Combinatorial Optimization – Problem Set 6

You can hand in one of the following problems at the start of Tuesday's problem session. Please explain your solution carefully. Don't forget to put your name.

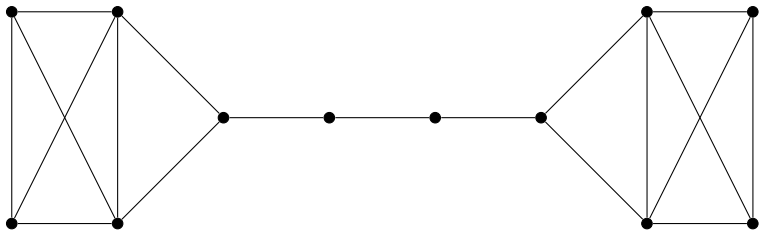
General Matchings

1. *Consider the following greedy algorithm for matchings:*

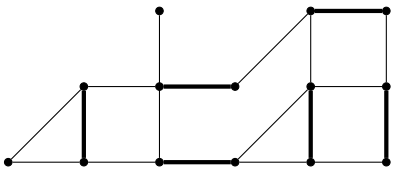
   Repeatedly add uv such that $\deg(u) + \deg(v)$ is minimal, then remove $\delta(u) \cup \delta(v)$.

   Give an example for which this algorithm does cannot return a maximum cardinality matching.

   This is the simplest example I could come up with. All edges have $d(uv) \geq 3$, except for the one in the middle, which has $d(uv) = 2$. After removing that one, and its two touching edges, we'd be left with two odd graphs, which we can't match perfectly. But we can match the whole graph perfectly if we avoid that middle edge.



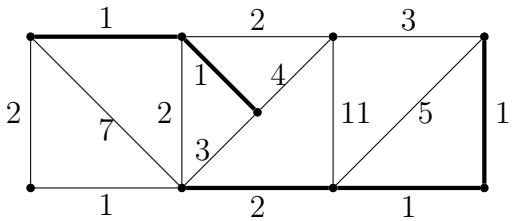
2. *Execute the blossom algorithm to find a maximum cardinality matching for the following example, starting with the given matching.*



   No way I'm doing that in LaTeX.
   I have set it up so that if you really take a shortest path in every step, then you will need 2 or 3 blossom shrinkings.

3. *Execute the postman algorithm to find a minimum postman tour in the following example (find the minimum weight perfect matching by inspection).*

   There are 4 odd-degree vertices. Compute all 6 shortest paths between those (just by inspection...), take the minimum weight perfect matching (by inspection), and for those paths double the edges (they're thicker in the picture here). Now trace an Euler tour, using the thick edges twice (by trial and error).
   The weight of the tour is the sum of all the weights, and then the double ones added. I think that's 52.



4. *Show that Dijkstra's algorithm can be used to find a shortest st-path in an undirected weighted graph, if all weights are nonnegative. Why does this fail when there are negative weights?*
Give an algorithm for shortest st-paths in an undirected graph G with arbitrary weights, but without negative cycles.
~~*(Hint: Find a minimum weight perfect matching in a graph constructed from two copies of G.)*~~

- **Dijkstra works for nonnegative weights:** Given an undirected graph G, define a directed graph D_G with $V(D_G) = V(G)$ and

$$D_G = \{ab, ba : \{a, b\} \in E(G)\}.$$

Define weights w' on D_G by $w'(ab) = w(\{a, b\})$. So we replace an edge $\{a, b\}$ by two directed edges ab and ba, both with weight $w(\{a, b\})$.
Then a shortest st-path in D_G, found by Dijkstra, will directly give a shortest st-path in G.

- **Dijkstra fails for negative weights:** When there are negative weights, D_G will have negative cycles. Indeed, if $w(\{a, b\}) < 0$, then aba is a cycle in D_G with weight $w'(aba) = 2w(\{a, b\}) < 0$. So Dijkstra's or Ford's algorithm will not work here.

