Design and Analysis of Algorithms
Fall 2021

Assignment 1

Submission: Oct. 13

Question 1. Below is the pseudo-code for a divide and conquer algorithm (binary search) to search a given element in a sorted array. Suppose that the input sorted array, *A,* is of size *n,* and the element to search is *x*. Analyze the computational cost of this algorithm in the form of *T(n)* and prove your conclusion.

```
binarySearch (A, left, right, x)
    if (left <= right) {
        mid = left + (right - left) / 2
        if (A[mid] == x) {
            return mid
        }
        if (A[mid] > x) {
            return binarySearch (A, left, mid – 1, x)
        }
        return binarySearch (A, mid+1, r, x)
    }
    return -1 // element not found
```

Question 2. Solve the following recurrence relations and give the asymptotic upper bound for each of them.

(a) $T(n) = 2T\left(\frac{n}{2}\right) + O(n),\ T(1) = 1$

(b) $T(n) = 2T\left(\frac{n}{4}\right) + O(n),\ T(1) = 1$

(c) $T(n) = 4T\left(\frac{n}{2}\right) + 1,\ T(1) = 1$

(d) $T(n) = T\left(n^{\frac{1}{2}}\right) + 1,\ T(1) = 1\ ,\ T(2) = 1$

Question 3. Analyze the time cost of the following algorithms using Θ notation. You need to give the details and prove your answer.

1).
```
int func1(int n) {
    int sum = 0
    for (int i = 1; i < n; i *= 2)
        sum += i
    return sum
}
```

2).
```
int func2(int n) {
    int sum = 0
    for (int i = n; i > 0; i--) {
        for (int j = i; j < n; j*=2) {
            for (int k = 0; k < j; k++) {
                sum += 1
            }
        }
    }
    return sum
}
```

3).
```
int func3(int n) {
    if (n <= 1)
        return n
    return n*func3(n-1)
}
```

Question 4. Design a divide-and-conquer algorithm to determine whether a given number *n* is a perfect square. A number is a perfect square if it can be expressed as the product of two equal integers. For example, *9* is a perfect square because *9 = 3 * 3*.

Note that your algorithm should finish in *O*(*log*(n)) time and it should not use any math functions. Show the pseudocode of your algorithm and analyze its time complexity.

Question5. Given a staircase with *n* steps, a person standing at the bottom wants to reach the top. The person can climb 1 stair or 2 stairs or 3 stairs at a time. There are different ways for the person to reach the top. For example, if there are totally 3 stairs, there are 4 ways: (1step, 1step, 1step), (1step, 2steps), (2steps, 1step) and (3steps).

1) Use divide and conquer to design an algorithm to count the number of ways the person can reach the top.

2) Analyze the time cost of your algorithm and give the asymptotic upper bound.

Question6. In the linear-time divide-and-conquer algorithm taught in class for the selection problem, recall that we divided the input elements into groups of 5. Analyze the running time of this algorithm, assuming that the input elements are divided into groups of 4. Does your algorithm run in linear time?

[Use a similar proof as given in the lecture slides "Divide & Conquer: Linear Time Selection", page 8.]