COMS20010 — Problem sheet 8

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:

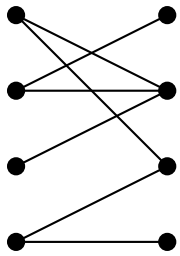
- ⋆ You'll need to understand facts from the lecture notes.
- ⋆⋆ You'll need to understand and apply facts and methods from the lecture notes in unfamiliar situations.
- ⋆⋆⋆ You'll need to understand and apply facts and methods from the lecture notes and also move a bit beyond them, e.g. by seeing how 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. At most 10% 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 9, focusing on network flows and NP-hardness.

1. [⋆⋆] In lectures, we showed how to use edge capacities to simulate vertex capacities in flow networks. Explain how to do this the other way round, using vertex capacities to simulate edge capacities. In other words, given a flow network (G, c_E, s, t) with source s, sink t and edge capacity function c_E, explain how to construct a network (G', c_V, s', t') with a vertex capacity function c_V such that flows in (G, c_E, s, t) correspond to flows in (G', c_V, s', t') of the same value, and vice versa.

2. (a) [⋆⋆] Turn this (bipartite) matching problem into a flow network problem using the reduction described in lectures.

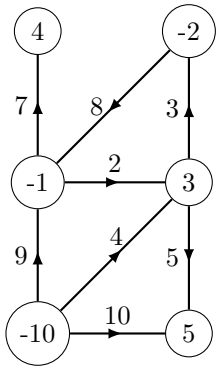


   (b) [⋆⋆] Use Ford-Fulkerson to calculate a maximum flow for your flow network from part (a).

   (c) [⋆⋆] How does this maximum flow give you a maximum matching for the earlier bipartite graph?

   (d) [⋆⋆⋆] Using the max-flow min-cut theorem, prove that if G is a bipartite graph with bipartition (A, B), and $|A| = |B|$, then G contains a perfect matching if and only if and for all $X \subseteq A$ we have $|N(X)| \geq |X|$. (**Hint:** Remember the reduction given in lectures from finding a maximum matching to finding a maximum flow.)

3. (a) [⋆⋆] Reduce this circulation problem to a maximum flow problem using the reduction in lectures.



   (b) [⋆⋆] Use Ford-Fulkerson to find a maximum flow in your reduction problem.

   (c) [⋆⋆] Explain how this shows us there is no valid circulation in this network.

4. [⋆⋆] In lectures, we said that every propositional logic formula (such as $\neg x \wedge \neg((y \vee z) \wedge \neg z)$) can be expressed in conjunctive normal form. In other words, for any formula F, there is a CNF formula F' on the same variables which evaluates to `True` if and only if F does. Give an exponential-time algorithm to find such a formula. (**Hint:** Try enumerating the assignments on which F is false.)

5. [⋆⋆⋆] In lectures, we said that search problems are often exactly as hard as their equivalent decision problems. This question gives two examples.

   (a) Give a Cook reduction from the problem of finding a satisfying assignment in a CNF to SAT, which asks whether a satisfying assignment exists. (**Hint:** Try coming up with an exponential-time recursive algorithm, then using the oracle to speed it up.)

