XJTLU | Computing

CPT108: Data Structures and Algorithms

Semester 2, 2023-24

Tutorial 1

Recursive Problem Solving

1. Suppose you are climbing the stairs. It takes you n steps to get to the roof ($n \geq 1$). You can climb 1 or 2 steps at a time. How many different ways can you get to the top?

   (a) Consider the base case. How many ways can you get to the first step? How about the second step?

   (b) Consider the state transition. Suppose there are $f(n-2)$ ways to get to the $(n-2)^{th}$ step, and $f(n-1)$ ways to get to the $(n-1)^{th}$ step, how many ways can we get to the n^{th} floor? Compute $f(n)$ with $f(n-2)$ and $f(n-1)$.

   (c) Complete the following code with above conclusions.

```
public int climbStairs(int n) {
    if (n==1) return _______;
    if (n==2) return _______;
    return _______;
}
```

   (d) Give an integer array cost, where `cost[i]` is the amount you pay to climb up from the i^{th} step of the staircase. Once you pay this fee, you can choose to climb one or two steps up. You can choose to climb the stairs starting from step 0 or 1. Complete the following code to calculate and return the minimum cost to reach the top of the stairs.

```
public int minCostClimbingStairs(int[] cost) {
    return this.minCostClimbingStairsRecursive(cost, cost.length);
}

public int minCostClimbingStairsRecursive(int[] cost, int n){
  // your implementation



}
```

Code Tracing

2. The following method implements a bubble sort algorithm.

```
1   public void bubbleSort(int[] array){
2       for(int i=0; i<array.length; i++){
3           for(int j=0; j<array.length-1; j++){
4               if(array[j]>array[j+1]){
5                   int tmp = array[j+1];
6                   array[j+1] = array[j];
7                   array[j] = tmp;
8               }
9           }
10      }
11  }
```

(a) Given the input array as $[10, 5, 4, 3]$, trace the array after each iteration.

i	j	array
0	0	5, 10, 4, 3
0	1	5, 4, 10, 3
0	2	
1	0	
1	1	
1	2	
2	0	
2	1	
2	2	
3	0	
3	1	
3	2	

(b) In terms of the above bubble sort algorithm, given an array with length as n, for each iteration of the outer loop, how many times does the inner loop iterate? Observing the table, is it necessary for the inner loop to iterate so many times? If not, try to find the minimum number of iterations of the inner loop, and ensure that any array can be sorted successfully under this number of iterations (Suppose the index of outer loop is i).

3. The following code is intended to find the position of the number equals to the target within an ascending array.

```java
public int binarySearch(int[] nums, int target) {
    int left = 0, right = nums.length - 1;
    while (left <= right) {
        int mid = (right - left) / 2 + left;
        int num = nums[mid];
        if (num == target) {
            return mid;
        } else if (num > target) {
            right = mid;
        } else {
            left = mid;
        }
    }
    return -1;
}
```

Trace the code with `nums` as $[2, 4, 5, 7, 10, 13]$ and `target` as 13, what errors do you find, and how you fix the code?

4. The following method 'search' is intended to find the position of the first number that larger than the target within an ascending array (e.g., given an array $[0, 2, 5, 7]$ and target as 3, the method should output 2 since 5 is the first element that larger than 3).

```java
public int search(int[] nums, int target) {
    return firstLarger(nums, 0, nums.length-1, target);
}

public int firstLarger(int[] nums, int left, int right, int
   target){
    if (nums.length == 1){
        if (nums[0] > target)
            return 0;
        else
            return -1;
    }
    int mid = (right-left)/2 + left;
    if (nums[mid] <= target){
        return firstLarger(nums, mid+1, right, target);
    } else {
        if(nums[mid-1] <= target)
            return mid;
        return firstLarger(nums, left, mid-1, target);
    }
}
```

(a) Trace the code with:
   (1) `nums` as $[-1, 0, 3, 5, 9, 12]$ and `target` as 8;
   (2) `nums` as $[1, 3, 4]$ and `target` as 0.

   What error do you find, and how you fix the code?

(b) What is the time complexity of this method?