

## CSE3358 Problem Set 9

04/15/05

Due 04/22/05

### Problem 1: Relaxed string matching (20 points)

We have seen in class a number of algorithms for the string matching problem where, given a text  $T[1..n]$  over some finite alphabet  $\Sigma$  and a pattern  $P[1..m]$ , we need to determine all positions in  $T$  where  $P$  occurs (called valid shifts). The Naive algorithm runs in  $O(mn)$  time. The Rabin-Karp algorithm runs in  $O(n + mv)$  expected time where  $v$  is the number of valid shifts. The suffix tree algorithm runs in  $O(n + m + v) = O(n + m)$  time. In this problem we consider a relaxed version of the string matching problem that can be solved in linear time.

Let  $T_i^m = T[i..i + m - 1]$ . We say  $T_i^m$  is an **anagram** of  $P$  if there is a way of permuting symbols of  $T_i^m$  so that the resulting array is equal to  $P$ .

(a) (10 points) Give an algorithm that, given an index  $i$ , determines whether  $T_i^m$  is an anagram of  $P$ . Your algorithm should be asymptotically as efficient as possible.

(b) (10 points) Give an algorithm that, given  $T$  and  $P$ , reports all  $i$ 's such that  $T_i^m$  is an anagram of  $P$ . Your algorithm should run in  $O(n + m)$  time.

### Problem 2: Finding maximal repeats (20 points)

Consider a string  $T[1..n]$ . Define  $T[0]$  and  $T[n + 1]$  as a special symbol that does not occur in  $T$ . Consider a string  $P[1..m]$ . We say  $P$  is a maximal repeat of  $T$  if  $\exists 0 < i, j \leq n$  such that:

- $P = T_i^m = T_j^m, i \neq j$
- $T_{i-1}^{m+1} \neq T_k^{m+1} \forall k$
- $T_i^{m+1} \neq T_k^{m+1} \forall k$

(a) (5 points) Based on the above definition, explain in plain English what a maximal repeat is.

(b) (10 points) Consider the suffix tree of  $T$ . Show that  $P$  is a maximal repeat of  $T$  is equivalent to  $\exists v$  in the suffix tree of  $T$  such that:

- $v$  is not a leaf
- the path from root to  $v$  spells  $P$
- there are two leaf nodes in  $v$ 's subtree, say  $i$  and  $j$ , such that  $T[i - 1] \neq T[j - 1]$

(c) (5 points) Show that a string  $T$  of length  $n$  has  $O(n)$  maximal repeats.

**Problem 3: The Thief and the Knapsack problem... and You (20 points)**

A thief robbing a store finds a set  $S$  of  $n$  items 1 to  $n$ . Item  $i$  has a value  $v_i$  dollars and a weight  $w_i$  pounds, where  $v_i$  and  $w_i$  are integers. He wants to take as valuable a load as possible, but he can carry at most  $W$  pounds in his knapsack for some integer  $W$ . Unfortunately, he did not take CSE3358; otherwise, he would not be stealing (would he?). But he needs an algorithm to find a set of items  $A \subseteq S$  of maximum total value such that  $\sum_{i \in A} w_i \leq W$ .

Now you have a moral issue here... Should you help the thief in solving his problem and in return get some points for this homework, or should you just refuse and lose the points? It's up to you... :) But if you decide not to help him, you definitely get 5 points for honesty.

(a) (10 points) This problem satisfies the optimality condition for subproblems: If  $A$  is an optimal solution for  $S$  with weight at most  $W$ , and  $i \in A$ , then  $A - \{i\}$  is an optimal solution for  $S - \{i\}$  with weight at most  $W - w_i$  (cut and paste argument). Give a Dynamic Programming formulation for this problem based on the above idea and an algorithm to solve it in  $O(nW)$  time. More precisely, let  $c(i, w)$  be the optimal solution for the set  $\{1, \dots, i\}$  with a weight at most  $w$ . Obviously, if  $w_i > w$ , then  $i$  cannot be part of the solution and  $c(i, w) = c(i - 1, w)$ . Express  $c(i, w)$  in terms of  $c(i - 1, w)$  and  $c(i - 1, w - w_i)$  and describe how the dynamic programming table is computed, and finally how to obtain the solution itself from the table.

(b) (10 points) Assume that any two items  $i$  and  $j$  satisfy the following:  $w_i < w_j \Rightarrow v_i > v_j$ . In other words, lighter items are more valuable. Describe an efficient greedy way for solving the knapsack problem in this case with a running time independent of  $W$ . Use a “cut and paste” argument that justifies the greedy choice.

**Problem 4: Largest common circuit (20 points)**

Define a circuit as a logic consisting of **AND**, **OR**, and **NOT** gates connected together such that:

- every gate has a fan in of 2 (a gate can have at most two inputs)
- every gate has a fan out of 1 (the output of a gate can be input to at most one gate)
- no feedback loops

The problem: We have two circuits as described above, one with  $m$  gates and one with  $n$  gates, we would like to obtain the largest circuit (in terms of the number of gates) that is a sub-circuit of both.

(a) (5 points) Give a brute force algorithm for finding the largest common circuit of two circuits and analyze its running time.

(b) (15 points) Give a dynamic programming formulation of the problem and design an algorithm (based on dynamic programming) for finding the largest common circuit of two circuits that runs in  $O(mn)$  time.

**Problem 5: Spanning trees (20 points)**

(a) (10 points) Let  $w_{max}$  be the maximum weight of an edge in a spanning tree. A bottleneck spanning tree is a spanning tree whose  $w_{max}$  is minimum over all spanning trees. Use a “cut and paste” argument to show that a minimum spanning tree is a bottleneck spanning tree.

(b) (10 points) Consider the following divide and conquer algorithm for computing a minimum spanning tree: Given a graph  $G = (V, E)$ , partition the set  $V$  of vertices into two sets  $V_1$  and  $V_2$  such that  $|V_1|$  and  $|V_2|$  differ by at most 1. Let  $E_1$  be the set of edges that are incident only on vertices in  $V_1$ , and let  $E_2$  be the set of edges that are incident only on vertices in  $V_2$ . Recursively solve a minimum spanning tree problem on each of the two subgraphs  $G_1 = (V_1, E_1)$  and  $G_2 = (V_2, E_2)$ . Finally, select the minimum weight edge in  $E$  that crosses from  $V_1$  to  $V_2$ , and use this edge to unite the resulting two minimum spanning trees into a single spanning tree.

Either argue that the algorithm correctly computes a minimum spanning tree and give an implementation for it with an analysis of its running time, or provide a counter example for which the algorithm fails.