Elifnur Kabalcı CSE321 Hw1
1801042617

1)

a) $f(n) = 2^n$, $g(n) = 2^{2n}$

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = \frac{2^n}{2^{2n}} = \frac{2^n}{2^n \cdot 2^n} = \frac{1}{2^n} = 0$$ g(n) grow faster

O positive c for $f(n) \leq c \cdot g(n)$ for all $n \geq n_0$

$\left.\begin{matrix} c=1 \\ n_0=1 \end{matrix}\right\}$ possible ✓ $f \in O(g)$

Ω no positive c for $f(n) \geq c \cdot g(n)$ for all $n \geq n_0$

$f \notin \Omega(g)$

$f \notin \Theta(g)$

b) $f(n) = n^2$, $g(n) = n^3$

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = \frac{n^2}{n^3} = \frac{1}{n} = 0$$ g(n) grow faster

O positive c for $f(n) \leq c \cdot g(n)$ for $n \geq n_0$

$\left.\begin{matrix} c=1 \\ n_0=1 \end{matrix}\right\}$ possible ✓

Ω no positive c for $f(n) \geq c \cdot g(n)$ for $n \geq n_0$

$f \notin \Omega(g)$

$f \notin \Theta(g)$

c) $f(n) = 3n+1$, $g(n) = 2n-1$

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = \frac{3n+1}{2n-1} \Rightarrow \frac{3}{2}$$

→ Result of limit is equal a constant number. So $f \in \Theta(g)$.

O positive c for $f(n) \leq c \cdot g(n)$ for $n \geq n_0$

$\left.\begin{matrix} c=2 \\ n_0=10 \end{matrix}\right\}$ possible ✓ $f \in O(g)$

Ω positive c for $f(n) \geq c \cdot g(n)$ for $n \geq n_0$

$\left.\begin{matrix} c=1 \\ n_0=1 \end{matrix}\right\}$ possible ✓ $f \in \Omega(g)$

d) $f(n) = 4n^2$, $g(n) = n^2$

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = \frac{4n^2}{n^2} = 4$$

-> Result of limit is equal a constant number. So $f \in \Theta(g)$

(O) positive c for $f(n) \leq c \cdot g(n)$ for $n \geq n_0$

$\left.\begin{array}{l} c=4 \\ n_0=1 \end{array}\right\}$ possible ✓ $f \in O(g)$

(Ω) positive c for $f(n) \geq c \cdot g(n)$ for $n \geq n_0$.

$\left.\begin{array}{l} c=1 \\ n_0=1 \end{array}\right\}$ possible ✓ $f \in \Omega(g)$

e) $f(n) = \log_2(n)$, $g(n) = \log_{10}(n)$

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = \frac{\log_2(n)}{\log_{10}(n)} = \log_{10}^{2}$$ (It's approximately 0,3)

-> Result of limit is equal a constant number. So $f \in \Theta(g)$.

(O) positive c for $f(n) \leq c \cdot g(n)$ for $n \geq n_0$.

$\left.\begin{array}{l} n=1 \\ c=1 \end{array}\right\}$ possible (This situation, $f(n) = g(n)$) $f \in O(g)$

(Ω) positive c for $f(n) \geq c \cdot g(n)$ for $n \geq n_0$.

$\left.\begin{array}{l} n=1 \\ c=1 \end{array}\right\}$ possible (This situation $f(n) = g(n)$) $f \in \Omega(g)$

f) $f(n) = 2^n$, $g(n) = 3^n$

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = \frac{2^n}{3^n} = \left(\frac{2}{3}\right)^n => 0$$

(O) positive c for $f(n) \leq c \cdot g(n)$ for $n \geq n_0$.

$\left.\begin{array}{l} n=1 \\ c=1 \end{array}\right\}$ possible (for every positive n and c number) $f \in O(g)$

(Ω) no positive c for $f(n) \geq c \cdot g(n)$ for $n \geq n_0$.

$f \notin \Omega(g)$

$f \notin \Theta(g)$

g) $f(n) = n^3$, $g(n) = 1000n^2$

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = \frac{n^3}{1000n^2} = \frac{n}{1000} \Rightarrow \infty$$

O positive c for $f(n) \leq c \cdot g(n)$ for $n \geq n_0$

$\left.\begin{matrix} c = 1 \\ n_0 = 1000 \end{matrix}\right\}$ possible ✓ (This situation $f(n) = g(n)$) $f \in O(g)$

Ω positive c for $f(n) \geq c \cdot g(n)$ for $n \geq n_0$.

$\left.\begin{matrix} c = 1 \\ n_0 = 1000 \end{matrix}\right\}$ possible ✓ (This situation $f(n) = g(n)$) $f \in \Omega(g)$

$f \in \Theta(g)$

→ Generally, $n > 1000$ $f > g$; $n < 1000$ $f < g$ } but, we take c number. So, It can be changable.

h) $f(n) = 5n + 4$, $g(n) = 2n + 2$

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = \frac{5n+4}{2n+2} \Rightarrow \frac{5}{2}$$

→ Result of limit is equal a constant number. So $f \in \Theta(g)$.

