ÖMER GEVİK
161044004

CSE 321

Introduction to Algorithm Design
HOMEWORK 1

1 a) The running time of algorithm A is at least $O(n^2)$ doesn't sound a scientifically. $O(n^2)$ means the big-Oh notation. So it is an upper bound. Inside of big-Oh notation n^2 function causes at most quadratic time complexity. The "at least" keyword doesn't suit of definition of big oh notation so it is not scientifically.

b) i) $2^{n+1} = O(2^n)$? ✓

$\Rightarrow 2^{n+1} = 2^1 . 2^n$ and from definition of big-Oh $\rightarrow 2^1.2^n \leq c.2^n$, if $c=2$, then $n \geq 0$ for all

So $2^{n+1} = O(2^n)$ is true.

ii) $2^{2n} = O(2^n)$?

$\Rightarrow 2^{2n} \leq c.2^n$ from definition of big-Oh $\rightarrow 2^n \leq c \rightarrow$ c is constant, left hand side is always bigger.

So $2^{2n} = O(2^n)$ is not true.

c) $\max(f(n), g(n)) = \Theta(f(n)+g(n))$?

$\Theta(f(n)+g(n)) = O(f(n)+g(n))$ and $\Omega(f(n)+g(n))$

$O(f(n)+g(n)) = f(n)+g(n) \leq c.(f(n)+g(n))$, $1 \leq c$ if $c=1$ for all $n \geq 0$ it is true ✓

$\Omega(f(n)+g(n)) = f(n)+g(n) \geq c.(f(n)+g(n))$, $1 \geq c$, if $c=1$ for all $n \geq 0$ it is true ✓

So if $O(f(n)+g(n))$ and $\Omega(f(n)+g(n))$ are true then also $\Theta(f(n)+g(n))$ is true. ✓

---

2 a) $n^{1.01} = \Theta(n\log^2 n)$? $\Rightarrow \lim_{n\to\infty} \frac{n^{1.01}}{n\log^2 n} = \lim_{n\to\infty} \frac{n^{0.01}}{\log^2 n} \xrightarrow{\text{L'Hopital}} \lim_{n\to\infty} \frac{0.01.n^{-0.99}}{2\log n . \frac{1}{\ln n}}$ ... L'Hopital In the result we see $\log^2 n$ grows faster than $n^{0.01}$

$\Rightarrow \lim_{n\to\infty} \frac{n^{1.01}}{n\log^2 n} = \frac{1}{\infty} = 0 \Rightarrow n^{1.01} = O(n\log^2 n)$

b) $n! = \Theta(2^n)$? $\Rightarrow$ Stirling's formula: $n! \cong \sqrt{2\pi n}.\left(\frac{n}{e}\right)^n$ for large n.

$\lim_{n\to\infty} \frac{n!}{2^n} = \lim_{n\to\infty} \frac{\sqrt{2\pi n}.\left(\frac{n}{e}\right)^n}{2^n} = \lim_{n\to\infty} \sqrt{2\pi n}.\left(\frac{n}{2e}\right)^n = \infty \Rightarrow n! = \Omega(2^n)$

c) $\sqrt{n} = \Theta((\log n)^3)$? $\Rightarrow \lim_{n\to\infty} \frac{\sqrt{n}}{(\log n)^3} \xrightarrow{\text{L'Hopital}} \lim_{n\to\infty} \frac{\frac{1}{2\sqrt{n}}}{\frac{3\log^2 n}{n\ln 2}} = \lim_{n\to\infty} \frac{n.\ln 2}{2\sqrt{n}.3\log^2 n} \xrightarrow{\text{L'Hopital}} \frac{\ln 2}{6}\lim_{n\to\infty} \frac{1}{\frac{\log^2 n}{2\sqrt{n}} + \frac{2\log n}{\sqrt{n}\ln 10}} = \infty$

$\Rightarrow$ So we see that $(\log n)^3$ grows faster than $\sqrt{n}$. $\sqrt{n} = \Omega((\log n)^3)$

d) $n.2^n = \Theta(3^n)$? $\Rightarrow \lim_{n\to\infty} \frac{n.2^n}{3^n} \Rightarrow 2^n$ grows faster than n and 3^n grows faster than 2^n and $n.2^n$. Using L'Hospital we can prove it.

