Amir Mizrahi, ID 311128763
Yagil Sofer, ID 205433808

מטלה תכנותית 1 – ניתוח זמן ריצה

1. Insertion Sort:

```
1   void algorithmClass::insertionSort(doubleClass& arrClass)
2   {
3       int k, j;
4       double* _arr = arrClass.getDoubleArr(), key;           θ(1)
5       for (k = 1; k < arrClass.getPhySize(); k++)
6       {
7           key = _arr[k];                                     θ(1)
8           j = k - 1;
9           while (j >= 0 && _arr[j] > key)                          θ(n)
10          {
11              _arr[j + 1] = _arr[j];                         θ(k)
12              j = j - 1;
13          }
14          _arr[j + 1] = key;                        θ(1)
15      }
16  }
```

נחשב את המקרה הגרוע – בו המערך מסודר בסדר ממוין יורד של איברים, למשל:

Index (k)	1	2	3	...	9
Value	k	k-1	k-2	...	k-j
הסבר – הערך בתא מס' 1 הוא הגדול ביותר, אחריו – 2 וכך הלאה. כאשר הערך בתא ה- n הוא הקטן ביותר					

ראשית, נוכיח מדוע מדובר בקלט הגרוע ביותר:

א. <u>לכל קלט בגודל n, זמן הריצה הוא לכל היותר $O(n^2)$</u> – קל להבחין שלכל קלט בגודל n, כמות המירבית של איטרציות בלולאה החיצונית היא n. בלולאה הפנימית, אנו נבצע לכל היותר n-1 פעולות swap, כלומר – בעבור כל קלט, זמן ריצתו המירבי $n * O(n) = O(n^2)$.

ב. <u>קיים קלט בגודל n עם זמן ריצה לפחות $\Omega(n^2)$</u> – נראה כי בעבור המערך הנ"ל, זמן הריצה הינו $\Omega(n^2)$.

קל להבחין שהלולאה החיצונית (5-16) רצה n איטרציות (עבור קלט כלשהו בגודל n).

שורה 7 – ערכו של המשתנה key הינו ערך המערך במקום ה-k, כאשר k רץ מ-1 ועד n.

שורה 8 – ערכו של j הינו כאינדקס אחד לפני k (התא לפניו במערך). מכאן ניתן להסיק, שבמקרה הגרוע, הלולאה הפנימית (9-14) תרוץ בדומה לטור חשבוני – כלומר, באיטרציה הראשונה יבוצעו 1 פעולות בלולאה הפנימית, באיטרציה השניה – יבוצעו 2 פעולות (כמספר פעולות ההחלפה), וכך הלאה... עד לערך המינימלי במערך.

בהתחלה, נגדיר משתנה t = מספר האיטרציה:

Index (t)	0	1	2	...	t
≈Actions per iteration	1	1	1	...	1

Amir Mizrahi, ID 311128763
Yagil Sofer, ID 205433808

נחפש חוקיות בעזרת הרץ j בלולאה הפנימית.

Index (t)	0	1	2	...	t
j=	k-1	k-2	k-3	...	k-t-1

נחשב את תנאי העצירה:

$$j < 0$$
$$k - t - 1 < 0$$
$$t > k - 1$$

$$\sum_{t=0}^{k-1} 1 = \underbrace{1 + 1 + 1 \ldots. +1}_{\text{פעמים } k} = \theta(k)$$

כעת נחשב את זמן הריצה של הלולאה החיצונית:

$$\sum_{k=1}^{n} k = 1 + 2 + 3 \ldots. +n = \theta(n^2)$$

לסיכום, זמן ריצת התוכנית בעבור המקרה הגרוע - $\theta(n^2)$.

2. Selection:

א. תחילה ננתח את זמן הריצה של Partition:

```
int algorithmClass::MyPartition(int left, int right, int _pivot, double* _arr)
{
    int pivot = left, index = right;
    bool direction = true;
    doubleSwap(_arr[_pivot], _arr[left]);

    while (pivot != index)
    {
        if ((_arr[pivot] > _arr[index]) && direction || (_arr[pivot] < _arr[index]) && !direction)
        {
            doubleSwap(_arr[pivot], _arr[index]);
            int tmpInd = pivot;
            pivot = index;
            if (direction)
            {
                direction = false;
                index = tmpInd + 1;
            }
            else
            {
                direction = true;
                index = tmpInd - 1;
            }
        }
        else if (direction)
            index--;
        else
            index++;
    }
    return pivot + 1;
}
```

$\theta(1)$ $\theta(1)$ $\theta(n)$ $\theta(1)$

נניח שגודל הקלט הוא n.

