Abdullah Çelik
171044002

CSE 321
Homework 1

1) $f_1(n) = 1$, $f_2(n) = \log(\log n)$, $f_3(n) = \log n$, $f_4(n) = n^a$, $f_5(n) = n$, $f_6(n) = n\log n$, $f_7(n) = n^2$, $f_8(n) = n^3$, $f_9(n) = b^n$, $f_{10}(n) = n.b^n$, $f_{11}(n) = n!$, $f_{12}(n) = b^{n^c}$. $0<a<1$, $b>1$, $c>1$. We can sort these functions in increasing order of asymptotic growth as

$$f_1 < f_2 < f_3 < f_4 < f_5 < f_6 < f_7 < f_8 < f_9 < f_{10} < f_{11} < f_{12}$$

we can sort the functions given in the question as follows.

$$T_2 < T_1 < T_4 < T_5 < T_3 < T_8 < T_6 < T_7$$

$$* \lim_{n\to\infty} \frac{f(n)}{g(n)} = \begin{cases} 0\,; & \text{if } f(n) \in o(g(n)) \\ c\,; & c>0 \text{ if } f(n) \in \Theta(g(n)) \\ \infty\,; & \text{if } f(n) \in w(g(n)) \\ 1\,; & \text{if } f(n) \sim (g(n)) \end{cases}$$

* For $i=2$, $j=1$, $T_2 < T_1$. Prove:

$\lim_{n\to\infty} \frac{T_2(n)}{T_1(n)} = \lim_{n\to\infty} \frac{4\log(\log n)}{3\log n + 3} = \frac{\infty}{\infty}$. I can use L'Hospital rule.

$$\lim_{n\to\infty} \frac{(4\log(\log n))'}{(3\log n + 3)'} = \lim_{n\to\infty} \frac{\frac{4}{n.(\ln 2)^2.\log n}}{\frac{3}{n.\ln 2}} = \lim_{n\to\infty} \frac{4}{3.\ln 2.\log n} = \frac{1}{\infty} = 0.$$

$$T_2(n) \in o(T_1(n)) \Rightarrow T_2(n) \in O(T_1(n)).$$

* For $i=1$, $j=4$, $T_1 < T_4$. Prove:

$\lim_{n\to\infty} \frac{T_1(n)}{T_4(n)} = \lim_{n\to\infty} \frac{3\log n + 3}{2000n + 1} = \frac{\infty}{\infty}$. I can use L'Hospital rule.

$$\lim_{n\to\infty} \frac{(3\log n + 3)'}{(2000n + 1)'} = \lim_{n\to\infty} \frac{\frac{3}{n.\ln 2}}{2000} = \lim_{n\to\infty} \frac{3}{2000.n.\ln 2} = \frac{1}{\infty} = 0$$

$$T_1(n) \in o(T_4(n)) \Rightarrow T_1(n) \in O(T_4(n))$$

* For $i=4$, $j=5$, $T_4 < T_5$. Prove:

$$\lim_{n\to\infty} \frac{T_4(n)}{T_5(n)} = \lim_{n\to\infty} \frac{2000n+1}{\left(\frac{n}{6}\right)^2} = \frac{\infty}{\infty}.$$ I can use L' Hospital rule.

$$\lim_{n\to\infty} \frac{(2000n+1)'}{\left(\left(\frac{n}{6}\right)^2\right)'} = \lim_{n\to\infty} \frac{2000}{\frac{n}{18}} = \frac{1}{\infty} = 0$$

$$T_4(n) \in o(T_5(n)) \Rightarrow T_4(n) \in O(T_5(n))$$

* For $i=5$, $j=3$, $T_5 < T_3$. Prove:

$$\lim_{n\to\infty} \frac{T_5(n)}{T_3(n)} = \lim_{n\to\infty} \frac{\left(\frac{n}{6}\right)^2}{n^5+8n^4}.$$ Divide n^2 by both sides.

$$\lim_{n\to\infty} \frac{\frac{n^2}{36}\cdot\frac{1}{n^2}}{(n^5+8n^4)\cdot\frac{1}{n^2}} = \frac{\frac{1}{36}}{n^3+8n^2} = \frac{1}{\infty} = 0$$

$$T_5(n) \in o(T_3(n)) \Rightarrow T_5(n) \in O(T_3(n))$$

* For $i=3$, $j=8$, $T_3 < T_8$. Prove:

$$\lim_{n\to\infty} \frac{T_3(n)}{T_8(n)} = \lim_{n\to\infty} \frac{n^5+8n^4}{2^n+n^3} = \frac{\infty}{\infty}.$$ I can use L' Hospital rule.

