Name : Kiran Tulsulkar NUID : 001057882

EECE 7205 : Fundamentals of Computer Engineering

Assignment 6

Problem 1 (30 Points)

In a Fibonacci sequence, each number is the sum of the two preceding ones, starting from 0 and 1. The first 12 numbers of the sequence are: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, ...

Write a C++ program to find the n^{th} number in a Fibonacci sequence using two different approaches. Each approach is implemented by a function that takes *n* as its only parameter. One function solves the problem only recursively. The other function solves the problem recursively but with the support of dynamic programming. For the dynamic programming implementation, you are allowed to define the needed array as a global variable. You are not allowed to use any loop in your functions. The only loop you will need in your program is the one in the `main()` function, right before calling the dynamic version function. You need this loop to initialize all elements in the array to an invalid Fibonacci number (e.g., -1) to keep track of the array elements that do not have a Fibonacci number calculated yet.

Test your code with *n* values 5, 10, 20, 30, 40

Submit your test program and in your report answer the following questions:

a) Comment on the running time of each function.
b) For each function, find the big *O* asymptotic notation of its running time growth rate.

```
-bash-4.2$ g++ as6q1.cpp -o as6q1 -std=c++11
-bash-4.2$ ./as6q1
Enter value of n = 5
Recursive Aproach : 5
Time taken for execution : 0
Dynamic Aproach : 5
Time taken for execution : 0
-bash-4.2$ ./as6q1
Enter value of n = 10
Recursive Aproach : 55
Time taken for execution : 0
Dynamic Aproach : 55
Time taken for execution : 0
-bash-4.2$ ./as6q1
Enter value of n = 20
Recursive Aproach : 6765
Time taken for execution : 0
Dynamic Aproach : 6765
Time taken for execution : 0
-bash-4.2$ ./as6q1
Enter value of n = 30
Recursive Aproach : 832040
Time taken for execution : 0.01
Dynamic Aproach : 832040
Time taken for execution : 0
-bash-4.2$ ./as6q1
Enter value of n = 40
Recursive Aproach : 102334155
Time taken for execution : 1.08
Dynamic Aproach : 102334155
Time taken for execution : 0
```

Name : Kiran Tulsulkar NUID : 001057882

Q1 a)

Running time of recursive Fibonacci is more than that of dynamic approach because you are calculating Fibonacci(4) you need Fibonacci(3) and Fibonacci(2), Now for Fibonacci(3), you need Fibonacci (2) and Fibonacci (1) but you notice you have calculated Fibonacci(2) while calculating Fibonacci(4) and again calculating it. So, we are solving many sub-problems again and again. In case of dynamic approach, we store these values and reduce the running time. Also it is clearly visible from the output as the value of n increase the recursive approach's running time goes on increasing.

Q1 b)

Recursive Approach :

T(n) = T(n-1) + T(n-2) + c
= 2T(n-1) + c //from the approximation T(n-1) ~ T(n-2)
= 2*(2T(n-2) + c) + c
= 4T(n-2) + 3c
= 8T(n-3) + 7c
= 2^k * T(n - k) + (2^k – 1)*c
Let's find the value of k for which: n - k = 0
k = n*T(n) = 2^n * T(0) + (2^n - 1)*c
= 2^n * (1 + c) – c

i.e. T(n) ~ 2^n

Dynamic approach :

T(n) = T(n-1) + c
= T(n-2) + 2c
= T(n-3) + 3c
= T(n-k) + kc
Let's find the value of k for which: n - k = 0
k = n*T(n) = T(0) + n*c
= n*c + 1

i.e. T(n) ~ n

Name : Kiran Tulsulkar NUID : 001057882

Problem 2 (40 Points)

Write a C++ program to compare the *Recursive* implementation of the *Rod Cutting* problem with the *Dynamic Programming* implementation of the same problem. The comparison should be in terms of each implementation running time for rod sizes: 5, 10, 15, …… , 40, 45, and 50 inches.

