

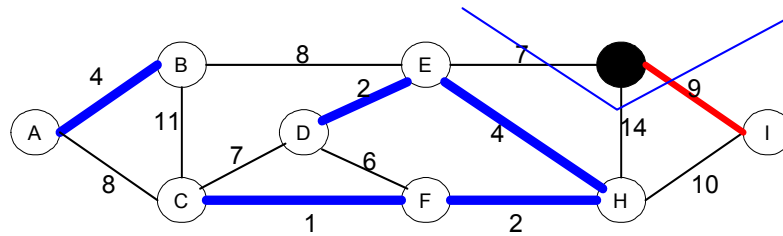
Assignment #6:

1. 23.1-2; 23.1-3; 23.1-4

Solutions:

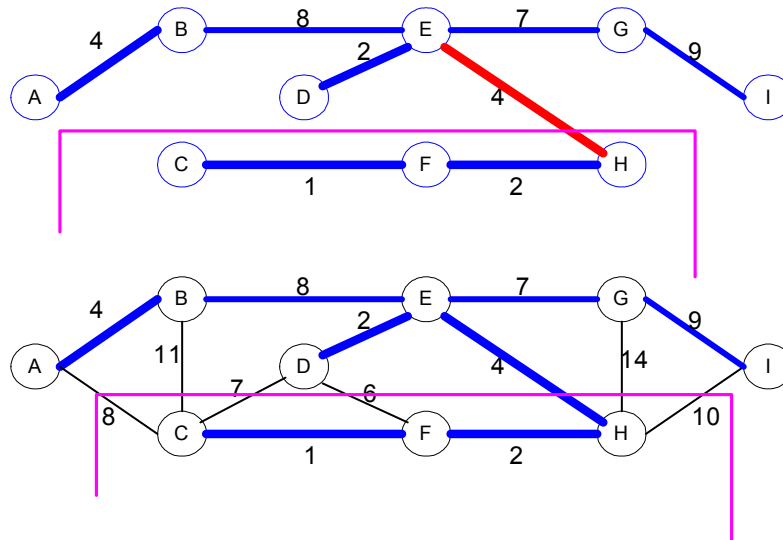
23.1-2:

The set A are the blue edges; edge (u, v) is the red edge. The cut is marked in blue.



23.1-3:

When an edge of a spanning tree is removed from the tree, it creates two disconnected components of the tree. The nodes corresponding to these two trees corresponds to a cut in the original graph. We do this with the edge (u, v) ($= (E, H)$) in the tree as shown below;



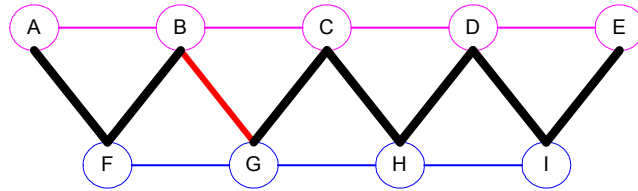
For this cut, the edge (u, v) is a light edge; else exchanging the light edge crossing this cut with edge (u, v) will provide a better tree which contradicts optimality of this tree.

23.1-4:

Take any graph with all weights equal and which has cycles. $\{(u, v) \in E : \exists(S, V - S) \ni (u, v) \text{ is a light edge crossing } (S, V - S)\} = E$. Since this contains a cycle, it is not a tree.

2. **23.2-8**

Consider the graph shown below: $V_1 = \{A, B, C, D, E\}$; $V_2 = \{F, G, H, I\}$. Minimal spanning tree on $G_1 = [V_1, E_1]$ is shown by pink edges and that on $G_2 = [V_2, E_2]$ is shown by blue edges. The red edge is minimum-weight edge that crosses the cut (V_1, V_2) . The weights of blue and pink edges is 1000; that of the red edge is 1; that of remaining edges is 2. Professor Toole's spanning tree consists of pink, blue, and red edges. The unique minimum weight spanning tree is black and red edges. So The algorithm fails.