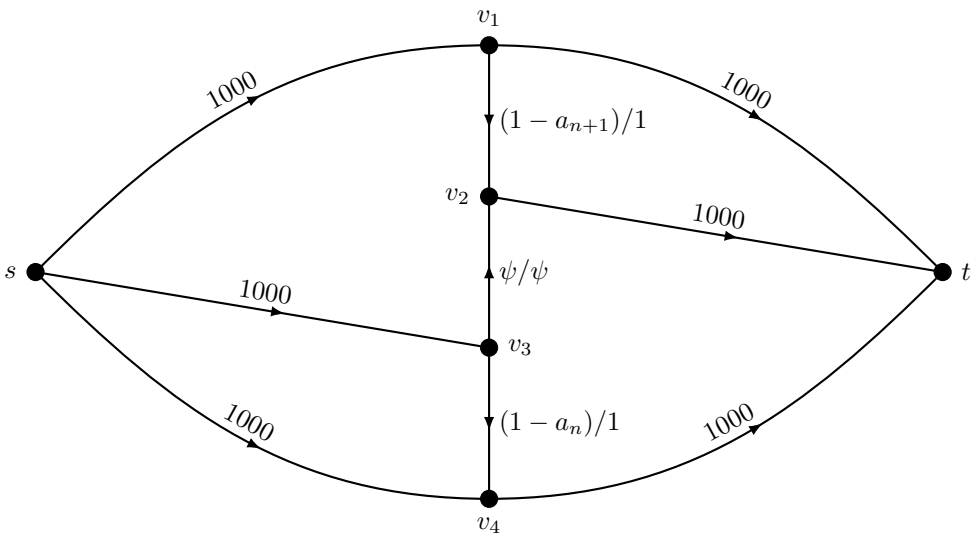
   (b) Give a Cook reduction from the problem of finding a maximum matching in a graph to the decision problem which asks: given a graph G and an integer k, does G contain a matching of size at least k? (Note that the graph need not be bipartite.)

6. (a) [⋆⋆] Solve the recurrence

$$\begin{aligned} a_0 &= 1, \\ a_1 = \psi &:= \frac{\sqrt{5}-1}{2}, \\ a_{n+2} &= a_n - a_{n+1} \text{ for all } n \geq 0. \end{aligned}$$

   Hint: in COMS10007 last year we saw that for recurrences of this form, solutions of the form Ax^n are often a good starting point.

   (b) [⋆⋆⋆] Consider the following flow network and the pictured flow, where the unspecified flows (on the edges incident to s and t) are arbitrary but substantially below the capacity.



Find a sequence of four augmenting paths such that, if flow is pushed along each path in sequence, then the flow on e_1 will become $1 - a_{n+3}$, the flow on e_2 will stay ψ, and the flow on e_3 will become $1 - a_{n+2}$.

(c) [⋆⋆⋆] Using the previous two parts, prove that the Ford-Fulkerson algorithm need not terminate when edge weights are irrational, and moreover that terminating it early may yield an answer which is very far from optimal.

(d) [⋆⋆⋆] What would go wrong with the above argument if the capacity of (v_3, v_2) were anything other than ψ?

7. Given literals $x_1, \ldots, x_k$ with $k \geq 2$, a *not-all-equal* or *NAE* clause $\text{NAE}(x_1, \ldots, x_k)$ evaluates to true if and only if the values of $x_1, \ldots, x_k$ are not all equal, i.e. at least one x_i is true and at least one x_i is false. The *NAE-SAT* problem asks, given a formula of the form

$$NAE(x_{1,1}, \ldots, x_{1,k_1}) \wedge NAE(x_{2,1}, \ldots, x_{2,k_2}) \wedge \cdots \wedge NAE(x_{\ell,1}, \ldots, x_{\ell,k_\ell}),$$

whether it is satisfiable — that is, whether there is any assignment of truth values to variables which makes the formula true. In other words, it is like SAT, except that we have NAE clauses instead of AND clauses. (Note we allow literals to appear multiple times in the formula, so we may have e.g. $x_{1,1} = x_{1,2} = x_{2,1}$.)

(a) [⋆⋆⋆⋆] 3-NAE-SAT is the version of NAE-SAT in which $k_i = 3$ for all $i \in [\ell]$. Give a Cook reduction from NAE-SAT to 3-NAE-SAT.

(b) [⋆⋆⋆⋆] Using the first part of the question, prove that 3-NAE-SAT is NP-complete under Cook reductions.

(c) [⋆⋆⋆⋆] *Monotone 3-NAE-SAT* is the version of 3-NAE-SAT in which all literals are un-negated variables, so we do not allow e.g. a clause $\text{NAE}(x_1, \neg x_2, x_3)$. Using the second part of the question, prove that Monotone 3-NAE-SAT is NP-complete under Cook reductions.