CS311: Homework #5

Due on November 11, 2013 at 4:30pm

Professor Lathrop Section 3

Josh Davis

Problem 1

Give algorithms for the following operations.

Solution

Algorithms for each part are below.

Part A

Give an algorithm that multiples two degree-1 polynomials with only three multiply operations.

```
function MultiplySingleDegreePolynomials(a, b, c, d)
    return 0
end function
```

Algorithm 1: Multiply 1 degree polynomial

Part B

Give a divide-and-conquer algorithm for multiplying two polynomials of degree n. Prove using the master theorem that your algorithm runs in $\Theta(n^{\log_2 3})$. You may assume that $n + 1$ is a power of 2.

```
function MultiplyNDegreePolynomials
    return 0
end function
```

Algorithm 2: Multiply n degree polynomial

Problem 2

Give an $O(\log n)$ time algorithm that computes the following function:

MEDIAN-OF-TWO(l_1, l_2)

Input: l_1 and l_2 are two sorted lists of integers. Each list has n elements ($2n$ elements in total) and the value of each element in the lists is unique.

Output: The value of the n^{th} smallest integer in the set of $2n$ integers is l_1 and l_2.

Solution

```
function Mid(x, y)
    return (y − x)/2 + x
end function

function Median-Of-Two(l1, l2)
    start1 ← 0
    start2 ← 0
    end1 ← l1.length
    end2 ← l2.length
    mid1 ← Mid(start1, end1)
    mid2 ← Mid(start2, end2)

    while mid1 < end1 and mid2 < e2 do
        if l1[mid1] < l2[mid2] then
            start1 ← mid1
            end2 ← mid2
        else
            end1 ← mid1
            start2 ← mid2
        end if

        mid1 ← Mid(start1, end1)
        mid2 ← Mid(start2, end2)
    end while

    if mid1 ≥ end1 then
        return l1[mid1]
    else
        return l2[mid2]
    end if
end function
```

Algorithm 3: Value of the n^{th} smallest integer in either list

Problem 3

Give an $O(n)$ average case running time algorithm that computes the following:

Kth-SMALLEST(*list*, *k*)

Input: An unsorted list *list* of unique integers and an integer k
Output: The value of the k^{th} smallest integer from the list

Solution

The algorithm is based off of **QuickSort** and is often called **QuickSelect**. The principle idea is to use the partitioning function of **QuickSort** because the element that is selected to be a partition will be placed in its correct place in the list. The position then will tell us where to look for the kth item.

```
function KTH-SMALLEST(list, k)
    index ← KTH-SMALLEST-REC(list, k, 0, list.length)
    return list[index]
end function

function QUICKSELECT(list, k, start, end)
    if start >= end then
        return end
    end if

    partition ← PARTITION(list, k, start, end)

    if k < partition then
        return QUICKSELECT(list, start, partition − 1)
    else if k > partition then
        return QUICKSELECT(list, partition + 1, end)
    end if

    return list[k]
end function
```

Algorithm 4: Give the k^{th} smallest integer from the list

Where the **Partition** algorithm is dependent on the type of partitioning scheme used.

Problem 4

Let T be a tree with n vertices. We say that a vertex v is a `minimal separator of T` if its removal splits T into two or more subtrees, each with at most $n/2$ nodes.

Solution

Part A
Show that ever finite tree has at least one minimal separator.

Proof. □

Part B
Show an $O(|V|)$ algorithm for identifying a minimal separator in a given tree.

```
function MinimalSeparator(T)
    return 0
end function
```

Algorithm 5: Identify a minimal separator in the given tree

Problem 5

Give an algorithm that computes the following:

BST-Reconstruction(*traversal*)

Input: An array of elements generated by a pre-order traversal of some binary search tree.
Output: A binary search tree identical to the original tree.

Solution

Solution for the function is below:

```
function BST-Reconstruction(traversal)
    return 0
end function
```

Algorithm 6: Binary search tree reconstruction

Problem 6

Let $G = (V, E)$ be a connected, undirected graph.

Prove or disprove:

$$\exists v \in V \mid G = (V \setminus \{v\}, E) \text{ is connected}$$

Solution

Problem 7

A *mother* vertex in a directed graph $G = (V, E)$ is a vertex v such that all other vertices in G can be reached by a directed path from v.

Solution

Part A

Give an $O(n + m)$ algorithm to test whether a given vertex v is a mother of G, where $n = |V|$ and $m = |E|$.

```
function IsMother(v, G)
    return 0
end function
```

Algorithm 7: Determine if a given vertex is a mother of a graph

Part B

Give an $O(n + m)$ algorithm to test whether a graph G contains a mother vertex.

```
function ContainsAMother(G)
    return 0
end function
```

Algorithm 8: Determine if a given graph has a mother

Problem 8

Suppose we are given the minimum spanning tree T of a given graph G and a new edge $e = (u, v)$ of weight w that we will add to G.

Solution

Give an $O(n)$ algorithm to find the minimum spanning tree of the graph $G + e$.

function $\text{ADDMST}(T, e, G)$
    return 0
end function

Algorithm 9: Add an edge to a given minimum spanning tree

Problem 9

Problem 6–7 from the text.

Solution

Part A

Let T be a minimum spanning tree of a weighted graph G. Construct a new graph G' by adding a weight of k to every edge of G. Do the edges of T form a minimum spanning tree of G'? Prove the statement or give a counterexample.

Proof. □

Part B

Let $P = \{s, \ldots, t\}$ describe the shortest weighted path between vertices s and t of a weighted graph G. Construct a new graph G' by adding a weight of k to every edge of G. Does P describe a shortest path from s to t in G'? Prove the statement or give a counterexample.

Counterexample

P still does not necessarily describe a shortest path from s to t in G'. This is weight added to each path is dependent on how many sections are in the path. If we were to let $k = 5$ and add it to each path, a path that has 1 edge will have 5 added to it, but a path with 2 edges will get 10 added to it. This is illustrated in the below example.

The first figure, Figure 1 is G and that path is the three edges straight across the line of nodes, from s to t then to u. However the second graph is Figure 2 has had k added to each edge, the shortest path is now the single edge from s to u.

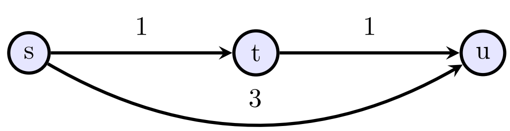

Figure 1: The graph G.

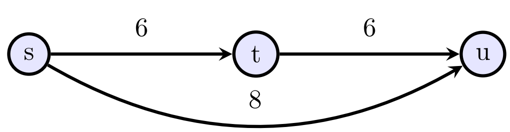

Figure 2: The graph G'.