CMSC27200-1 Theory of Algorithms **Winter 2006**

Practice problems for midterm: February 7

Lecturer: Pedro Felzenszwalb *Scribe: Parinya Chalermsook*

1. Describe an algorithm which, given n numbers, output the list of k numbers whose sum is maximum. Your algorithm should run in $O(n \log k)$ time. *(***Hint:** *Use some data structure)*

2. Given a list of integers $a_1, a_2, \ldots a_n$, describe the algorithm that checks if there is a pair a_i, a_j that adds up to to exactly M in $O(n \log n)$ time. *(***Hint:** *Sort them first)*

3. Consider the following pseudocode that computes k-th power of x.

   power(x,k)

   1 if $k = 0$ then

   2 return 1

   3 else if k is even then

   4 return $sqr(power(x, k/2))$;

   5 else

   6 return $x \times sqr(power(x, k/2))$;

   (a) Write down the running time recurrence $T(k)$ where x is fixed.

   (b) Analyze the running time when k is the power of two.

   (c) Prove the same running time for any k.

4. Consider an array of numbers (possibly negative) $a[1], a[2], \ldots, a[n]$. Design an algorithm to find i, j such that $\sum_{k=i}^{j} a[k]$ is maximized. Your algorithm should run in linear time. (**Hint:** Dynamic Programming)

5. The following statement is true or false. Explain!!!

   To find a shortest path for a graph with negative weights, we can eliminate negative edges by adding the large enough constant to every edge so that they all become non-negative. Then use Dijkstra's algorithm.

6. **Minimum Spanning Trees**

   (a) Show that the edge with minimum weight in G is in some minimum spanning tree.

   (b) Let T be any **MST** and $e \in T$, show that there exists a cut $(S, V - S)$ such that, e is minimum edge accross this cut.

7. **MST with update**

   Let T be **MST** of $G = (V, E)$.

   (a) If we add a new edge e to G, our T might not be MST anymore. Can you obtain a new MST from the old one without recomputing everything? Do it in time $O(|V|)$.

   (b) Suppose we add a node v to G. Explain how to compute a new MST in $O(deg(v)|V|)$.

8. Suppose we have a directed graph $G = (V, E)$ without cycles. Given $s, t \in V$, We want to count the number of all possible paths to go from s to t. We are going to solve this problem using dynamic programming.

   (a) Prove that a directed graph G has no cycle if and only if we can renumber the vertices in G as $v_1, v_2, \ldots, v_n$ such that if $(v_i, v_j) \in E$ then $i \leq j$.

   (b) Suppose that we are given such ordering, write a recurrence of this problem.

   (c) Solve this problem using dynamic programming. What is the running time of your algorithm?

9. Given a list L of integers, $a_1, a_2, \ldots, a_n$, and an integer M, describe an algorithm that finds the largest subset of L whose sum is at most M. Your algorithm should run in linear time. You can assume that you are given an algorithm which finds median in linear time to be used as a subroutine. (**Hint:** Divide and Conquer).

10. Given a set S of n numbers, find an algorithm that partitions S into two disjoint subsets S_1 and S_2 whose union is S, that minimizes the difference of the sum of these groups, i.e. minimizing

$$|\sum_{x \in S_1} x - \sum_{x \in S_2} x|$$

    (**Hint:** Dynamic programming)

11. Given a directed graph $G = (V, E)$ with non-negative length function $l : E \to R$. Given a source s, recall that we can use Dijkstra's algorithm to find shortest paths from s to every nodes.

    Suppose that there is a further condition enforcing each edge as follows. We first consider the "length" of each edge as traveling time. We start from s at time $t = 0$. For each edge $e = (u, v)$, we have parameters $p(e)$ and $r(e)$, which define the possible time one can enter the edge at u to move to v (in other word it defines the time when the edge is opened). The edge is close at time $t = 0$. e is opened from time $r(e)$ to $r(e) + p(e) - 1$ and will be close again from time $r(e) + p(e)$ to $r(e) + 2p(e) - 1$. In general, the edge is open in time interval $t \in [r(e) + 2ip(e), r(e) + (2i + 1)p(e))$ for integer i and close when $t \in [r(e) + (2i + 1)p(e), r(e) + (2i + 2)p(e))$ for integer i. When the edge is close, anyone traveling on the edge continues to move, but no one can enter the edge until it is open again. We want to compute the shortest traveling time from s to every node.

    For example, consider the following graph.

$$s - - - - - - - - > u - - - - - - - - - > t$$

    The parameters are $l(s, u) = l(u, t) = 10, r(s, u) = 2, p(s, u) = 3$ and $r(u, t) = 0, p(u, t) = 8$. The shortest traveling time to u is 12 and to t is 26.

    Show how to modify Dijkstra's algorithm to solve this problem. And what is the running time of your algorithm?