פתרונות

קוד

שאלה 1

1. ממשו את אלגוריתם חיפוש בינארי – איטרטיבי (השלימו את הקוד הבא)
2. ממשו את אלגוריתמי המיון שלמדתם בהרצאה ובתרגול :

Merge Sort, Bubble Sort, Insertion Sort, Selection Sort

ספקו חסם עליון וחסם תחתון וחסם הדוק במידת הצורך עבור כל אחד מהאלגורתמים.

פתרון:

```
	// Method to test above
	public static void main(String[] args) {
		int[] arr = {8,7,6,5,4,3,2,1};
		// Note: That Array must be sorted!!!
		InsertionSort(arr);
		//                  [1, 2, 3, 4, 5, 6, 7, 8]
		// Index:    0, 1, 2, 3, 4, 5, 6 ,7
		int key = 7;
		int result = BinarySerach(arr,key);
		System.out.println(result != -1 ? "Found at index " + result : "Not Found");
	}
	// A function to implement Insertion sort
	public static void InsertionSort(int[] arr) {
		int n = arr.length;
		for(int i=1; i<n; i++) {
			int j = i - 1;
			int key = arr[i];
			while( j >= 0 && arr[j] > key )
			{
				arr[j+1] = arr[j];
				j--;
			}
			arr[j+1] = key;
		}
	}
	// Java implementation of iterative Binary Search
    // Returns index of key if it is present in arr[],
    // else return -1
	public static int BinarySerach(int[] arr, int key) {
		int left = 0;
		int right = arr.length - 1;
		while( left <= right) {
			int middle = left + (right - left) / 2;
		// Check if key is present at mid
			if(arr[middle] == key)
				return middle;
		// If key greater, ignore left half
			if(arr[middle] < key)
				left = middle + 1;
		// If key is smaller, ignore right half
			else
				right = middle - 1;
		}
	  // if we reach here, then element was
	  // not present
		return -1;
}
```

2. בתוך " קטעי קוד במודל"

שאלה 2

נתון מערך A בגודל n וידוע כי $n-\lceil\sqrt{n}\rceil$ האיברים הראשונים שלו ממויינים
תאר/י אלגוריתם שממיין את A בסיבוכיות לינארית

פתרון:
קודם כל נמיין את האיברים $\lceil\sqrt{n}\rceil$ האחרונים במערך בעזרת אחד המיונים שלמדנו בתרגול, לדוגמא מיון בועות - נקבל כי הסיבוכיות היא $O\left(\left(\lceil\sqrt{n}\rceil\right)^2\right)=O(n)$.
לאחר מכן נמזג את 2 תתי המערכים

$$A\left[1\ldots\left[n-\sqrt{n}\right]\right]$$
$$A\left[n-\lceil\sqrt{n}\rceil+1\ldots n\right]$$

נשים לב כי הסיבוכיות למיון היא $O(n)$
והסיבוכיות למיזוג הינה $\Theta\left(n-\lceil\sqrt{n}\rceil+\lceil\sqrt{n}\rceil\right)=\Theta(n)$ סה"כ $\Theta(n)$
הערה: סיבוכיות מיזוג של 2 מערכים **ממויינים** בגודל n ומערך בגודל m היא $\Theta(n+m)$ כאשר הפלט הוא מערך ממויין – האלגוריתם הינו Merge אשר נלמד כחלק מ-MergeSort. נסו לממש זאת בעצמכם!

שאלה 3

נתון מערך A בגודל n של מספרים ממשיים ומספר ממשי נוסף z
א. הציעו אלגוריתם שמכריע האם קיימים שני אינדקסים שונים i,j כך ש-$A[i]+A[j]=z$
ב. מה צריך לשנות באלגוריתם כדי שיחזיר את i,j הנ"ל האם הם קיימים

פתרון:

תמיד רצוי לנסות קודם כל לפתור בעיות בצורה הנאיבית, לאחר שמצאנו פתרון נאיבי לחשוב האם יש דרך לשפר את הביצועים.
הפתרון הנאיבי הוא לעבור על כל הזוגות האפשריים במערך ולבדוק אם קיים זוג אשר נותן את הערך הרצוי, יש סה"כ $\binom{n}{2}$ זוגות אפשריים ולכן נקבל $\Theta(n^2)$
(במידה ולא הכרתם את המושג, תלמדו זאת בבדידה – בכל אופן יש מקסימום n^2 זוגות אפשריים ולכן נקבל אותו סדר גודל)

מימוש היותר יעיל:

א. קודם כל נמיין את המערך בעזרת אחד מאלגורתמי המיון שנלמדו בהרצאה ובתרגול.
לאחר מכן, נעבור על כל האיברים במערך ועבור כל איבר i במערך נחפש האם קיים אינדקס j כך ש $A[i]+A[j]=z$, זאת ע"י חיפוש בינארי עבור $z-i$
סה"כ סיבוכיות הינה הסיבוכיות מיון + $O(n\cdot logn)$ עבור הלולאה הפנימית.
ניתן למיין בעזרת $MergeSort$ ולכן נקבל סה"כ $O(n\ logn)$

ב. בגלל שאנחנו ממיינים את במערך, האינדקסים של המערך הממויין לא תואמים לאינדקסים של המערך הישן, ולכן נעזר במערך עזר על מנת למפות אינדקס חדש לאינדקס ישן.

