מבני נתונים
תרגיל 1 - פתרונות

סיבוכיות זמן ריצה

1. עבור כל אחת מהבעיות, כתבו תוכנית קטנה שפותרת אותה ונתחו את זמן הריצה במקרה הגרוע של התוכנית. חשבו זאת קודם ע"י ספירה מדויקת של מספר הפעולות ואז פשטו את הביטוי שקיבלתם וכיתבו אותו במונחים של $\Theta(\cdot)$. נסו לתת תוכנית יעילה ככל האפשר. בנוסף הסבירו מה n מייצג.

(א) מציאת מקסימום במערך לא ממוין.

פתרון:

```
int max(int[] a)
{
  int m = a[0];
  for (int i=1; i<a.length; i++)
    if (a[i] > m)
      m = a[i];
  return m;
}
```

n שלנו יהיה גודל המערך.

- השורה הראשונה לוקחת פעולה אחת: 1
- אחריה יש לולאה שלוקחת $n-1$ צעדים, בכל אחד מתבצעת או פעולה אחת או שתיים, פלוס נספור 1 בשביל התקדמות הלולאה כל פעם. סך הכל במקרה הגרוע: $3(n-1)$
- בסוף יש עוד פעולה אחת: 1.

הכל ביחד: $1 + 3(n-1) + 1 = 3n - 1 = \Theta(n)$

(ב) חישוב ממוצע ערכים במערך.

פתרון:

```
int average(int[] a)
{
```

```
  int sum = 0;
  for (int i=0; i<a.length; i++)
    sum += a[i];
  return sum / a.length;
}
```

n שלנו יהיה גודל המערך.

- שורה ראשונה לוקחת 1.
- הלולאה היא של n צעדים כל אחד יקח 1 בשביל קידום הלולאה ועוד 1 בשביל החישוב. סך הכל: $2n$.
- שורה אחרונה לוקחת 1.

סך הכל: $1 + 2n + 1 = 2n + 2 = \Theta(n)$.

(ג) מציאת מקסימום במערך ממוין.

פתרון:

```
int max2(int[] a)
{
  return a[a.length-1];
}
```

n שלנו יהיה גודל המערך. יש רק פעולה אחת פה. לכן זמן הריצה הוא $\Theta(1) = 1$.

2. בכל סעיף נתונות שתי פונקציות f, g. בכל אחד אמרו האם $f = O(g)$, $f = \Omega(g)$, או $f = \Theta(g)$. השתמשו בלמה העיקרית שלמדנו בכיתה.

(א) $f(n) = n$, $g(n) = n \cdot \log(n) + n$

פתרון:

$$\frac{f(n)}{g(n)} = \frac{n}{n\log(n) + n} = \frac{1}{\log(n) + 1} \longrightarrow 0$$

כלומר $f = O(g)$.

(ב) $f(n) = n^3 + n + 1$, $g(n) = 4n^3 + 8$

פתרון:

$$\frac{f(n)}{g(n)} = \frac{n^3 + n + 1}{4n^3 + 8} = \frac{n^3 + 2}{4n^3 + 8} + \frac{n - 1}{4n^3 + 8} = \frac{1}{4} + \frac{n - 1}{4n^3 + 8} \longrightarrow \frac{1}{4}$$

כלומר $f = \Theta(g)$.

(ג) $f(n) = 2^n$,$g(n) = 3^n$

פתרון:

$$\frac{f(n)}{g(n)} = \frac{2^n}{3^n} = (\frac{2}{3})^n \longrightarrow 0$$

כלומר $f = O(g)$.

(ד) מנקודה זו והלאה יש צורך בכלל לופיטל, לכן בסעיפים אלה רק כתבו מה אתם חושבים שהתשובה, אין צורך בהוכחה מדויקת. בפתרונות אכתוב את ההוכחה המלאה ולפני הבחינה עברו שוב על שאלות אלה וראו שאתם יודעים לפתור גם אותן.

$f(n) = \sqrt{n}$, $g(n) = \log^2(n)$

פתרון: $\star$ מסמן שימוש בלופיטל.

$$\frac{f(n)}{g(n)} = \frac{\sqrt{n}}{\log^2(n)} =_\star \frac{\frac{1}{2\sqrt{n}}}{2\log(n)\cdot\frac{1}{n}} = \frac{\sqrt{n}}{4\log(n)} =_\star \frac{\frac{1}{2\sqrt{n}}}{\frac{4}{n}} = \frac{\sqrt{n}}{8} \longrightarrow \infty$$

כלומר $f = \Omega(g)$.

