HW7 Solution Sketches

April 27, 2001

Bo graded the first 7 problems; Don graded the last 3.

24.1-1 Consider $G = (V, E)$ and consider the cut of G of the form $(\{u\}, V - \{u\})$. Let $A = \emptyset$. Trivially, A is included in some MST and $(\{u\}, V - \{u\})$ respects A. Since (u, v) is a light edge, since it is a minimum-weight edge in G. Thus, by Theorem 24.1, (u, v) is safe for A, so $A \cup \{(u, v)\} = \{(u, v)\}$ is contained in some MST.

24.1-2 Consider a weighted undirected graph $G = (V, E)$ with $V = \{1, 2, 3\}$, $E = \{(1, 2), (1, 3)\}$, $w(1, 2) = 1$, and $w(1, 3) = 2$. So G looks like this:

```
        1
  1   /   \   2
     /     \
    2       3
```

Clearly the only MST for G is all of G. Let $A = \emptyset$ and consider the cut $(\{1\}, \{2, 3\})$. Clearly the cut respects A and both (1,2) and (1,3) are safe for A. However, (2,3) is not a light edge crossing the cut.

24.1-3 Suppose (u, v) is contained in the MST T. Since T is a tree, removing (u, v) from T disconnects it into two pieces, say T_1 and T_2. Let V_i be the vertices contained in T_i, for $i = 1, 2$. Clearly (V_1, V_2) is a cut of the graph that respects $T - \{(u, v)\}$. Suppose that (u, v) is not a light edge crossing this cut. Then there is another edge (u', v') such that $w(u', v') < w(u, v)$. Let $T' = T - \{(u, v)\} \cup \{(u', v')\}$. Clearly the total weight of T' is less than that of T (since we have replaced (u, v) by (u', v')). T' is also a spanning tree, since there is a path in $T - \{(u, v)\}$ between any pair of edges in T_1 and between any pair of edges in T_2, and (u', v') gives us a bridge between T_1 and T_2. But this contradicts the assumption that T is a MST. So (u, v) must be a light edge.

Comment: Note that Theorem 24.1 only says that a light edge should be safe. You can't conclude directly from that theorem that if an edge is safe for a cut, then it must be a light edge for that cut (in fact, you proved the contrary in 24.1.2). If an edge is safe, then it is light for *some* cut, but that's what you need to prove.

24.2-1 Given T, sort the edges so that if there are ties, we always put edges in T ahead of the ones not in T. (That is, if there are two edges (u, v) and (u', v') such that $w(u, v) = w(u', v')$, and $(u, v) \in T$, $(u', v') \notin T$, we put (u, v) ahead of (u', v') in the list.

Comment: You can prove by induction that if you sort edges in this way, Kruskal will return T. For this homework, you got full credit as long as you gave the correct way of sorting the edges of G. No proof was required.

24.2-4 Kruskal's algorithm as discussed in the book (and in class) takes time $O(|E| \lg |E|)$:

- $O(|V|)$ to do the initialization
- $O(|E| \lg |E|)$ to sort the edges by weight
- $O((|V| + |E|) \lg^*(|V|) + |E|)$ to process the edges using UNION-FIND.

If all the edges in a graph are in the range 1 to $|V|$, then we can do the sort using counting sort, which takes time $O(|V| + |E|) = O(|E|)$. In that case, running time of the algorithm is $O((|V| + |E|) \lg^*(|V|) + |E|)$. (It's also OK to say $O(|E|\alpha(|E|, |V|))$, using the notation of Ackermann's function, as in the text.)

If all the edges in the graph are in the range 1 to W, where W is constant, then counting sort takes time $O(|E|)$ (since W is a constant, $O(|E| + W) = O(|E|)$, so again, the running time is $O((|V| + |E|) \lg^*(|V|) + |E|)$

[Grading:] 3 points for each part. -1 if $O(|E| + W)$ is not simplified to $O(|E|)$.

[Comments:]

1. $lg^*|E|$ (or $lg^*|V|$) *cannot* be taken to be a "constant for all practical purposes and thus ignored in a complexity expression". -1 every time you did this.
2. The third step of Kruskal's algorithm (involving the Unions and Finds) takes time $O((|V| + |E|) \lg^*(|V|) + |E|)$ and not $O(|E| lg |E|)$ (as given in the book). However, no points were deducted for this.

25.1-7 Consider the subgraph formed in this way: it consists of all the vertices in V reachable from s and contains an edge from u to v if there is a shortest path from s to v such that the last edge on that path is the edge from u to v. It's easy to show that this subgraph is a tree with s at the root. There are at most $|V|$ vertices on the tree altogether and, since it's a tree, there are at most $|V| - 1$ edges. Relax the edges (u, v) on this tree in order of increasing distance of u from s. That is, if $d[u] < d[u']$, then we relax all edges on the tree going out from u before the edges going out from u'. It doesn't matter what you do in case of ties. It is easy to check that this sequence of $|V| - 1$ relaxes results in $d[v] = \delta(s, v)$ for all vertices v on the tree.