אזי, קל להבחין שבאלגוריתם זה – כל איטרציה של לולאת ה- while, האינדקס מתקדם לכיוון ה-pivot. למשל, אם ה-index מימין ל-pivot, ה-index יתקדם לכיוון ה- pivot "צעד" אחד שמאלה, ולהיפך. לכן, הלולאה תגיע לתנאי העצירה כאשר ה- index וה-pivot שווים. בנוסף, בכל איטרציה מתבצעות סדר גודל של $\theta(1)$ פעולות.

Amir Mizrahi, ID 311128763
Yagil Sofer, ID 205433808

נסכם: $\theta(1) * n$ פעולות בלולאת ה- while, ולכן זמן הריצה של אלגוריתם זה הינו $\theta(n)$.

ב. Selection

```
1  double algorithmClass::Selection(int left, int right, int i, double* _arr)
2  {
3      int pivot, leftOverPart;
4      pivot = MyPartition(left, right, left, _arr) - 1;
5      leftOverPart = pivot - left + 1;
6      if (i == leftOverPart)
7          return _arr[pivot];
8      if (i < leftOverPart)
9          return Selection(left, pivot - 1, i, _arr);
10     else
11         return Selection(pivot + 1, right, i - leftOverPart, _arr);
12 }
13
```

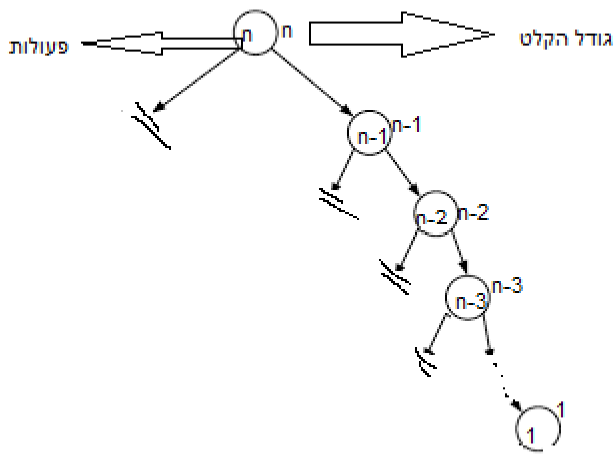
$\theta(input\ size)$

$\theta(t)$

לפנינו אלגוריתם רקורסיבי, נמצא נוסחת נסיגה:
בסעיף א' הראינו כי זמן הריצה של אלגוריתם Partition הוא $\theta(input\ size)$.
נסמן t = הקלט הגרוע ביותר (נסביר בהמשך את אופיו ונוכיח כי הוא אכן הקלט הגרוע ביותר).
נסתכל על שתי הקריאות הרקורסיביות בשורות 9 ו-12. נשים לב שרק אחת מהקריאות הרקורסיביות תבוצע. אנו כאמור מעוניינים בניתוח הזמן הגרוע, ולכן –

$$T(n) = \theta(t) + \theta(n)$$

א. <u>לכל קלט בגודל n, זמן הריצה הוא לכל היותר $O(n^2)$</u> - למקרה הגרוע הוא עץ רקורסיה עם לכל היותר n רמות ולכל היותר n פעולות בכל רמה, כלומר $T(n) = O(n^2)$..
לכל היותר n רמות – כיוון שבכל רמה נחסר לכל הפחות את ה- pivot. לכל היותר n פעולות – כיוון שברמה העליונה נבצע n פעולות, ובכל רמה אחרת נבצע לפחות פעולה אחת פחות.



ב. <u>קיים קלט בגודל n עם זמן ריצה לפחות $\Omega(n^2)$</u> - המקרה הגרוע דורש למעשה קלט שבעבורו זמן הריצה הוא הגרוע ביותר. נשאף למצוא קלט בזמן ריצה $T = \Omega(n^2)$.

