Pre-tutorial questions

Do you know the basic concepts of this week's lecture content? These questions are only to test yourself. They will not be explicitly discussed in the tutorial, and no solutions will be given to them.

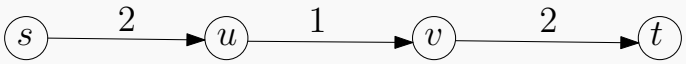
1. Flow network $G = (V, E)$.
   (a) Is G directed or undirected?
   (b) What does it mean that an edge has a capacity?
   (c) What is a flow f in G?
   (d) Why is the flow of an edge bounded by the capacity of an edge?
   (e) What are the capacity and conservation constraints?
2. Min Cut (A, B) of G.
   (a) What is a cut (A, B) of G?
   (b) What is the capacity of a cut?
   (c) What is the flow of a cut?
3. Min cut vs. max flow
   (a) Why is the value of a cut an upper bound on the maximum flow?
   (b) What does the Min Cut - Max Flow theorem state?
4. Ford-Fulkerson
   (a) The Ford-Fulkerson algorithm iteratively increases the flow. How is this done in each iteration?
   (b) Can you upper bound the number of iterations performed by the Ford-Fulkerson algorithm?

Tutorial

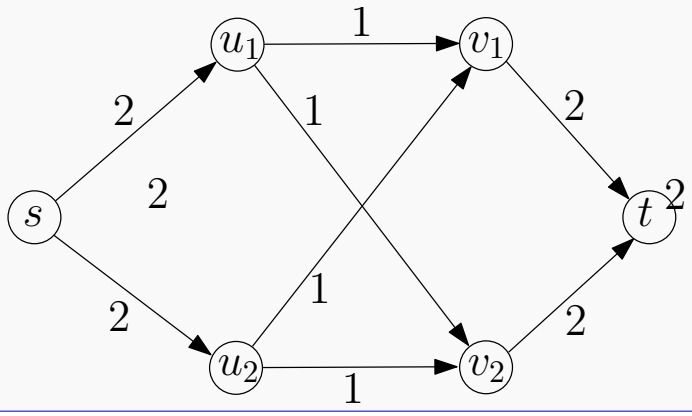
Problem 1
Let $G = (V, E)$ be an arbitrary flow network, with source s, sink t, and edge capacities $c : E \to \mathbb{Z}^+$. Decide whether each of the following statements are true of false. If it is true, give a short explanation. If it is false, give a counterexample.

1. If f is a maximum s-t flow in G, then f saturates every edge out of s; that is, $f(e) = c(e)$ for each edge e coming out of s.
2. Let (A, B) be a minimum s-t cut with respect to c. Define new capacities $\hat{c}(e) = c(e) + 1$ for each $e \in E$. Then (A, B) is a minimum s-t cut with respect $\hat{c}$.

Solution: 1. This is False. The following graphs is a counter example.



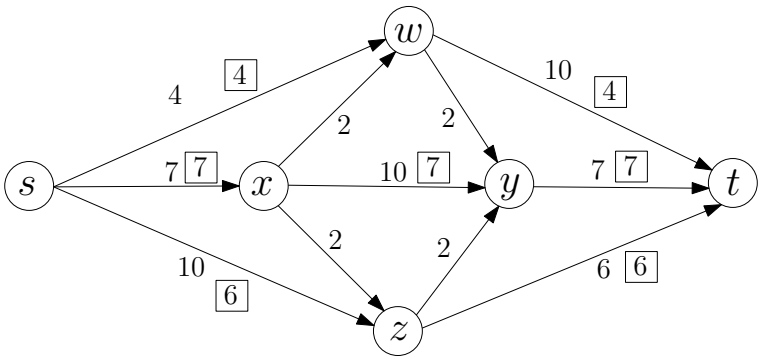
2. This is False. The following graph is a counter example. The cut $(\{s, u_1, u_2\}, \{v_1, v_2, t\})$ is minimum before the update but not after.



Problem 2

The figure below shows a flow network on which an $s - t$ flow is shown. The capacity of each edge appears as a label next to the edge, and the numbers in boxes give the amount of flow sent on each edge. (Edges without boxed numbers have no flow being sent on them.)