Comment: It takes a little work to fill in all the details, but since this is a 3-point problem, you got full credit if you present the right idea.

25.2-1 Here's the result of running Dijkstra's algorithm with s as the source. At the end of each row, I've listed the vertex added to S.

k	$d(s)$	$\pi(s)$	$d(u)$	$\pi(u)$	$d(v)$	$\pi(v)$	$d(x)$	$\pi(x)$	$d(y)$	$\pi(y)$	S
1	0	NIL	3	s	∞	NIL	5	s	∞	NIL	s
2	0	NIL	3	s	9	u	5	s	∞	NIL	u
3	0	NIL	3	s	9	u	5	s	11	x	x
4	0	NIL	3	s	9	u	5	s	11	x	v
5	0	NIL	3	s	9	u	5	s	11	x	y

Now here it is with y as the source:

k	$d(s)$	$\pi(s)$	$d(u)$	$\pi(u)$	$d(v)$	$\pi(v)$	$d(x)$	$\pi(x)$	$d(y)$	$\pi(y)$	S
1	3	y	∞	NIL	7	y	∞	NIL	0	NIL	y
2	3	y	6	s	7	y	8	s	0	NIL	s
3	3	y	6	s	7	y	8	s	0	NIL	u
4	3	y	6	s	7	y	8	s	0	NIL	v
5	3	y	6	s	7	y	8	s	0	NIL	x

25.2-2 Almost any graph with negative-weight edges will cause problems for Dijkstra. For example, consider a directed graph with three vertices, s, t, and u, three edges, (s,t), (s,u), and (t,u), with $w(s,t) = 2$, $w(s,u) = 1$, and $w(t,u) = -2$. If we run Dijkstra's algorithm with source s, we first add s to S, with $d[s] = 0$. At that point, we have $d[t] = 2$ and $d[u] = 1$, so we add u to S. This doesn't affect $d[t]$. Finally, we add t to S. The final distances computed by the algorithm will be $d[s] = 0$, $d[u] = 1$, and $d[t] = 2$. But then $d[u] \neq \delta(s,u)$, since $\delta(s,u) = 0$.

The problem in the proof of Theorem 25.10 occurs on the last line on p. 529 and the material immediately following, where it says that "because ... all edge weights are nonnegative, we have that $\delta(s,y) \leq \delta(s,u)$. Because this is false, the rest of the argument (in particular, the conclusion that $d[y] \leq d[u]$ is false.

This is all you needed to say for the problem, but to understand it better, it is helpful to look at the rest of the proof more carefully and at the counterexample. In the second paragraph of the proof of Theorem 25.10, they take u to be the first vertex added to S for which $d[u] \neq \delta(s,u)$. In the counterexample above, u is indeed the first vertex added to S with this property. The proof then looks at the shortest path from s to u—that would be the path (s,t,u) in the counterexample above—and considers the first vertex y on that path that's in $V - S$ and the predecessor x of y on that path. In the counterexample, y is t and x is s.

The third paragraph of the proof starts with the claim that $d[y] = \delta(s,y)$ when u is inserted into S. That's actually true in the counterexample, since $d[t] = 2 = \delta(s,t)$ when u is inserted into S. They then want to get a contradiction by showing that $d[u]$ must be $\delta(s,u)$. This is where they need the fact that $\delta(s,y) \leq \delta(s,u)$. But in the counterexample above where $y = t$, it's not true that $\delta(s,t) \leq \delta(s,u)$, since $\delta(s,t) = 2$ and $\delta(s,u) = 0$.

[Grading: 3 points for a counterexample and 3 points for spotting where the problem lies in the proof of the theorem.]

Common mistakes:

1. Many students forgot to point out the problem with the proof of Theorem 25.10.

2. Some students claimed that $\delta(s,u) \le \delta(s,y) + w(y,u)$ doesn't hold if there are negative-weight edges. Unfortunately, $\delta(s,u) \le \delta(s,y) + w(y,u)$ always holds according to the definition of $\delta(s,*)$.

