Graph theory - problem set 7

October 31, 2019

1. Show that Hall's theorem can be derived from Knig's theorem.

   Hall from Knig: Suppose there is no matching that matches A in its entirety. Using Knigs Theorem, one can show that there is a vertex cover C with $|C| < |A|$ vertices. There is no edge between $A - C$ and $B - C$ because otherwise such an edge would not be covered by the cover C. This means $N(A-C) \subset C \cap B$. But then we get $|N(A-C)| \leq |C \cap B| = |C| - |C \cap A| < |A| - |C \cap A| = |A - C|$, which is a contradiction with Halls condition.

2. Show that Knig's theorem can be derived from Hall's theorem.

   Knig from Hall: Let C be a minimum cover. Our goal is to show that G has a matching of size $|C|$. Since there are no edges between $A - C$ and $B - C$ from a similar argument as in the first part, every edge either goes from $C \cap A$ to B or from $C \cap B$ to A. We can thus define two disjoint bipartite subgraphs of G: a graph H_1 with vertex sets $C \cap A$ and $B - C$, and a graph H_2 with vertex sets $C \cap B$ and $A - C$. We will show that H_1 has a matching that matches $C \cap A$ and H_2 has a matching that matches $C \cap B$. Their union would be a matching of size $|C|$, which would end the proof.

   Consider H_1 and let $D \subset C \cap A$. If $|N_{H_1}(D)| < |D|$, then we could replace $D \subset C$ by $N_{H_1}(D)$ to get a smaller vertex cover, contradicting our minimality assuption on C. Indeed, if an edge e is covered by $v \in D$, then either it is in H_1 and also covered by a vertex in $N_{H_1}(D)$, or it is not in H_1 and incident to a vertex in $C \cap B$, so it is covered anyway. This proves that Halls condition holds for H_1 and that there exists a matching in it that matches $C \cap A$. Similarly, we can prove the existence of a matching in H_2 that matches $C \cap B$. Thus the union of our matchings has cardinal $|C|$, which lets us conclude.

3. (*Flow Decomposition*) Show that we can decompose any feasible flow on a network $G = (V, A)$ into at most $|A|$ paths and cycles.

   Solution: Given a flow f, we can use the following algorithm to decompose it into paths and cycles.

   Input: A graph $G = (V, A)$ and a flow function f on this graph.

   Output: A set P of paths in G and a set C of cycles in G such that the union of these paths and cycles forms the flow f.

   (a) Find a positive flow s-t-path. If there aren't any, then the flow is zero everywhere.
   (b) Decrease the flow on this path until an edge has flow 0.
   (c) Add this path to P.
   (d) Repeat a-c until there is no more positive flow s-t-path in G.
   (e) If there exists still edges with strictly positive flow, find a positive flow cycle by following strictly-positive flow edges until you come back to the starting edge.
   (f) Decrease the flow along this cycle until one of its edges has flow 0.
   (g) Add this cycle to C.
   (h) Repeat e-g until the resulting flow is the zero flow and return P and C.

   The following algorithm works because of basic flow properties, and returns less than $|A|$ elements because to each path and each cycle corresponds the single edge where the flow hit 0 when decreasing, thus $|A| \geq |P| + |C|$.

4. (*Halloweeeeen*) On the 31st of October, enthusiastic partying at a prominent Swiss university has unfortunately resulted in the arrival at the university's medical clinic of 169 students in need of emergency treatment. Each of the 169 students requires a transfusion of one unit of blood. The clinic has supplies of 170 units of blood. The number of units of blood available in each of the four major blood groups and the distribution of patients among the groups in summarized below:

