COMS20010 — Problem sheet 5

You don't have to finish the problem sheets before the class — focus on understanding the material the problem sheet is based on. If working on the problem sheet is the best way of doing that, for you, then that's what you should do, but don't be afraid to spend the time going back over the quiz and videos instead. (Then come back to the sheet when you need to revise for the exam!) I'll make solutions available shortly after the problem class. As with the Blackboard quizzes, question difficulty is denoted as follows:

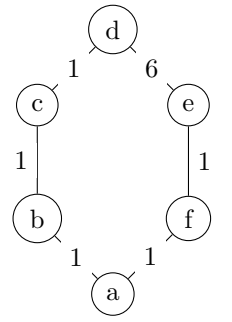
- ⋆ You'll need to understand facts from the lecture notes.
- ⋆⋆ You'll need to understand and apply facts and methods from the lecture notes in unfamiliar situations.
- ⋆⋆⋆ You'll need to understand and apply facts and methods from the lecture notes and also move a bit beyond them, e.g. by seeing how to modify an algorithm.
- ⋆⋆⋆⋆ You'll need to understand and apply facts and methods from the lecture notes in a way that requires significant creativity. You should expect to spend at least 5–10 minutes thinking about the question before you see how to answer it, and maybe far longer. Only 20% of marks in the exam will be from questions set at this level.
- ⋆⋆⋆⋆⋆ These questions are harder than anything that will appear on the exam, and are intended for the strongest students to test themselves. It's worth trying them if you're interested in doing an algorithms-based project next year — whether you manage them or not, if you enjoy thinking about them then it would be a good fit.

If you think there is an error in the sheet, please post to the unit Team — if you're right then it's much better for everyone to know about the problem, and if you're wrong then there are probably other people confused about the same thing.

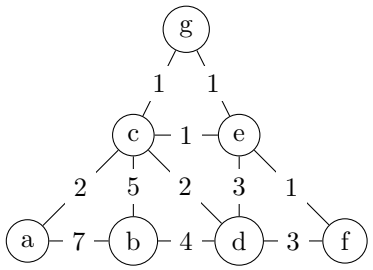
This problem sheet covers week 5, focusing on matchings and spanning trees.

1. (a) [⋆⋆] For each of these graphs give a minimum spanning tree as well as a spanning tree which is not minimum.

i.



ii.

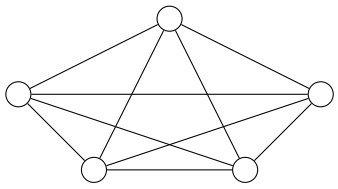


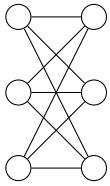
(b) [⋆⋆] How is Kruskal's algorithm different to Prim's algorithm?

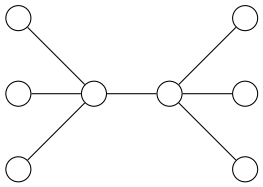
(c) [⋆⋆] Which earlier ideas in the course are these algorithms based on?

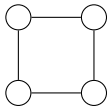
2. [⋆⋆] Let $G = ((V, E), w)$ be a weighted graph, and let T be a minimum spanning tree for G. Give an example to show that the path between two vertices in T may not be a shortest path in G.

3. [⋆⋆] Which of these graphs are bipartite?

(a)


(b)


(c)


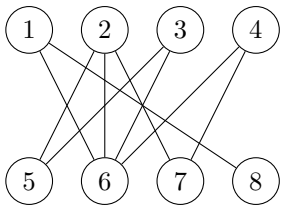
(d)


4. [⋆⋆⋆] Give an algorithm to find a bipartition for any graph with no odd-length cycle in polynomial time. **Hint:** Try using one of the algorithms covered last week.

   Together with the fact that an odd-length cycle is not bipartite, proved in lectures, this implies a graph is bipartite if and only if it has no odd-length cycle.

5. [⋆⋆⋆] Show that a bipartite graph G with bipartition (A, B) has at most $|A||B|$ edges. Using this, show that any n-vertex graph with more than $n^2/4$ edges is not bipartite, and hence contains an odd cycle. (You may assume n is even.)

6. (a) [⋆⋆] For the following graph, run the naive greedy algorithm from the start of video 5-2 to compute a matching which you can't add any more edges to (which might or might not be maximum). Start with the vertices of lowest index, and break ties by choosing edges which connect to vertices of lower index.



   (b) [⋆⋆] Find an augmenting path for this matching.

   (c) [⋆⋆] Use the augmenting path you have found to turn the matching into a maximum matching.