$$\lim_{n\to\infty} \frac{(n^5+8n^4)'}{(2^n+n^3)'} = \lim_{n\to\infty} \frac{5n^4+32n^3}{2^n.\ln 2+3n^2} = \frac{\infty}{\infty}.$$ when we derive four times, equation is

$$\lim_{n\to\infty} \frac{\text{constant}}{2^n.\ \text{constant}} = \frac{1}{\infty} = 0$$

$$T_3(n) \in o(T_8(n)) \Rightarrow T_3(n) \in O(T_8(n))$$

* For $i=8$, $j=6$, $T_8 < T_6$. Prove:

$$\lim_{n\to\infty} \frac{T_8(n)}{T_6(n)} = \lim_{n\to\infty} \frac{2^n + n^3}{3^n + n^2}$$. Divide both sides by 3^n

$$\lim_{n\to\infty} \frac{\left(\frac{2}{3}\right)^n + \frac{n^3}{3^n}}{1 + \frac{n^2}{3^n}} = \frac{\lim_{n\to\infty} \left(\frac{2}{3}\right)^n + \frac{n^3}{3^n}}{\lim_{n\to\infty} 1 + \frac{n^2}{3^n}} = \frac{\lim_{n\to\infty} \left(\frac{2}{3}\right)^n + \lim_{n\to\infty} \frac{n^3}{3^n}}{1 + \lim_{n\to\infty} \frac{n^2}{3^n}}$$

$$= \frac{0 + \lim_{n\to\infty} \frac{n^3}{3^n}}{1 + \lim_{n\to\infty} \frac{n^2}{3^n}}.$$ Note: $\lim_{x\to\infty} \frac{x}{e^x} = \frac{\infty}{\infty} \Rightarrow \lim_{x\to\infty} \frac{x'}{(e^x)'} = \lim_{x\to\infty} \frac{1}{e^x} = \frac{1}{\infty} = 0$

Using above rule, equation is: $\frac{0+0}{1+0} = \frac{0}{1} = 0$

$T_8(n) \in o(T_6(n)) \Rightarrow T_8(n) \in O(T_6(n))$

* For $i=6$, $j=7$, $T_6 < T_7$. Prove:

$$\lim_{n\to\infty} \frac{T_6(n)}{T_7(n)} = \lim_{n\to\infty} \frac{3^n + n^2}{n^n + 1000n}$$. Divide both sides by n^n.

$$\lim_{n\to\infty} \frac{\frac{3^n}{n^n} + \frac{n^2}{n^n}}{1 + \frac{1000n}{n^n}} = \frac{\lim_{n\to\infty} \frac{3^n}{n^n} + \lim_{n\to\infty} n^{2-n}}{1 + \lim_{n\to\infty} \frac{1000}{n^{n-1}}}$$

$$\lim_{n\to\infty} \frac{1000}{n^{n-1}} = \frac{1}{\infty} = 0$$

$$\lim_{n\to\infty} n^{2-n} = \lim_{n\to\infty} \left(\frac{1}{n}\right)^{n-2} = \frac{1}{\infty} = 0$$

$$\lim_{n\to\infty} \frac{3^n}{n^n} = \lim_{n\to\infty} \left(\frac{3}{n}\right)^n = \lim_{n\to\infty} e^{n \ln \frac{3}{n}}, \quad \lim_{n\to a} g(n) = b \Rightarrow \lim_{n\to\infty} n \cdot \ln \frac{3}{n} = -\infty$$

and $f(v) = e^v \Rightarrow \lim_{v\to\infty} e^v = 0$

$T_6(n) \in o(T_7(n)) \Rightarrow T_6(n) \in O\, T_7(n)$

* If $a<b$ and $b<c$ then $a<c$. I don't need to prove $a<c$ again. So in increasing order of asymptotic growth is

$T_2 < T_1 < T_4 < T_5 < T_3 < T_8 < T_6 < T_7$

2) a) $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{99n}{n} = 99$. So I can say that

