Computer Science 385
Design and Analysis of Algorithms
Siena College
Spring 2024

Problem Set 5

Due: 4:00 PM, Friday, March 15, 2024

You may work alone or with a partner or in a group of three on this assignment. However, in order to make sure you learn the material and are well-prepared for the exams, you should work through the problems on your own before discussing them with your partner, should you choose to work with someone. In particular, the "you do these and I'll do these" approach is sure to leave you unprepared for the exams.

There is a significant amount of work to be done here, and you are sure to have questions. It will be difficult if not impossible to complete the assignment if you wait until the last minute. A slow and steady approach will be much more effective.

Submitting

Please submit a hard copy (typeset preferred, handwritten OK but must be legible) for all written questions. Only one submission per group is needed.

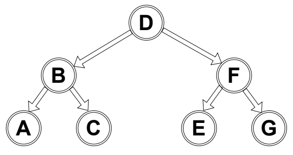
Tree Traversals

Recall the idea of tree traversals, specifically in the context of binary trees. Three types of binary tree traversals can be specified recursively, and are shown below, along with an example binary tree, which happens to be a binary search tree.

```
ALGORITHM Preorder($T$)
    if $T$ not empty then
        visit the root of $T$
        Preorder($T_L$)
        Preorder($T_R$)
```

```
ALGORITHM Inorder($T$)
    if $T$ not empty then
        Inorder($T_L$)
        visit the root of $T$
        Inorder($T_R$)
```

```
ALGORITHM Postorder($T$)
    if $T$ not empty then
        Postorder($T_L$)
        Postorder($T_R$)
        visit the root of $T$
```



Question 1: For each traversal type, give the order in which tree nodes are visited on the tree above. (3 points)

Question 2: How many times is INORDER called if initially called on the root of the sample tree above as input? Be sure to count all the base cases because it gets called on a lot of empty trees... (2 points)

For the next four questions, Suppose INORDER is called on a tree with n nodes, y of which have exactly one child and z of which are leaf nodes. This means $n - (y + z)$ nodes have two children.

Question 3: Exactly how many base case calls to INORDER are made? Express your answer in terms of n, y, and/or z. (2 points)

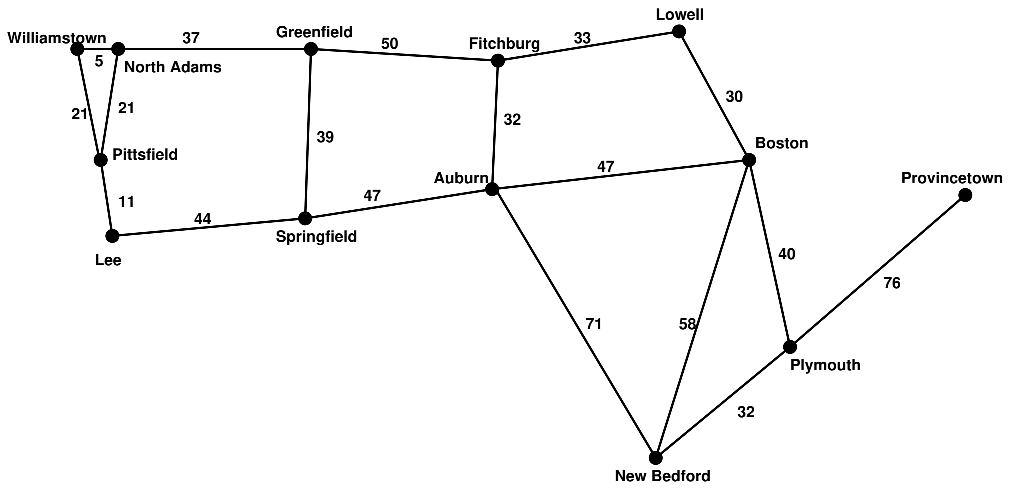
Question 4: How many total calls to INORDER are made (count all base case calls and non-base case calls)? Express your answer in terms of n, y, and/or z. (2 points)

Question 5: Give an upper bound on the total number of calls to INORDER in terms of just n and using Θ-notation. Note that y and z are both less than or equal to n. (2 points)

Question 6: For each call to INORDER, a constant number of operations are done testing for the base case and printing the data. Knowing an upper bound on the total number of calls made and that a constant number of operations are done per call, what is the Big-Θ running time of INORDER when called on a tree of n nodes? (2 points)

Graph Traversals

For the next two questions, please refer to the graph below. You may ignore the edge weights. Break ties by alphabetical order.