(ה) $f(n) = n\log(n)$,$g(n) = n\log(n) + n^2$

פתרון: אקח את היחס הפוך כדי שיהיה יותר קל לחשב את הגבול:

$$\frac{g(n)}{f(n)} = \frac{n\log(n)+n^2}{n\log(n)} = 1+\frac{n^2}{n\log(n)} = 1+\frac{n}{\log(n)} =_\star 1+\frac{1}{\frac{1}{n}} = 1+n \longrightarrow \infty$$

כלומר $f = O(g)$.

(ו) $f(n) = n + \log^2(n)$,$g(n) = n$

פתרון:

$$\frac{f(n)}{g(n)} = \frac{n+\log^2(n)}{n} = 1 + \frac{\log^2(n)}{n} =_\star 1 + \frac{2\log(n)\frac{1}{n}}{1} =$$
$$1 + \frac{2\log(n)}{n} =_\star 1 + \frac{\frac{2}{n}}{1} = 1 + \frac{2}{n} \longrightarrow 1$$

כלומר $f = \Theta(g)$.

(ז) $f(n) = 2^n$,$g(n) = n^2$

פתרון:

$$\frac{f(n)}{g(n)} = \frac{2^n}{n^2} =_\star \frac{\ln(2)2^n}{2n} =_\star \frac{\ln^2(2)2^n}{2} = const\cdot 2^n \longrightarrow \infty$$

כלומר $f = \Omega(g)$.

3. כתבו אלגוריתם המקבל מערך (לאו דוקא ממוין) ומחזיר את המספר שמופיע בו הכי הרבה פעמים. למשל על הקלט:

$$1, 4, 3, 6, 5, 3, 3, 2, 4, 6, 3, 5$$

הוא יחזיר את המספר 3. האלגוריתם שלכם צריך לרוץ בזמן $O(n\log(n))$. אל תשכחו לנתח את זמן הריצה שלו.

פתרון: שלב ראשון בפתרון יהיה להשתמש באלגוריתם מיון טוב כמו מיון מיזוג, שיקח בעצמו $O(n\log(n))$. לאחר מכן נעבור ונמצא מה המספר שמופיע הכי הרבה - וזה בזמן לינארי: $O(n)$. סך הכל נקבל סיבוכיות: $O(n+n\log(n)) = O(n\log(n))$.

```
int solution(int[] a)
{
  mergeSort(a);
  int bestNum = a[0];
  int bestCount = 1;
  int count = 1;
  for (int i = 1; i < a.length; i++) {
    if (a[i] == a[i - 1])
      count++;
    else
      count = 1;
    if (count > bestCount) {
      bestCount = count;
      bestNum = a[i];
    }
  }
  return bestNum;
}
```

כיון שיש פה רק לולאה אחת, שרצה על כל המערך, וכל שאר הפעולות הן בזמן קבוע, אז מלבד המיון, זמן הריצה שלנו הוא $O(n)$ כמו שרצינו.

4. נתון מערך ממוין, מלבד מספר אחד בו, שנמצא במקום הלא נכון. כתבו אלגוריתם הרץ בזמן $O(n)$ הממיין מערך כזה שהוא מקבל כקלט. שוב, נתחו את זמן הריצה של האלגוריתם שלכם.

פתרון: נרוץ פעם אחת קדימה על המערך ונבעבע קדימה ואז נרוץ פעם אחת אחורה ונבעבע למטה במערך. זה יפתור לנו את כל המקרים.

```
int solution(int[] a)
{
```

```
  for (int i = 0; i < a.length; i++)
    if (a[i] > a[i+1])
      swap(a, i, i+1)
  for (int i = a.length - 1; i >= 0; i--)
    if (a[i] < a[i-1])
      swap(a, i, i-1)
}
```

כל לולאה רצה ב $O(n)$ ולכן סך הכל $O(2n) = O(n)$.