ניתן לעשות זאת באמצעות טבלת מיפוי (HashMap – ילמד בהמשך הקורס)
בעזרת מערך עזר או בעזרת מערך עזר של זוגות סדורים (יצירת מחלקה עבור זוג סדור)

```java
public static void main(String[] args) {
	int[] arr = {8,7,6,5,4,3,2,1};
	// Note: That Array must be sorted!!!
	InsertionSort(arr);
	int z = 10;
	int[] results = Question3(arr,z);
	System.out.println(results[0] != -1 ?
		"Found at (" + results[0] +"," + results[1] + ")" : "Not Found");
}
// A function to implement  InsertionSort
public static void InsertionSort(int[] arr) {
	int n = arr.length;
	for(int i=0; i<n-1; i++) {
		int min_index = i;
		for(int j=i+1; j<n; j++)
		{
			if(arr[min_index] > arr[j])
				min_index = j;
		}
		int temp = arr[i];
		arr[i] = arr[min_index];
		arr[min_index] = temp;
	}
}
public static int[] Question3(int[] arr, int z) {
	int[] ans = {-1,-1}; // O(1)
	for(int i=0,n=arr.length; i<n; i++) { // O(n)
		int j = Arrays.binarySearch(arr, z - arr[i]); // O(n logn)
		if(j > 0) { // O(1)
			ans[0] = i;
			ans[1] = j;
			break;
		}
	}
return ans;
}
```

<u>מימוש היותר יעיל</u> (בסדר גודל אותו דבר, אלא ההבדל הוא בקבועים)
אותו מימוש, רק המימוש של המתודה Question3 שונה, באופן הבא:

```java
public static int[] Question3(int[] arr, int z) {
	int[] ans = {-1,-1}; // O(1)
	int left = 0;
	int right = arr.length - 1;
	while(left <= right) { // O(n)!
		if(arr[right] + arr[left] == z) {
			ans[0] = left;
			ans[1] = right;
			break;
		}
		else if(arr[right] + arr[left] < z)
			left ++;

		else right --;
```

```
        }
        return ans;
}
```

סיבוכיות

שאלה 4

סדרו את הפונקציות הבאות לפי גודלן – העזרו באתר https://www.geogebra.org/graphing

פתרון:

מהגדול לקטן:

$f(x) = x^x$

$g(x) = x!$

$h(x) = 2^x$

$p(x) = x^2$

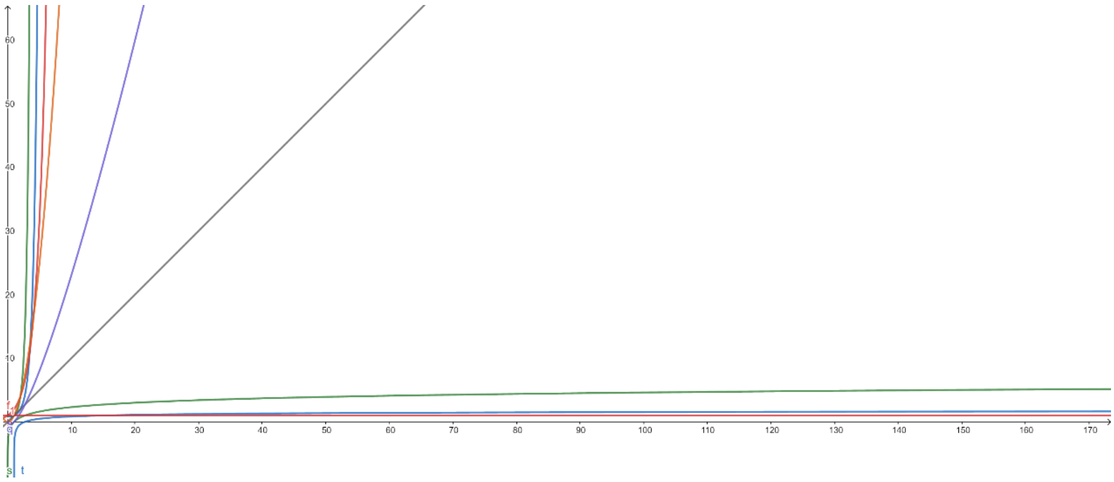
$q(x) = x \cdot \ln(x)$

$r(x) = x$

$s(x) = \ln(x)$

$t(x) = \ln(\ln(x))$

$f_1 : y = 1$



שאלה 5

מצאו חסם הדוק לנוסחאות הבאות:

