# CSE 321
# HW-1

1) a) $f(n)=2^n \quad g(n)=2^{2n}$

$\lim_{n\to\infty} \frac{f}{g} = \lim_{n\to\infty} \frac{2^n}{2^{2n}} = \lim_{n\to\infty} \frac{1}{2^n} = 0, \; f \in O(g)$

b) $f(n)=n^2 \quad g(n)=n^3$

$\lim_{n\to\infty} \frac{f}{g} = \lim_{n\to\infty} \frac{n^2}{n^3} = \lim_{n\to\infty} \frac{1}{n} = 0 \quad f \in O(g)$

c) $f(n)=3n+1 \quad g(n)=2n-5$

$\lim_{n\to\infty} \frac{f}{g} = \lim_{n\to\infty} \frac{3n+1}{2n-5} = \frac{3}{2}, \; f \in \Theta(g)$

d) $f(n)=4n^2 \quad g(n)=n^2$

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

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

Different bases doesn't change growth rate in logarithm, $f \in \Theta(g)$

f) $f(n)=2^n \quad g(n)=3^n$

$2^n$ growths slower than $3^n$, $f \in O(g)$

g) $f(n)=n^3 \quad g(n)=1000n^2$

$\lim_{n\to\infty} \frac{f}{g} = \lim_{n\to\infty} \frac{n^3}{1000n^2} = \lim_{n\to\infty} \frac{n}{1000} = \infty, \; f \in \Omega(g)$

h) $f(n)=5n+4 \quad g(n)=2n+2$ $\qquad \log_2 x : \frac{1}{x \ln 2}$

$\lim_{n\to\infty} \frac{f}{g} = \lim_{n\to\infty} \frac{5n+4}{2n+2} = \frac{5}{2}, \; f \in \Theta(g)$

i) $f(n)=\sqrt{n} \quad g(n)=\log_2 n$ 1. L'Hospital 2. L'Hospital

$\lim_{n\to\infty} \frac{f}{g} = \lim_{n\to\infty} \frac{\sqrt{n}}{\log_2 n} = \lim_{n\to\infty} \frac{\frac{1}{2\sqrt{n}}}{\frac{1}{n}\cdot \ln 2} = \lim_{n\to\infty} \frac{\cancel{n}\sqrt{n}}{2\cdot\cancel{\sqrt{n}}\cdot \ln 2} = \infty, \; f \in \Omega(g)$

j) $f(n)=2^n \quad g(n)=2^{n+1}$

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

2) I will make limit comparison test

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

1) $\frac{1}{2n}$ and $\log n$

$$\lim_{n\to\infty} \frac{\frac{1}{2n}}{\log n} = \lim_{n\to\infty} \frac{1}{2n.\log n} = 0$$

$\frac{1}{2n}$ grows slower than $\log n$

2) $\log n$ and $\sqrt{n+5}$

$$\underbrace{\lim_{n\to\infty} \frac{\log n}{\sqrt{n+5}} = \lim_{n\to\infty}}_{1.\ L\grave{}Hospital} \frac{\frac{1}{n \ln 10}}{\frac{1}{2\sqrt{n+5}}} = \underbrace{\lim_{n\to\infty} \frac{\cancel{\ln 10}.2\sqrt{n+5}}{n} = \lim_{n\to\infty}}_{2.\ L\grave{}Hospital} \frac{\frac{1}{2\sqrt{n+5}}}{1} = 0$$

$\log n$ grows slower than $\sqrt{n+5}$

3) $\sqrt{n+5}$ and $n+1$

$$\underbrace{\lim_{n\to\infty} \frac{\sqrt{n+5}}{n+1} = \lim_{n\to\infty}}_{1.\ L\grave{}Hospital} \frac{\frac{1}{2\sqrt{n+5}}}{1} = 0$$

$\sqrt{n+5}$ grows slower than $n+1$

4) $n+1$ and $10^n$

$$\underbrace{\lim_{n\to\infty} \frac{n+1}{10^n} = \lim_{n\to\infty}}_{1.\ L\grave{}Hospital} \frac{1}{1.10^n.\ln 10} = 0$$

