FDS Assignment 1

Q1. Define algorithm and its characteristics.

Ans. An algorithm is a set of steps required to solve a problem.

→ characteristics:-

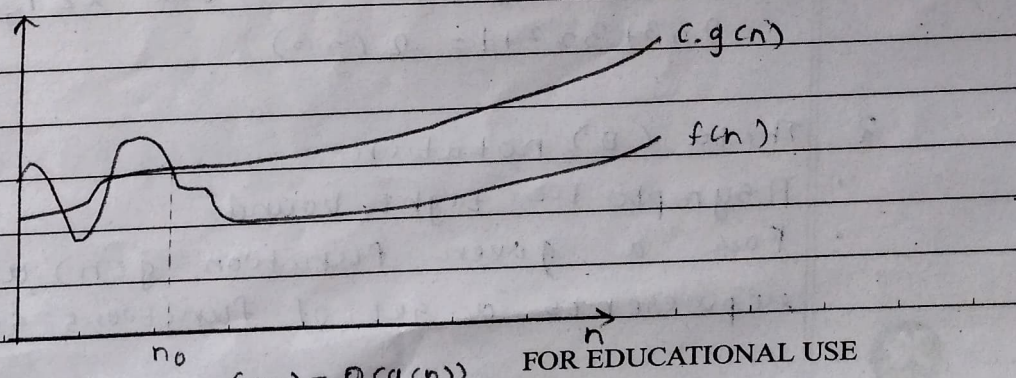
1) Input: Input data, supplied externally (zero or more)
2) Output: Result of the program
3) Finiteness: In every case, algorithm terminates after a finite number of steps.
4) Definiteness: The steps should be clear and unambiguous.
5) Effectiveness: An algorithm should be written using basic instructions. It should be ~~feas at~~ feasible to convert the algorithm in a computer program.

Q2. Explain asymptotic notations Big O, Theta and Omega with one example of each.

Ans 1. ~~Asym~~ O- notation (Big Oh)

- Asymptotic Upper bound of algorithm running time
- For a given function g(n), we denote O(g(n)) as the set of functions:

$O(g(n)) = \{ f(n) \mid$ there exists positive constants c and n_0 such that $0 \leq f(n) \leq c\,g(n)$ for all $n \geq n_0 \}$



eg: if $f(x) = 2x^3 + 3x^2 + 1$

then $f(x) \leq 6x^3$ for all $x \geq 1$

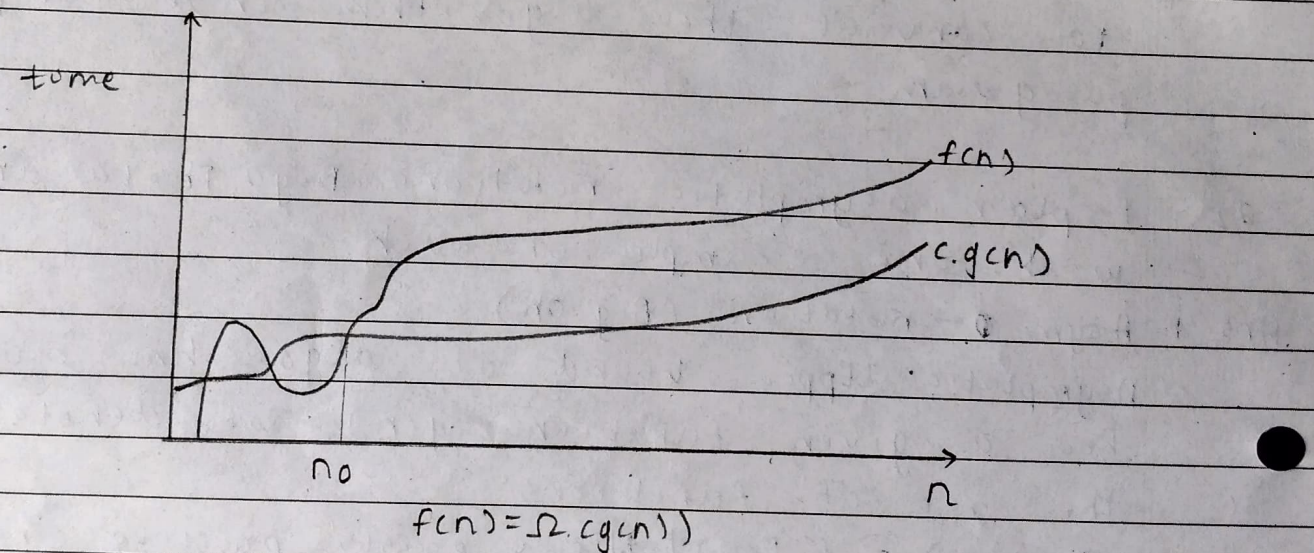
$\therefore 2x^3 + 3x^2 + 1 = O(x^3)$ as

