CPSC 320 Proofs: Exam Edition

1 How to do 320 proofs

In the following examples, we've taken proof questions from old CPSC 320 exams and provided some common **incorrect** answers. See if you can find what's missing from these proofs, and try to fix them!

Lesson 1: Capture the key points

Practice question (Q2.1, 2017W1 midterm 1)[1]**:** You're given an SMP instance where two men have the same preference list. Prove that the Gale-Shapley algorithm (with men proposing) run on this instance will take **more than** n iterations.

- An incomplete proof: *Since two men share the same preference list, they will both propose to the same woman first. She can only accept one of them, so one of the men will have to propose again to someone else.*
- A not-fully-justified proof: *To complete in n iterations, we need each man to propose exactly once and have him be accepted. When two men have the same preference list, we need at least one more proposal, and hence at least $n + 1$ iterations.*

What are the necessary points for a complete proof?

Lesson 2: Be precise

Practice question (Q4.3, 2016W1 midterm): Prove that an articulation point in a simple, unweighted, undirected graph is **never** diametric.

- An on-track but vague proof: *An articulation point can't be diametric, because articulation points are in the middle of a graph (because they connect two components), while diametric nodes are at the edges.*

How can we "translate" the above proof attempt into the language of a proof?

A general lesson: avoid unjustified statements! (These are proofs. Explaining and justifying your thinking process is kind of the whole point.)

[1] You also had to prove something very similar in your first assignment – hopefully this looks familiar to you!

2 Exchange arguments (for greedy optimality proofs)

Exchange arguments are a technique for proving the optimality of greedy algorithms. Assume that we have an optimal solution to a problem that's different from the greedy solution. In an exchange argument, we show that we can gradually transform the optimal solution into our greedy solution without its quality ever decreasing. This shows that the optimal solution is no better than the greedy solution and the greedy solution is in fact optimal.

Steps in an exchange argument:

1. Define the greedy solution $\mathcal{G} = \{g_1, g_2, \ldots g_j\}$ and optimal solution $\mathcal{O} = \{o_1, o_2, \ldots o_k\}$.
2. Describe where $\mathcal{G}$ differs from $\mathcal{O}$. This could mean elements that appear in different orders in each solution, or elements that appear in one solution but not the other, etc.
3. Perform an exchange to transform the solution $\mathcal{O}$ into solution $\mathcal{G}$, proving that the quality will not decrease by doing so.

(Note: the swap described in steps 2 and 3 may need to be repeated multiple times, as in general there will be more than one swap that needs to be made. You need to be sure that you can exchange **repeatedly** without hurting the solution quality.)

Practice question (Q5, 2016W2 midterm)

Predatory banks take the debits to an account that occur over the day and reorder them to maximize the fees they can charge. For each debit that results in taking an account into overdraft (having negative balance in that account) or where the account is already in overdraft, the bank charges the customer an overdraft fee.

So, for example, you may have spent \$3, \$7, \$2, and \$1 in a day when your account balance started at \$4. If we keep the debits in the order they're given, the \$3 debit takes your balance to \$1. The \$7 debit takes your balance to -\$6 and is the first to go into overdraft. All subseuqent debits are also in overdraft. The \$2 debit takes your balance to -\$8. The \$1 debit takes your balance to -\$9.

Imagine that an overdraft fee on a debit of d dollars that goes into overdraft by $k \leq d$ dollars is $\frac{k^2}{100}$. (k is the amount of debit that cannot be paid. So, for the \$7 debit in the example above, $k = 6$. For the \$2 debit $k = 2$ and for the \$1 debit $k = 1$.)

In this problem, the bank wants to generate an order of the debits that will maximize the fees it collects.

1. Give a small but non-trivial instance of the problem along with its optimal solution and the value (total fees) of that solution.

2. Determine an ordering of debits for a greedy algorithm to solve this problem optimally.

3. Complete the following exchange argument to prove correctness of your algorithm.

We compare the greedy algorithm's debit ordering $\mathcal{G}$ against an optimal ordering $\mathcal{O}$. If $\mathcal{O}$ and $\mathcal{G}$ are the same, then $\mathcal{G}$ is optimal. Otherwise, let d_i and d_{i+1} be a pair of neighbouring debits that are in one order in $\mathcal{O}$ and the opposite order (and not necessarily neighbouring) in $\mathcal{G}$. Let the total of all the debits before d_i in $\mathcal{O}$ be T and the initial account balance be B. We now show that we can swap d_i and d_{i+1} without decreasing the overall value of the solution in four cases:

Case 1, $B \leq T$ **(i.e., both debits are entirely in overdraft):** After swapping, both are still entirely in overdraft and so contribute the same amount to the fees collected.

Case 2, $T < B < T + d_i$ **(i.e.,** d_i **is the debit that takes the account into overdraft):** LEFT AS BONUS (not required to answer).

Case 3, $T + d_i \leq B < T + d_i + d_{i+1}$ **(i.e.,** d_{i+1} **is the debit that takes the account into overdraft):**

Case 4, $T + d_i + d_{i+1} \leq B$ **(i.e., neither debit is in overdraft)**

Thus, we can start from $\mathcal{O}$ and repeatedly swap neighbouring debits that are in the opposite order to their order in $\mathcal{G}$ until the solution is identical to $\mathcal{G}$ without reducing the value (fees) of the solution, and so $\mathcal{G}$ is optimal. QED