$n+1$ grows slower than $10^n$

$\frac{n+1}{n^2\log n}$ $\log n$ $\frac{1}{n.\ln 10}$

5) $10^n$ and $n^2 \log n$

$$\lim_{n\to\infty} \frac{10^n}{n^2\log n} = \infty$$

$n^2\log n$ grows slower than $10^n$

6) $n^2\log n$ and $2^n$

$$\lim_{n\to\infty} \frac{n^2\log n}{2^n} = 0$$

$n^2\log n$ grows slower than $2^n$

7) $2^n$ and $n!$

$$\lim_{n\to\infty} \frac{2^n}{n!} = 0$$

$2^n$ grows slower than $n!$

8) $n!$ and $n^{2^n}$

$$\lim_{n\to\infty} \frac{n!}{n^{2^n}} = 0$$

$n!$ grows slower then $n^{2^n}$

Slowest to fastest

$\frac{1}{2n} - \log n - \sqrt{n+5} - n+1 - n^2\log n - 2^n - 10^n - n! - n^{2^n}$

3) Code files uploaded as seperatly on Teams.

a) Merge BTS

- "inorderTraversal" function has $O(n)$ time complexity
- "mergeBSTHelper" function divides list into two halfs in all recursive calls. The number of recursive calls is "$\log n$". Thats why this helper function has $O(\log n)$ time complexity
- So this function has $O(n)$ time complexity.

b) Finding kth smallest element

- It starts from root and traverses to the leftmost node by pushing all encountede nodes onto the stack. This part has $O(\log n)$ time complexity where $n$ is $h$ and height of tree.

- So the time complexity is $O(\log n)$

c) Balancing BST

- "inorder Traversal" has $O(n)$ time complexity.
- "listToBST" recursively divides the list into two half and selects midde as root. The time complexity of this function is $O(n)$.
- So the time complexity of this function is $O(n)$.

d) Finding in range

- It traverses the tree by moving to the left until it reaches the leftmost node. This part has a $O(h)$ time complexity.
- The function pops the element from stack one by one and checks it. This part takes $O(1)$ time.
- And the function goes right child of the popped node also have a $O(1)$ time complexity.
- So as h is n the time complexity is $O(n)$.

4)
```
i=2
while i<=n:
  if(i%2!=0):
    i=i-1
  else:
    i=i*i
    i=i+1
Print(i)
```

| | | |
|---|---|---|
| First iteration | i=2 ⟶ | i=5 |
| Second " | i=5 ⟶ | i=4 |
| third " | i=4 ⟶ | i=17 |
| fourth " | i=17 ⟶ | i=16 |
| fifth " | i=16 ⟶ | i=257 |

- If i is even i is squared and incremented by 1.
- If i is odd i is decremented by 1.

• It is not possible for this function loops exactly n-2 times, as it will enter the "else" block every two rounds. Instead it will increase faster and return by logn.
• So the time complexity is $O(\log n)$

5) function findFirstEven(arr):

```
for each element in array, do:
    if current is even
        return current
    end if
end for
return null
end function
```

- This code iterates through an array of elements until it finds the first even element. If it doesn't find an even element it returns "null" element.
- The probability distribution is %20 even and %80 odd. On average an even element can be found after checking approximatly 5 elements.
- In the worst case scenario, where there are no even elements in the list, the algorithm would need to check all elements in the list. Time complexity is O(n).
- In the best scenario, where the first element is even. Time complexity is O(1).
- For an average case I will use expected value formula.
  - First element is even (Prob %020), X = 1
  - Second " " even (Prob. %80 * %20), X = 2
  - Third " " " (Prob. %80 * %80 * %20), X = 3

  ⋮

$E(X) = 1 \cdot P(1) + 2 \cdot P(2) + 3 \cdot P(3) \ldots$

$E(X) = 1.(0.20) + 2.(0.16) + 3.(0.128) + \ldots$

- E(X) is reaches to 1. This means, on average, we would check only 1 element before finding the first even element. So the average time complexity is O(1).