CS 38 Introduction to Algorithms **Spring 2014**

Final Exam Solutions

Posted: June 10

If you have not turned in the final, obviously you should not consult these solutions.

1. As in the greedy approximation algorithm for set cover, we repeatedly pick the set that covers the largest number of remaining uncovered items, until we've selected k sets. Let $U' \subseteq U$ be a subset of maximum size that can be covered by k sets.

   Let x_i be the number of elements covered after i rounds, and let $y_i = |U'| - x_i$. We note that $x_0 = 0$ and $y_0 = |U'| = OPT$.

   We claim that $y_i \le (1 - 1/k)^i OPT$. The proof is by induction on i. Clearly it holds for $i = 0$. Now in the i-th step, there are at least y_{i-1} elements of U' that remain uncovered, and yet k sets cover all of U'. Thus there is some set that covers at least $1/k$ fraction of the remaining elements. Thus in round i we cover at least $1/k$ of the y_{i-1} remaining elements, and so $y_i \le (1 - 1/k)y_{i-1}$ which by induction yields $y_i \le (1 - 1/k)^i OPT$.

   Using the suggested inequality, we have that $y_k \le OPT/e$, and because $x_k = OPT - y_k$ we find that $x_k \ge (1 - 1/e)OPT$. In other words, we achieve coverage of $OPT/(e/(e-1))$, as required.

2. We pick an arbitrary node r as the root, and perform a DFS from r. We will fill in the dynamic programming table as we finish processing each vertex in this order (this ensures that all children are processed before the parent).

   Define $M_0[u]$ to be the size of the maximum matching of the subtree rooted at u in which u is not matched, and $M_1[u]$ to be the size of the maximum matching of the subtree rooted at u in which u is matched. We observe that

   $$M_0[u] = \sum_{\text{children } v \text{ of } u} \max(M_0[v], M_1[v])$$

   since u is to be unmatched, so we just form the best possible matching at each subtree below u. And,

   $$M_1[u] = \max_{\text{children } v \text{ of } u} M_0(v) + \sum_{\text{children } v' \text{ of } u, v' \neq v} \max(M_0[v], M_1[v]),$$

   since we must choose one child v of u to match to u, and then v cannot be matched in its subtree, and the remaining subtrees are matched in the best possible way as above.

   We initialize $M[v, 0] = M[v, 1] = 0$ for leaves v and then fill in the table in post-order via the DFS as described above. At each node, we need to do work proportional to the number of children. Summing through the entire tree, we find that the total work in $O(E)$ as required. The size of the maximum matching overall is the maximum of $M_0[r]$ and $M_1[r]$.

To actually report the matching, we can store an extra value at each node u saying which of $M_0[u]$ or $M_1[u]$ was greater, and if it was $M_1[u]$, which of the children v we chose to match to u. With one further DFS, we can then traverse the tree and output those edges selected by the optimum matching.

3. In linear time, we can compute $A_{i,n/2}$ for $i = 1, 2, \ldots, n/2$ and $A_{n/2,j}$ for $j = n/2, n/2 + 1, \ldots, n$. We work out from the middle, so once we have computed $A_{i,n/2}$ we only need to add to it a_{i-1} to obtain $A_{i-1,n/2}$, and so on. As we produce these sums, we keep track of the maximum in each sequence; i.e., the i^* for which $A_{i^*,n/2}$ is maximum among $A_{i,n/2}$ and the j^* for which $A_{n/2,j^*}$ is maximum among $A_{n/2,j}$.

   Now, the optimum subsequence is either contained in the left half entirely, or in the right half entirely, or it straddles the middle. Each of the first two possibilities leads to a recursive call on a sequence half as long. The maximum subsequence that straddles the middle is clearly A_{i^*,j^*} which we have already computed.

   If $T(n)$ is the number of operations to solve the problem on an input sequence of n integers, we obtain the recurrence: $T(1) = O(1)$ and

$$T(n) = 2T(n/2) + O(n)$$

   which has as its solution $T(n) = O(n \log n)$ as required.

