**1.Write short notes on optimization**

Optimization is the process of finding the conditions that give the maximum or minimum value of the function. An act, process, or methodology of making something (such as design, system or decision) as fully perfect, functional or effective or possible. One common application of calculus is calculating the minimum or maximum value of a function. For example, companies often want to minimize production costs or maximum revenue. In manufacturing, it is often desirable to minimize the amount of material used to package a product with a certain volume.

In an optimization problem, the types of mathematical relationships between the objective and constrains and the decision variables determine how hard it is to solve, the solution methods or algorithms that can be used for optimization, and the confidence you can have that the solution is truly optimal. In mathematics, computer science and economics, an optimization problem is the problem of finding the best solution from all feasible solutions. An optimization problem with discrete variables is known as a discrete optimization.

**2.What are the different algorithms you know**

I know some algorithms which is the most fundamental types of algorithms.

#Recursive Algorithm, #Dynnamic Programing Algorithm, #Backtracking Algorithm, #Greedy Algorithm, #Brute Force Algorithm.

**Recursive Algorithm:** A Recursive algorithm is an algorithm which calls itself with smaller input values and which obtain the result for the current input by applying simple operations to the returned value for the smaller input. Generation of Fectorial, Fibonacci number series are the examples of Recursive algorithm.

**Dynamic programing algorithm:** In terms of mathematical optimization, Dynamic programing usually refers to simplifing a decision by breaking it down into a sequence of decision steps over time.Knapsack is the example of Dynamic programing algorithm.

**Greedy Algorithm:** Mostly fail to find the globally optimal solution because they usually do not operate exhaustively on all the data. Huffman coding, Kruskal’s algorithm, Prime’s algorithm are the examples of greedy algorithm.

**3.Why are you learning so many algorithms**

We are learning so many algorithms to use those basic methods to give more efficient solution for codes or complex problems we do everyday.

Algorithms are the solution steps for a particular problem. Writing algorithm helps me speed up my thinking ability for other programing problems. Also I can dry run the program through algorithms to be sure if would it work or not.

**4. Show analysis of a recursive algorithm.**

Factorial(n)

{

If(n=0)

Return 1;

Else

n\*factorial(n-1)

}

**Time complixity analysis:**

Let’s assume ==,\*,-, operatos’s cost constant 1 time.

T(n)= T(n-1)+1+1+1

=T(n-1)+3

=T(n-2)+6

=T(n-3)+8

=T(n-k)+3k

T(0)=1

Interms of T(0)

n-k=0

n=k

T(n)=T(0)+3n

T(n)=3n+1

The Time complexity in the worst case: O(n)

**5.Design an iterative and recursive algorithm and prove that your algorithm works.**

**Recursive Fibonacci vs Iterative method**

Int fib int(n)

{

If(n<=1)

Return n;

Return fibo (n-1) + fibo(n-2);

}

**Iterative method:**

Int fib(int n)

{

Int f(n+2);

Int i;

f[0]=0;

f[1]=1;

for(i=2; i<=n; i++)

{

F[i]= f[i-1] + f[i-2];

}

Return f[n];

}

};

**Time Complexity analysis:**

T(n)= T(n-1)+ T(n-2)+ 1

We can assume that T(n-2)= T(n-1).

Substituting the value of T(n-1) = T(n-2) into our relation T(n), we get:

T(n)= T(n-1)+ T(n-1)+1= 2\*T(n-1)+1

T(n)= 2\*[2\*T(n-2)+1]+1= 4\*T(n-2)+3

Next, we can substitute in T(n-2)= 2\*T(n-3)+1:

T(n)= 2\*[2\*[2\*T(n-3)+1]+1]+1]=8\*T(n-3)+7

And again for T(n-3)= 2\*T(n-4)+ 1:

T(n)= 2\*[2\*[2\*[2\*T(n-4)+1]+1]+1]+1= 16\*T(n-4)+ 15

We can see a patten starting to emerge here, so let’s attempt to form a general solution forT(n). It appers to stand that:

T(n)= 2^k\*T(n-k)+ (2^k-10

In this function 2^k is the term that has the highest rate of change for differentt values. That’s why we can assume that

Time complexity is (2^n)

While the iterative method is O(n).

**Name: Bidyut Sharma**

**Id: 191-15-12720**

**Section: O-14**