HW - 3

Sushrutha Reddy Mogili
1002164706.

1. Pseudocode for insertion sort is given in the book on Pg 18. Find the runtime for
   a) Best case
   b) worst case, your answer should use Θ notation.

Sol:-

a) <u>Runtime for Best case for insertion sort</u>-
In the best case situation for insertion sort, the inner loop will only checks the other value or element in the array until it finds the small element in the array then reaches the beginning of the array, where the array was actually sorted. And hence there is no need to exchange the element inside the inner loop because each and every element is already sorted.
Hence the algorithm needs only one comparision, since the outer loop iterates each element only once. Hence the total number of iteration for outer loop is linear hence it is expressed as $O(n)$, where n is the array's length.

b) <u>Runtime for worst case for insertion sort</u>-
The worst case complexity for insertion sort forms if the input array is arranged in reverse order. Then the each element for unsorted part of array need to be compared. Then move the element to the sorted part until it finds the correct position to place that element.

$$T(n) = 1+2+3+\cdots+(n-1)$$
$$= \frac{n(n-1)}{2} = \frac{n^2-n}{2}$$ [where sum of $1+2+\ldots+n-1$ can be written using the formula for sum of first $n-1$ numbers]

$\therefore T(n) = O(n^2)$ = worst complexity.
where the number of comparison increases exponentially.

2. For Bubble sort what is
a) worst case time complexity and why
b) best case time complexity and why?
Hint use HW from ch 1 and 2.

Sol:-
a) worst case time complexity for the Bubble sort-
In the Bubble sort algorithm the given array is arranged in reverse order for that the Bubble sort algorithm need to check how many maximum number of comparisons are needed and then it re-arranges the values in the array accordingly. Hence the largest element of the array will move to the bottom of the array, hence the ~~largest~~ loop runs $n-1$ times.
$\therefore$ Hence the worst case time complexity for bubble sort algorithm is $O(n^2)$.

b) Best case time complexity for the Bubble sort-
In this algorithm, when the array is already sorted, then this algorithm pass each and every element without requiring any swaps. Then after the first pass then the bubble sort algorithm finds the array is already sorted. Hence the number of comparisons and swaps are less. Hence the algorithm finds the array is already in order.

$\therefore$ Hence the Best case time complexity for bubble sort algorithm is $O(n)$

3. For selection sort what is
a) worst case time Complexity and why?
b) Best case time Complexity and why?
Hint use HW from ch. 1 and 2

Sol:-

a) worst case time complexity for Selection Sort -
The worst case complexity for selection sort occurs when the input array is sorted in reverse order. In the unsorted array for each and every element the algorithm must compare with the other elements in the array. Then the inner loop iteration the algorithm performs $n-1$ comparisons for the first pass, and $n-2$ comparisons for the second pass and so on for $\frac{n(n-1)}{2}$ for final result

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

∴ Hence the worst case time complexity for selection sort is $O(n^2)$.

b) Best case time Complexity for selection sort -
The Best Case time complexity for selection sort occurs when the input array is already sorted in correct position. But still this algorithm will goes the complete array to compare each and every element. And also swapping is not required because the array is already sort. But the no. of comparisons for the Best case is same for the worst case.
∴ Hence the Best case time Complexity for Selection sort is $O(n^2)$.

4. Find the runtime Complexity using Θ-notation of :

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

sol:- The runtime complexity for $T(n) = \log n + n + 1$ is $O(n)$.

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

sol:- The runtime Complexity for $T(n) = \log n + \lg n + 10^{1000}$ is $O(\log n)$

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

sol:- The runtime Complexity for $T(n) = n\log n + \lg n + n + 3$ is $O(n\log n)$.

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

sol:- Hence the runtime Complexity for $T(n) = 2^n + n!$ is $O(n)$.

5. prove the order of growth.

Hint : use limits

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

sol:- Now let $P = \lim_{n\to\alpha} \left(\frac{1}{2} n(n-1)\right) = n^2$

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

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

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

divide n^2 for both numerator and denominator

$$\lim_{n\to\alpha} \frac{1}{2} \times \frac{\frac{n^2-n}{n^2}}{\frac{n^2}{n^2}} \Rightarrow \lim_{n\to\alpha} \frac{1}{2} \times \frac{n-1}{n}$$

$P = \frac{1}{2} - 0 = 1/2$ //

Hence the limit for $\frac{1}{2} n(n-1)$ is n^2

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

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

sol:- now let $P = \lim_{n \to \infty} \log \left(\frac{n}{\sqrt{n}}\right)$

$$P = \lim_{n \to \infty} \log \left(\frac{n}{n^{(1/2)}}\right)$$

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

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

$P = 0$ $\quad \therefore \frac{1}{n}$ is zero has $n \to \infty$.

