ASSIGNMEMT

COURSE CODE: CSE 214

COURSE NAME: Algorithm

Submitted To

Subroto Nag Pinku

Department of CSE

Daffodil International University

Submitted By

Md. Foysal Ahmed

ID: 191-15-12486

Section: O-14

1. Write short notes on Optimization

Ans:

Optimization means finding an alternative way to achieve highest performance within very low cost by maximizing desired factors and minimizing undesired ones. In comparison, maximization means trying to maintain the highest or maximum result or outcome without regard to cost or expense.

Optimization in algorithms refers to a procedure which is executed iteratively by comparing various solutions till an optimum or a satisfactory solution is found. Sometimes a naïve solution can be un-optimized or costly to solve. In that scenario optimization algorithms comes in handy which gives an optimal solution of the problem within low cost. Though optimization is used for maintain highest result but in real world optimization, there could be more than one objective that the designer may want to optimize simultaneously. The multiple objective optimization algorithms are complex and computationally expensive. Therefore, the most important objective is chosen as the objective function and the other objectives are included as constraints by restricting their values within a certain range.

2. What are the different algorithms you know?

Ans:

An algorithm is a procedure or formula for solving a particular problem, based on conducting a sequence of specified actions. A computer program can be viewed as an elaborate algorithm. In mathematics and computer science, an algorithm usually means a small procedure that solves a recurrent problem.

There are various algorithms exists for various problems. One particular problem can be solved by using many algorithms. Some of them are for Sorting, and some for searching, optimizing and so on. Among of them **Linear search** and **Binary search** are the searching algorithms that I know. On the other hand, **Bubble sort, Marge sort, Quick sort** are the sorting algorithms that I know. In addition to this, I also know the **Greedy algorithm** for optimization including Bin packing, Knapsack Problem, Hoffman coding and some dynamic programming concepts.

Md. Foysal Ahmed
ID: 191-15-12486
Section: O-14

3. Why are you learning so many algorithms?

Ans:

In the world of computers, an algorithm is the set of instructions that defines not just what needs to be done but how to do it. Algorithms are the building blocks of computer programs. They are as important to programming as recipes are to cooking. The algorithms we use in programming are any series of steps that produce an output we are interested in, for example, searching for an item in a list.

Algorithms are important for everything. Computer scientists learn by experience. As future computer engineer we learn by seeing others solve problems and by solving problems by ourselves. Being exposed to different problem-solving techniques and seeing how different algorithms are designed helps us to take on the next challenging problem that we are given. By considering a number of different algorithms, we can begin to develop pattern recognition so that the next time a similar problem arises, we are better able to solve it. Algorithms are often quite different from one another. One algorithm may use many fewer resources than another. One algorithm might take 10 times as long to return the result as the other. For example, simple sorting algorithm like bubble sort and merge sort. Both are for sorting data elements in a given list. But their computational time is different. We would like to have some way to compare these two solutions. Even though they both work, one is perhaps "better" than the other. By knowing these algorithm, we can decide what will be best for a particular problem. As we study algorithms, we can learn analysis techniques that allow us to compare and contrast solutions based on their characteristics. In the worst case scenario, we may have a problem that is intractable, meaning that there is no algorithm that can solve the problem in a realistic amount of time. It is important to be able to distinguish between those problems that have solutions, those that do not, and those where solutions exist but require too much time or other resources to work reasonably.
To understand and solve all these cases mentioned above having knowledge on algorithm is must and that is the reason behind knowing various algorithms.

4 . Show analysis of a recursive algorithm

Ans:

Finding nth Fibonacci number using recursive approach given bellow:

IMPLEMENTATION

```
#include <bits/stdc++.h>

using namespace std;

int fibo(int n)
{
   if(n<=1)
   {
      return n;
   }
   else
   {
      return fibo(n-1)+fibo(n-2);
   }

}
int main()
{
   int n,number;
   cin >> n;

   number = fibo(n);

   cout<<number<<endl;

}
```

ANALYSIS

	Cost	Time
`fib(n):`		
`if n <= 1`	c1	1
`return 1`	c2	1
`return fib(n - 1) + fib(n - 2)`	T(n-1)+T(n-2)	