Blood type	*A*	*B*	*O*	*AB*
Supply	46	34	45	45
Demand	39	38	42	50

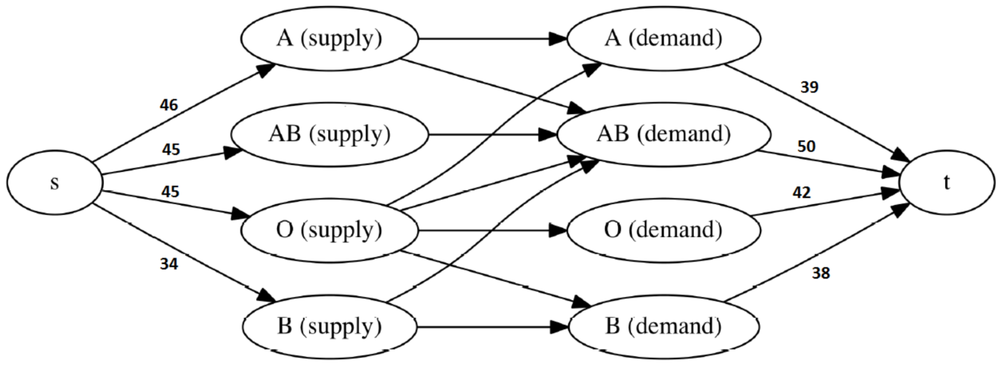
A patients can only receive A or O blood. B patients can only receive B or O blood. O patients can receive only O blood. AB patients can receive any of the four blood types.

(a) Give a max-flow formulation that determines a distribution that satisfies the demands of a maximum number of patients. You should draw a directed graph with edge capacities such that a feasible flow corresponds to a feasible choice for the transfusion.

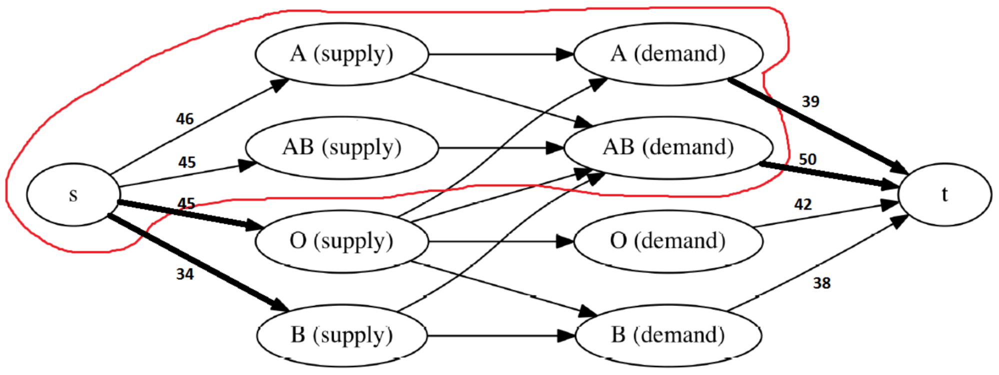
(b) Find a cut in your graph of value smaller than 169. Use it to give an explanation of why not all of the patients can receive blood from the available supply. Try to make your explanation understandable to the clinic's staff, who do not know network flow theory.

Solution:

We formulate the problem as follows: there are two nodes for each blood type, representing the supply and the demand, and there is a source node s and a sink node t. The arcs represent which type of blood in the supply can be donated to which group. The capacities represent the supply or the demand of a blood type. Note that in the edges between a supply and a demand node the capacity is set to ∞. This is an arbitrary choice. On one hand, this represents the fact that we dont constrain the amount of blood that can be given to a group. On the other hand, this will ensure that the minimum cut will not cross any of these arcs, which makes easier to look for a small cut in the graph. To find such a cut we can use the algorithm seen in class to compute the maximum flow (of value 168) and then find a cut of the same capacity.



The cut gives an intuition of why we cannot satisfy the totality of the demand. The problem lies in the blood types O and B, which are out of the cut. This means that the flow from s to O, B fills all the capacity, but it is still not enough. In fact, the demand for O is 42 units, that can only come from



the supply of O, which has 45 units. On the other hand, the demand for B is 38 and the supply of B blood is only 34, hence 4 units should come from O supply, but ther¡e are only 3 left. Alternatively, we could have come up with the latter explanation and use the intuition behind it to derive the cut.

5. Let $G = (V, A)$ be a directed graph and let us fix an origin node $s \in V$ and a destination node $t \in V$. We define the *connectivity* of a graph as the maximum number of vertex-disjoint (besides s and t) directed paths from s to t. We define the *vulnerability* of the graph as the minimum number of vertices (besides s and t) that need to be removed so that there exists no directed path from s to t. Prove that connectivity is equal to vulnerability.

   Solution: Some min-flow max cut argument. (Menger's theorem)

6. The *Edmonds-Karp algorithm* is the same as Ford-Fulkerson, except it always chooses an augmenting path of shortest length (this can be found using BFS). In this problem we show that this small requirement can significantly improve the running time of the algorithm: no matter what the capacities are (large or irrational), it finds a max flow in polynomial time (in $O(|V||E|^2)$ steps, to be precise).

