Name :- Aashi Ashokbhai Goyani
ID :- 1002205247
HOMEWORK - 3

Q 3-1 Problem 3-1

$$P(n) \sum_{i=0}^{d} a_i n^i$$

a) If $K \geq d$, then $f(n) = O(n^k)$

Ans We must show there exist positive constants c & n.

$$0 \leq f(n) \leq cn^k \quad \forall \quad n \geq n_0$$
$$0 \leq \sum_{i=0}^{d} a_i n_i \leq cn^k \quad \forall \quad n \geq n_0$$

$$0 \leq a_0 + a_1 n^1 + \ldots\ldots + a_d n^d \leq cn^K \quad \forall \; n \geq n_0$$
$$0 \leq a_0 + a_1 n^1 + \ldots\ldots + a_d n^d \leq \sum_{i=0}^{d} n^k \quad \forall \; n \geq 1$$

$\therefore$ by setting $c = \sum_{i=0}^{d} a_i$ and $n_0 = 1$; we

can conclude $K \geq d \longrightarrow f(n) = o(n^K)$

b) If $k \leq d$, then $p(n) = \Omega(n^k)$

Ans Positive constants c, n_0

$$0 \leq cn^k \leq p(n) \quad \forall\ n \geq n_0$$

$$0 \leq cn^k \leq \sum_{i=0}^{d} a_i n^i \quad \forall\ n \geq n_0$$

$$0 \leq cn^k \leq a_0 + a_1 n^1 + \ldots + a_d n^d \quad \forall\ n \geq n_0$$

$$0 \leq a_k n^k \leq a_0 + a_1 n^1 + \ldots + a_d n^d \quad \forall\ n \geq n_0$$

Setting $c = a_k$ satisfies inequality for all $n_0 \geq 1$

$\therefore k \leq d$

$$p(n) = \Omega(n^k)$$

c) if $K = d$, then $P(n) = \Theta(n^K)$

Ans In the answers (a) and (b) it is proved $p(n) = \Omega(n^K)$ and $p(n) = O(n^k)$ which is sufficient to show that $p(n) = \Theta(n^k)$

d) if $k > d$, then $p(n) = O(n^k)$

Ans. This can be proved by removing equality from @.

Another way is to "set limits".

$$\lim_{n\to\infty} \frac{p(n)}{n^k} \Rightarrow \lim \sum_{i=0}^{d} \frac{a_i n_i}{n^k} \Rightarrow \lim_{n\to\infty} \frac{n^d}{n^k} = 0$$

e) if $k < d$, then $p(n) = \omega(n^k)$

Ans This can be proved by removing equality from ⓑ.

Another way is to Set limits

$$\lim_{n\to\infty} \frac{p(n)}{n^k} \Rightarrow \sum_{i=0}^{d} \frac{a_i n_i}{n^k} \Rightarrow \lim_{n\to\infty} \frac{n^d}{n^k} = \infty$$

Q 3.1-4 Is $2^{n+1} = O(2^n)$? Is $2^{2n} = O(2^n)$?

Is $2^{n+1} = O(2^n)$?

Ans Yes, $2^{n+1} = 2 \cdot 2^n \leq C\, 2^n$ where $C \geq 2$

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

Ans No, $2^{2n} = 2^n \cdot 2^n$

Suppose $2^{2n} = O(2^n)$. then there is a constant $c > 0$ such that $c > 2^n$

Since 2^n is unbounded, no such C can exist.

Name :- Aashi Ashokbhai Goyani
ID :- 100220547
HOMEWORK - 3

Q1 Insertion Sort

Find runtime for A) Best Case
B) Worst Case

Use Θ-notation

Ans :- a) In the best-case scenario, the array is already sorted, so there's no need to move any element around. The algorithm will perform a single comparison for each element in the array. Therefore, the runtime is linear, or $O(n)$, where n is the length of the array.

∴ Insertion sort runtime for best case is $\Theta(n)$

$$T(n) = \sum_{j=2}^{n} 1 = n$$

b) In the worst case scenario, the array is sorted in reverse order. Therefore, each element in array has to be compared to all elements before it, in order to find its correct position, leading to (n-1) Comparisons for the second element and so on. This results in a total of $n(n-1)/2$ comparisons and a quadratic runtime ($O(n^2)$), where n is the length of array.

$$O(n^2) = T(n) = \sum_{j=2}^{n} j = \frac{(n(n+1))}{2} = \frac{n^2}{2} + \frac{n}{2}$$

Q2. For bubble sort what is

a) worst case time complexity and why
b) best case time complexity and why

Ans. a) In worst case scenario for Bubble Sort, the array is sorted in reverse order. So, the algorithm needs to perform swap for almost every pair of adjacent element during each pass through the array.

In that case, algorithm will have runtime at $O(n^2)$ where n is number of element in input.

It makes (n-1) comparisons for the first pass, (n-2) comparisons for the second pass and so on.

Resulting total of $n(n-1)/2$ comparisons.

$$\therefore O(n^2) = \frac{n^2}{2} = \frac{1}{2}(n)$$

b) Best case time complexity:

In the best-case scenario, the array is already sorted. So, no swaps are needed ~~be~~ during each pass through the array because all elements are in correct order. This means that the algorithm only has to go through the array once to make sure that it is sorted.

In this case, Algorithm have $O(n)$ runtime, n = no. of input elements.