נראה כי בעבור מערך ממוין בסדר עולה, מתקיים כי:

Index (t)	0	1	2	…	n-1
Value	1	2	3	…	n
Complexity	<u>$\Omega(n)$</u>	<u>$\Omega(n)$</u>	<u>$\Omega(n)$</u>	…	<u>$\Omega(n)$</u>

Amir Mizrahi, ID 311128763
Yagil Sofer, ID 205433808

<u>נסביר:</u>

האלגוריתם Partition מבצע בעבור כל קריאה $\theta(n)$ פעולות כתלות בגודל הקלט. למעשה במקרה זה, Partition ממקם את האיבר ה-i בגודלו במקומו ה- i, ולכן משאיר את המערך ללא שינוי וממשיך לקדם את index לכיוון ה- pivot, בכל אחת מהאיטרציות.

ניתן להסיק כי כיוון שהמערך ממוין, האלגוריתם Selection יבצע n פעמים את האלגוריתם Partition. כלומר,

$$T(n) = \sum_{i=0}^{n} \theta(n) = \theta(n^2)$$

הראינו באמצעות חסם עליון ותחתון שזמן הריצה במקרה הגרוע הוא - $\theta(n^2)$.

3. Fifth Algorithm:

```
double algorithmClass::fivesAlgorithem(double* arr, int n, int i)
{
    if (n <= 5)//halt point
    {
        doubleBubbleSort(n, arr);
        return arr[i - 1];
    }
    int remain = n % 5;
    int BsizeWithRemainder = (n/5);
    if (remain != 0)
        BsizeWithRemainder++;

    double* B = new double[BsizeWithRemainder];
    assert(B);
    for (int t = 0; t < n / 5; t++)
    {
        doubleBubbleSort(5, arr + t * 5);
        B[t] = (arr + t * 5)[2];
    }

    if (remain != 0)
    {
        doubleBubbleSort(remain, arr + n- remain);
        int medianLocation = ceil(static_cast<double>(remain) / 2);
        B[BsizeWithRemainder - 1] = (arr + n -remain)[medianLocation-1];
    }

    int remain2 = BsizeWithRemainder % 5;
    int BsizeWithRemainder2 = (BsizeWithRemainder / 5);
    if (remain2 != 0)
        BsizeWithRemainder2++;
    double median = fivesAlgorithem(B, BsizeWithRemainder2, ceil(static_cast<double>(BsizeWithRemainder2)/2));
    delete[] B;
    int medianIndex = findIndByNumInDoubleArr(median, arr, n);
    int k = MyPartition(0, n - 1, medianIndex, arr);
    if (k > i)
        return fivesAlgorithem(arr, k - 1, i);
    else if (k < i)
        return fivesAlgorithem(arr + k, n - k, i - k);
    else
        return arr[k - 1];
}
```

$\theta(1)$

$\theta(n)$

$\theta(1)$

$\theta(1)$

$T(\frac{n}{5})$

$\theta(n) - worst\ case$

$\theta(n)$

$T(k)$

$T(n-k)$

לפי הנחיות העזר המצורפות לתרגיל, נראה את נכונות אי השוויון הבא:

(3n/10) - c ≤ k ≤ (7n/10) + c

Amir Mizrahi, ID 311128763
Yagil Sofer, ID 205433808

נראה שכתוצאה מבחירת ה- *pivot*, הערך *k* באלגוריתם החמישיות מקיים את התכונות הבאות עבור *c>0*.

נתבונן במערך *A* – כאשר *A1-...-An* הינם ערכים כלשהם.

Index	0	1	2	...	n-1
	A1	A2	A3	...	An

נתבונן במערך החציונים של *A*, נסמנו *B*:

Index	0	1	2	...	n/5 – 1
	B1	B2	B3	...	B(n/5)

במערך *B*, עלינו למצוא את החציון.

- טענת עזר: נוכיח שלכל איבר ב-*b* במערך *B* מתקיים שלפחות 2 איברים במערך *A* קטנים ממנו, ולפחות 2 איברים במערך *A* גדולים ממנו.
  הוכחה: לפי האלגוריתם, מערך *B* נוצר אמ"מ גודלו של *A* גדול מ-5 (אחרת, סיימנו).
  כלומר, אנו מניחים שבמערך *A* לפחות 6 איברים. כלומר, במערך *B* קיים חציון שגדול לפחות מ-2 איברים וקטן מלפחות 2 איברים במערך *A*.