5. שנו את האלגוריתם שלמדנו בתרגול למיון בועות, כך שזמן הריצה שלו במקרה הכי טוב הוא $O(n)$. הנה האלגוריתם שלמדנו:

```
void swap(int[] a, int i, int j)
{
  int temp = a[i];
  a[i] = a[j];
  a[j] = temp;
}
void bubbleSort(int[] a) {
  int temp;
  for (int i = 1 ; i < arr.length ; i++)
    for (int j = 0; j < arr.length-i; j++)
      if(a[j] > a[j+1])
        swap(a, j, j+1);
}
```

פתרון:

```
void bubbleSort(int[] a) {
  int temp;
  for (int i = 1 ; i < arr.length ; i++)
    boolean found = false;
    for (int j = 0; j < arr.length-i; j++)
      if(a[j] > a[j+1]) {
        swap(a, j, j+1);
        found = true;
      }
    if (!found)
      return;
}
```

האלגוריתם הזה עובד כמו מיון בועות, אבל אם באחד המעברים שלו הוא רואה שהוא לא החליף אף שני אברים זה אומר שהמערך כבר מסודר, ואפשר לעצור. המקרה הטוב פה יהיה כאשר המערך ממוין כבר מראש ואז כבר אחרי מעבר אחד - $O(n)$ צעדים, האלגוריתם יעצור.

6. נתון הקוד הבא:

```
boolean check(int[] a, int[] b)
{
  for (int i = 0; i < a.length; i++)
    for (int j = 0; j < b.length; j++)
      if (a[i] < b[j])
        return false;
  return true;
}
```

(א) מה עושה השיטה?

פתרון: מחזירה אמת אם כל אברי a גדולים שווים מכל אברי b , ומחזירה שקר אחרת.

(ב) מה סיבוכיות זמן הריצה שלה? הניחו שגדלי המערכים שווים שניהם ל n.

פתרון: הלולאה החיצונית רצה n פעמים. על כל ריצה שלה הלולאה הפנימית רצה n פעמים ומבצעת פעולה אחת או שתיים. סך הכל: $\Theta(n^2)$

(ג) כתבו שיטה שמבצעת את אותה פעולה אך בסיבוכיות זמן טובה יותר. נתחו את סיבוכיות הזמן החדשה.

פתרון:

```
void check2(int[] a, int[] b)
{
  int minA = a[0];

  for (int i = 1; i< a.length; i++)
    minA = Math.min(minA, a[i]);
  for (int i = 1; i< b.length; i++)
    if (minA < b[i])
      return false;
  return true;
}
```

סיבוכיות זמן הריצה היא $\Theta(n)$, כי יש שתי לולאות, כל אחת רצה n פעמים ובתוכה לוקחת $\Theta(1)$.

פתרון משוואות רקורסיביות

1. פתרו כל אחת מנוסחאות הנסיגה הבאות בעזרת שיטת האב, או הסבירו מדוע משפט האב לא מתאים לדוגמא. נמקו היטב את השימוש במשפט האב ובדקו את כל התנאים.

(א) $T(n) = 4T(n/2) + n$

פתרון: מקרה 1 מתאים פה:

- $n^{\log_b(a)} = n^{\log_2(4)} = n^2$
- $f(n) = n$
- ניקח $\epsilon = \frac{1}{2}$ ונקבל $f(n) < n^{\log_b(a)-\epsilon}$, וזה כי: $n < n^{2-\frac{1}{2}}$
- ואז התוצאה היא: $T(n) = \Theta(n^{\log_n(a)}) = \Theta(n^2)$

(ב) $T(n) = 2T(n/3) + n^2$

פתרון: מקרה 3 מתאים פה:

- $n^{\log_b(a)} = n^{\log_3(2)} \approx n^{0.63}$
- $f(n) = n^2$
- ניקח $\epsilon = 1$ ונקבל $f(n) > n^{\log_b(a)+\epsilon}$, וזה כי: $n > n^{0.63+1}$
- בשביל מקרה 3 יש לבדוק את התנאי: קיים איזשהו $c < 1$ כך ש $af(\frac{n}{b}) < cf(n)$.
  ניקח $c = \frac{1}{2}$, ואז: $2f(\frac{n}{3}) = 2\left(\frac{n}{3}\right)^2 = \frac{2}{9}n^2$ שזה באמת קטן מ $cf(n) = \frac{1}{2}n^2$
- ואז התוצאה היא: $T(n) = \Theta(f(n)) = \Theta(n^2)$