25.3-1 Here's the result of running the Bellman-Ford algorithm with y as the source. In each row, I've put in the result of doing one pass of relaxing the edges in lexicographic order, that is (u,v), (u,x), (u,y), (v,u), (x,v), (x,y), (y,v), (y,z), (z,u), (z,x).

k	$d(u)$	$\pi(u)$	$d(v)$	$\pi(v)$	$d(x)$	$\pi(x)$	$d(y)$	$\pi(y)$	$d(z)$	$\pi(z)$
1	8	z	7	y	9	z	0	NIL	2	y
2	5	v	6	x	9	z	0	NIL	2	y
3	4	v	6	x	9	z	0	NIL	2	y
4	4	v	6	x	9	z	0	NIL	2	y

Now here it is again using z as the source and changing the weight of (y,v) to 4. Notice we stil go through the edges in the same order.

k	$d(u)$	$\pi(u)$	$d(v)$	$\pi(v)$	$d(x)$	$\pi(x)$	$d(y)$	$\pi(y)$	$d(z)$	$\pi(z)$
1	6	z	∞	NIL	7	z	∞	NIL	0	NIL
2	6	z	4	x	7	z	2	u	0	NIL
3	2	v	4	x	7	z	2	u	0	NIL
4	2	v	2	x	7	z	-2	u	0	NIL
5	2	v	2	y	7	z	-2	u	0	NIL

Because the 5th round of relaxation changes the weights, the graph has a negative-weight cycle: (u,y,v).

[Grading:] 3 points for each table.

27.1-3 Call S the set of sources of a multiple-source, multiple-sink flow. To extend the value of a flow to a multi-source situation, sum over the sources, making $|f| = \sum_{s\in S}\sum_{v\in V} f(s,v)$. Other properties extend analogously, particularly flow conservation, which would be for all $u \in V - S \cup T$, where T is the set of sinks.

Take a multi-sink-source flow, f_{ms}, and add a supersource ss. Call the new flow f. Since the members of S now must obey flow conservation in f, and only a flow between each member of S (call it s_i) and ss has been added, it must now be that $f(s_i, ss)$ provides flow conservation. So $f(s_i, ss) = -\sum_{v\in V} f_{ms}(s_i, v)$. By skew symmetry, $f(ss, s_i) = \sum_{v\in V} f_{ms}(s_i, v)$. Since ss only has direct flow to members of S, the value of the new flow $|f| = \sum_{v\in V} f(ss,v) = \sum_{s_i\in S} f(ss, s_i) = \sum s_i \in S \sum_{v\in V} f_{ms}(s_i, v) = |f_{ms}|$, as we wanted.

"Vice versa": Now, take a single-source flow, f, and remove the source (call it ss), making all vertices with direct flow from the old source into sources, members of S. Since in flow f, the members of S obeyed flow conservation, and only flow from the source has been removed, $\sum_{v\in V} f_{ms}(s_i, v) = f(ss, s_i)$. So the value of the new flow f_{ms} is $\sum_{s_i\in S}\sum_{v\in V} f_{ms}(s_i, v)$, which equals $\sum_{s_i\in S} f(ss, s_i)$. Because ss only had direct flow to the members of S, this equals $|f|$.

27.1-6 Skew Symmetry: If $f_1(u,v) = -f_1(v,u)$ and $f_2(u,v) = -f_2(v,u)$, then $f_1(u,v) + f_2(u,v) = -f_1(v,u) - f_2(v,u)$, thus $(f_1 + f_2)(u,v) = -(f_1 + f_2)(v,u)$.

Flow conservation: Since the identities of the source and sink are identical in f_1 and f_2, the set $V - s, t$ is the same in f_1 and f_2. If $\sum_{x \in V} f_1(u, x) = 0$ and $\sum_{x \in V} f_2(u, x) = 0$, then $\sum_{x \in V} f_1(u, x) + f_2(u, x) = 0$, and $\sum_{x \in V} (f_1 + f_2)(u, x) = 0$.

Capacity constraint: $f_1(u, v) \leq c(u, v)$ and $f_2(u, v) \leq c(u, v)$ do **not** imply that $(f_1 + f_2)(u, v) \leq c(u, v)$. Simple counterexample: $f_1(u, v) = f_2(u, v) = c(u, v)$, but $(f_1 + f_2)(u, v) > c(u, v)$.

In short, flow conservation and skew symmetry hold, but the capacity constraint does not.

Extra Problem:

$$\begin{aligned}
f(X,Y) &= \textstyle\sum_{x\in X}\sum_{y\in Y} f(x,y) && \text{[by def. of f[X, Y]} \\
&= \textstyle\sum_{y\in Y}\sum_{x\in X} f(x,y) && \\
&= \textstyle\sum_{y\in Y}\sum_{x\in X} -f(y,x) && \text{[by skew symmetry]} \\
&= -\textstyle\sum_{y\in Y}\sum_{x\in X} f(y,x) && \\
&= -f(Y,X) && \text{[by def. of f(Y, X)]}
\end{aligned}$$