6.854J / 18.415J Advanced Algorithms
Fall 2008

18.415/6.854 Advanced Algorithms

Problem Set 3

If you have any doubt about the collaboration policy, please check the course webpage.

Problem 1. Consider the symmetric traveling salesman path problem. We are given a set V of vertices, a (symmetric) distance $d(i,j) = d(j,i) \geq 0$ for every pair i, j of vertices, 2 special vertices s and t, and we would like to find a path from s to t passing through all vertices (a *Hamiltonian* path) and of minimum total length. This problem is NP-hard, and so we will not try to solve it exactly in polynomial time.

One simple heuristic is known as 2-OPT. Start from any ordering $v_1 = s, v_2, \cdots, v_n = t$ and remove 2 non-consecutive edges of the Hamiltonian path, say (v_i, v_{i+1}) and (v_j, v_{j+1}) where $i+1 < j$. Now, there is a unique way to form a new Hamilonian path by adding 2 other edges, namely (v_i, v_j) and (v_{i+1}, v_{j+1}). If this results in a path of shorter length, this so-called 2-swap is performed, and this is repeated until no more improvements are possible.

Now suppose we would like to create a data structure to maintain the ordering of the vertices on the path. For any vertex v, we would like to be able to find NEXT(v), the vertex following v on the way to it, and similarly PREVIOUS(v), the vertex preceding v (i.e. closer to s). The tricky thing is that when we perform a 2-swap, the vertices v_{i+1} to v_j are now visited in the *reverse* order (from v_i we go to v_j, then to v_{j-1} and continue all the way to v_{i+1}, and then continue at v_{j+1}). This means we would also like to have an operation REVERSE(v, w) that reverses the ordering from v to w; in our case above we would perform REVERSE(v_{i+1}, v_j). If we were maintaining the ordering as a doubly-linked list (with pointers *next* and *previous*), a REVERSE operation would require $\Theta(n)$ time in the worst-case. Show how to use splay trees to maintain the ordering and perform any sequence of m operations (NEXT, PREVIOUS, or REVERSE) in $O((m+n)\log n)$ time.

(If you augment the splay tree with additional information at every node, you must indicate how this information is maintained while performing operations.)

Problem 2. In lecture, we argued the static optimality property of splay trees by showing that the time T required for a splay tree to access element i $m_i \geq 1$ times for $i = 1, \cdots, n$ is within a constant of the time required by any static BST. In this problem, you need to argue that this time T for splay trees is

$$O\left(\sum_i m_i \left(1 + \log \frac{m}{m_i}\right)\right),$$

where $m = \sum_i m_i$ is the total number of accesses.

Problem 3. In the blocking flow problem (which arises in the blocking flow algorithm for the maximum flow problem), we are given a directed *acyclic* graph $G = (V, E)$, 2 vertices s and t and capacities on the edges. The goal is to find a flow f such that for every path P in E from s to t at least one of the edges of P is saturated (i.e. $f(v, w) = u(v, w)$).

1. Is the following argument correct?

   > A blocking flow f is maximum since, if we take S to be the set of vertices reachable from s in (V, E) by non-saturated edges, we get a cut $(S : \bar{S})$ whose value equals the blocking flow, and hence the blocking flow must be optimal.

   If the argument is fallacious, show a blocking flow which is not maximum.

2. Show how to find a blocking flow in a graph $G = (V, E)$ with n vertices and m edges in $O(m \log n)$ time. (Your solution can be quite short.)

(FYI, Dinitz showed that one can solve a maximum flow problem in a general directed graph by solving at most n blocking flow problems in directed acyclic graphs.)

Problem 4. A team of n members would like to travel a distance d from A to B as quickly as possible. All of them can walk and have also one scooter (which can carry only one person at a time) that they can use. For each person i ($1 \leq i \leq n$), we know his/her walking speed w_i and his/her speed s_i when travelling on the scooter. The goal is to find a way to bring all n people to destination so as to minimize the time at which the last person arrives. The scooter can be left by any member of the group on the side of the road, and picked up by anyone else of the group. Members of the group can also walk or use the scooter backwards (towards A) if that helps.

1. Consider the case where $n = 3$, $w_1 = w_2 = 1$, $s_1 = s_2 = 6$, $w_3 = 2$, $s_3 = 8$ and $d = 100$. Find the fastest way for everyone to travel the distance d.

2. For a general instance (general n and arbitrary speeds), write a linear program whose value is always a lower bound on the time needed for the n-person team to travel a distance d. This should be a small linear program; the number of variables and constraints should be $O(n)$ (and not dependent on d, or the number of 'legs' of the solution).

3. Use the linear programming routines in matlab (or any LP software like CPLEX) to compute your lower bound for the instance given in 1. (You don't have to use a linear programming software, provided you can exhibit an optimum solution, with a proof of optimality.)

If the bound you obtain is not equal to the value you found in 1., either improve your solution to 1., or find a stronger linear program in 2.

Matlab is available on athena, see `http://web.mit.edu/olh/Matlab/Matlab.html` for more info. Type `help linprog` for information on how to use the LP routine.

Problem 5. We will rederive the max-flow min-cut theorem from linear programming duality. Consider the maximum flow problem on a directed graph $G = (V, E)$ with source $s \in V$, sink $t \in V$ and edge capacities $u : E \to \mathbb{R}$. The max-flow problem is a linear program:

$$\max w$$

subject to

$$\sum_j x_{ij} - \sum_j x_{ji} = \begin{cases} w & i = s \\ 0 & i \neq s, t \\ -w & i = t \end{cases}$$

$$x_{i,j} \leq u_{i,j} \qquad (i,j) \in E$$

$$0 \leq x_{i,j} \qquad (i,j) \in E.$$

The variables are the x_{ij}'s (one per edge) and w.

1. Show that its dual is equivalent to:

   $$\min \sum_{(i,j)\in E} u_{ij} y_{ij}$$

   subject to

   $$z_i - z_j + y_{ij} \geq 0 \quad (i,j) \in E$$

   $$z_s = 0, z_t = 1$$

   $$y_{ij} \geq 0 \qquad (i,j) \in E.$$

2. Given a cut $(S : \overline{S})$ with $s \in S$ and $t \notin S$, show that a feasible solution y, z to the dual can be obtained of value equal to the capacity of the cut: $U(S : \overline{S}) = \sum_{(i,j)\in E} u_{ij} y_{ij}$.

3. Given any (not necessarily integral!) optimal solution y^*, z^* of the dual linear program and an optimal solution x^* of the primal linear program, show how to construct from z^* a cut separating s from t of value equal to the maximum flow.

   This shows the max-flow–min-cut theorem.