$f(n) \in \Theta(g(n))$

b) $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{2n^4+n^2}{(\log n)^6} = \frac{\infty}{\infty}$. So I can use L'Hospital rule

$\lim_{n\to\infty} \frac{(2n^4+n^2)'}{((\log n)^6)'} = \lim_{n\to\infty} \frac{8n^3+2n}{6.(\log n)^5 . \frac{1}{n.\ln 2}}$. Constants are negligible. Equation is

$\lim_{n\to\infty} \frac{n^4+n^2}{(\log n)^5} = \frac{\infty}{\infty}$. If i derive four times, equation is

$\lim_{n\to\infty} \frac{n^4+n^2}{\log n} = \frac{\infty}{\infty}$. $\lim_{n\to\infty} \frac{(n^4+n^2)'}{(\log n)'} = \lim_{n\to\infty} \frac{4n^3+2n}{\frac{1}{n.\ln 2}} = \lim_{n\to\infty} \ln 2.(4n^4+2n^2)$

$= \infty$. So $f(n) \in w(g(n)) \Rightarrow f(n) \in \Omega(g(n))$

c) $f(n) = \sum_{x=1}^{n} x = \frac{n.(n+1)}{2}$

$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{n^2+n}{8n+2\log n} = \frac{\infty}{\infty}$. Applying L'Hospital rule

$\lim_{n\to\infty} \frac{(n^2+n)'}{(8n+2\log n)'} = \lim_{n\to\infty} \frac{2n+1}{8+\frac{2}{n.\ln 2}} = \lim_{n\to\infty} \frac{2n+1}{\frac{8n+2\log e}{n}} = \lim_{n\to\infty} \frac{2n^2+n}{8n+2\log e} = \frac{\infty}{\infty}$

Applying L'Hospital rule, $\lim_{n\to\infty} \frac{4n+1}{8} = \infty$

$f(n) \in w(g(n)) \Rightarrow f(n) \in \Omega(g(n))$

d) $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{3^n}{5^{\sqrt{n}}} = \lim_{n\to\infty} \frac{3^{\sqrt{n}} . 3^{\sqrt{n}}}{5^{\sqrt{n}}} = \lim_{n\to\infty} \left(\frac{9}{5}\right)^{\sqrt{n}} = \infty$

$f(n) \in w(g(n)) \Rightarrow f(n) \in \Omega(g(n))$

3) a) It looks from the first element to the last element of the array and finds the element that repeats more than half the number of elements of the array.

Inputs. int nums[] → address of the starting position of the array.

int n → size of the array.

Output: int → integer number. If this number is -1, it means there is no such element is found or found element is -1.

b)

```
int myFunction (int nums[], int n)                Cost:
{
    for (int i=0; i<n; i++){   - - - - - - - -    Θ(n)
        int count = 1;         - - - - - - - - -  Θ(1)
        for (int j=i+1; j<n; j++)  - - - - - -    Θ(n)
            if (nums[j] == nums[i])  - - - - -    Θ(1)
                count ++;       - - - - - - - -   Θ(1)
        if (count > n/2)        - - - - - - - -   Θ(1)
            return nums[i];     - - - - - - - -   Θ(1)
    }
    return -1;                  - - - - - - - -   Θ(1)
}
```

$T_{avg} \in O(n^2)$

- Best case is to find item in the first index. It means looking at all elements of the array only once. $T_{best} \in \Theta(n)$
- Worst case is not find item. It means looking at elements $\sum_{x=1}^{n} x = \frac{n.(n+1)}{2}$ times. $T_{worst} \in \Theta(n^2)$

4) a) A different version of the one implemented in question 3. Finds elemet that repeats more than half the number of the array.

Inputs: int nums[] → address of the starting position of the array.
int n → size of the array

Output: int → integer number. If this number is -1, it means there is no such element is found or found item is -1.

b)

	Cost
`int myFunction2 (int nums[], int n)`	
`{`	
`    int i, *map, max=0;`	$\Theta(1)$
`    for (i=0; i<n; i++)`	$\Theta(n)$
`        if (nums[i] >max)`	$\Theta(1)$
`            max = nums[i];`	$\Theta(1)$
`    map = (int*) calloc (max+1, sizeof(int));`	$\Theta(1)$
`    for (i=0; i<n; i++)`	$\Theta(n)$
`        map [num[i]] ++;`	$\Theta(1)$
`    for (i=0; i<n; i++)`	$\Theta(n)$
`        if (map[nums[i]] > n/2)`	$\Theta(1)$
`            return nums[i];`	$\Theta(1)$
`    return -1;`	$\Theta(1)$
`}`	

$T(n) \in \Theta(n)$

Checks all elements in both best case and worst case.