So $\lim_{n\to\infty} \frac{n.2^n}{3^n} = \frac{1}{\infty} = 0 \Rightarrow n.2^n = O(3^n)$

e) $\sum_{i=1}^{n} i^k = \Theta(n^{k+1})$? $\Rightarrow \sum_{i=1}^{n} i^k = 1^k+2^k+3^k+\dots+n^k \Rightarrow \lim_{n\to\infty} \frac{1^k+2^k+\dots+n^k}{n^{k+1}} = \lim_{n\to\infty} \frac{\overbrace{1^k+2^k+\dots+n^k}^{n \text{ terms}}}{n.n^k} \leq \frac{n.n^k}{n.n^k} \in O(n^{k+1})$

$\sum_{i=1}^{n} i^k = \sum_{i=1}^{\frac{n}{2}} i^k + \sum_{i=\frac{n}{2}+1}^{n} i^k \geq \sum_{i=\frac{n}{2}+1}^{n} i^k = ([\frac{n}{2}]+1)^k + ([\frac{n}{2}]+2)^k + \dots + n^k \geq \frac{n}{2}.\left(\frac{n}{2}\right)^k = \frac{n^{k+1}}{2^{k+1}} \Rightarrow ([\frac{n}{2}]+1)^k \geq \left(\frac{n}{2}\right)^k$

$[\frac{n}{2}] \geq \frac{n}{2} \Rightarrow \sum_{i=1}^{n} i^k \in \Omega(n^{k+1}) \rightarrow \sum_{i=1}^{n} i^k \in O(n^{k+1})$ & $\sum_{i=1}^{n} i^k \in \Omega(n^{k+1}) \Rightarrow \sum_{i=1}^{n} i^k \in \Theta(n^{k+1})$

(2)

f) $2^n = \Theta(2^{n+1})$? $\Rightarrow$ $2^n \leq c.2^{n+1} \Rightarrow 2^n \leq c.2^n.2$, if $c=\frac{1}{2}$ then for all $n\geq 0$ it's $\underline{2^n \in O(2^{n+1})}$

$\Rightarrow$ $2^n \geq c.2^{n+1} \Rightarrow 2^n \geq c.2^n.2$, if $c=\frac{1}{2}$ then for all $n \geq 0$ it's $\underline{2^n \in \Omega(2^{n+1})}$

So $2^n \in O(2^{n+1})$ & $2^n \in \Omega(2^{n+1})$ then $\underline{2^n \in \Theta(2^{n+1})}$

g) $n^{\frac{1}{2}} = \Theta(5^{\log_2 n})$? $\Rightarrow \lim_{n\to\infty} \frac{n^{\frac{1}{2}}}{5^{\log_2 n}} \Rightarrow \lim_{n\to\infty} \frac{n^{\frac{1}{2}}}{n^{\log_2 5}} \Rightarrow \lim_{n\to\infty} n^{\overbrace{(\frac{1}{2}-\log_2 5)}^{>1}}_{\text{negative}} = \lim_{n\to\infty} \frac{1}{n^{\cdot}} = \frac{1}{\infty} = 0 \Rightarrow \underline{n^{\frac{1}{2}} \in O(5^{\log_2 n})}$

h) $\log_2 2n = \Theta(\log_2 3n)$? $\Rightarrow \lim_{n\to\infty} \frac{\log_2(2n)}{\log_2(3n)} \xRightarrow{\text{L'Hopital}} \lim_{n\to\infty} \frac{\frac{1}{\ln 10 \cdot 2n}\cdot 2}{\frac{1}{\ln 10 \cdot 3n}\cdot 3} = \lim_{n\to\infty} \frac{\frac{1}{\ln 10 \cdot n}}{\frac{1}{\ln 10 \cdot n}} = 1 \overset{\text{constant}}{\Rightarrow} \underline{\log_2(2n) \in \Theta(\log_2(3n))}$

---

(3) $\log n$, $\sqrt{n+10}$, $n+10$, 10^n, 100^n, $n^2\log n$, $32^{\log n}$, n^6

$\frac{f(n+1)}{f(n)} \Rightarrow \frac{\log(n+1)}{\log(n)}$, $\frac{\sqrt{n+11}}{\sqrt{n+10}}$, $\frac{n+11}{n+10} = 1+\frac{1}{n+10}$, $\frac{10^{n+1}}{10^n} = 10$, $\frac{100^{n+1}}{100^n} = 100$, $\frac{(n+1)^2\log(n+1)}{n^2\log n}$, $\frac{32^{\log n+1}}{32^{\log n}} = 32^{\log\frac{n+1}{n}}$, $\frac{(n+1)^6}{n^6}$

$32^{\log n} \Rightarrow n^{\log 32}$

$\sqrt{n+10} > \log n$ because $\frac{\sqrt{n+11}}{\sqrt{n+10}} > \frac{\log(n+1)}{\log(n)}$ when $n \geq 2$

$n+10 > \sqrt{n+10}$ because $\frac{n+11}{n+10} = 1+\frac{1}{n+10} > \frac{\sqrt{n+11}}{\sqrt{n+10}}$ when $n \geq 0$