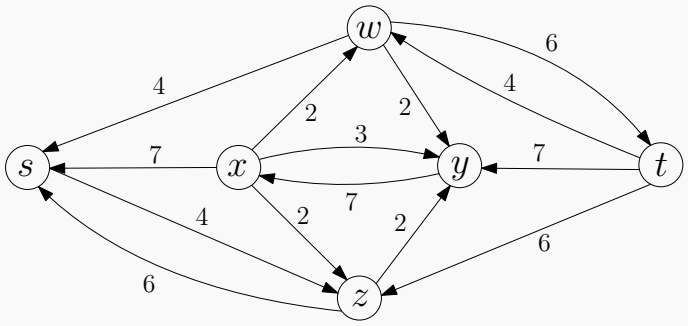
1. What is the value of this flow?
2. Is this a maximum $s - t$ flow in this graph? If not, find a maximum $s - t$ flow.
3. Find a minimum $s - t$ cut. (Specify which vertices belong to the sets of the cut.)



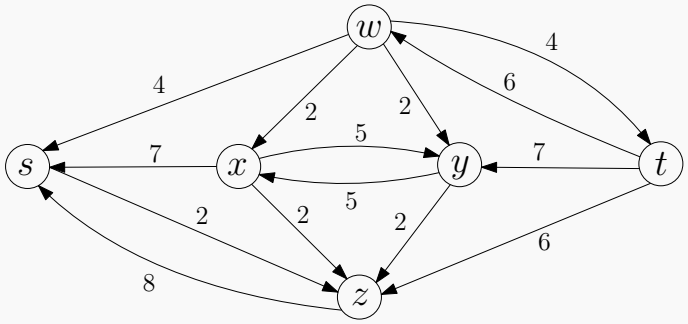
Solution:

1. The value of this flow is 17. (The total amount of flow passing from s to t.)

2. No, this is not a maximum flow. We can find a maximum flow by looking at the residual network. In this case, from the given flow, we can get a max flow with only a single augmenting path, as shown in the picture below. So the maximum flow has value 19. The residual network for the given network is as shown in the figure:



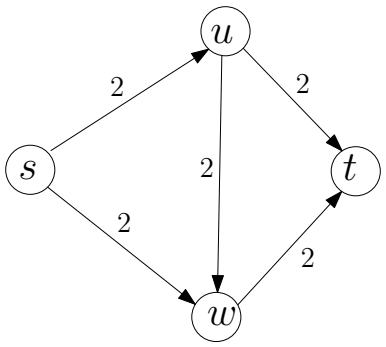
   Augment flow with path s, z, y, x, w, t. Capacity of this path is 2 so send 2 new units of flow on this path. The new residual flow shows that no augmenting paths exists, hence, the maximum flow is 19.



3. As explained in class, we can find a minimum cut using the residual network for the final maximum flow. Starting at s, find all vertices that are reachable from s in the residual network. This forms one set in the minimum cut. The other vertices form the other set part of the cut. So a minimum cut in this case are the two sets $\{s, z\}$ and $\{x, w, y, t\}$. You can check that the capacity of this cut (i.e. the total capacity of the edges from $\{s, z\}$ to $\{x, w, y, t\}$, consisting of the directed edges $(s, w), (s, x), (z, y)$, and (z, t)) is 19, as the Max Flow/Min Cut Theorem guarantees.

Problem 3

Find all minimum $s - t$ cuts in the following graph. The capacity of each edge appears as a label next to the edge.



Solution: First, there are exactly four possible cuts in the graph. These are (recalling that the cut must separate s and t):

(a) $\{s\}, \{u, w, t\}$,

(b) $\{s, u\}, \{w, t\}$,

(c) $\{s, w\}, \{u, t\}$, and

(d) $\{s, u, w\}, \{t\}$.

It is a simple matter to determine the capacity of each of these cuts, and find those that are minimum cuts. In the order given above, the capacities of the cuts are 4, 6, 4, and 4. So we see that there are three minimum cuts in this graph, namely

(a) $\{s\}, \{u, w, t\}$,

(b) $\{s, w\}, \{u, t\}$, and

(c) $\{s, u, w\}, \{t\}$.

Problem 4

Design a linear time algorithm that given a flow f verifies that f is maximum. If the flow is maximum your algorithm should output "yes", otherwise, it should output "no". No other action is required.