לפי טענת העזר, ניתן להסיק שהחציון של *B* (נסמנו כ- *pivot*) הוא האיבר ה- *n/10* בגודלו במערך *B*.
מכאן, ניתן להסיק:
ישנם *n/10 -1* איברים קטנים מ-*pivot* ו-*n/10* איברים שגדולים מ- *pivot*. לפי טענת העזר קיימים לפחות שני איברים שהם בעצמם קטנים מהאיבר ה- *n/10-1*. כלומר, ניתן להגיד ש:

$$pivot = \frac{n}{10} >= \ 3 * \left(\frac{n}{10} - 1\right) = \frac{3n}{10} - 3$$

כמו כן, באופן סימטרי ולפי טענת העזר, מתקיים:

$$pivot = \frac{n}{10} <= \ 3 * \left(\frac{n}{10} + 1\right) = \frac{3n}{10} + 3$$

כלומר,
הראינו שלחציון *pivot* קיימים לפחות *3n/10 +3* איברים שהוא קטן מהם ולפחות *3n/10-3* איברים שהוא גדול מהם. לכן, ניתן להסיק ש- *3n/10<=pivot<=7n/10* נסמן *k* כמיקום ה- *pivot*, כלומר – האיבר ה-*k* בגודלו במערך *A*.

3n/10

7n/10

Index	0	1	k	...	n-1
	A1	A2	Pivot	...	An

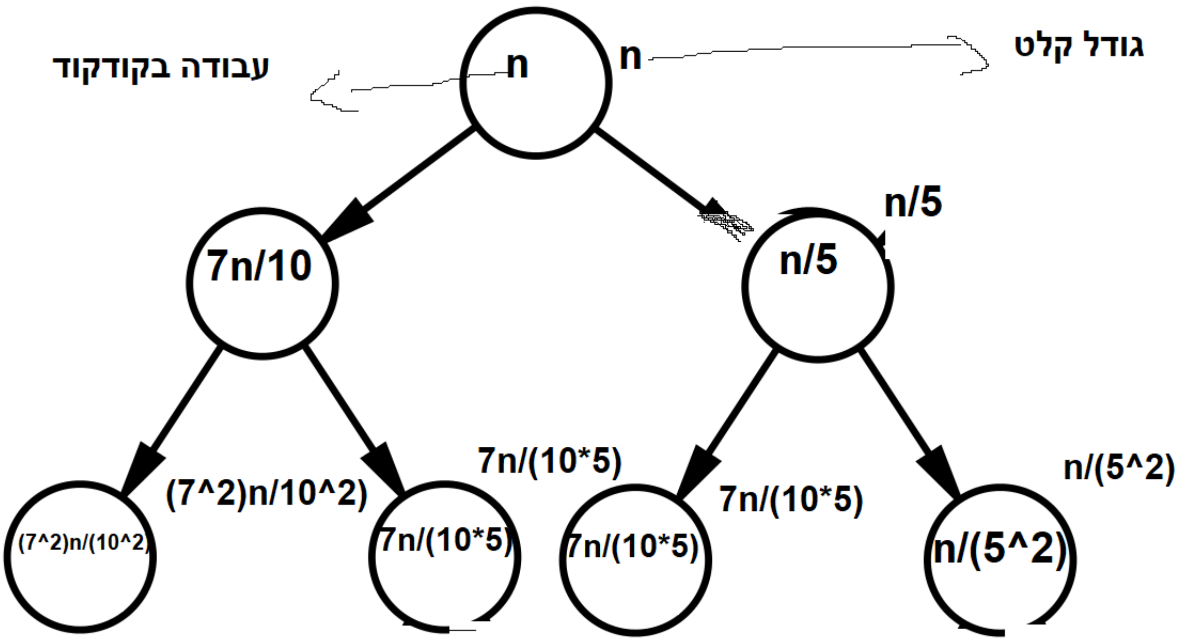
הערה – ההמחשה באמצעות המערך הנ"ל מתקיימת גם אילו המערך הוצג "הפוך".