- **Negative weights without negative cycles**

Note: *The solution to the third part that I originally had in mind does not quite work, and so the hint was not so helpful. Ask me if you want to know what was wrong with it. The construction below does sort of consist of two copies, but with more structure.*

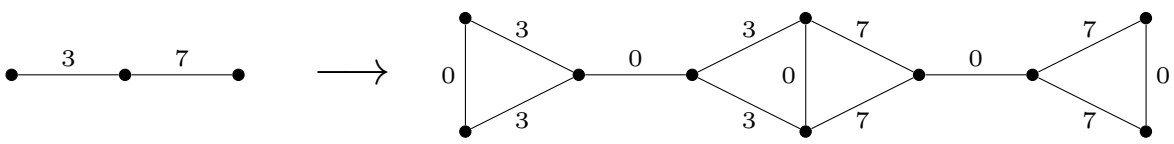
- We define a new graph G' from G as follows.
We leave s and t as they are. For every other vertex v of G, we take two copies v_1 and v_2, with an edge v_1v_2 of weight $w'(v_1v_2) = 0$.
For every edge uv in G, we add the following: Two vertices x_{uv} and y_{uv}, with edges

$$u_1x_{uv}, \quad u_2x_{uv}, \quad x_{uv}y_{uv}, \quad y_{uv}v_1, \quad \text{and} \quad y_{uv}v_2,$$

that have weights

$$w'(u_1x_{uv}) = w'(u_2x_{uv}) = w(uv), \quad w'(x_{uv}y_{uv}) = 0, \quad w'(y_{uv}v_1) = w'(y_{uv}v_2) = w(uv).$$

For an edge sv we add vertices x_{sv} and y_{sv}, with edges $sx_{sv}, x_{sv}y_{sv}, y_{sv}v_1, y_{sv}v_2$, and weights $w'(x_{sv}y_{sv}) = 0, w'(sx_{sv}) = w'(y_{sv}v_1) = w'(y_{sv}v_2) = w(sv)$. For edges tv we do the analogous thing. This completes G'. Here's what it looks like for two edges (not involving s or t, and leaving out vertex labels):



Now we claim that a minimum weight perfect matching M in G' will give a shortest st-path P in G, with $w'(M) = 2w(P)$.

Given a minimum weight perfect matching M in G', we construct a path P as follows. One of the edges sx_{sv} must be in M. Then for that v the edge $x_{sv}y_{sv}$ is not in M, so one of the edges $y_{sv}v_1$ and $y_{sv}v_2$ must be in M. Then v_1v_2 is not in M, and for some neighbor u of v one of the edges v_1x_{vu} and v_2x_{vu} must be in M. It is a bit tricky to describe, and a picture would do miracles, but this must go on like this, and the only way it can finish is at t. So we have an alternating path from s to t. Then all the original vertices v that we used copies of in this path together form an st-path P. Observe that the total weight of the matching edges in the alternating path in M equals $2w(P)$.
Now consider any edge M not on this alternating path. If it is not of type v_1v_2 or $x_{uv}y_{uv}$, then it will force an alternating path in the same way, but now it can only finish as an alternating cycle, and it similarly gives a cycle in G. Since there are no negative cycles, the total weight of this cycle is ≥ 0. But then it must be $= 0$, because the matching is minimum, and we can replace the cycle by edges of type v_1v_2 or $x_{uv}y_{uv}$, which have weight 0.
So this shows that because M is minimum, the total weight of edges of M outside the alternating path is 0, which means that $w'(M) = 2w(P)$.

We are done if every st-path P in G comes from a matching M in this way, with $w'(M) = 2w(P)$. But this is pretty obvious, by inverting the construction above: The path gives an alternating path from s to t, and every other vertex we match by edges of type v_1v_2 or $x_{uv}y_{uv}$.
So if M is minimum, then P must be shortest, because if there were a shorter path, then it would give a smaller matching.