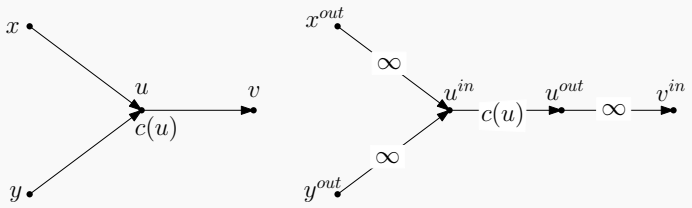
Solution: Given the input graph G and f, construct G_f in linear time and check if there is an s-t path. If there is an s-t path then we could use the path to push one extra unit of flow, so f is not maximum. Otherwise, we reach the termination criterion of the Ford-Fulkerson algorithm, which means that f is maximum.

Problem 5

Consider a generalization of the maximum flow problem where vertices have capacities. An instance is defined by a directed graph $G = (V, E)$, a pair of vertices s and t, and vertex capacities $c : V \to Z^+$. A flow $f : E \to \mathbb{Z}^+$ is feasible if $f^{out}(u) \leq c(u)$. The objective is to find a maximum feasible s-t flow.

Design an efficient algorithm for this problem.

Solution: The high level idea is to enforce the vertex capacity with a standard edge capacity. To achieve this, we split every vertex u into two nodes (u^{in} and u^{out}) and connect them using and edge with capacity $c(u)$. For every edge (u, v) in the input graph, we create an edge (u^{out}, v^{in}) with capacity $+\infty$. The following picture shows how a vertex u and its incoming and outgoing edges are transformed by the reduction.
Let $G = (V, E)$ be the input graph and G' the graph obtained by the reduction. Let f be an s-t flow in G obeying the vertex constraints. We define a flow f' in G' as follows. For each $(u, v) \in E$ set $f'(u^{out}, v^{in}) = f(u, v)$; for each vertex $u \in V$ set $f'(u^{in}, u^{out}) = f^{out}(u)$. It is easy to show that this is a feasible s^{out}-t^{in} flow in G' with the same value as f.
Similarly, given a feasible flow s^{out}-t^{in} flow f' in G', we can find a feasible s-t flow in G with the same value as f'. Therefore, solving the edge-capacitated maximum flow problem in G' is equivalent to the vertex-capacitated maximum flow problem in G.



Problem 6

Consider a generalization of the minimum cut problem where we want to separate two sets of vertices. An instance is defined by a graph $G = (V, E)$, a pair of disjoint subsets of vertices $S, T \subseteq V$, and edge capacities $c : E \to \mathbb{Z}^+$. The objective is to find a minimum capacity cut (A, B) such that $S \subseteq A$ and $T \subseteq B$.

Design an efficient algorithm for this problem.

Solution: Let G' be the graph obtained from G by adding two new vertices s and t and edges $\{(s, u) : u \in S\} \cup \{(v, t) : v \in T\}$ with capacity ∞. Let (A', B') be an s-t cut in G'. If $S \subseteq A'$ and $T \subseteq B'$ the capacity of (A', B') equals the capacity of $(A' - s, B' - t)$; otherwise, the capacity is $+\infty$. Therefore, there is a one-to-one correspondence between minimum s-t cuts in G' and minimum S-T cuts in G.

Problem 7

We want to wirelessly connect a set of mobile computers to a set of stationary base stations. Each computer u has an effective transmission radius r_u. Suppose we know the spatial location of each computer and each base station. Our goal is to assign each computer to some station within its transmission radius. Define the load of a station to be the number of computers assigned to it.

Your task is to design a polynomial time algorithm that will produce a feasible computer-station assignment minimizing the maximum station load.

Solution: We reduce the problem to a maximum flow computation. Consider a bipartite graph where the vertices on the left correspond to computers and vertices on the right correspond to base stations. For every computer-station pair (u, v), we check if v if within the transmission radius of u; if so, we create a directed edge with capacity 1 from u to v. In addition, we create a node s, which we connect to all the computers with an edge of capacity 1, and we create a node t, which we connect from all the stations with an edge of capacity X, where X is a parameter that we can tune.
First notice that any integral flow that sends n units of flow effectively encode a feasible assignment with maximum load X or less. Therefore, the problem now is to find the smallest X such that the max flow instance has value n. For this we can use binary search in the range $[1, \ldots, n]$. Therefore, with only $\log_2 n$ maximum flow computation we can find the feasible assignment with minimum maximum load.