כעת, נניח ש- *k* הוא בדיוק *7n/10* ושהרקורסיה של האלגוריתם הנ"ל ימשיך עם המערך הגדול יותר.
ניתן לראות כי במקרה זה, אם *k=7n/10*, הרקורסיה תמשיך עם המערך הגדול ביותר, ולכן מדובר בקלט הגרוע ביותר.
לכן גודל המערך הגדול הוא *7n/10*. מכאן, נבנה את נוסחת הנסיגה על סמך ההנחות הנ"ל:

Amir Mizrahi, ID 311128763
Yagil Sofer, ID 205433808

$$T(n) = T\left(\frac{7n}{10}\right) + T\left(\frac{n}{5}\right) + \Theta(n) \quad ; \quad T(5) = 1$$

נפתור את נוסחת הנסיגה באמצעות עץ רקורסיה:



רמה	מס' קודקודים	גודל קלט	עבודה ברמה
0	1	n	n
1	2	$\left(\frac{7n}{10}\right), \left(\frac{n}{5}\right)$	$\left(\frac{9n}{10}\right)$
2	4	...	$\left(\frac{81n}{100}\right)$
	...		
l	$2^\wedge i$	...	$\left(\frac{9n}{10}\right)^i$

$$\left(\frac{7}{10}\right)^i n \leq 5$$
$$\frac{n}{5} \leq \left(\frac{10}{7}\right)^i$$
$$\log_{10/7} \frac{n}{5} \leq i$$

$$\left(\frac{1}{5}\right)^i n \leq 5$$
$$\frac{n}{5} \leq (5)^i$$
$$\log_5 \frac{n}{5} \leq i$$

חסם עליון: $T(n) \leq \sum_{i=0}^{\log_{\frac{10}{7}} \frac{n}{5}} \left(\frac{9}{10}\right)^i * n = n * \sum_{i=0}^{\infty} \left(\frac{9}{10}\right)^i = n * \theta\left(\frac{1}{\frac{1}{10}}\right) = O(n)$

חסם תחתון: $T(n) \geq \sum_{i=0}^{\log_5 \frac{n}{5}} \left(\frac{9}{10}\right)^i * n = n * \sum_{i=0}^{\log_5 \frac{n}{5}} \left(\frac{9}{10}\right)^i \geq n * \left(\frac{9}{10}\right)^0 = \Omega(n)$

ולכן, לכל קלט בגודל n זמן הריצה הוא $T(n) = \Theta(n)$.

Amir Mizrahi, ID 311128763
Yagil Sofer, ID 205433808

נראה עתה כי **קיים** קלט גרוע:

נתבונן במערך *A*:

Index	**0**	**1**	**2**	**...**								**n-1**
	1	2	3	*n*	*n-1*	4	5	6	*n-2*	*n-3*	...	...

ובמערך החציונים של *A*, מערך *B*:

Index	**0**	**1**	**2**	**...**			**n/5-1**
	3	6	9	*12*	*15*	...	*3n/5*

קל להבחין שב-*B* (מערך החציונים) יש *n/5* איברים. כעת עלינו למצוא את החציון במערך זה, כיוון שהמערך אינו ממוין של איברים עוקבים זה לזה. נגדיר משתנה *t* "ונתרגם" את איברי המערך כאילו היו איברים עוקבים ונביע את החציון. בכך נוכל למצוא אותו במערך זה ולאחר מכן נציב *t=(n/5)/2* = *n/10*.