$T_{best} \in \Theta(n)$

$T_{worst} \in \Theta(n)$

$T_{avg} \in \Theta(n)$

5) As we can see, the 4th algorithm is much faster than the 3rd algorithm. But it also uses a lot of memory (from heap). In large arrays:

- If speed is more important than memory, algorithm 4 should be used. Otherwise algorithm 3 should be used.

• What makes them better?

- The returned value in the worst case of both algorithms may not fully explain its situation. (If array contains negative elements.)
- In algorithm 3, the first loop can be $i < n/2$, not $i < n$. Because when $i > n/2$, count can never be greater than $n/2$.
- In algorithm 4, where the elements of the array are less than 0, the max value will be a negative number and an error is occurred in allocation. Where max value is positive, map[nums[i]] will cause out of index if array contains negative value.
- In an array where the elements in it are repeated frequently, most of the space we allocate will not be used. The reason for this is that we allocate max +1 space. Here we can allocate as much as space as the number of unique items and the rest of the process can be done accordingly.

6) a) function (a[0... n-1], b[0... m-1])

		Cost
max1 = a[0] , max2 = b[0]	----	$\Theta(1)$
for i=1 to n do	-----	$\Theta(n)$
if max1 > a[i]	------	$\Theta(1)$
max1 = a[i]	-----	$\Theta(1)$
for i=1 to m do	----	$\Theta(m)$
if max2 > b[i]	-----	$\Theta(1)$
max2 = b[i]	-----	$\Theta(1)$
return max1 * max2	----	$\Theta(1)$

- $T(n) \in \Theta(n)$
- It will look all the elements of both arrays in best case and worst case.

$T_{best} \in \Theta(n)$, $T_{worst} \in \Theta(n)$, $T_{ave} \in \Theta(n)$

b) function (a[0... n-1], b[0... n-1])

		cost
arr = new int[n * m]	----	$\Theta(1)$
i=0, temp=0	----	$\Theta(1)$
for i=0 to n do	----	$\Theta(n)$
arr[i] = a[i]	----	$\Theta(1)$
for j=0 to m do	----	$\Theta(m)$
arr[i+j] = b[j]		
for i=0 to n * m do	----	$\Theta(n*m)$
for j=i+1 to n * m do	----	$\Theta(n*m)$
if arr[i] > arr[j]	---	$\Theta(1)$
temp = arr[i]	-----	$\Theta(1)$
arr[i] = arr[j]	----	$\Theta(1)$
arr[j] = temp	----	$\Theta(1)$
return arr	-----	$\Theta(1)$
		$\Theta(1)$

- n and m is a number. We can a more general expression, $T(n) \in \Theta(n^2)$
- It will place all elements and sorts. So $T_{best} \in \Theta(n^2)$

$T_{worst} \in \Theta(n^2)$

c) function (arr [0 ... n-1] , item) — Cost

```
arr2 = new int [n+1]          --- Θ(1)
i=0                           --- Θ(1)

for i=0 to size do            --- Θ(n)
    arr2 [i] = arr [i]        --- Θ(n)
arr2 [i] = item               --- Θ(n)
return arr2                   --- Θ(n)
```

- $T(n) \in \Theta(n)$
- It will place all items both best case and worst case. $T_{best} \in \Theta(n)$
  $T_{best} \in \Theta(n)$

d) function (arr [0 ... n-1] , index) — Cost

```
if index < 0 or index >= size  -- Θ(1)
    return arr                 --- Θ(1)
arr2 = new int [size - 1]      --- Θ(1)

for i=0 to index do            --- Θ(n)
    arr2 [i] = arr [i]         --- Θ(1)
for i = index +1 to size do    --- Θ(n)
    arr2 [ i-1 ] = arr [i]     --- Θ(1)
return arr2                    --- Θ(1)
```

- $T(n) \in \Theta(n)$
- It will place all items both best and worst case. $T_{best} \in \Theta(n)$
  $T_{worst} \in \Theta(n)$