0. $T(n) = T(n-1) + 1, T(0) = 0$

1. $T(n) = \begin{cases} 3T\left(\frac{n}{3}\right) + n, & n \geq 2 \\ 1, & n = 1 \end{cases}$

2.. $T(n) = \sum_{k=1}^{n-1} (T(k) + T(n-k) + 1), \quad T(0) = 1$

פתרון:

.0

$$T(n) = T(n-1) + 1 = T(n-2) + 1 + 1 = T(n-3) + 1 + 1 + 1 = \cdots T(n-k) + k$$

ולכן עבור $k = n$ נקבל כי $T(n) = T(0) + n = n \in \Theta(n)$

.1

$$T(n) = 3T\left(\frac{n}{3}\right) + n = 3 \cdot \left[3 \cdot T\left(\frac{n}{3^2}\right) + \frac{n}{3}\right] + n = 3^2 \cdot T\left(\frac{n}{3^2}\right) + 2n = \cdots = 3^k \cdot T\left(\frac{n}{3^k}\right) + kn$$

עבור $k = \log_3 n$ נקבל כי:

$$T(n) = 3^{\log_3 n} \cdot T(1) + \log_3 n \cdot n = n + \log_3 n \cdot n = \Theta(n \cdot logn)$$

.2

$$2.. \qquad T(n) = \sum_{k=1}^{n-1} (T(k) + T(n-k) + 1) = 2 \cdot \sum_{\text{ך}=1}^{n-1} T(k) + (n-1)$$

נשים לב כי:

$$T(n-1) = 2 \cdot \sum_{k=1}^{n-2} T(k) + (n-2)$$

ולכן

$$T(n) - T(n-1) = 2 \cdot \sum_{\text{ך}=1}^{n-1} T(k) + (n-1) - 2 \cdot \sum_{k=1}^{n-2} T(k) - (n-2) = 2T(n-1) + 1$$

ולכן

$$T(n) = 3T(n-1) + 1$$

ולכן

$$T(n) = \cdots = 3^k T(n-k) + kn$$

ולכן עבור $k = n$ נקבל כי $T(n) = 3^n + n^2 = O(3^n)$

שאלה 6

נתונות 4 פונקציות הבאות:

$$f_1(x) = \log(x^x x!), \quad f_2(x) = x\log(x), \quad f_3(x) = 3^{\log_4 x}, \quad f_4(x) = 9^{\log_3 x}$$

סדרו את הפונקציות לפי סדר אסימפטוטי $O(\ldots)$, מן ה"קטנה" ל- "גדולה". אם מתקיים $f_i = \theta(f_j)$ ציינו זאת. הוכיחו את תשובתכם.

פתרון:

נשים לב כי:

$$f_4(x) = 9^{\log_3 x} = 9^{\frac{\log_9 x}{\log_3 x}} = 9^{2\log_9 x} = 9^{\log_9 x^2} = x^2$$

$$f_3(x) = 3^{\log_4 x} = 3^{\frac{\log_3 x}{\log_3 4}} = \left(3^{\log_3 x}\right)^{\frac{1}{\log_3 4}} = x^{\frac{1}{1.261}}$$

$$f_2(x) = xlogx$$

$$f_1(x) = \log(x! \cdot x^x) = \log(x!) + \log(x^x) = \log(x!) + xlog(x)$$

נוכיח כי $f_3(x) = O\big(f_2(x)\big)$

$f_2(x) = \Theta\big(f_1(x)\big)$

$f_2(x) = O\big(f_4(x)\big)$

ולכן הסדר הינו:

$$f_3(x) = x^{\frac{1}{1.261}}$$

$$f_2(x) = xlogx, \qquad f_1(x) = \log(x!) + xlog(x)$$
$$f_4(x) = x^2$$

טענה: $f_3(x) = O\big(f_2(x)\big)$

הוכחה:

נראה כי קיימים c, x_0 כך שלכל $x > x_0$ מתקיים כי $|f_3(x)| \le c \cdot |f_2(x)|$ ולכן $f_3(x) = O(f_2(x))$

נשים לב כי :

$$f_3(x) = x^{\frac{1}{1.261}} \underbrace{\le}_{x > x_0 = 1} x \underbrace{\le}_{x > x_1 = 10} x \cdot \log(x) = f_2(x)$$