O positive c for $f(n) \leq c \cdot g(n)$ for $n \geq n_0$.

$\left.\begin{matrix} c = 3 \\ n = 1 \end{matrix}\right\}$ possible ✓ $f \in O(g)$

Ω positive c for $f(n) \geq c \cdot g(n)$ for $n \geq n_0$.

$\left.\begin{matrix} n = 1 \\ c = 1 \end{matrix}\right\}$ possible $f \in \Omega(g)$

i) $f(n) = \sqrt{n}$, $g(n) = \log_2(n)$

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = \frac{\sqrt{n}}{\log_2 n}$$ ⇒ $\log_2^n$ grow faster ⇒ 0

O positive c for $f(n) \leq c \cdot g(n)$ for $n \geq n_0$.

$\left.\begin{matrix} c = 1 \\ n = 16 \end{matrix}\right\}$ possible (This situation $f(n) = g(n)$) $f \in O(g)$

Ω positive c for $f(n) \geq c \cdot g(n)$ for $n \geq n_0$.

$\left.\begin{matrix} c = 1 \\ n = 16 \end{matrix}\right\}$ possible (This situation $f(n) = g(n)$) $f \in \Omega(g)$

$f \in \Theta(g)$

Ω: $f(n) = 2$, $g(n) = 2$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \frac{2^n}{2^{n+1}} = \frac{1}{2}$$

-> Result of limit is equal a constant number. So $f \in \Theta(g)$.

(O) positive c for $f(n) \leq c \cdot g(n)$ for $n \geq n_0$.

$\left.\begin{matrix} c=1 \\ n_0=1 \end{matrix}\right\}$ possible $f \in O(g)$

(Ω) no positive c for $f(n) \geq c \cdot g(n)$ for $n \geq n_0$.

$f \notin \Omega(g)$

⚠ Result of limit is constant : Asymtotic grows. Two function's grow are similar

2)

a) $f(n) = 1/2n$, $g(n) = \log(n)$ * Compare all functions.

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \frac{1}{2n \cdot \log(n)} \Rightarrow 0$$ g(n) grow faster

b) $f(n) = \log(n)$, $g(n) = \sqrt{n+5}$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \frac{\log(n)}{\sqrt{n+5}} = \frac{1/n}{1/2\sqrt{n+5}} = \frac{2\sqrt{n+5}}{n} \Rightarrow 0$$ g(n) grow faster

c) $f(n) = \sqrt{n+5}$, $g(n) = n+1$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \frac{\sqrt{n+5}}{n+1} \Rightarrow 0$$ g(n) grow faster

d) $f(n) = n+1$, $g(n) = 10^n$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \frac{n+1}{10^n} \Rightarrow 0$$ (Exponential grow fast) g(n) grow faster

e) $f(n) = 10^n$, $g(n) = n^2 \cdot \log(n)$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \frac{10^n}{n^2 \cdot \log n} \Rightarrow 0$$ (Exponential grow fast) f(n) grow faster

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

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \frac{n^2 \cdot \log(n)}{2^n} \Rightarrow 0$$ (Exponential grow fast) g(n) grow faster

g) $f(n) = 2^n$, $g(n) = n!$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \frac{2^n}{n!} = \frac{2^n}{\underbrace{\sqrt{2\pi n} \cdot (n/e)^n}_{\text{Stirling formula}}} = \frac{(2e)^n}{n^n \cdot \sqrt{2\pi n}} \Rightarrow 0$$ (n grow faster than 2e)

g(n) grow faster

h) $f(n) = 10^n$, $g(n) = n!$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \frac{10^n}{n!} = \frac{10^n}{\sqrt{2\pi n}\cdot(n/e)^n} = \frac{(10e)^n}{n^n \cdot \sqrt{2\pi n}} \Rightarrow 0$$ (n grow faster than 10e)

g(n) grow faster

i) $f(n) = n!$, $g(n) = n^{2n}$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \frac{n!}{n^{2n}} = \frac{\sqrt{2\pi n}\cdot(n/e)^n}{n^{2n}} = \frac{\sqrt{2\pi n}}{(n\cdot e)^n} \Rightarrow 0$$ g(n) grow faster

⚠ Slower to faster grow list

$\frac{1}{2n}$, $\log(n)$, $\sqrt{n+5}$, $n+1$, $n^2 \cdot \log(n)$, 2^n, 10^n, $n!$, n^{2n}

4)

1. i=2
2. while i<=n:
3. if i%2 != 0:
4. i = i-1;
5. else
6. i = i*i
7. i = i+1
8. print(i)

⇒ We think that in every term $i \leq n$, for worst case.