- Use the C++ clock function to measure the time consumed by each implementation in **micro-seconds**. (Reference: http://www.cplusplus.com/reference/ctime/clock/)
- Use the following formula to calculate the price of each cut of length *L* inches:

$$\text{Price} = \begin{cases} 2 & \text{if } L = 1 \\ \textbf{floor}(L*2.5) & \text{if } 1 < L < \text{rod size} \\ (L*2.5)-1 & \text{if } L = \text{rod size} \end{cases}$$

where **floor** is a C++ function that returns the largest integral value that is not greater than the value of its parameter.
(Reference: http://www.cplusplus.com/reference/cmath/floor/)

Based on the results from your program, fill the following table:
(*Note*: if a run takes more than 2 minutes to finish, just terminate it and report in the table "no solution").

Rod Size	Recursive Time	Recursive Time Revenue	Dynamic Time	Dynamic Time Revenue
5	0	12	0	12
10	0	25	0	25
15	0	37	0	37
20	0	50	0	50
25	0.33	62	0	62
30	9.6399	75	0	75
35	>2mins	-	0	87
40	>2mins	-	0	100
45	>2mins	-	0	112

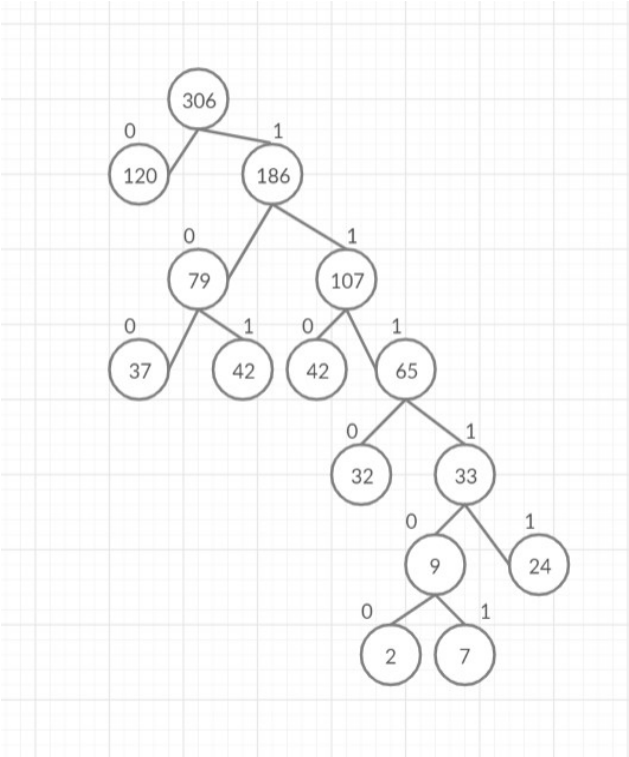
```
-bash-4.2$ g++ as6q2.cpp -o as6q2 -std=c++11
-bash-4.2$ ./as6q2
Enter Rod Size: 25
Price Array : 2 5 7 10 12 15 17 20 22 25 27 30 32 35 37 40 42 45 47 50 52 55 57 60 61
Recursive Aproach
Maximum Possible Revenue is: 62
Time taken for execution : 0.33

Dynamic Programming Aproach
Maximum Possible Revenue is: 62
Time taken for execution : 0
```

Name : Kiran Tulsulkar NUID : 001057882

Problem 3 (30 Points)

Find the Huffman encoding tree and hence the Huffman code for each of the letter shown in the following letter-frequency table:

Letter	Z	K	M	C	U	D	L	E
Frequency	2	7	24	32	37	42	42	120



Huffman Tree

Letter	Frequency	Huffman Code
E	120	0
L	42	101
D	42	110
U	37	100
C	32	1110
M	24	11111
K	7	111101
Z	2	111100

Letter Frequency Table