

CS 312 - Algorithm Design

Midterm - April 19

1. The Fibonacci series is defined by the recurrence: $F(n) = F(n-1) + F(n-2)$ for all $n \geq 2$. Additionally, $F(0) = 0$ and $F(1) = 1$. This can be translated straightforwardly into the following recursive function:

```
int fib(n) {
    if (n == 0) {
        return 0;
    }
    if (n == 1) {
        return 1;
    }
    return fib(n-1) + fib(n-2);
}
```

Rewrite this function to use dynamic programming.

2. On the last exam, we looked at a greedy algorithm to produce change, shown below. On that exam, the question was whether this algorithm always produced change using the fewest number of coins. The answer was that for the coin system we are accustomed to, it does produce an optimal answer at all times. However, for more unusual coin systems, it is not always optimal, like having coins valued at 4, 3 and 1. If we needed to give 6 coins in change, the greedy algorithm would give $4 + 1 + 1$. An optimal solution would be $3 + 3$. Here is the greedy algorithm from the last exam.

```
giveChange (amount) {
    numQuarters = amount / 25
    amount = amount % 25
    numDimes = amount / 10
    amount = amount % 10
    numNickels = amount / 5
    amount = amount % 5
    numPennies = amount
}
```

- a. Which algorithm design technique would you use to solve this problem for an arbitrary coin system and arbitrary amount of change: divide-and-conquer or dynamic programming?
- b. Using the strategy you selected, design an algorithm that will produce the fewest number of coins as change, given coins valued at c_1 to c_k and producing change in any value from 1 to n . You may assume that the smallest-valued coin has a value of 1 so that any amount of change can be produced. You can also assume that you have an unlimited supply of each type of coin.
- c. Work through the example of producing change worth 10 for a coin system where the coins are valued at 6, 5 and 1 using the algorithm you designed.
- d. What is the $O()$ cost of the algorithm that you designed.

3. You are given a pile of coins that all look the same. They are the same size, the same color and the same engravings. You are also told that exactly one of the coins is fake. The fake coin can be distinguished from real coins because it weighs less than real coins. You are also given a balance that you can use to compare the weights of coins. Each side of the balance is large enough to hold the entire pile of coins.
 - a. Which algorithm design technique would you use to solve this problem: divide-and-conquer or dynamic programming?
 - b. Using the strategy you selected, design an algorithm to solve the problem of identifying the fake coin.
 - c. Using $O()$ notation, how many weighings does your algorithm require in the worst case to find the fake coin.