

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 multiplies two degree-1 polynomials with only three multiply operations.

```
1: function MULTIPLYSINGLEDegreePolynomials
2:   return 0
3: 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.

```
1: function MULTIPLYNDegreePolynomials
2:   return 0
3: 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

```
1: function MEDIAN-OF-TWO( $l_1, l_2$ )  
2:   return 0  
3: 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 *k*th item.

```
1: function KTH-SMALLEST(list, k)
2:   index  $\leftarrow$  KTH-SMALLEST-REC(list, k, 0, list.length)
3:   return list[index]
4: end function
5:
6: function QUICKSELECT(list, k, start, end)
7:   if start  $\geq$  end then
8:     return end
9:   end if
10:
11:   partition  $\leftarrow$  PARTITION(list, k, start, end)
12:
13:   if k < partition then
14:     return QUICKSELECT(list, start, partition - 1)
15:   else if k > partition then
16:     return QUICKSELECT(list, partition + 1, end)
17:   end if
18:
19:   return list[k]
20: 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 every 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.

```
1: function MINIMALSEPARATOR( $T$ )  
2:   return 0  
3: 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:

```
1: function BST-RECONSTRUCTION(traversal)  
2:   return 0  
3: 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 $m = |E|$.

```
1: function ISMOTHER( $v, G$ )  
2:   return 0  
3: 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.

```
1: function CONTAINSAMOTHER( $G$ )  
2:   return 0  
3: 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 $G + e$.

```
1: function ADDMST( $T, e, G$ )  
2:   return 0  
3: 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 $P = \{s, \dots, t\}$ 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.

Proof.

□