Practice Questions

BLM19307E / BLM22311E / SEN22311E

Algorithm Analysis and Design

Question #1

What is the time complexity of the following function? Indicate your answer in Θ(.) form.

```
void func(int n){
      int  i = n;
      int x = 0;
      while (i > 1){
            x = x + 2;
            i = i/3;
      }
      for(int j=1;j<=x;j++)
            for(int k=1;k<=x;k++)
                  print(“Hello”);
}
```

Question #2

```
void func(int n)
    if n=0 print('Hello');
    else
      for i=1 to 2^n
          func(n-1);
```

$p(n)$: number of prints when the input is n

Question #3

```
Algorithm: Secret(x)
// Input: x is a non-negative integer
// Output: ?
if x = 1:
    return 1
else
    return Secret(x - 1) + x*x*x
```

- Which problem is this algorithm designed to solve? Provide the recursive definition of problem (function Secret) and solve the recursion.
- Set up a recurrence relation **for the number of multiplications, i.e., basic operation,** made by the algorithm and solve it.
- Can you design a better algorithm by using the brute-force search technique? If you can, write the pseudo-code of this algorithm and the time efficiency class that this algorithm belongs to? Explain the reason

Question #4

Consider the clique problem: given a graph G and a positive integer k, determine whether the graph contains a clique of size k, i.e., a complete subgraph of k vertices. Design an exhaustive-search algorithm for this problem.

A clique, C, in an undirected graph G = (V, E) is a subset of the vertices, $C \subseteq V$, such that every two distinct vertices are adjacent. This is equivalent to the condition that the induced subgraph of G induced by C is a complete graph. In some cases, the term clique may also refer to the subgraph directly.

Question #5

The following is a recursive version of InsertionSort. Write down the recurrence relation that describes the number of write accesses to the array made in the worst case.

```
public static void sort(int[] array, int n) {
    // sorts the first n elements of array
    if(n == 0) {
         return;
    }
    else {
        int tmp = array[n-1];
        sort(array,n-1);
        int j;
        for (j = n-1; (j > 0) && (array[j-1] > tmp); j--) {
         array[j] = array[j-1];
        }
        array[j] = tmp;
    }
    return;
}
```

Question #6

Design a decrease-by-half algorithm for computing $\lfloor \log_2 n \rfloor$ and determine its time efficiency.

Question #7

Given two 64-bit integers a; n, here is an algorithm to compute a^n:

Power(a,n):
1. If $n = 0$: return 1.
2. Return $a \times$ Power$(a,n-1)$.

- Let T(n) denote the running time of Power(a,n). Write a recurrence relation for T(n).
- What is the solution to your recurrence from part (a)? Use $\Theta()$ notation.

Question #8

```
PARTA(n){
    steps that cost O(1)
    PARTA(n/4)
    steps that cost O(1)
    PARTA(n/4)
    steps that cost O(1)
    PARTA(n/4)
}
```

T(n) = ?

Question #9

You are given an array A of n numbers. A[i] is the price of a specific stock on day i. Your goal is to figure out the best days to buy and sell that stock in order to maximize profit. That is, you must find two days i, j such that $i < j$, and A[j] - A[i] is maximized. Brute-force algorithm for this problem has a complexity of $O(n^2)$, but we are seeking for a more efficient algorithm.

(a) Here is a divide and conquer algorithm for this problem:

1. If array has at most 2 elements, solve the problem trivially, return.

2. Solve the problem in first n/2 elements (recursively). Let p_1 be the maximum profit.

3. Solve the problem in last n/2 elements (recursively). Let p_2 be the maximum profit.

4. .....

.....

5. Return the maximum of p_1, p_2 and p_3.

Fill the fourth step of the above algorithm.

Question #10

Divide and Conquer Example – Longest Common Prefix

Given n strings, the problem is to find the longest common prefix of these strings. As an example, if the input is {"abdullah", "abdi", "abdal", "abdulkerim"}, output would be "abd". If the input is {"kelam", "kelime", "kemal", "kemik"}, output would be "ke".

(a) Design a divide-and-conquer algorithm for this problem. Please provide a step by step description of your algorithm.

(b) What is the time complexity of your algorithm? Write a recurrence relation and solve it. Provide an answer in terms of n and m, where n is the number of strings and m is the length of the largest string.