(ג) $T(n) = 9T(n/3) + n^2$

פתרון: מקרה 2 מתאים פה:

- $n^{\log_b(a)} = n^{\log_3(9)} = n^2$
- $f(n) = n^2$
- ונקבל $f(n) = n^{\log_b(a)}$
- ואז התוצאה היא: $T(n) = \Theta(n^{\log_n(a)}\log(n)) = \Theta(n^2\log(n))$

(ד) $T(n) = \frac{1}{2}T(\frac{1}{2}n) + \frac{1}{n}$

פתרון: לא מתאים למשפט האב כיון ש $a < 1$.

(ה) $T(n) = T(\frac{1}{2}n) + \log n$

פתרון: לא מתאים למשפט. עקרונית מקרה 3 מתאים פה, אבל התנאי הנוסף אינו מתקיים: $af(\frac{n}{b}) = \log(\frac{n}{2}) = \log(n) - 1$. ניקח קבוע $c < 1$, ואז צריך להתקיים:

$$\begin{array}{lcl} \log(n) - 1 & < & c\log(n) \\ (1-c)\log(n) & < & 1 \\ \log(n) & < & \frac{1}{1-c} \end{array}$$

אבל צד ימין הוא סך הכל איזשהו קבוע וצד שמאל שואף לאינסוף כש n גדול. לכן זה לא יכול להתקיים.

2. פתרו את נוסחאות הרקורסיה הבאות בעזרת שיטת ההצבה:

(א)

$$T(n) = 2T(n-1)$$
$$T(1) = 5$$

פתרון:

$$\begin{aligned} T(n) &= 2T(n-1) \\ &= 2(2T(n-2)) = 2^2T(n-2) \\ &= 2^2(2T(n-3)) = 2^3T(n-3)) \\ &= 2^kT(n-k) \end{aligned}$$

נציב $k = (n-1)$, ונקבל:

$$2^{n-1}T(1) = 5 \cdot 2^{n-1}$$

(ב)

$$T(n) = T(n-2) + 4$$
$$T(1) = 3$$
$$T(0) = 0$$

פתרון:

$$\begin{aligned} T(n) &= T(n-2) + 4 \\ &= T(n-4) + 4 + 4 \\ &= T(n-6) + 4 + 4 + 4 \\ &= T(n-2k) + 4k \end{aligned}$$

אם n זוגי, אז נציב $k = \frac{n}{2}$ ונקבל:

$$T(0) + 2n = 2n$$

אם n אי זוגי, נציב $k = \frac{n-1}{2}$ ונקבל:

$$T(n - 2\frac{n-1}{2}) + 4\frac{n-1}{2} = T(1) + 2(n-1) = 3 + 2n - 2 = 2n + 1$$

(ג)

$$T(n) = 3T(\tfrac{1}{2}n) + 1$$
$$T(1) = 2$$

פתרון:

$$\begin{aligned} T(n) &= 3T(\tfrac{1}{2}n) + 1 \\ &= 3(3T(\tfrac{1}{2^2}n) + 1) + 1 = 3^2T(\tfrac{1}{2^2}n) + 3 + 1 \\ &= 3^2(3T(\tfrac{1}{2^3}n) + 1) + 3 + 1 = 3^3T(\tfrac{1}{2^3}n)3^2 + 3 + 1 \\ &= 3^kT(\tfrac{1}{2^k}n) + 3^{k-1} + 3^{k-2} + ... + 1 = 3^kT(\tfrac{1}{2^k}n) + \tfrac{3^k-1}{2} \end{aligned}$$

נציב $k = \log_2(n)$, ונקבל:

$$3^{\log_2(n)}T(1)+\frac{3^{\log_2(n)}-1}{2} = 2\cdot 3^{\log_2(n)}+\frac{3^{\log_2(n)}-1}{2} = 2\frac{1}{2}\cdot 3^{\log_2(n)}-\frac{1}{2}$$

אבל:

$$3^{\log_2(n)} = \left(2^{\log_2(3)}\right)^{\log_2(n)} = 2^{\log_2(3)\log_2(n)} = n^{\log_2(3)}$$

ולכן התוצאה:

$$2\frac{1}{2}\cdot n^{\log_2(3)} - \frac{1}{2}$$

(ד)

$$T(n) = aT(\tfrac{n}{b}) + c$$
$$T(1) = d$$

כאשר a, b, c, d הם קבועים גדולים מ־0. כמו כן, $a > 1, b > 1$. אחרי שאתם מקבלים את התוצאה המדויקת, כיתבו אותה ב־ Θ, והשוו אותה למה שהייתם מקבלים אילו הייתם משתמשים במשפט האב.