$n^2\log n > n+10$ because $\left(\frac{n+1}{n}\right)^2 \log_n (n+1) > \frac{n+11}{n+10} = 1+\frac{1}{n+10}$ when $n \geq 2$

$32^{\log n} > n^2\log n$ because $n^{\log 32} > n^2\log n$ when $n \geq 2$

$n^6 > 32^{\log n}$ because $n^6 > n^{\log 32}$ when $n \geq 0$

$10^n > n^6$ because $\frac{10^{n+1}}{10^n} = 10 > \frac{(n+1)^6}{n^6}$ when $n \geq 0$

$100^n > 10^n$ because $\frac{100^{n+1}}{100^n} = 100 > \frac{10^{n+1}}{10^n} = 10$ when all n

For the result:

$$\log n < \sqrt{n+10} < n+10 < n^2\log n < 32^{\log n} < n^6 < 10^n < 100^n$$

---

(4) a)

```
Find Min (root):
    while root.left != null do
        root = root.left
    end while
  return root
end
```

Height of tree is n. So time complexity of FindMin is T(n)

$T(n) \in \underline{O(n)}$ for all left nodes.

b)

```
Search (root, node):
    if root == null || root.val == node do
        return root  } 4
    end if
    if root.val < node do
        return Search(root.right, node)
    end if
  return Search(root.left, node)
end
```

(braces over the condition parts labelled 1, 2, 3)

Height of tree is n. So time complexity of Search is T(n)

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

In Master Theorem: $T(n) = aT(n/b) + f(n)$

$a=2$, $b=2$, $d=0$

$a \quad b^d \Rightarrow 2 > \overset{1}{2^0} \Rightarrow T(n) = \Theta(n^{\log_b a})$

$\Rightarrow T(n) = \Theta(n^{\log_2 2}) = \Theta(n) \supset \underline{O(n)}$

Ömer ÇEVİK
161044004

c) Delete(root, node):

```
if root == null do                                             }1
    return null                                                }1
end if
root.left = Delete(root.left, node)                            }1
root.right = Delete(root.right, node)                          }1
if root.data == node && root.left == null && root.right == null do
    return null                                                }1
end if
return root                                                    }1
end
```

Height of tree is n. So time complexity of Delete is T(n).

$T(n) = 2T(n/2) + 11$

Master Theorem: $T(n) = aT(n/b) + f(n)$ $\left.\right\}$ $a=2$, $b=2$, $d=0$

$a \quad b^d \Rightarrow 2 > \underbrace{2^0}_{1}$ $\quad T(n) = \Theta(n^{\log_b a})$

$\Rightarrow T(n) = \Theta(n^{\log_2 2})$

$= \Theta(n)$

$\Theta(n) \supset \underline{O(n)}$

d) Merge(firstTree, secondTree):

```
if firstTree == null do                  }1
    return secondTree                    }1
end if
if secondTree == null do                 }1
    return firstTree                     }1
end if
list1 = firstTree.toList();  }n
list2 = SecondList.toList(); }n
while ( i < list1.size() && j < list2.size() ) do
    if ( list1[i] < list2[j] ){ list3.add(list1[i]); ++i; }
    else{ list3.add(list3.add(list2[j]); ++j;}                      } 2*n
while ( i < list1.size()) { list3.add( list1[i++]); end while       } <n
while ( j < list2.size()) { list3.add (list2[j++]); end while       } <n
return list3.toTree(); }n
end
```

Height of each tree are n. So time complexity of Merge is T(n)

$T(n) = 4 + n + n + 2n + n + c$

$= 5n + c = 5n = n \in \underline{O(n)}$

5

```
void function(int n)
{
    int count = 0;               }1
    for(int i = 2; i <= n; i++)
        if (i % 2 == 0)
        {
            count++;
        }
        else
        {
            i = (i-1)*i;
        }
}
```

for i=2 in for loop if case happens. and i=3 becomes. and count=1

for i=3 in for loop else case happens and i= (3-1)*3+1= 7 becomes

for i=7 in for loop else case happens and i= (7-1)*7+1 = 43 becomes

⋮

for i=n in for loop else case happens. Because i=(n-1)*n-1

So $i = n^2 - n - 1$ and this shows us if n is odd or even it will be evaluated always an odd number and always else case will be evaluated

$T(n) = \underbrace{\log n}_{\text{else case}} + \underbrace{1}_{\substack{i=2\\ \text{if case}}} + \underbrace{4}_{\substack{\text{other}\\ \text{initializations}}} \Rightarrow \underline{T(n) = O(\log n)}$

Ömer GEVİK
161044004