For n>1 there occurs 1 comparison, 2 subtractions, 1 addition.
Therefore, the time function can be written as,
$T(n) = T(n-1) + T(n-2) + 4$ where 4 is some constant that can be replaced with c

$T(n) = T(n-1) + T(n-2) + c$
Let's try to establish a lower bound by approximating that $T(n-1) \sim T(n-2), though\ T(n-1) \geq T(n-2)$, hence lower bound

$T(n) = 2T(n-2) + c$ [from the approximation T(n-1) ~ T(n-2)]

Md. Foysal Ahmed
ID: 191-15-12486
Section: O-14

Now, $T(n-2) = 2T(n-4) + c$

Substituting $T(n)$,
$T(n) = 2 * (2T(n-4) + c) + c$
$T(n) = 4T(n-4) + 2c + c$
$T(n) = 4T(n-4) + 3c$
Again,
$T(n-4) = 4T(n-4-4) + 3c$
$T(n-4) = 4T(n-8) + 3c$
Now substitute,
$T(n) = 4 * (4T(n-8) + 3c) + 3c$
$T(n) = 16T(n-8) + 12c + 3c$
$T(n) = 16T(n-8) + 15c$
…………………………………………….
…………………………………………….
…………………………………………….
$T(n) = 2^k T(n-2k) + (2^k - 1)c$
Let's find the value of k for which: $n - 2k = 0$
$k = n/2$

Therefore,
$T(n) = 2^{\frac{n}{2}} T(0) + (2^{\frac{n}{2}} - 1)c$
$T(n) = 2^{\frac{n}{2}} T(0) + 2^{\frac{n}{2}} * c - c$
$= 2^{\frac{n}{2}}(1 + c) - c$

$i.e. T(n) = 2^{n/2}$

now for the upper bound, we can approximate $T(n-2) \sim T(n-1)$ as $T(n-2) \leq T(n-1)$
$T(n) = 2T(n-1) + c$ [from the approximation $T(n-1) \sim T(n-2)$]

Now, $T(n-1) = 2T(n-2) + c$

Substituting $T(n)$,
$T(n) = 2 * (2T(n-2) + c) + c$
$= 4T(n-2) + 3c$
Again,
$T(n-2) = 4T(n-4) + 3c$
Now substitute,
$T(n) = 4 * (4T(n-4) + 3c) + 3c$
$T(n) = 16T(n-4) + 15c$

Md. Foysal Ahmed
ID: 191-15-12486
Section: O-14

……………………………………………
……………………………………………
……………………………………………

$T(n) = 2^k T(n-k) + (2^k - 1)c$
Let's find the value of k for which: $n - k = 0$
$k = n$
Therefore,
$T(n) = 2^n T(0) + (2^n - 1)c$
$\quad = 2^n * (1 + c) - c$

$i.e. T(n) = 2^n$

Hence the time taken by recursive Fibonacci is $O(2^n)$ or exponential.

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

Ans:
Let's consider a program finding the number of digits of a given number n.

Iterative approach given bellow:

IMPLEMENTATION

```
#include <bits/stdc++.h>
using namespace std;

int countDigit(int n)
{
   int count = 0;
   while (n != 0) {
      n = n / 10;
      ++count;
   }
   return count;
}

int main(void)
{
   int n;
   cin>>n;
   int res = countDigit(n);
   cout << "Number of digits : "<<res;
}
```

Md. Foysal Ahmed
ID: 191-15-12486
Section: O-14

If the input n = 1234 then the while loop will continue till n becomes 0 and in the meantime n will be divided by 10 each time which will eliminate the last digit of n.

When n = 1234 which is not equal to 0
Then n = n/10 which gives 123 and count will increase to one
Again n = 123 and n = n/10 which is 12 and count increases to one.
Again n = 12 and n = n/10 which is 1 and count increases to one.
Lastly n = 1 and n = n/10 which is 0 and count increases to one.
Now n becomes 0 which will break the loop and returns the number of count happened that is **4**.

