COMP 251 Winter 2014 Mid-Term Exam 2 (solutions)

1.

a) Yes, it is bipartite. {A, C, E, G, I} and {B, D, F, H} are two sets such that all edges are crossing edges.

b) The stable matching is $(\alpha 1, \beta 1)$, $(\alpha 2, \beta 3)$, $(\alpha 3, \beta 2)$.

The only rejection occurs when the matching is $\{(\alpha 1,\beta 2), (\alpha 2,\beta 3)\}$ and then $\alpha 3$ proposes to $\beta 2$, who rejects $\alpha 1$in favor of $\alpha 3$. This yields the matching $\{(\alpha 3,\beta 2), (\alpha 2,\beta 3)\}$. Then $\alpha 1$ proposes to $\beta 1$ who accepts.

c) It is unstable because $\alpha 2$ prefers $\beta 3$ over $\beta 2$, and $\beta 3$ prefers $\alpha 2$ over $\alpha 3$.

GRADING SCHEME:

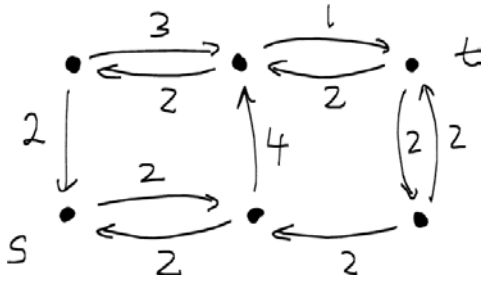
For (a) and (b), we simply looked for the answer. No explanation was required for (b).

For (c), we some students wrote out an answer which described the Gale-Shapley algorithm or some situation that could or could not occur within the running of that algorithm. However, we already know what the G-S algorithm does from b). The question in c) is not about G-S.

Some students got half of c), e.g. $\alpha 2$ prefers $\beta 3$ over $\beta 2$, but then they made comments about say $\alpha 3$' a preferences which are not relevant.

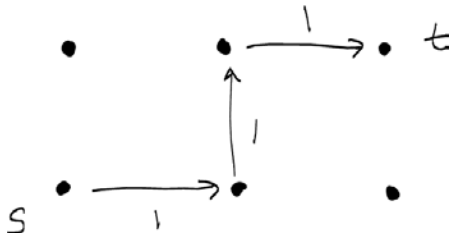
2.

a.



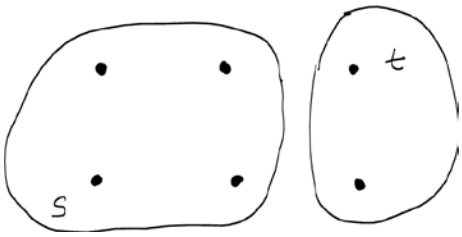
GRADING SCHEME: 3 points. Some students only put the forward edges. If correct they got 1.5.

b.



Some students put the new flow after graph (the flow from the augmenting path + the original flow). We decided not to take marks here, since the question asks specifically for the path.

c.



Many students marked the "cut set", namely the crossing edges, rather than the cut. We gave full marks for that, but you should be careful. A "cut" is not the same thing as a "cut set".

3.

a. You can describe it in words as "take as many of the most valuable coin as you can, then take as many of the second most valuable coin, etc"

b. You can do it with four coins: 1x16, 2x12, 1x3.

c. f(n) = 1 + min_{ i} { f(n – vi) : vi <= n }, and you also need a base case f(0) = 0.

Note that this problem was part of the Exercises, namely E17 Q7.

4.

a)

#edges	a	b	c	d	E
0	0	infinity	infinity	infinity	Infinity
1	0	infinity	infinity	-2	5
2	0	8	0	-2	-5
3	0	-2	-10	-2	-5
4	-1	-2	-14	-2	-5

vertex	a	b	c	d	E
prev[vertex]	b	e	b	a	D

GRADING SCHEME for (a): total of 7 points. We took off 1 point for every mistake. (We tried to avoid taking off more than one point for what was essentially the same mistake.)

GRADING SCHEME for (b): total of 3 points. There were a range of answers. Here is roughly what we wanted (which is not a complete solution – see below):

> Check if the minimum cost path to any vertex decreases from iteration n-1 to iteration n. If it does, then this path contain a cycle and it must be a negative cycle. The reason is must contain a cycle is that such a path would contain n+1 vertices, but there are only n vertices in the graph. The reason it must be a negative cycle is that positive cycles can be pruned to get a shorter path, so Bellman Ford would never have found a path that contains a positive cycle.

Students had many answers that sort of said the above but if they were a bit vague or they added statements that were irrelevant then we took off points.

I wrote about that the above "solution" is not a complete answer. Why not? The claim here that you are supposed to prove really is the following: "at least one vertex will have its cost go down from iteration n-1 to n **if and only if** the graph contains a negative cycle." The above solution only proves that claim in the forward direction (→). The proof in the other direction (←) is a bit more subtle. You would need to show that if the graph contains a negative cycle, then the cost of some vertex must decrease on nth iteration. While this might seem obvious at first glance, its actually a bit tricky to prove. If you want to see a proof, check out Tim Roughgarden's video on it: https://class.coursera.org/algo2-2012-001/lecture/139

By the way, Roughgarden points out this claim is true only if all vertices are reachable from the starting vertex. This is obvious once we remind ourselves that this situation can happen. For example, if there is a negative cost cycle that is not reachable from the starting vertex, then running BF for n iterations obviously won't find any of the vertices in this cycle and so we won't be able to detect it.