1. term

1	i=2
2	true
5	true
6	i=4
7	i=5
8	print ->5

2. term

2	true
3	true
4	i=4
8	print ->4

3. term

2	true
5	true
6	i=16
7	i=17
8	print ->17

4. term

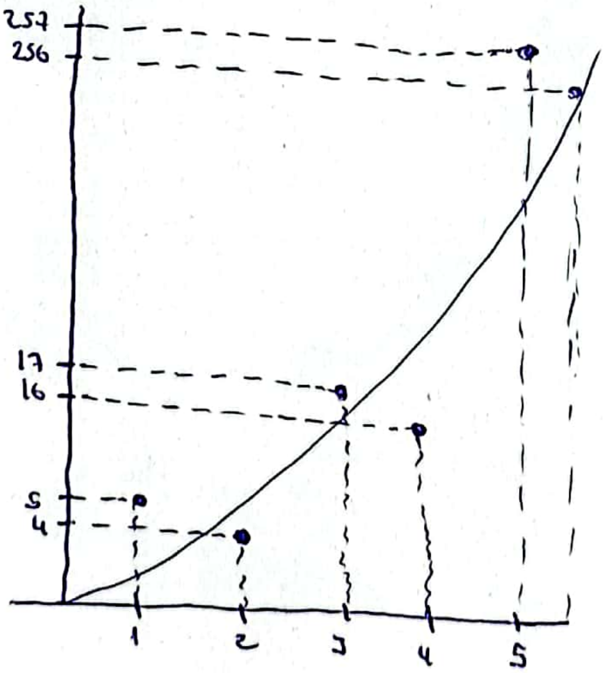
2	true
3	true
4	i=16
8	print ->16

5. term

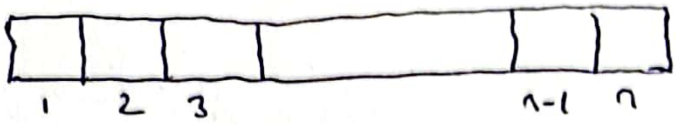
2	true
5	true
6	i=256
7	i=257
8	print ->257

⚠ If we see in general graph is grow exponential.

so big O-notation is $\in O(n^2)$



S)



best case: first element is even. $O(1)$

worst case: last element / does not exist $\Big\}$ $O(n)$

$P(1) = \underbrace{0{,}2}_{\text{First}}$

$P(2) = \underbrace{0{,}8}_{\text{First}} \cdot \underbrace{0{,}2}_{\text{second}} = 0{,}16$

$P(3) = \underbrace{0{,}8}_{\text{first}} \cdot \underbrace{0{,}8}_{\text{second}} \cdot \underbrace{0{,}2}_{\text{third}} = 0{,}128$

$\vdots$

$P(n) = (0{,}8)^{n-1} \cdot 0{,}2$

Linear Search Algorithm

```
// n is size of L
// i is counter
// L is list
// temp is template int (or)

i=0;
while (temp%2!=0 && i<n){
    temp = L[i];
    i++;
}
if(temp%2==0){
    return temp;
}
else{                 //does not exist
    return -1;
}
```

$\sum_{i=1}^{n} n \cdot P(i)$ (all previous elements are odd and current element is even)

$\Rightarrow 0{,}2 + 2 \cdot 0{,}2 \cdot 0{,}8 + \text{-----} + n \cdot (0{,}8)^{n-1} \cdot 0{,}2$

$= 0{,}2 \cdot \underbrace{\left(1 + 2 \cdot 0{,}8 + 3 \cdot (0{,}8)^2 + \text{----} + n \cdot (0{,}8)^{n-1}\right)}_{\left(1/(1-0{,}8)\right)}$

* Sum of geometric series $(a/1-r)$

$= (0{,}2) \cdot \frac{1}{(0{,}2)} = 1$

$\Rightarrow$ This means that it's complexity is constant. $\in \Theta(1)$.

3)

insert_node Method : This method inserts a node into a tree by recursively.
(a)
So, worst case time complexity for every insertion is O(log n).

merge_bst Method : This method merge two tree. In while part worst case complexity is O(n). But in while loop, we call insert_node method. So method's worst case is O(n. log n).
(a)

inorder_traversal Method : This method adds node's values into an array. by recursively. Method's worst case time complexity is O(n).
(helper)

kth_smallest Method : k is a constant value. This method finds the kth smallest value. The worst case scenerio is; first smallest is end of the tree, second is before that_etc. We need to check all list. So, worst case time complexity is O(n²).
(b)

Construct_balanced_bst Method : This method return the root that balanced right to left. Method takes values from an array so it needs to divide two. (for left-right). So worst case time complexity is O(log n).
(c)

balanced_bst Method : This method turns into root to array with using inorder traversal. Call the construct_balanced_bst. So it become O(n) + O(log n), the worst case is O(n).
(c)

inorder_traversal_with_range_check Method : This method find_elements_in_range Method's recursive method. There is more than one condition for calling byself but. in worst case scenerio it goes until big-O notation O(n).
(d)

a) O(n.log n)

b) O(n²)

c) O(n)

d) O(n)