$f(x) \leq C.x^3$ for all $x \geq K$ where $C = 6$ and $K = 1$

2. Omega (Ω) notation

- Asymptotic lower bound of algorithm running time
- For a given function $g(n)$, we denote $\Omega(g(n))$ represents a set of functions such that:

$\Omega(g(n)) = \{ f(n) :$ there exist positive constants c and n_0 such that $0 \leq c\,g(n) \leq f(n)$ for all $n \geq n_0 \}$



$f(n) = \Omega(g(n))$

eg: if $f(x) = 2x^3 + 3x^2 + 1$

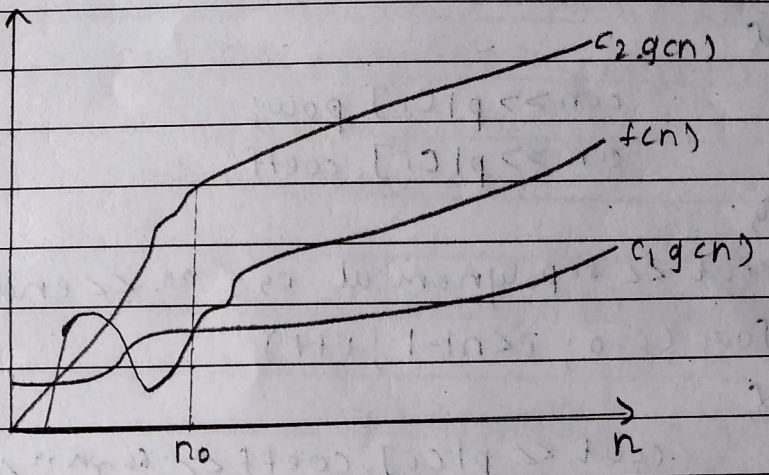
then $f(x) \geq C x^3$ for all $x \geq 1$, $C = 1$

$\therefore 2x^3 + 3x^2 + 1 = \Omega(x^3)$

3. Theta (Θ) notation

- Asymptotic tight bound
- For a given function $g(n)$, we denote $\Theta(g(n))$ represents a set of functions such that:

$\Theta(g(n)) = \{f(n)$: there exist positive constants c_1, c_2 and n_0 such that $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$ for all $n \geq n_0\}$



$f(n) = \Theta(g(n))$

eg: For any two functions $f(x)$ and $g(x)$, $f(x) = \Theta(g(x))$ if and only if $f(x) = O(g(x))$ and $f(x) = \Omega(g(x))$

Q3. Write a c++ code to perform polynomial multiplication using arrays.