7. [⋆⋆⋆] In lectures, we discussed a purely greedy algorithm for finding a maximum matching — forming a matching by iterating over all edges and adding as many as possible — and noted that it doesn't always

work. Show that if a maximum matching has size k, then this greedy algorithm returns a matching of size at least $k/2$. Why might this be useful?

8. [★★★★★] Suppose you are trying to design a clustering algorithm. You are given a set P of n photos of k objects, with $n > k$, and an algorithm which is capable of evaluating a "distance" $d(x, y)$ between any two photos x and y. You may assume that $d(x, y) > 0$ whenever x and y are distinct, and that $d(x, y) = d(y, x)$ for all $x, y \in P$; however, you may not assume that d satisfies the triangle inequality. You expect that if $d(x, y)$ is low, then x and y are likely to be photos of the same object, and vice versa. Your objective is to use this information to categorise the photos according to which object they depict — that is, to partition P into non-empty subsets $P_1, \ldots, P_k$ such that if x and y lie in two different sets P_i, then $d(x, y)$ is large. Specifically, you are trying to maximise the *spacing*:

$$\min\{d(x, y) \colon x \text{ and } y \text{ are not in the same } P_i\}.$$

Find an algorithm which finds $P_1, \ldots, P_k$ in $O(|P|^2 \log |P|)$ time, given P and d, and prove it works. **Hint:** Try to find a way of applying Kruskal's algorithm.

9. [★★★ and a half] You are teaching a class of schoolchildren $C_1, \ldots, C_n$ in the pre-COVID era. The end of term is approaching, and you have bought them a large tub of chocolates — you plan to give them four chocolates each, and eat the rest yourself. However, there are many types $T_1, \ldots, T_k$ of chocolate in the tub, and not every child likes every type of chocolate — a given child C_i is willing to accept only chocolates from a set $S_i \subseteq \{T_1, \ldots, T_k\}$. The tub is also not bottomless — for each $i \in [k]$, there are only $t_i > 0$ chocolates of type T_i. Give an algorithm which, given $T_1, \ldots, T_k$, $t_1, \ldots, t_k$ and $S_1, \ldots, S_n$, assigns four chocolates to every child in polynomial time if possible and returns `Impossible` otherwise.

10. [★★★★] Suppose you're trying to schedule meetings across a company, with multiple rooms (from a set R) available. For simplicity, suppose that all meetings are an hour long and occur on the hour, and write T for the set of available times. A set P of people are trying to book rooms for meetings, and each person $p \in P$ has a set T_p of available times and a set R_p of rooms that will meet their needs. (For example, some people may need a projector, some people may need large seating capacity, some people may need a nicely-furnished room to impress a visitor, and so on.) Describe a polynomial-time algorithm to assign as many rooms and times as possible while satisfying everyone's preferences and avoiding double-booking. In other words, it should return a set $\mathcal{S}$ of k triples $(p, r, t) \in P \times R \times T$ such that:

    (i) each person p appears in at most one triple in $\mathcal{S}$;

    (ii) each pair $(r, t) \in R \times T$ appears in at most one triple in $\mathcal{S}$;

    (iii) for all $(p, r, t) \in \mathcal{S}$, we have $r \in R_p$ and $t \in T_p$; and

    (iv) k is as large as possible subject to the remaining constraints.

    Hint: Try to reduce this problem to finding a maximum matching in an auxiliary graph.

11. Suppose you are given coordinates for n cities, and you wish to connect their electrical networks as cheaply as possible. The local terrain may not be flat, so cost may not scale linearly with distance. However, it is still true that all costs are non-negative, and that it costs at least as much to build a cable from u to v and then a cable from v to w as it does to build a cable from u to w directly.

    (a) [⋆] Explain how to use Prim's algorithm to solve the problem, under the assumption that power cables can only connect to each other inside cities.

    (b) [★★★★★] Prove that the additional assumption in part (a) increases the cost by at most a factor of 2. **Hint**: Consider an optimum network without the assumption, and try to replace it with one that only branches in cities. You may use the fact that depth-first search on a tree generates a walk that crosses every edge twice.

12. [$\star\star\star\star$] Let G be a **3-regular** graph (also known as a **cubic** graph). Show that if a maximum matching in G has size k, then the purely greedy algorithm of Question 7 returns a matching of size at least $3k/5$. (**Hint:** How many edges are in the matching returned by the greedy algorithm?)

13. [$\star\star\star$] Let G be an n-vertex bipartite graph, and let M be a matching in G. Let k be the size of a maximum matching in G. In lectures, we proved Berge's lemma: $|M| = k$ if and only if M has no augmenting paths. Prove that for all odd integers $\ell \geq 3$, if M has no augmenting paths of length less than ℓ, then $|M| \geq \frac{\ell-1}{\ell+1}k$. **Hint:** Consider the proof of Berge's lemma.