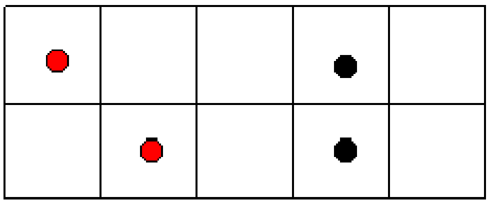
חידה לחימום

נתונה מטריצה בגודל $2*N$ (2 שורות ו-N עמודות). על המטריצה מונחים בכל שורה אסימון שחור ואסימון אדום במקומות כלשהם, כך שהאסימון האדום נמצא משמאל לאסימון השחור. בסך הכול מונחים על-גבי הלוח 2 אסימונים שחורים ו-2 אסימונים אדומים.



במשחק שני שחקנים, שחקן אחד משחק באסימונים השחורים והאחר באסימונים האדומים.

כל שחקן מזיז בתורו אסימון אחד שלו לכוון אסימון היריב כל מספר משבצות שירצה - לפחות משבצת אחת ולכל היותר מספר משבצות השווה למרחק שבינו לבין אסימון היריב באותה שורה. התנועה היא קדימה ובאותה שורה בלבד.

המנצח הוא השחקן שלאחר תורו לא ניתן יותר להזיז אסימונים.

המטרה היא לפתח אסטרטגיה כך שבהינתן המצב ההתחלתי של הלוח, אבחר האם להיות השחקן הראשון או השני, ואשחק כך שאנצח תמיד את היריב.

* * *

נניח כי חוקי המשחק שונו, וכי כעת שחקן יכול לנוע גם אחורנית. כיצד תשתנה אסטרטגיית הניצחון?

<u>מבני נתונים ויעילות אלגוריתמים</u>

(25.2.2015)

<u>1. מיון מהיר (Quick Sort)</u>

האלגוריתם למיון מהיר – שפותח על-ידי מדען המחשב Hoare בשנת 1962, מסתמך על שיטת הפרד-ומשול (Divide-and-Conquer) לפיתוח אלגוריתמים רקורסיביים. הרעיון האלגוריתמי עובד כך:

הפרד:

בוחרים איבר מסוים להיות איבר הציר (pivot) אשר סביבו מבצעים חלוקה. הוא מועבר למקום pivot_index.

המערך A[low..high] מחולק לשני תתי-מערכים A[low..pivot_index-1] ו-A[pivot_index+1..high].

מתקיים שכל איבר בתת-המערך A[low..pivot_index-1] קטן-או-שווה ל-pivot.

כמו כן מתקיים, שכל איבר בתת-המערך A[pivot_index+1..end] גדול ממש מ-pivot.

האיבר pivot נמצא כעת, כאמור, במקום pivot_index.

משול:

את שני התת-מערכים A[low..pivot_index-1] ו-A[pivot_index+1..high] ממיינים באמצעות שתי קריאות רקורסיביות למיון-מהיר. כעת, כל אחד משני תתי-המערכים ממוין, ולכן – המערך A ממוין כולו.

כך תיראה פונקצית המיון, שעושה שימוש בפונקצית-עזר בשם partition שיש לה שלושה תפקידים: (1). לקבוע מי יהיה איבר הציר; (2). לסדר מחדש את האיברים במערך כך שהוא יהיה מחולק לשלושה חלקים (קודם תת-מערך המורכב כולו מאיברים הקטנים-או-שווים לאיבר הציר, ואח״כ – איבר הציר עצמו, ואח״כ – תת-מערך המורכב כולו מאיברים הגדולים ממש מאיבר הציר); (3). להחזיר את האינדקס של איבר הציר.

כדי למיין מערך A המכיל n איברים, נזמן את הפונקציה לראשונה באופן הבא quick_sort(A,0,n-1).

```
void quick_sort (int a[], int low, int high)
{
 int pivot_index;

 if (low < high)
 {
  pivot_index = partition(a, low, high);
  quick_sort(a, low, pivot_index - 1);
  quick_sort(a, pivot_index + 1, high);
 }
}
```

לב האלגוריתם זו הפונקציה המבצעת חלוקה של המערך. נכתוב אותה כך :

```
int partition (int a[], int low, int high)
{
 int i = low;   /* האינדקס השמאלי ביותר */
 int j = high;  /* האינדקס הימני ביותר */

 /* נבחר בתור איבר הציר את האיבר השמאלי ביותר */
 int pivot = a[low];

 while (i < j)
 {
  /* נזיז את האינדקס השמאלי ימינה - עד שניתקל באיבר הגדול מאיבר הציר */
  while (i <= high-1 && a[i] <= pivot)
      i++;

  /* נזיז את האינדקס הימני שמאלה - עד שניתקל באיבר הקטן-או-שווה לאיבר הציר */
  while (j >= 0 && a[j] > pivot)
      j--;

  /* על-ידי ביצוע החלפה - נגדיל כל אחד משני תתי-המערכים */
  if (i < j)
      swap(&a[i],&a[j]);
 }

 /* נעביר את איבר הציר למקומו המתאים, ונחזיר את האינדקס שלו */
 swap(&a[low],&a[j]);
 return j;
}
```

<u>שאלה</u>

הדגימו כיצד תתנהג פונקצית partition, כאשר תופעל על המערך :

```
a = {5,3,2,6,4,1,3,7};
```

<u>שאלה</u>

איזה ערך תחזיר הפונקציה partition אם כל האיברים במערך שווים זה לזה?