Ans.
```
#include<iostream>
using namespace std;
struct poly
{
    int coeff;
    int pow;
};
struct poly p1[10], p2[10], p3[10]
int main()
{
    int n1, n2, i;
```

```
cout<<"Enter no. of terms: ";
cin >>n1;
cout<<"Enter power and coeff of 1st polynomial:"
    <<endl;
for(i=0; i<n1; i++)
{
    cin>>p1[i].pow;
    cin >>p1[i].coeff;
}
cout <<"polynomial is: "<<endl;
for(i=0; i<n1-1; i++)
{
    cout <<p1[i].coeff<<"x^"<<p1[i].pow<<"+";
}
cout <<p1[i].coeff <<"x^"<<p1[i].pow<<endl;
cout<<"Enter power and coeff of 2nd
      polynomial : "<<endl;
for(i=0; i<n2;i++)
{
    cin>>p2[i].pow;
    cin>>p2[i].coeff;
}
cout <<"polynomial is: "<<endl;
for(i=0; i<n2-1; i++)
{
    cout <<p2[i].coeff<<"x^"<<p2[i].pow<<"+";
}
cout <<p2[i].coeff<<"x^"<<p2[i].pow<<endl;
int j,c,p,k=0;
i=0;
```

```
while (i<n1)
{   j=0;
    while (j<n2)
    {
        c = p1[i].coeff * p2 [j]. coeff;
        p = p1[i].pow + p2 [j]. pow;
        int temp=0;
        while (temp <k)
        {
            if (p3 [temp]. pow == p)
            {
                p3 [temp]. pow == p;
                p3 [temp]. coeff = p 3[temp]. coeff+c;
                break;
            }
            else
            { temp++; }
        }
        if (temp ==k)
        {
            p3 [k]. pow =p;
            p3[k]. coeff = c;
            k++;
        }
        j++;
    }
    i++;
}
cout<< "multiplication of polynomials is: "<<endl;
for (i=0; i<k-1; i++)
{
    cout <<p3[i]. coeff<<"x^"<<p 3 [i]. pow<< "+";
}
```

```
cout << p3[i].coeff << "x^" << p3[i].pow << endl;
return 0;
}
```

Q4. Define and explain: Sparse matrix.

Ans. • A sparse matrix is a matrix that has many zero elements.

• For example, the following 4×4 matrix A is a sparse matrix.

$$A = \begin{bmatrix} 0 & 0 & 9 & 0 \\ 0 & 1 & 0 & 0 \\ 5 & 0 & 9 & 0 \\ 8 & 0 & 0 & 6 \end{bmatrix}$$

• Conventional method of representation of such a matrix is not space efficient.

• It will be prudent to store non-zero elements only.

• If this is done, then, the matrix may be thought of as an ordered list of non-zero elements only.

• Information about non-zero elements have 3 parts:

1) Row 2) Column 3) value

No. of columns in original matrix

No. of rows in original matrix

No. of Non-zero elements

4	4	6
0	2	9
1	1	1
2	0	5
2	2	9
3	0	8
3	3	6

Row no. | column no. | value

Q5. Show the various passes of bubble sort on an unsorted list 11, 15, 2, 13, 6

ns.

Pass No.	Data at the end of the pass
1.	~~15, 2, 13, 6, 11~~ 11, 2, 13, 6, 15
2.	2, 11, 6, 13, 15
3.	

Q 5. Show the various passes of bubble sort on an unsorted list 11, 15, 2, 13, 6

Ans. First pass (i=1)

j=0 11 15 2 13 6

j=1 11 15 2 13 6

j=2 11 2 15 13 6

j=3 11 2 13 15 6

Second pass (i=2)

j=0 11 2 13 6 15

j=1 2 11 13 6 15

j=2 2 11 13 6 15

j=3 2 11 6 13 15

Thoard pass (i=3)

j=0 2 11 6 13 15

j=1 2 11 6 13 15

j=2 2 6 11 13 15

j=3 2 6 11 13 15

~~Third~~ Fourth pass (i=4)

j=0 2 6 11 13 15

j=1 2 6 11 13 15

j=2 2 6 11 13 15

j=3 2 6 11 13 15

∴ Sorted array → 2 6 11 13 15