ולכן עבור $c = 1$ ו- $x_0 = 10$ נקבל כי $f_3(x) = O(f_2(x))$

טענה: $f_2(x) = \Theta\big(f_1(x)\big)$

הוכחה:

$$0 \le f_2(x) = xlog(x) = \log(x^x) \underbrace{\le}_{x \ge x_0 = 1} \log(x!) + \log(x^x) = f_1(x) \le \log(x^x) + \log(x^x)$$
$$= 2xlog(x) = 2 \cdot f_2(x)$$

ולכן קיימים שלושה קבועים $c_1 = 1$ ו- $c_2 = 2$ ו- $x_0 = 1$

כך שלכל $x > x_0$

מתקיים $0 \le 1 \cdot |f_2(x)| \le f_1(x) \le 2 \cdot |f_2(x)|$

ולכן ולכן לפי הגדרה של Θ מתקיים כ $f_2(x) = \Theta(f_1(x))$

טענה: $f_2(x) = O\big(f_4(x)\big)$

$$f_2(x) = xlogx \underbrace{\le}_{x > x_0 = 0} x^2 = f_4(x)$$

ולכן עבור $c = 1$ ו- $x_0 = 0$ נקבל כי $f_2(x) = O(f_4(x))$

- ניתן לפתור זאת גם ע"י לימטים

שאלה 7

הוכח או הפרך :

1. אם $f(x), g(x) > 1$ אזי $f(x) + g(x) = O\big(f(x) \cdot g(x)\big)$
2. אם $f(n) = O\big(g(n)\big)$ אזי $2^{f(n)} = O\big(2^{g(n)}\big)$

פתרון:

נוכיח טענה זו.

נגדיר $h(x) = f(x) + g(x)$ ונגדיר $k(x) = f(x) \cdot g(x)$

מספיק להוכיח כי $h(x) = O(k(x))$

כלומר צ"ל כי קיימים קבועים c, x_0 כך שלכל $x > x_0$ מתקיים:

$$0 \le h(x) \le c \cdot k(x)$$

נניח בה"כ כי $f(x) \le g(x)$

ולכן עבור $c = 2$ ו- $x_0 = 1$ נקבל כי:

$$0 \le h(x) \le 2 \cdot k(x)$$

היות ומתקיים כי: (לפי טענת עזר *)

$$0 \le f(x) + g(x) \le 2 \cdot f(x) \cdot g(x)$$

כי לפי ההנחה $f(x) \le g(x)$ ובנוסף $f(x), g(x)$ פונקציות חיוביות גדולות מ-1.

טענת עזר * : $f(x) + g(x) \le 2 \cdot f(x) \cdot g(x)$

נניח בשלילה כי $2\cdot f(x)\cdot g(x) < f(x)+g(x)$
ולכן $2\cdot f(x)\cdot g(x) - g(x) < f(x)$
ולכן $g(x)\cdot(2\cdot f(x)-1) < f(x)$
אבל ידוע כי $2\cdot f(x)-1>1$ ולכן קבלנו כי $g(x)\cdot c < f(x)$ כאשר $c>1$
בסתירה לכך ש- $f(x)\le g(x)$

2. **הטענה אינה נכונה.**
יהי $f(n)=2\log(n)$
ו- $g(n)=\log(n)$
נשים לב כי $f(n)=O\big(g(n)\big)$
עבור $c=2$ ו-$n_0=0$
כי $f(n)=2\log(n)\le 2\cdot g(n)=2\cdot\log(n)$
אולם, $2^{f(n)}=2^{2\log(n)}=2^{\log(n^2)}=n^2$
ו- $2^{g(n)}=2^{\log(n)}=n$
ו- $n^2\ne O(n)$

<u>שאלה 8</u>

נשים לב כי סיבוכיות חיפוש בינארי מוגדרת באופן הבא:

$$T(n)=\begin{cases}T\left(\frac{n}{2}\right)+2, & n\ge 2\\ 1, & n\le 2\end{cases}$$

בעוד שהסיבוכיות של חיפוש טרנרי הינה מוגדרת באופן הבא:

$$T(n)=\begin{cases}T\left(\frac{n}{3}\right)+4, & n\ge 2\\ 1, & n\le 2\end{cases}$$

היות ויש 4 השוואות בכל שלב ברקורסיה.
נקבל כי עבור חיפוש בינארי:

$$T(n)=\cdots=k\cdot 2+T\left(\frac{n}{2^k}\right)=2\log_2(n)$$

בעוד שעבור חיפוש טרנרי נקבל:

$$T(n)=\cdots=4\cdot k+T\left(\frac{n}{3^k}\right)=4\log_3(n)$$

ידוע כי $2\log_2(n)=\frac{2\log_3(n)}{\log_3(2)}=3.17\log_3(n)$
ולכן נקבל כי עדיף להשתמש בחיפוש בינארי מבחינת סיבוכיות.