King Fahd University of Petroleum and Minerals

College of Computer Science and Engineering
Computer Engineering Department
COE 466: Quantum Architecture and Algorithms

Problem Set 2

Due date: Wednesday 7-10-2020 (11:59 PM)

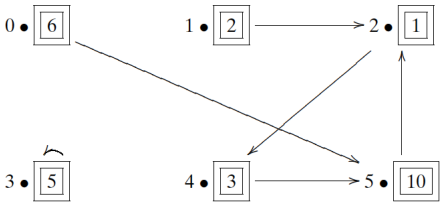
Problem Sets

1. Given matrix M that represents the dynamics of the marble system in Figure 1.

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

   Calculate M^2 and M^3. If all the marbles start at vertex 2, where will all the marbles end up after 5 time steps?

Figure 1: Marble system



2. Consider the following hypothetical situation at a hypothetical college. Thirty percent of all math majors become computer science majors after one year. Another 60% become physics majors after one year. After a year, 70% of the physics majors become math majors and 10% of the physics majors become computer science majors. In contrast to the other

departments, computer science students are usually very happy: only 20% of them become math majors and 20% become physics majors after a year.

(a) Draw a graph that describes the situation.

(b) Give the corresponding adjacency matrix. Notice that it is a doubly stochastic matrix.

(c) Consider three students. If each student is majoring in one of these three areas, indicate the student's probable major after 2 and 4 years. For example, if a student is majoring in Math, what is the probability that he ended up majoring in CS or Phys in 2 and 4 years. Calculate the same probability of a student majoring in Physics or CS.

3. (Programming assignment) Write a Python program that takes two inputs: a list of numbers and a target value. Your program should check if the target value exists in the provided list of numbers. If the target value exists, the program should return three outputs:

   (a) The location of the (first) occurrence of target value in the list

   (b) The time it took to find the target number

   (c) The number of steps (comparisons) it took to find the target number

   You are requested to implement two searching algorithms:

   (a) Linear search: A typical pseudocode for linear search is in Algorithm 1.

```
Data: list, value
Result: position of value
begin
    for each item in list do
        if item == value then
            Return item's location
        end
    end
end
```

Algorithm 1: Linear Search

   (b) Binary search: A typical pseudocode for binary search is is in Algorithm 2. Note that binary search must take the sorted list as input.

   Your program should first read each (sorted) list from the file *lists.txt*. The format of each list is as the following: $[value : x_1x_2x_3x_4, \ldots, x_n]$, where value is the target value you should search for and $x_1, x_2, \ldots, x_n$ are the

```
Data: sorted list A, value
Result: position of value
begin
    lowerBound = 1
    lowerBound = size(list)
    while value is not found do
        if upperBound < lowerBound then
            Value doesn't exist in the list
            Return -1
        end

        midPoint = lowerBound + (upperBound − lowerBound)/2

        if A[midPoint] < value then
            lowerBound = midPoint + 1
        end

        if A[midPoint] > value then
            upperBound = midPoint − 1
        end

        if A[midPoint] == value] then
            Value is found
            Return midPoint
        end
    end
end
```

Algorithm 2: Binary Search

list items separated with a space. Figure 2 depicts the expected output of your program. Compare the number of steps for both algorithms, what do you observe? Explain their performance in terms of the algorithms complexity.

After that, you should run the same program with the other lists (*worstCase_lists.txt*). Please compare the output of the two lists. What do you observe?

Figure 2: Expected output for each search algorithm

```
List 1 -> postition:19 # of steps:20
List 2 -> postition:41 # of steps:42
List 3 -> postition:13 # of steps:14
List 4 -> postition:28 # of steps:29
List 5 -> postition:18 # of steps:19
List 6 -> postition:49 # of steps:50
List 7 -> postition:28 # of steps:29
List 8 -> postition:38 # of steps:39
List 9 -> postition:2 # of steps:3
List 10 -> postition:10 # of steps:11
```

Useful Resources

1. Python Tutorial
2. Data Structure and Algorithms