   (a) Let $l_a(v)$ denote the length of the shortest augmenting path from s to v after a steps. Show that if the algorithm chooses the augmenting s-t path $v_0 \dots v_k$ after step a, then $l_a(v_{i+1}) = l_a(v_i) + 1$ for every $0 \leq i \leq k - 1$.

   (b) Prove that these distances cannot decrease, i.e., $l_a(v) \geq l_{a-1}(v)$ for every a and v.
   [Hint: for fixed a, choose a v such that $l_a(v) < l_{a-1}(v)$ and $l_a(v)$ is minimum]

   (c) Show that if $l_i(t) = l_a(t)$ for some $i \geq a$, then after step i the algorithm saturates an edge uv such that $l_a(v) = l_i(v) = l_i(u) + 1 = l_a(u) + 1$ and does not unsaturate any other edge with this property. We get $l_{a+|E|}(t) > l_a(t)$ by noting that there are at most $|E|$ edges to saturate.
   [Hint: Look at the augmenting path. Use (b) on l_i and on the analogously defined $m_i(v)$ for the shortest length of an augmenting v-t path (i.e., that $m_a(v) \geq m_{a-1}(v)$)]

   (d) Deduce that the algorithm stops after at most $|V||E|$ improvements. We get a $O(|V||E|^2)$ running time using a $O(|E|)$-time BFS to find augmenting paths.

   Solution:

   (a) We know that $l_a(v_{i+1}) \leq l_a(v_i) + 1$ by taking the shortest path from s to v_i and the edge $v_i v_{i+1}$. The other inequality comes from the fact that $v_i v_{i+1}$ lies on the shortest s-t path and a strict inequality would give a shorter s-t path.

   (b) Suppose by contradiction that this is not true. Fix such an a, and let v be such that $l_a(v) < l_{a-1}(v)$ and $l_a(v)$ minimal among those. Let p_a be the path along which we augmented to get from step $a-1$ to a. Observe that v is the first vertex on a new shortest path, which appeared by augmenting along p_a. Thus v is on p_a and by (a), we know that $l_a(v) = l_a(u) + 1$ where u is the preceding vertex on p_a. Contradiction because $l_a(v)$ was supposed to be the minimal such vertex.

   (c) Let the augmenting path be p_i. Since p_i saturates uv this is an edge on a shortest s-t path between step a and step i. So by (a) we get $l_a(v) = l_i(v) = l_i(u) + 1 = l_a(u) + 1$. Now suppose by contradiction that p_i unsaturated an edge wx which was saturated earlier. Then we have by (b) that $l_i(w) \geq l_a(w)$ and $m_i(x) \geq m_a(x)$. If p_i goes from s to w to x to t it has at least length $l_i(w) + 1 + m_i(x) \geq l_a(w) + 1 + m_a(x)$ and by (a) we have $l_a(w) = l_a(x) + 1$. Together we get $l_i(w) + 1 + m_i(x) \geq l_a(x) + m_a(x) + 2 > l_a(t) = l_i(t)$, meaning that p_i would not be a shortest path.

   (d) By (c) we get that for each possible length of a shortest path, there are at most $|E|$ iterations, since there are at most $|E|$ edges to saturate. A path in G has a length between 1 and $|V|$, so there are $|V||E|$ iterations.

7. Suppose you are given an oracle that, given a graph G, tells you whether G has a perfect matching or not. Show how to use this oracle to determine the maximum cardinality matching of a graph $G(V, E)$. The total number of calls should be at most $|V| + |E|$. *Hint: modify the graph at each call of the oracle.*

Solution: We define $G + k$ to be the graph by adding k dummy vertices joined to every vertex of G to the original graph. A graph can only have a perfect matching if its vertex set has even cardinality. Thus, for every k such that $|V| + k$ is even, starting with 0 or 1 depending on parity, we can use the oracle on $G+k$. When the oracle returns true for the first time, this means that there exists a matching of cardinality $\frac{n-k}{2}$. Then, for each edge e we can call the oracle on $G-e$. If it returns true, this means $e \notin G$ and we can just delete e. If it returns false, then $e \in G$ and we can keep applying the same principle on $G - \{x, y\}$ where x, y are vertices such that $e = xy$. This method called the oracle at most $k + |E| \leq |V| + |E|$ times thus the required condition is satisfied.