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 Borden’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.

\[\text{A} \rightarrow \text{B} \rightarrow \text{C} \rightarrow \text{D} \rightarrow \text{E} \]
\[\text{F} \rightarrow \text{G} \rightarrow \text{H} \rightarrow \text{I} \]

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 Borden’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 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. Will be done in class

5. Will be done in class
6. 26.2-9

Solution:
Consider the graph all of whose edge capacities are equal to 1. The first figure shows \( f \) and the second shows \( f' \). It should be clear that \( f \uparrow f' \) is not feasible.

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;

(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 $G = [V, \vec{E}]$ where $\vec{E} = \{(u, v) : f(u, v) > 0\}$ 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.