<u>שאלה</u>

מהי סיבוכיות זמן הריצה של הפונקציה partition?

כעת, נרצה לנתח את סיבוכיות זמן הריצה של הפונקציה quick_sort.

נשים לב שזמן הריצה תלוי באיבר הנבחר לשמש בתור איבר ציר, ובחלוקה הנוצרת כתוצאה מכך. אם החלוקה מאוזנת (שני תתי-המערכים שווים בגודלם) אזי סיבוכיות זמן הריצה של האלגוריתם תהיה $\Theta(n \cdot \log n)$, בדיוק כמו מיון-מיזוג, מיון-עץ או מיון-ערימה. אם, לעומת זאת, החלוקה אינה מאוזנת (אחד מבין שני תתי-המערכים הוא גדול מאוד, והשני – קטן מאוד), אזי סיבוכיות זמן הריצה תהיה $\Theta(n^2)$, בדיוק כמו מיון-בועות, מיון-הכנסה או מיון-בחירה.

החלוקה הגרוע ביותר:

מיון-מהיר מתנהג באופן הגרוע ביותר כאשר הפונקציה partition מחלקת את המערך לשלושה חלקים: תת-מערך המכיל n-1 איברים הקטנים-או-שווים לאיבר הציר, ואז איבר הציר עצמו, ולבסוף – תת-מערך המכיל 0 איברים הגדולים-ממש מאיבר הציר.

נוסחת הנסיגה במקרה הזה היא: $T(n) = T(n-1) + \Theta(n)$

ואם נפתור אותה (בשיטת ההפרשים או בשיטת האיטרציה), נקבל: $T(n) = \Theta(n^2)$

החלוקה הטובה ביותר:

אם הפונקציה partition מחלקת את המערך לשני תתי-מערכים שגודל כל אחד מהם כמחצית המערך המקורי, אז האלגוריתם למיון-מהיר ירוץ בזמן קצר יותר.

במקרה כזה, נוסחת הנסיגה היא: $T(n) = 2T(\frac{n}{2}) + \Theta(n)$

ואם נפתור אותה (לפי משפט 1 או לפי משפט האב), נקבל: $T(n) = \Theta(n \cdot \log n)$

ניתן להבטיח שנשיג את החלוקה הטובה ביותר, אם בתור איבר הציר נבחר את החציון (median). אפשר להריץ את האלגוריתם למציאת איבר מיקום בזמן לינארי Select, ולקחת בתור איבר הציר את החציון, במקום את האיבר השמאלי ביותר. החולשה בפתרון זה, הוא שהקבועים המסתתרים בפונקצית זמן הריצה של Select הם, לרוב, גדולים מאוד, ולכן – מבחינה מעשית נעדיף שלא ליישם כך את מיון-מהיר.

המקרה הממוצע:

ניתן להוכיח (ולא נעשה זאת), שבמקרה הממוצע, גם אז זמן הריצה של מיון-מהיר יהיה $\Theta(n \cdot \log n)$

<u>**שאלה**</u>

מה תהיה סיבוכיות זמן הריצה של מיון-מהיר אם...

א. לכל האיברים במערך יש אותו ערך?

ב. כל איברי המערך שונים זה מזה, והמערך ממוין בסדר יורד?

2. פתרון משוואות נסיגה לינאריות אי-הומוגניות

למדנו לפתור משוואות נסיגה **לינאריות אי-הומוגניות**:

$$f(n)-c_1\cdot f(n-1)-c_2\cdot f(n-2)-c_3\cdot f(n-3)-...-c_k\cdot f(n-k)=g(n)$$

כאשר $g(n)\neq 0$. השיטה אודה למדנו לפתרון משוואות כאלה נקראת "שיטת ההפרשים".