Recursive approach given bellow:

IMPLEMENTATION

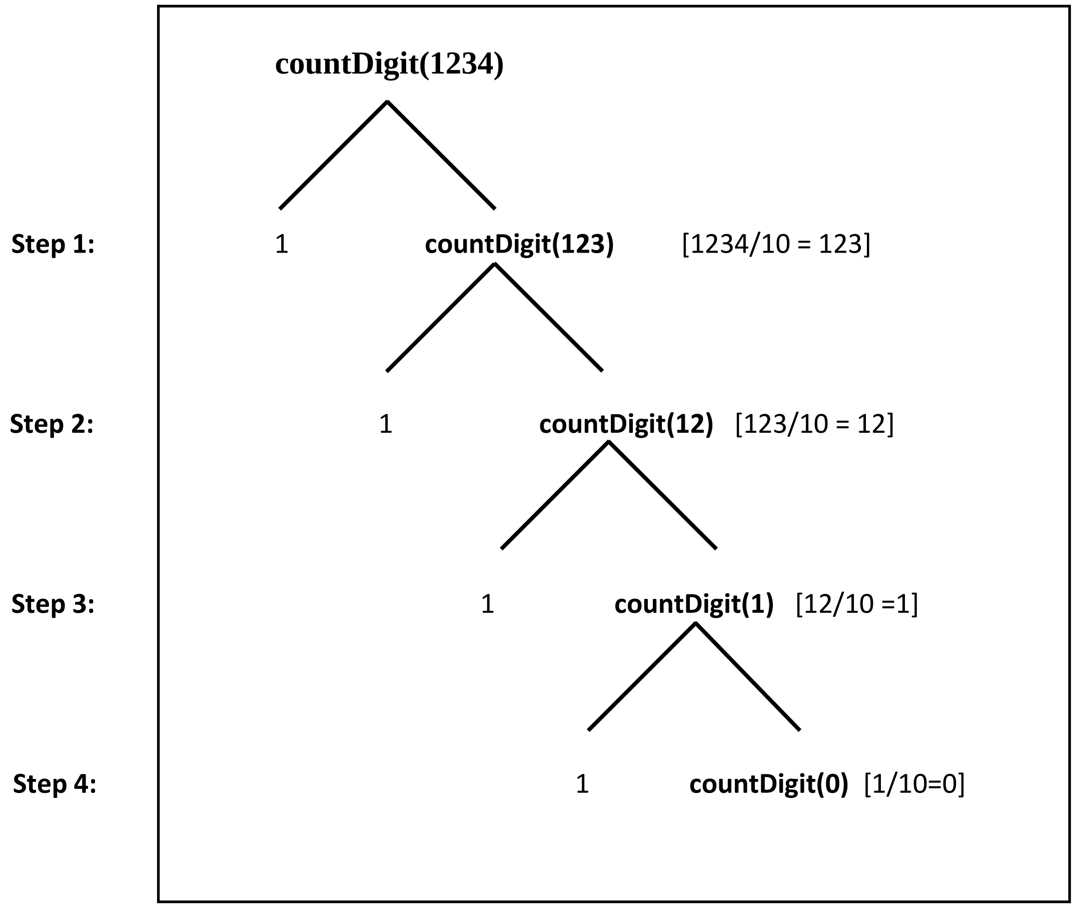
```
#include <bits/stdc++.h>
using namespace std;

int countDigit(long long n)
{
   if (n == 0)
      return 0;
   return 1 + countDigit(n / 10);
}

int main(void)
{
   int n;
   cin>>n;
   cout << "Number of digits :" << countDigit(n);
   return 0;
}
```

As it is a recursive algorithm that means the countDigit method will call itself till we reach to the base case which is n = 0.

If the input n = 1234 then the recursive tree will look like this;



After reaching the base case condition that is n=0 then step 4 will return 1 and resume step 3 which will return 1+1=2 and resume step 2 which will return 1+2 = 3 and again resume step 1 and it will return 1+3 = 4 and return to the main function.

Therefore, both the iterative and recursive algorithm the desired output is same which is 4.

Now we can say that this algorithm works for both iterative and recursive approach.

****

[THE END]

Md. Foysal Ahmed
ID: 191-15-12486
Section: O-14