3. **23-4**

- (a) This is precisely Kruskal's Algorithm B. It works. I am leaving the proof to you. Implementation: Keep a spanning tree in T at all times. This will show if the edge under consideration creates a circuit with other edges or not when the edge is not a tree edge. If the edge under consideration is a tree edge, then its removal disconnects the tree and we need to check if there is any other edge across this cut.
- (b) Do the graph in 23.2-8 in this order: pink, blue, red, black. The tree you get is Professor Toole's which is not optimal as shown above. Implementation: Do the same as for Kruskal's Algorithm A.
- (c) This algorithm works because it has removed maximum weight edge of a circuit (one of if there are many), it produces the same result as Kruskal's algorithm B. Implementation: Do the same for Kruskal Algorithm A but modified as follows. When $\text{FIND-SET}(u) = \text{FIND-SET}(v)$, we need to trace the first common ancestor and the path to it from both nodes. This gives the cycle and now we can remove the edge of maximum weight. Clever way is to also keep maximum edge weight while tracing the common ancestor.

- 4. Consider the minimum spanning tree problem on a connected undirected graph. Show that Boruvka algorithm produces a spanning tree if all

weights are distinct (equally if they are totally ordered). Give a counterexample to show that if edges have equal weight, we may not get a spanning tree.

Solution:

Proof by induction on the number of steps in the evolution of the algorithm. Initially since the set of blue edges is empty, we are O.K. Now suppose that the algorithm has no blue cycles before step k , and at step k creates one or more cycles. Let one of these cycles connect trees T_1, T_2, \dots, T_p . Note this requires p edges at this step of the Boruvka algorithm. Among the edges chosen at this step one of them is the least weight edge (or has the least index in the ordering). This edge must be chosen by both the trees which its end points connect. This implies that Boruvka algorithm chooses no more than $p - 1$ edges at this step. This is a contradiction to the above. Thus, there are no cycles in the final tree of this algorithm.

For the second part, consider the three node complete graph with equal weights. Each node selects a different edge and we get a cycle.

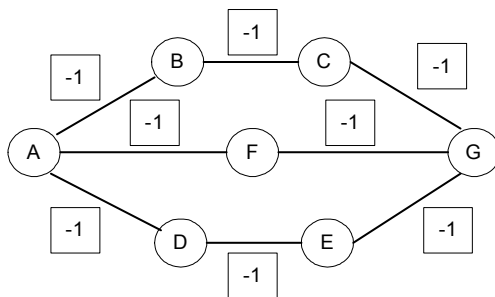
5. Let $G = [V, E]$ be a directed graph and $w[e]$ ($= w[u, v]$ if $e = (u, v)$) (not necessarily nonnegative) be weight on edge $e \in E$. Let K be a constant satisfying the condition that

$$r[e] = w[e] + K > 0 \quad \forall e \in E$$

- (a) Give an example to show that the shortest path in G from s to all other nodes depends on whether we use the weights $w[e]$ or $r[e]$.

Solution:

Consider the graph shown below; all edges are directed from left to right. Let $K = 3$. The shortest paths for (A, G) on $w[e]$ is either $A - B - C - G$ or $A - D - E - G$. For the same pair on $r[e]$ it is $A - F - G$.



- (b) We know that lengths of the shortest path from s satisfy the relations:

$$\delta(s, v) \leq \delta(s, u) + w(u, v) \quad \forall (u, v) \in E$$

Suppose $\{x_v\}; v \in V$ satisfy the relations:

$$x_v \leq x_u + w(u, v) \quad \forall (u, v) \in E$$

Does this imply $x_v = \delta(s, v)$ for all $v \in V$?

Solution:

NO. For example if $w(u, v) \geq 0$ we can take $x_v = 0$ for all $v \in V$.

- (c) Let $r(u, v) = w(u, v) + x_u - x_v$ for $(u, v) \in E$ with the above $\{x_v\}$. Now $r(u, v) \geq 0$ for all $(u, v) \in E$. So we can apply Dijkstra algorithm to the problem with r . Are these paths also shortest paths with w ?

Solution:

Let $P_{s,t}$ be any path from s to t .

$$\begin{aligned} \sum_{(u,v) \in P_{s,t}} r(u, v) &= \sum_{(u,v) \in P_{s,t}} \{w(u, v) + x_u - x_v\} \\ &= x_s - x_t + \sum_{(u,v) \in P_{s,t}} w(u, v) \end{aligned}$$

Since $x_s - x_t$ is independent of $P_{s,t}$, a shortest path with r is also a shortest path with w .

- (d) In case (c), do we get to do less work in determining the shortest paths from s to all other nodes?

Solution:

Yes, since Bellman-Ford is $\Theta(|E| |V|)$ and Dijkstra algorithm is $O(|E| \lg |V|)$ or lower depending on the implementation. .