פתרון:

$$\begin{aligned} T(n) &= aT(\tfrac{n}{b}) + c \\ &= a(aT(\tfrac{n}{b^2}) + c) + c = a^2T(\tfrac{n}{b^2}) + c(1 + a) \\ &= a^2(T(\tfrac{n}{b^2}) + c) + c(1 + a) = a^3T(\tfrac{n}{b^3}) + c(1 + a + a^2) \\ &= ... \\ &= a^kT(\tfrac{n}{b^k}) + c(1 + a + a^{k-1}) = \\ &= a^kT(\tfrac{n}{b^k}) + c\tfrac{a^k-1}{a-1} \end{aligned}$$

נציב $k = \log_b(n)$, ונקבל:

$$a^{\log_b(n)}T(1) + \frac{c}{a-1}(a^{\log_b(n)} - 1) = a^{\log_b(n)}\left(d + \frac{c}{a-1}\right) - \frac{c}{a-1}$$

אם נפעיל קצת חוקי לוגים:

$$a^{\log_b(n)} = a^{\frac{\log_a(n)}{\log_a(b)}} = n^{\frac{1}{\log_a(b)}} = n^{\log_b(a)}$$

ולכן נקבל:

$$n^{\log_b(a)}\left(d + \frac{c}{a-1}\right) - \frac{c}{a-1}$$

וזה שווה ל־

$$\Theta(n^{\log_b(a)})$$

3. שאלה זו היא לתרגול נוסף של משפט האב. אין צורך להגיש אותה.

פתרו כל אחת מנוסחאות הנסיגה הבאות בעזרת שיטת האב, או הסבירו מדוע משפט האב לא מתאים לדוגמא. נמקו היטב את השימוש במשפט האב ובדקו את כל התנאים.

(א) $T(n) = 6T(n/2) + n^2$

פתרון: מקרה 1 מתאים פה:

- $n^{\log_b(a)} = n^{\log_2(6)} = n^3$
- $f(n) = n^2$
- ניקח $\epsilon = \frac{1}{2}$ ונקבל $f(n) < n^{\log_b(a)-\epsilon}$, וזה כי: $n^2 < n^{3-\frac{1}{2}}$
- ואז התוצאה היא: $T(n) = \Theta(n^{\log_n(a)}) = \Theta(n^3)$

(ב) $T(n) = T(n/3) + n^2$

פתרון: מקרה 3 מתאים פה:

- $n^{\log_b(a)} = n^{\log_3(1)} = n^0 = 1$
- $f(n) = n^2$
- ניקח $\epsilon = 1$ ונקבל $f(n) > n^{\log_b(a)+\epsilon}$, וזה כי: $n^2 > n^{0+1}$
- בשביל מקרה 3 יש לבדוק את התנאי: קיים איזשהו $c < 1$ כך ש $af(\frac{n}{b}) < cf(n)$.
  ניקח $c = \frac{1}{2}$, ואז: $f(\frac{n}{3}) = (\frac{n}{3})^2 = \frac{1}{9}n^2$ שזה באמת קטן מ $cf(n) = \frac{1}{2}n^2$
- ואז התוצאה היא: $T(n) = \Theta(f(n)) = \Theta(n)$

(ג) $T(n) = 2T(\frac{1}{2}n) + \frac{n}{\log n}$

פתרון: לא מתאים למשפט. מקרה 2 לא מתאים כיון ש

$$n^{\log_b(a)} = n^{\log_2(2)} = n \neq \Theta(\frac{n}{\log(n)})$$

מקרה 1 נראה יותר מתאים אבל אז צריך להתקיים: $n > n^\epsilon \frac{n}{\log(n)}$, שזה $\log(n) > n^\epsilon$, אבל כמו שאמרנו, זה לא מתקיים לאף $\epsilon > 0$.