מתחלק ב 3 ללא שארית.

להלן אוטומט סופי דטרמיניסטי A_2 המקבל את L_2:



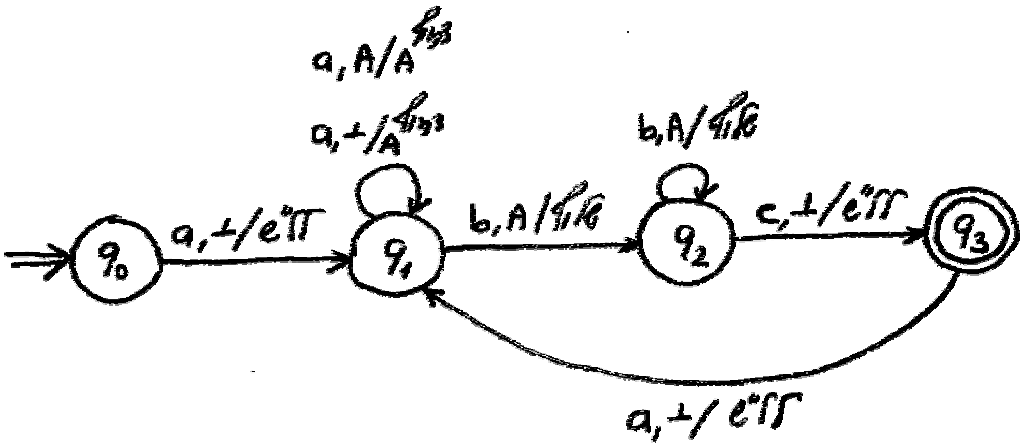
אוטומט החיתוך יוצא:



כתבו וערכו: איתן ראט ואתי מנשה. כל הזכויות שמורות ©. כל שימוש/העתקה/צילום/שכפול /ציטוט של עמוד זה או של חלקו למטרות מסחריות אסורים בהחלט ללא רשות בכתב מאיתן ראט ואתי מנשה.

שאלה 12

להלן אוטומט המחסנית המקבל את השפה המוגדרת בשאלה:



כתבו וערכו: איתן ראט ואתי מנשה. כל הזכויות שמורות ©. כל שימוש/העתקה/צילום/שכפול /ציטוט של עמוד זה או של חלקו למטרות מסחריות אסורים בהחלט ללא רשות בכתב מאיתן ראט ואתי מנשה.

פרק שני – מודלים חישוביים

שאלה 13

נגדיר את שתי השפות הבאות:

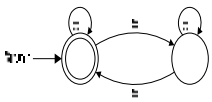
L_1 = { אוסף כל המילים מעל {a, b} אשר מספר ה a-ים בהן זוגי. }

L_2 = { אוסף כל המילים מעל {a, b} אשר מכילות את הרצף aba. }

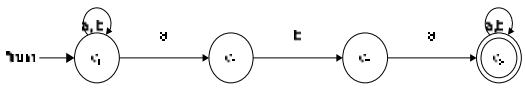
השפה הנתונה L היא בדיוק שפת החיתוך של שתי שפות אלו: $L = L_1 \cap L_2$.

נגדיר אוטומט סופי דטרמיניסטי עבור כל אחת מהשפות הנ"ל:

A_1 :



A_2:



A_1 הוא אוטומט סופי שמקבל את L_1, ולכן L_1 שפה רגולרית.

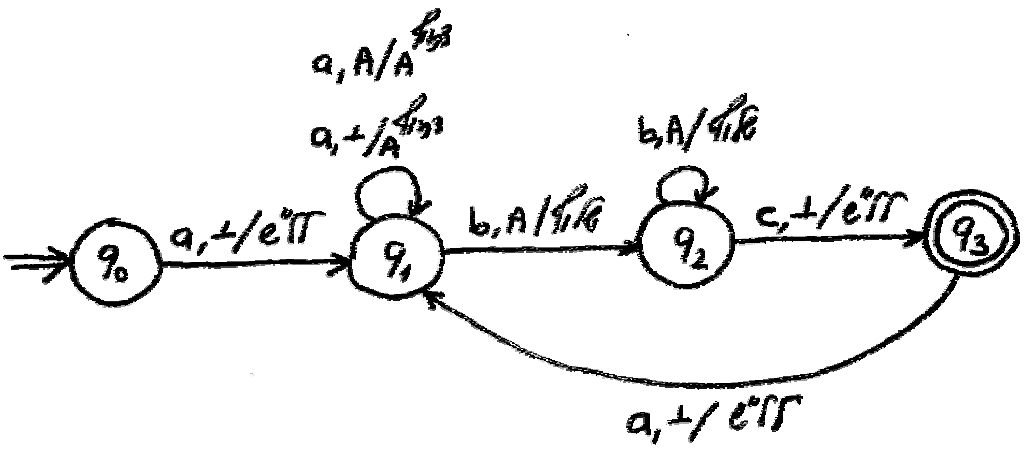
A_2 הוא אוטומט סופי שמקבל את L_2, ולכן L_2 שפה רגולרית.

מאחר ו $L = L_1 \cap L_2$, ובגלל סגירות משפחת השפות הרגולריות תחת חיתוך, גם L רגולרית. .

שאלה 14

שפה חופשית הקשר היא שפה אשר קיים אוטומט מחסנית המקבל אותה.

