Illinois Institute of Technology
Department of Computer Science

Solutions to Third Examination

CS 430 Introduction to Algorithms
Spring, 2012

8am–10am, Tuesday, May 1, 2012
104 Stuart Building

Exam Statistics

51 students took the exam. The range of scores was 12–97, with a mean of 60.25, a median of 63, and a standard deviation of 19.12. Very roughly speaking, if I had to assign final grades on the basis of this exam only, 80 and above would be an A (8), 65–79 a B (17), 45–64 a C (15), 35–44 a D (7), below 35 an E (4).

Problem Solutions

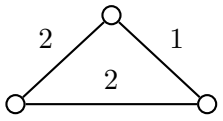
1. (a) Start at an arbitrary vertex, visiting the vertices using DFS and coloring them alternately red/green until we meet some visited (colored) vertex, if its color is different from its neighbor, continue; otherwise, the graph is not 2-colorable because we have found a cycle of odd length. A graph that can be 2-colored is called *bipartite*.

   (b) This is a minor embellishment of DFS and hence, like the original DFS, takes time $O(|V| + |E|)$.

   (c) If the algorithm is applied to an acyclic undirected graph, it will never find a back edge (that is, an edge to an already-colored vertex); that means it will never find an already-colored vertex, so it will always succeed in 2-coloring the graph. This proves that trees are bipartite.

2. This is a small part of Problem 23-3 on page 640 of CLRS.

   (a) Here is a small example:



   An MST must include the edge of weight 1, but the two edges of length 2 together form an MBST.

   (b) Suppose that T is an MST, but is not an MBST. Then consider the costliest edge e in T. e is costlier than in any MBST (or else T would be an MBST), in particular say MBST B. When you delete e from T you get two connected components (trees), say C_1 and

C_2. Now B is an spanning tree, so it must have at least one edge connecting a vertex in C_1 with a vertex in C_2; such an edge e' must have cost strictly less than the cost of e because e is costlier than any in an MBST. But then we could add edge e' to T and delete edge e, giving us a spanning tree of lower cost than T; that's impossible because T is an MST.

3. (a) The Bellman-Ford algorithm can detect the existence of a negative-weight cycle; this is almost what we need, but not quite. The weight of a cycle in the Bellman-Ford algorithm is the *sum* of the edge weights, but we need the product of the weights. Thus, following the hint, change each original edge weight $w(u, v)$ to $-\log w(u, v)$. Under this transformation, a negative weight cycle $v_1, v_2, v_3, \ldots v_k$ detected by Bellman-Ford would have weight

$$\sum_{i=1}^{k} -\log w(v_i, v_{i+1 \bmod k}) < 0,$$

or,

$$-\log \prod_{i=1}^{k} w(v_i, v_{i+1 \bmod k}) < 0,$$

and hence

$$\log \prod_{i=1}^{k} w(v_i, v_{i+1 \bmod k}) > 0,$$

so that

$$\prod_{i=1}^{k} w(v_i, v_{i+1 \bmod k}) > 1.$$

This means that the sequence of exchanges denoted by this cycle gives an overall exchange rate greater than 1, so that if we start with x currency units we end with strictly more than x currency units and we make money. If Bellman-ford says there are no negative-weight cycles, there is no way to make money by arbitrage.

(b) To include a 10% commission, we multiply all the original edge weights by 90%, that is, reweighting the edge (u, v) to $-\log 0.9w(u, v)$.

4. (a) Rephrase the problem as "Does graph G have a cycle of length greater than or equal to L?"

(b) Given a graph G and a cycle C, it is simple to verify in linear time that the length of C is at least L.

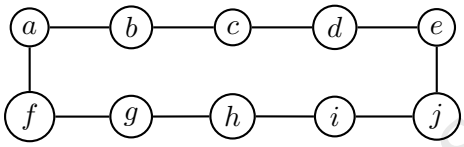
(c) Given a Hamiltonian circuit problem on an n-vertex graph, we could give every edge weight 1 and ask, does the graph have a cycle of length at least n? That is, if we could solve the Longest Cycle Problem in polynomial time, then P = NP (and we win a million dollars!)

(d) NP-complete means both NP-hard *and* in NP, so parts (b) and (c) prove the problem is NP-complete.

5. (a) To prove that the triangle inequality holds for the taxicab metric, consider three points $A = (x_1, y_1)$, $B = (x_2, y_2)$, and $C = (x_3, y_3)$. Then, because $|u - v| + |v - w| \geq |u - w|$,

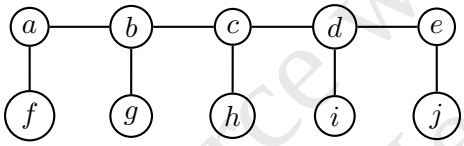
$$|AB| + |BC| = |x_1 - x_2| + |y_1 - y_2| + |x_2 - x_3| + |y_2 - y_3| \geq |x_1 - x_3| + |y_1 - y_3| = |AC|.$$

(b) In fact, *any* tour must cost at least n because every pair of points is separated by at least (taxicab) distance 1, a cycle of n vertices must be at least length n. Of course this means an optimal tour has at least length n. But there *is* a tour of that length,



which is therefore optimal.

(c) Following the hint, consider the spanning tree



The length of this spanning tree is $2k - 1 = n - 1$ (k teeth, plus the spine of lenth $k - 1$), so it must be an MST (we cannot connect n points in fewer than $n - 1$ edges and each edge has length at least 1).

What does the heursitic do starting from this MST? The heuristic begins with the walk around the periphery of the tree, visiting the vertices f, a, b, c, d, e, j, e, d, i, d, c, h, c, b, g, b, a, f; this walk has taxicab length $2n - 2 = 18$. Each time we apply the triangle inequality to get rid of vertex duplications *there is no decrease in the over cycle length* because we are using the taxicab distance. For example, when we change j, e, d to j, d (eliminating duplication when we go around the right end of the comb, the distance $\overline{je} + \overline{ed} = 2$ is the same as the resulting distance $\overline{jd} = 2$ in the taxicab world.