Question 7: Perform a breadth-first traversal of the graph starting at Williamstown. Following the example in Levitin Section 3.5, show the contents of the queue and the order in which they are added, and include cross edges in your breadth-first traversal tree. (5 points)

Question 8: Perform a depth-first traversal of the graph starting at Williamstown. Following the example in Levitin Section 3.5, show the contents of the stack and the order in which they are pushed, and include back edges in your depth-first traversal tree. (5 points)

Your Internship in Admissions

In college admissions, one measure of similarity between schools is the number of "cross admits" (*i.e.*, students who applied to and were offered admission to both). A large number of cross admits would mean the schools are seen by college-bound students as being similar. Your task is to take the lists of students admitted to college A and B and count the number of names that appear in both. Your input consists of two arrays: `A[0...n-1]` contains the names of n students admitted to college A, and `B[0...m-1]` contains the names of m students admitted to college B. For example, for the two unreasonably small arrays below, 2 should be returned as the answer, because students "D. Chambers" and "I. Jones" were the only ones in both arrays. You may assume there are no duplicate names in the arrays, and that no two students exist with the same name.

```
A[0...7] = {"S. Cooper", "D. Chambers", "H. Stone", "M. Brady",
            "A. Fonzarelli", "I. Jones", "P. Venkman", "L. Organa"}

B[0...4] = {"I. Jones", "M. McFly", "F. Gump", "D. Chambers", "M. Simpson"}
```

Question 9: You could easily write a brute force algorithm solving this problem in $\Theta(nm)$ time. But the number of students in both arrays is very large, and so an asymptotically more efficient algorithm is required. Write such an algorithm. (20 points)

```
ALGORITHM NumCrossAdmits($A, B$)
    //Input: $A[0..n-1]$, admits to college A
    //Input: $B[0..m-1]$, admits to college B
    //Output: the numbers of names that appear in both $A$ and $B$
```

Question 10: Give the Θ efficiency class of your algorithm and briefly explain how you arrived at this bound. (3 points)

Play Ball!

Major League Baseball teams are deep into spring training and will be starting the season while we're on Easter break, so let's consider a problem involving baseball statistics. Suppose you have a large array of numbers representing the batting averages of all players in a league who played in some minimum number of games, *e.g.*:

$$A = .312, .288, .185, .300, .276, .297, .307, .330.$$

Your task is to find the pair of batting averages that are closest together in value.

Question 11: Develop pseudocode for an algorithm to solve this problem the operates in better than $\Theta(n^2)$ time. (20 points)

Question 12: Give the Θ efficiency class of your algorithm and briefly explain how you arrived at this bound. (3 points)

Make Money Fast

For the remaining questions, consider the following problem.

Determining the best time to buy and sell a stock is a problem of interest to many people for obvious reasons. Perhaps examining past behavior might be a way to predict future behavior, leading to massive profits and a life of leisure. The problem we will consider is to find optimal buy and sell days given a sequence of known daily stock price changes over some period of days in the past. For example, consider the daily changes in value shown below for Roger Bacon Enterprises stock over 15 days. The values indicate the amount by which the stock price changed from the start of the trading day to the end of the trading day. Positive values indicate the stock increased in price on that day, and negative values indicate the price fell.

```
-6 -1 +5 +7 +5 +3 -5 -7 -1 +10 +1 +5 -20 +10 +1
```

We want to calculate the maximum amount of profit you could have earned assuming you buy exactly once (at the beginning of a day) and sell exactly once (at the end of a day, possibly the same day you bought) during the 15 day period. For example, if you bought on day 2 and sold on day 5, then you would have a net profit of 20. However, if you bought on day 2 and sold on day 11, then you would have a net profit of 23. (Note: we're all computer scientists here, so the first day is numbered 0.)

We could easily solve this problem in $\Theta(n^2)$ time using brute force by examining all pairs of buy and sell days, where n is the number days in the period being considered. But that's last month's news. We can do better!

Question 13: Improve upon this by giving a divide and conquer algorithm solving this problem that runs asymptotically faster than $\Theta(n^2)$. Start with the pseudocode below. It returns the maximum profit you could have made if you bought and sold exactly once within the range of days indicated. A complete and correct base case is provided. (20 points)

```
ALGORITHM MaxProfit($A, sday, eday$)
    //Input: an array of price changes $A[0..n-1]$
    //Input: start ($sday$) and end ($eday$) days of the subrange to consider
    //Output: the maximum possible profit from buying and selling within the range
    if $sday = eday$ then
        // only one day in the range, so must buy and sell on this day
        return $A[sday]$
```

Question 14: Once you have your algorithm, complete the recursive call tree started below, showing all the recursive calls your algorithm will make and the return value from each. (6 points)

The root of your tree should look like what you see below, with `MP` (short for `MaxProfit`) and the blank filled in with the answer that the call returns. All other nodes in your tree should be formatted similarly.

`MP(A,0,7)`
`returns ________`

Use `A = [6, -50, 2, 13, 5, -14, 1, 6]` as your input array.

Question 15: Give a recurrence formula for the worst case running time of your algorithm. (2 points)

Question 16: What is the Θ efficiency class based on your recurrences. (You may find the closed form of your recurrence in any way you like or apply the Master Theorem, if applicable.) (3 points)

Grading

This assignment will be graded out of 100 points.

Feature	Value	Score
Question 1	3	
Question 2	2	
Question 3	2	
Question 4	2	
Question 5	2	
Question 6	2	
Question 7	5	
Question 8	5	
Question 9	20	
Question 10	3	
Question 11	20	
Question 12	3	
Question 13	20	
Question 14	6	
Question 15	2	
Question 16	3	
Total	100	