t= 0	*t=1*	*t=2*	*t=3*	*...*	*t=n/5-1*
3	*6*	*9*	*12*		*3(t+1)*

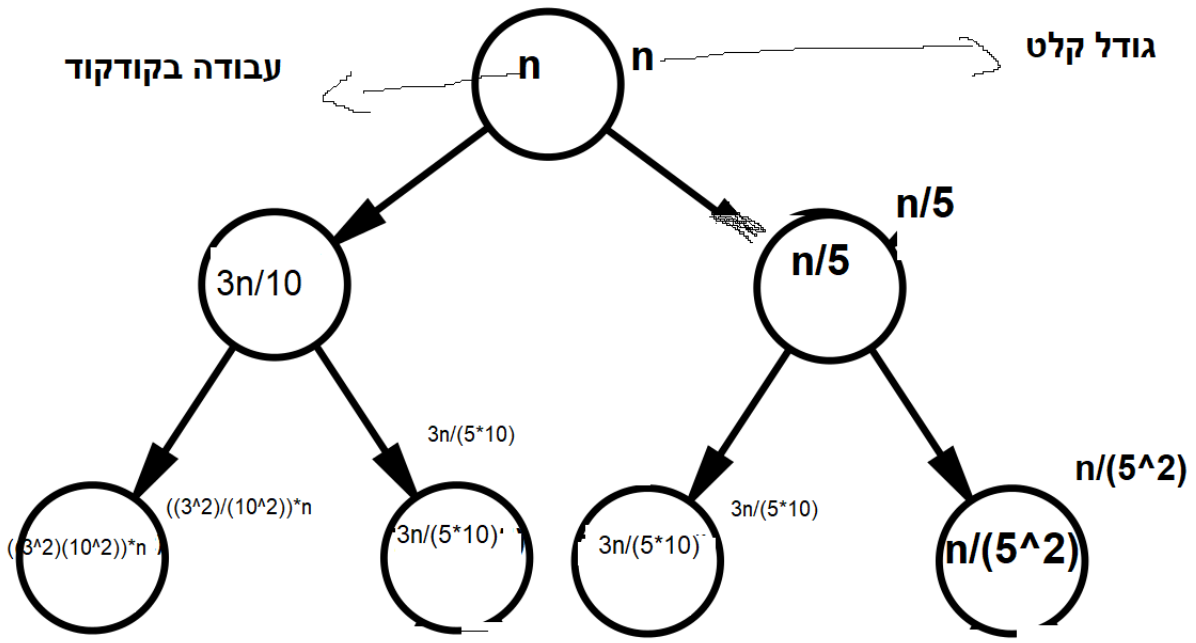
נציב *t=n/10-1* ונקבל שהחציון הוא *3((n/10) = (3n/10)*, שהוא ע"פ מה שהוכחנו למעלה – הקלט הגרוע ביותר. נסמן *k= the object at the '3n/10' size in the array*.

נציב:

$$\frac{-n}{5} \leq \left|k - \frac{n}{2}\right| = \left|\frac{3n}{10} - \frac{n}{2}\right| \leq \frac{n}{5}$$

נוסחת הנסיגה הינה:

$$T(n) = T\left(\frac{3n}{10}\right) + T\left(\frac{n}{5}\right) + \Theta(n) \quad ; \quad T(5) = 1$$



Amir Mizrahi, ID 311128763
Yagil Sofer, ID 205433808

רמה	מס' קודקודים	גודל קלט	עבודה ברמה
0	1	n	n
1	2	$\left(\frac{3n}{10}\right), \left(\frac{n}{5}\right)$	$\left(\frac{n}{2}\right)$
2	4	$max\left(\frac{3}{10}\right)^2 n, \dots, min, \left(\frac{n}{5^2}\right)$	$\left(\frac{n}{4}\right)$
		...	
I	$2^{\wedge}i$	$max\left(\frac{3}{10}\right)^i n, \dots, min, \left(\frac{n}{5^i}\right)$	$\left(\frac{n}{2^i}\right)$

$$\left(\frac{3}{10}\right)^i n \le 5$$
$$\frac{n}{5} \le \left(\frac{10}{3}\right)^i$$
$$\log_{10/3}\frac{n}{5} \le i$$

$$\left(\frac{1}{5}\right)^i n \le 5$$
$$\frac{n}{5} \le (5)^i$$
$$\log_5\frac{n}{5} \le i$$

חסם עליון: $T(n) \le \sum_{i=0}^{\log_{\frac{10}{3}}\frac{n}{5}} \left(\frac{1}{2}\right)^i * n = n * \sum_{i=0}^{\infty} \left(\frac{1}{2}\right)^i = n * \theta\left(\frac{1}{\frac{1}{2}}\right) = O(n)$

חסם תחתון: $T(n) \ge \sum_{i=0}^{\log_5\frac{n}{5}} \left(\frac{1}{2}\right)^i * n = n * \sum_{i=0}^{\log_5\frac{n}{5}} \left(\frac{1}{2}\right)^i \ge n * \left(\frac{1}{2}\right)^0 = \Omega(n)$

ולכן, הראינו כי קיים קלט בגודל n בזמן ריצה של $T(n) = \Theta(n)$.