Q3. For selection sort what is
a) Worst case time complexity and why?
b) Best case time complexity and why?

Ans a) Worst case time complexity.

The worst case for selection sort is when the ~~field~~ array is sorted in reverse order. In this case, selection sort needs to traverse the entire unsorted part to find the minimum or maximum element. Since, there are quadratic numbers of comparison and swaps, the time complexity is $O(n^2)$, where n is the length of array.

b) Best case time complexity

The best case for selection sort is when the array is already sorted.

Though the number of swaps reduces, the algorithm still has to make same number of comparisons for each element in the unsorted part.

Therefore, the time complexity is $O(n^2)$

where n is length of array.

Q4 Find the runtime complexity using Θ - notation of :

a) $T(n) = \log n + n + 1$

Ans = $\Theta(n)$

b) $T(n) = \log n + \lg n + 10^{1000}$

Ans = $\Theta(\log n)$

c) $T(n) = n \log n + \lg n + n + 3$

Ans = $\Theta(n \log n)$

d) $T(n) = 2^n + n!$

Ans = $\Theta(n!)$

Q5. Prove the order of growth.

a) $\frac{1}{2}n(n-1) = \Theta(n^2)$

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

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

$= \frac{1}{2}$ (Constant)

b) $\lg n \in O(\sqrt{n})$

↑

little O

Ans

$\lim_{n\to\infty} \left[\frac{\log n}{\sqrt{n}}\right] = \left[\frac{(\log e)\cdot \frac{1}{n}}{\frac{1}{2\sqrt{n}}}\right]$

$= 2\log e \lim_{n\to\infty} \frac{1}{\sqrt{n}}$

$= 0$ constant

c) $\lg n = O(\sqrt{n})$

↑ big-O

Ans

$$\lim_{n\to\infty}\left[\frac{\log n}{\sqrt{n}}\right] = \left[\frac{(\log e)\cdot \frac{1}{n}}{\frac{1}{2}\sqrt{n}}\right] = 2\log e \cdot \lim_{n\to\infty}\frac{1}{\sqrt{n}} = 0$$

Ans = O Constant

d) $n! = \Omega(2^n)$

Ans) $f(n) = n!$

$g(n) = 2^n$

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

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

$\therefore n! = \Omega(2^n)$

e) $2n = O(n^3)$

↓ big O

Ans $= \lim_{n\to\infty}\frac{2^n}{n^3} \Rightarrow 2\lim_{n\to\infty}\frac{1}{n^2} = 0$

f) $4n^3 \in \Omega(n)$

Ans $\lim_{n\to\infty} \frac{4n^3}{n} = 4 \lim_{n\to\infty} n^2 = \infty$

Q6 Find the runtime in (H) notation of

a) $T(n) = \sum_{i=1}^{n} i$

Ans $\Rightarrow \frac{n(n+1)}{2} \Rightarrow \frac{n^2+n}{2} \Rightarrow$ (H) (n^2)

b) $T(n) = \sum_{i=5}^{n-4} i$

Ans $\Rightarrow \sum_{i=1}^{n-4} i - \sum_{i=1}^{4} i$

$\Rightarrow \frac{n-4\ (n-4+1)}{2} - \frac{4(4+1)}{2}$

$\Rightarrow \frac{(n-4)(n-3)}{2} - \frac{20}{2}$

$\Rightarrow \frac{n^2-7n+12-20}{2} \Rightarrow \frac{n^2-7n-8}{2}$

$\Rightarrow O(n^2)$

c) $T(n) = \sum_{i=1}^{n} (i + 10^{100})$

Ans $\Rightarrow \sum_{i=1}^{n} i + \sum_{i=1}^{n} 10^{100}$

$\Rightarrow \frac{n(n+1)}{2} + 10^{100}(n-1+1)$

$\Rightarrow \frac{n(n+1)}{2} + 10^{100} n$

$\Rightarrow \Theta(n^2)$

d) $T(n) = \sum_{i=0}^{n} i^2$

Ans $\Rightarrow \frac{n(n+1)(2n+1)}{6}$

$\Rightarrow \Theta(n^3)$

e) $T(n) = \sum_{i=1}^{n} (i^2 + i + 2)$

Ans $\Rightarrow \sum_{i=1}^{n} i^2 + \sum_{i=1}^{n} i + \sum_{i=1}^{n} 2$

$\Rightarrow \frac{(n)(n+1)(2n+1)}{6} + \frac{n(n+1)}{2} + 2n$

$\Rightarrow \Theta(n^3)$

f) $T(n) = \sum_{i=2}^{n-4} \Theta(i^2)$

Ans $\Rightarrow \Theta(n^{2+1})$

$\Rightarrow \Theta(n^3)$

g) $T(n) = \sum_{i=1}^{n} i^3$

Ans $\Rightarrow \frac{n^2(n+1)^2}{n^4}$

$\Rightarrow \Theta(n^4)$

h) $T(n) = \sum_{i=1}^{n} 1$

Ans $\Rightarrow (n-1+1)$

$\Rightarrow n$

$\Rightarrow \Theta(n)$

i) $T(n) = \sum_{i=a}^{n} 1$

↑ constant

Ans $\Rightarrow 1\,(n-a+1)$

$\Rightarrow O(n)$

$\Rightarrow \Theta(n)$