4. (a) The constraint matrix has all $0, 1, -1$ entries as is necessary for total unimodularity. The proof is by induction on the size of the submatrix. Consider an $i \times i$ square submatrix. If it involves the row associated with constraint $y_s = 1$ then it has at most one 1 in that row, and we can reduce to the $i - 1 \times i - 1$ case not involving that row (either by expanding around that 1 or finding that the row has all zeros and so the determinant is 0). The same holds for submatrices that include the row associated with constraint $y_t = 0$. The same argument also holds for submatrices that include the column associated with variable x_e, as this has exactly one 1 in it. So, the only case left is that we have a submatrix of the matrix whose columns are associated with the y_v and whose rows are associated with first set of constraints, one for each edge e. This matrix has a 1 in entry (e, v) iff edge e enters vertex v and a -1 in entry (e, v) iff edge e leaves vertex v. Thus there are at most two non-zero entries per row. Now if our submatrix has a row with zero or 1 non-zeros, then we can reduce to the $i - 1 \times i - 1$ case as before. Otherwise, there are exactly two non-zeros in each row, since in a given row, there is a $+1$ for edge $e = (u, v)$ entering v and a -1 for edge e leaving u (if either of these were missing, we wouldn't have two non-zeros in that row). In this case if we add all the rows of the submatrix, we get the zero vector, which implies it is singular, and has determinant 0.

   We conclude that every square submatrix has determinant $0, 1$ or -1 and so the constraint matrix is totally unimodular, as required.

   (b) We make the following observations about the structure of an optimal solution to the LP.

   First, if y_v is smaller than all of its neighbors, then we can decrease the LP value (make it better) by increasing it to equal the minimum of its neighbors. This is because for any edge $e = (u, v)$ the constraint on x_e is $x_e \geq y_u - y_v$ and for any edge $e = (v, w)$ the constraint on x_e is $x_e \geq y_v - y_w$, which is subsumed by $x_e \geq 0$ as long as $y_v \leq y_w$. Thus we can decrease the value of x_e by increasing the value of y_v, up to the minimum of its neighbors.

Similarly, if y_v is larger than all of its neighbors, then we can decrease the LP value (make it better) by decreasing it to equal the maximum of its neighbors. This is because for any edge $e = (u, v)$ the constraint on x_e is $x_e \geq y_u - y_v$ which is subsumed by $x_e \geq 0$ as long as $y_v \geq y_u$, and for any edge $e = (v, w)$ the constraint on x_e is $x_e \geq y_v - y_w$. Thus we can decrease the value of x_e by decreasing the value of y_v, up to the maximum of its neighbors.

Finally we note that for an edge $e = (u, v)$, changing both y_u and y_v by the same amount doesn't alter the constraint on x_e (namely $x_e \geq y_u - y_v$).

Let $U = \{u : y_u = \max_u y_u\}$ and let $V = \{v : y_v = \min_v y_v\}$. Then if there is a connected component $U' \subseteq U$ that excludes y_s and y_t, we can decrease the value of the y_u by the same amount for all $u \in U'$, and by the observations above this allows the LP value to decrease. Similarly, if there is a connected component $V' \subseteq V$ that excludes y_s and y_t, we can increase the value of the y_v by the same amount for all $v \in V'$, and by the observations above this allows the LP value to decrease. When neither operation is available, we must have that every connected component of U contains y_s and every connected component of V contains y_t and thus $\max_u y_u = 1$ and $\min_v y_v = 0$. We conclude that at the optimum, we have y_v between 0 and 1 for all v.

Then by total unimodularity the optimum occurs at a 0/1 assignment to the y_v variables. We can thus think of this assignment as an $s - t$ cut (since $y_s = 1$ and $y_t = 0$). An edge $e = (u, v)$ crossing the cut from the s side to the t side has the associated constraint $y_v - y_u + x_e \geq 0$, which implies $x_e \geq 1$. Similarly, and edge contained on one side of the cut or the other has the constraint $x_e \geq 0$ and an edge crossing in the reverse direction has the constraint $x_e \geq -1$ which is subsumed by the constraint $x_e \geq 0$ which is there for all edges. Thus we have exactly the constraints that $x_e \geq 1$ for edges crossing the cut, and $x_e \geq 0$ otherwise, and since the c_e are positive and we are minimizing, the minimum occurs when all of these constraints are tight, and the value is thus the value of the cut. Since every $s - t$ cut is possible with a 0/1 assignment to the y_v variables, the minimum of the LP is exactly the value of the min $s - t$ cut in the graph.