להלן אוטומט מחסנית A המקבל את L:



כתבו וערכו: איתן ראט ואתי מנשה. כל הזכויות שמורות ©. כל שימוש/העתקה/צילום/שכפול /ציטוט של עמוד זה או של חלקו למטרות מסחריות אסורים בהחלט ללא רשות בכתב מאיתן ראט ואתי מנשה.

שאלה 15

L_1	**חופשית הקשר ולא רגולרית**. יש צורך לזכור בדיוק כמה a-ים היו ברצף שפתח את המילה, כדי לדעת כמה צריכים להיות ברצף שמסיים את המילה. מכיון שמספר זה יכול להיות אינסופי, אס"ד לא יכול לקבל שפה זו. קל, לעומת זאת, להגדיר אוטומט מחסנית שיקבל את השפה: על כל a ברצף הראשון נדחף A למחסנית, ועל כל a שני ברצף המסיים נשלוף A מהמחסנית. המילה תתקבל כאשר אין יותר קלט והמחסנית ריקה.
L_2	**חופשית הקשר ולא רגולרית**. $L_2 = (aa)^n b^n \cdot \{aa\}$. השפה $\{aa\}$ היא סופית, לכן רגולרית, ולכן חופשית הקשר. השפה $(aa)^n b^n$ היא חופשית הקשר בדומה ל $a^n b^n$, כפי שמוכח בספר הלימוד. שירשור שפות ח"ה הוא שפה ח"ה.
L_3	**רגולרית**. רצף ה a-ים בתחילת המילה צריך להיות באורך זוגי, ואחריו בא הרצף bbaa. אין שום בעיה לבנות אס"ד, מכיון שכל מה שהאוטומט צריך לזכור הוא מספר סופי – את שארית החלוקה של מס' ה a-ים ב- 2. השארית יכולה להיות רק 0 או 1, ולכן יספיק אס"ד בן שני מצבים עבור רצף ה a-ים, ועוד ארבעה מצבים עוקבים, אחד לכל אות, עבור ארבע האותיות המסיימות את המילה. אין צורך באוטומט מלא, ולכן זה יספיק.
L_4	**לא חופשית הקשר**. יש קשר משולש בחזקות, שערכו יכול להיות אינסופי. כל שלוש החזקות קשורות זו לזו, וערכן אינו מוגבל. לכן אוטומט מחסנית אינו מספיק, ולכן השפה אינה חופשית הקשר.
L_5	**רגולרית**. אין קשר בין n ל- m, ולכן גם החזקות אינן תלויות זו בזו. הדרישה היחידה היא לזוגיות, כלומר, האוטומט צריך לזכור את שארית החלוקה ב 2, ולכן מספר המצבים יהיה סופי, ולכן יספיק אוטומט סופי, ולכן השפה רגולרית.

כתבו וערכו: איתן ראט ואתי מנשה. כל הזכויות שמורות ©. כל שימוש/העתקה/צילום/שכפול /ציטוט של עמוד זה או של חלקו למטרות מסחריות אסורים בהחלט ללא רשות בכתב מאיתן ראט ואתי מנשה.

שאלה 16

i. הטענה אינה נכונה.

בשפת החיתוך, גם מספר ה a-ים וגם מספר ה b-ים הוא זוגי בכל מילה. במילה aba מספר ה b-ים הוא 1, כלומר מספר אי-זוגי, ולכן איננה שייכת לשפת החיתוך.

ii. הטענה אינה נכונה.

שפת המשלים של L_1, היא אוסף כל המילים בהן מספר ה a-ים הוא אי-זוגי. הואיל ומספר ה a-ים אינו קשור למספר ה b-ים (שהוא המאפיין את L_2), אז השיוויון אינו נכון.

דוגמא: המילה ab שייכת לשפת המשלים של L_1, אך לא שייכת ל L_2.

iii. הטענה נכונה.

L_3 מגבלה את צורת המילה, ו- L_1 מגבילה את מספר ה a-ים להיות זוגי. החיתוך על L_2 מגביל גם את מספר ה b-ים להיות זוגי. מקבלים את המילים שנראות כמו המילים ב L_3, אבל מספר ה a-ים וה b-ים בהן זוגי, ואכן כך קובעת הטענה.

iv. הטענה אינה נכונה.

לפי הגדרת L_3 ו- L_4, מספר האותיות a, וגם מספר האותיות b, יכול להיות 0. כל המילים שהן מהצורה a^k וגם מילים מהצורה b^k שייכות לשתי השפות, ולכן גם לשפת החיתוך.

דוגמא: המילה aaaa שייכת גם ל L_3 וגם ל L_4. לכן החיתוך אינו רק המילה הריקה.

v. הטענה אינה נכונה.

גם ב L_3 וגם ב L_4 אין קשר בין מספר ה a-ים למספר ה b-ים. לכן גם בשרשור שלהן לא ייתכן קשר שכזה.

לדוגמא, המילה abbaaaaaa שייכת לשפת השירשור, למרות שמספר ה b-ים לא שווה למספר ה a-ים הכולל במילה, כפי שעולה מהטענה.

כתבו וערכו: איתן ראט ואתי מנשה. כל הזכויות שמורות ©. כל שימוש/העתקה/צילום/שכפול /ציטוט של עמוד זה או של חלקו למטרות מסחריות אסורים בהחלט ללא רשות בכתב מאיתן ראט ואתי מנשה.