דרך אחרת לפתרון נוסחאות נסיגה לינאריות אי-הומוגניות, שעבור מקרים רבים היא יותר אפקטיבית, היא באמצעות המשפט הבא, אותו נכנה משפט 3 (בעקבות משפטים 1 ו-2, אותם כבר הכרנו):

משפט 3

בהינתן נוסחת נסיגה מהצורה

$$f(n)-c_1\cdot f(n-1)-c_2\cdot f(n-2)-c_3\cdot f(n-3)-...-c_k\cdot f(n-k)=b^n\cdot p(n)$$

כאשר b הוא קבוע כלשהו, ו-p(n) הוא פולינום שונה מ-0, שדרגתו d, אז המשוואה האופיינית היא:

$$(\alpha^k-c_1\cdot\alpha^{k-1}-c_2\cdot\alpha^{k-2}-c_3\cdot\alpha^{k-3}-...-c_k)(\alpha-b)^{d+1}=0$$

נראה דוגמאות לשימוש במשפט 3.

שאלה

נתונה נוסחת הנסיגה $T(n) = 2T(n-1) + n$. חשבו את סדר הגודל (אסימפטוטית) של $T(n)$.

תשובה

נרשום את נוסחת הנסיגה בתור: $T(n) - 2T(n-1) = n$.

הביטוי שבאגף ימין הוא מהצורה $b^n\cdot p(n)$, כאשר $b = 1$ ו-$p(n) = n$ (פולינום מדרגה $d = 1$).

לפי משפט 3, המשוואה האופיינית היא: $(\alpha-2)(\alpha-1)^2=0$

פתרונותיה הם: $\alpha_1=2$ ו-$\alpha_2=\alpha_3=1$.

לכן, צורת הפתרון היא: $T(n)=A_1\cdot 2^n+A_2\cdot 1^n+A_3\cdot n\cdot 1^n$.

קרי: $T(n)=A_1\cdot 2^n+A_3\cdot n+A_2$.

לכן, מבחינה אסימפטוטית, מתקיים $T(n)=\Theta(2^n)$.

שאלה

פתרו בעזרת משפט 3 את נוסחת הנסיגה $T(n) = 2T(n-1) + 1$, עם תנאי ההתחלה $T(1) = 1$, $T(2) = 3$, אותה פתרנו בשיעור קודם בעזרת שיטת ההפרשים.

תשובה

נרשום את נוסחת הנסיגה בתור: $T(n) - 2T(n-1) =1$.

הביטוי שבאגף ימין הוא מהצורה $b^n\cdot p(n)$, אם ניקח $b = 1$ ו-$p(n) = 1$ (פולינום מדרגה $d = 0$).

לפי משפט 3, המשוואה האופיינית היא: $(\alpha-2)(\alpha-1)=0$

וזו בדיוק המשוואה האופיינית שקיבלנו כשפתרנו נוסחת נסיגה זו באמצעות שיטת ההפרשים.
לכן, שאר הפתרון מתנהל בדיוק באותו האופן.

<u>שאלה</u>

נתונה נוסחת הנסיגה $T(n) = 2T(n-1) + 2^{n+1}$, עם תנאי ההתחלה $T(0) = 2$, $T(1) = 8$.

מצאו ביטוי מפורש עבור $T(n)$.

<u>תשובה</u>

נרשום את נוסחת הנסיגה באופן הבא: $T(n) - 2T(n-1) = 2\cdot 2^n$.

בדוגמא זו, $b = 2$ ו-$p(n) = 2$ (פולינום מדרגה $d = 0$).

לפי משפט 3, המשוואה האופיינית: $(\alpha - 2)(\alpha - 2) = 0$

לכן, שני הפתרונות שווים ל-2.

בעזרת תנאי ההתחלה, נוכל למצוא כי הביטוי המפורש הוא: $T(n) = (n+1)\cdot 2^{n+1}$ (בדקו שזה אכן כך!).

<u>שאלה</u>

נתונה נוסחת הנסיגה $T(n) = 2T(n-1) + 3^n(n+5)$. חשבו את סדר הגודל שלה.

<u>תשובה</u>

מכיוון ש-$b = 3$ ו-$p(n) = n+5$ (פולינום ממעלה $d = 1$), הרי שלפי משפט 3, המשוואה האופיינית היא:

$(\alpha - 2)(\alpha - 3)^2 = 0$, שפתרונותיה הם $\alpha_1 = 2$ ו-$\alpha_2 = \alpha_3 = 3$.

לכן, הפתרון הוא מהצורה $T(n) = A_1 \cdot 2^n + A_2 \cdot 3^n + A_3 \cdot n \cdot 3^n$, וסיבוכיותו היא $T(n) = \Theta(n \cdot 3^n)$.

<u>שאלה</u>

במישור מעבירים n קווים ישרים, שכל אחד מהם נחתך עם כל אחד אחר, ואף שלושה מהם אינם נחתכים בנקודה אחת. נסמן ב-$f(n)$ את מספר האזורים הנוצרים במישור על-ידי n קווים אלו.

א. הוכיחו כי מתקיים כלל הנסיגה $f(n) = f(n-1) + n$, וחשבו תנאי התחלה מתאימים.

ב. פתרו את נוסחת הנסיגה הלינארית האי-הומוגנית מסעיף א'.