Hence, $\log n \in o(\sqrt{n})$.

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

sol:- Now let $P = \lim_{n \to \infty} \log \left(\frac{n}{\sqrt{n}}\right)$

$$P = \lim_{n \to \infty} \log \left(\frac{n}{n^{(1/2)}}\right)$$

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

$P = 0$ $\quad \therefore \frac{1}{n}$ is zero has $n \to \infty$.

Hence, $\log n = \Theta(\sqrt{n})$.

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

Sol:- Now let $\underset{n \to \alpha}{lt} \frac{n!}{2}$

Now use l-hospital rule,

$\log\left(\frac{n!}{2^n}\right) = \log(n!) - \log(2^n)$

since $\log \frac{a}{b} = \log a - \log b$

$= \log(n!) - n\log 2$

$= \underset{n \to \alpha}{lt} \left(\frac{n!}{2^n}\right)$ $\left[\text{since } \log a^b = b \log a\right]$

$= 0$

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

Sol:- Now let $\lim_{n \to \alpha} \left(\frac{2n}{n^3} \right)$

$\therefore \lim_{n \to \alpha} \left(\frac{2}{n^2} \right)$

Since $n \to \alpha$, then the denominator n^2 will grows without the bound.

$\therefore \lim_{n \to \alpha} \left(\frac{2}{n^2} \right) = 0.$

Hence $2n = O(n^3)$.

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

Sol:- Now let $\lim_{n \to \alpha} \left(\frac{4n^3}{n} \right)$

$\Rightarrow \lim_{n \to \alpha} (4n^2)$

$\Rightarrow \alpha$

Hence $4n^3 \in \Omega(n)$.

6. Find the runtime in Θ-notation of:
Hint look up the formula's for Σ.

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

sol:- The formula for sum of first n natural numbers is,

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

Hence $T(n) = \Theta(n^2)$

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

sol: The formula for sum of first 4 natural numbers is,

$$T(n) = \frac{n(n+1)}{2} - \frac{4(4+1)}{2} \quad \left[\text{from the formula } \frac{n(n+1)}{2}\right]$$

$$\therefore T(n) = \frac{n^2+n}{2} - 10 = \frac{n^2+n-20}{2}$$

Hence the runtime in Θ-notation for $T(n) = \Theta(n^2)$.

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

sol:- from the formula,

$$T(n) = \frac{n(n+1)}{2} + n \cdot 10^{100}$$

$$= \frac{n^2+n}{2} + n \cdot 10^{100}$$

$$= \frac{n^2+3n+2(10^{100})}{2}$$

Hence runtime is $T(n) = \Theta(n^2)$.

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

Sol:- The formula for the sum of first n squares is,

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

$$T(n) = \frac{(n^2+n)(2n+1)}{6}$$

$$T(n) = \frac{2n^3+n^2+2n^2+n}{6}$$

$$T(n) = \frac{2n^3+3n^2+n}{6}.$$

Hence, $T(n) = O(n^3)$.

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

split it in to 3 parts,

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

use arithematic formula,

$$T(n) = \frac{n(n+1)(2n+1)}{6} + \frac{n(n+1)}{2} + 2n.$$

$$T(n) = \frac{2n^3+3n+n}{6} + \frac{n^2+n}{2} + 2n.$$

Hence $T(n) = O(n^3)$.

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

it will represents the sum of terms,

$\therefore T(n) = O(2^2) + O(3^2) + O(4^2) + \dots + O((n-4)^2)$

$T(n) = O(4) + O(9) + O(16) + \dots + O((n-4)^2)$

Hence, $T(n) = O((n-4)^2)$.

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

Here $T(n) = \frac{n^2 (n+1)^2}{4}$

here the biggest value is n^4

Hence, $T(n) = O(n^4)$.

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

it will represents the constant terms repeated n times.

$= 1(n-1+1)$

$= n$

Hence, $T(n) = O(n) = O(1)$.

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

hence $T(n) = 1(n-a+1)$

hence, $T(n) = O(n-a+1)$.

$= O(1)$