Assignment-1

Course Code: CSE 214/CSE 215

Course Title: Algorithms & Lab

Submitted To

Name:Subroto Nag Pinku

Department of CSE

Daffodil International University

Submitted By

Name: Md. Amirul Islam

Id: 191-15-12123

Sec: 0-14

1. Write a short note about Optimization?

Optimization means adding some mathematical terms in solution, reducing time complexity. The optimization solution takes less time. It gives the best performance.

Example:

Q: Write a program that prints the sum of all numbers from 1 to n?

Brute Force solution:

```c
#include<stdio.h> int
main( ){ int n, sum;
scanf("%d", &n); sum
= 0;
        for(int i = 1; i <= n; i++)
                sum = sum +  i;
        printf("%d", sum);
        return 0;
}
```

Time complexity: O(n)

Space complexity: O(1)

Optimal solution:

```c
#include<stdio.h> int main(
){ int n, sum; scanf("%d",
&n); sum = n * (n + 1)  / 2;
printf("%d", sum); return 0;
}
```

Time complexity: O(1)

Space complexity: O(1)

An optimal solution gives better performance.

2. Write the different algorithms you know?

        = I know some mathematical, sorting, greedy, and searching algorithms.

Searching type algorithm =binary search and linear search

Sorting algorithm=insertion,bubble,insertion ,merge ,quicksort

Mathematically =Euclidean gcd algorithm

Greedy algorithm=bin packing, partial knapsack and coin change These are the algorithms that I know.

3. Why are you learning so much algorithm?
=By an algorithm we can design any type of program.it is used in every sector like for networking signal,traffic system we can use it .also for computer version we use the geometric algorithm and in artificial intelligence we use backtrack search algorithm.BY using an algorithm we can find cases like the best case, worst case, and avg case. Always it give us a good solution but sometimes it gives the worst-case. We can use algorithms to solve ant problems. And it makes our work easier for solving any problem. Also, I think that it is an idea to make our problem easier to understand. Mainly it depends on theory and every algorithm is mathematically proved.

4. Show analysis of a recursive algorithm?
        Now I am gonna design a recursive algorithm and analyze the following problem.
The user will give you a number as an input and you have to print it reversely.
Analysis of a input and output:
Input:
5 1 2 3 4
5
Output:
5 4 3 2 1
Algorithm:

```
void reverse(int n ){
        int c;
        scanf("%d",
        &c); if(n == 0)
        return;
```

```
        reverse(n - 1);
        printf("%d ", c);
    }
```

Explanation: It first takes n as a number of numbers. After that, it is called the recursive function reverse( ). Then it takes an input and assigns in variable c. After that, it called again the reverse function and stored the previous function into the memory stack. This process repeats until its value of n becomes 0. When the function finds the value of n 0, then it starts returning. Where a function return source starts doing its unfinished work that id printing.

When we call a function a fast input number will be in the first called function and the last number is in the last called function. It starts returning from last to first that's why it will be printed reversely.

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

I am gonna design an algorithm for the Fibonacci number sequence. We will find out the nth Fibonacci number.

Fibonacci sequences are: 1, 1, 2, 3, 5, 8….

Iterative algorithm: For Fibonacci number, we will assign 0 in an array index 0 and 1 in array index 1.

$Fibonacci_0 = 0$

$Fibonacci_1 = 1$

After that will run a loop and generate Fibonacci number into that array.

$$\sum_{i=2}^{n} Fibonacci_{i-1} + Fibonacci_{i-2}$$

Time complexity O(n).

Code:

```
#include<stdio.h> int main( ){ int n, fibo[(int)
1e5]; scanf("%d", &n); fibo[ 0 ] = 0; fibo[ 1 ]
```

= 1; for(int i = 2; i <= n; i++){ fibo[i] = fibo[i
- 1] + fibo[i - 2];
        }
 printf("%d\n", fibo[n]); return 0;
}

It's working perfectly.

Recurrence Algorithm:

Base case if the recursive Fibonacci algorithm is:

T(0)   0

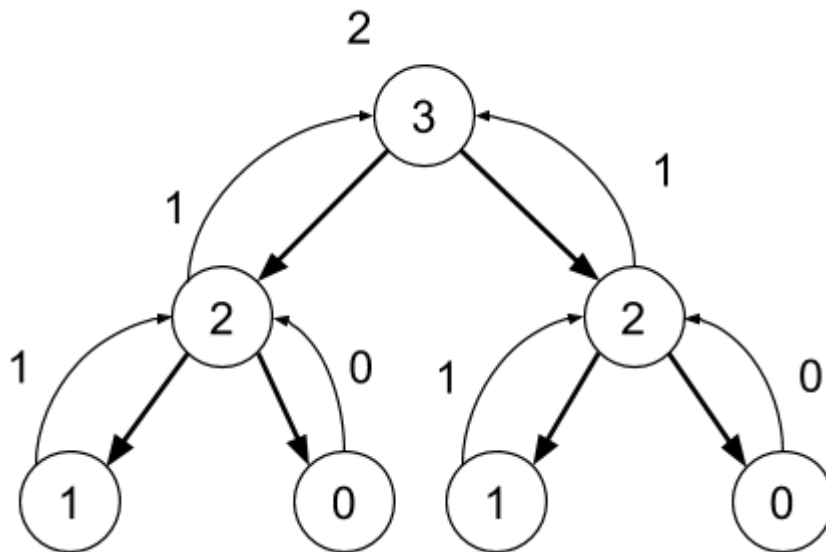T(1)   1

Function call of Fibonacci number:

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

Time complexity: O (2^n)

State-based tree diagram:

Let n = 3

Code:

```c
#include<stdio.h> int
fibonacci(int n){
        if(n == 0)
                return 0;
        else if(n == 1)
return 1;
        return fibonacci(n - 1) + fibonacci(n - 2);
}
int main( ){ int
        n, k;
        scanf("%d", &n); k = fibonacci(n);
        printf("%d fibonacci number:  %d\n", n,
        k); return 0;
}
```

It works perfectly.