- (e) In case (c), there was no mention of negative cycles in the problem with w – how come?

Solution:

There can be none since $r(u, v) \geq 0$ and hence there is no negative cycle with r . But if there is no negative cycle with r then there is no negative cycle with w either since for any cycle C :

$$\begin{aligned} \sum_{(u,v) \in C} r(u, v) &= \sum_{(u,v) \in C} \{w(u, v) + x_u - x_v\} \\ &= \sum_{(u,v) \in C} w(u, v) \end{aligned}$$

6. 26.2-9

Solution:

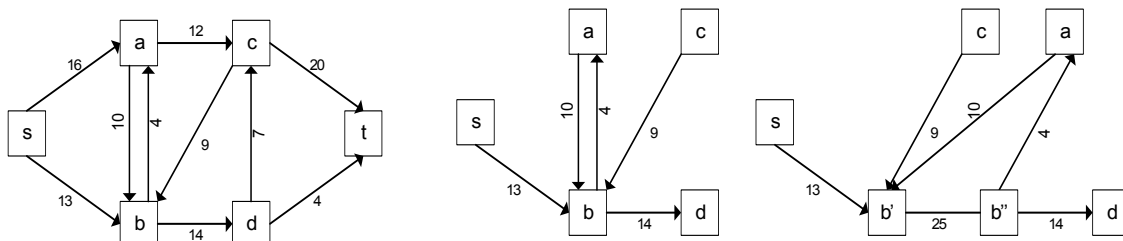
For a given pair of vertices s and t , let the maximum number of edge disjoint paths in G between these two nodes be $p_{s,t}[G]$. Edge connectivity of G equals $\min_{(s,t)} \{p_{s,t}[G]\}$. $p_{s,t}[G]$ can be obtained by doing a maximum flow problem on $G = [V, E]$ with edge capacities equal to 1 and s as the

origin and t as the destination (or t as origin and s as destination). Please note that by max-flow-min-cut theorem, $p_{s,t}[G]$ equals the capacity of a min-cut separating s and t . In this case, it is the number of edges in such a cut. Our method is to determine $p_{s,v}[G]$ for all $v \in V - \{s\}$. This requires $|V| - 1$ max-flow problems on the same flow network described above. This yields min-cuts separating s from each of the remaining vertices in G . One of these must be the overall min-cut in G since in such a cut s is on one side and there are other nodes on the other side. Let an overall mincut be $(S, V - S)$ with $s \in S$ and $v \in V - S$. Any min-cut separating s from v is also an overall min-cut. The value of this cut equals edge connectivity of the graph.

7. 26-1

Solution:

- (a) Given a flow network $G = [V, E]$ with node and edge capacities, recall G is a directed graph. To convert this to a problem on G' with edge capacities only we do the following trick at each node called "node-splitting": In this graph shown below node b has capacity 25;



This results in at most $2|V|$ vertices and $|V|$ additional edges in G' .

- (b) Think of the dark nodes as origins in a multiple origin setting and all nodes in rim as destinations. All edge and node capacities are equal to 1. Solve a maximum flow problem and check if the flow equals the number of dark nodes.

8. 26-4

Solution:

- (a) Given a maximum flow f in G with original capacities, we get a residual graph G_f . Make the change in capacity of edge (u, v) and increase it by 1 (if this edge is not in G_f introduce it with a capacity of 1). Now do BFS and if you find a path to t from s , this path flow is increased by 1. Increasing the capacity of an edge by 1 can make the new flow increase in value by at most 1. This part follows from max-flow-min-cut theorem. Since BFS takes $O(|E| + |V|)$ -time, we are done.

- (b) If in the optimal solution f in the original problem, $f(u, v) < c(u, v)$ in G , the original solution is still optimal. If not, in G_f , we try to find by BFS whether there is a path from u to v . If yes, increase flow on this path by 1 and decrease flow on edge (u, v) by 1. The total value of flow does not change. If not, do a BFS on $\hat{G} = [V, \hat{E}]$ where $\hat{E} = \{(u, v) : f(u, v) > 0\}$ starting at origin. This finds a path $p_{s,u}$ from s to u with positive flow. Do another BFS from v and this produces a path $p_{v,t}$ from v to t with positive flow. Now reduce flows on these paths and the edge (u, v) by 1. A constant number of BFS takes $O(|E| + |V|)$ -time and we are done.