Tutorial 7 solutions

1. Let's look first at why the two proofs above aren't entirely correct. The first one is clearly on the right track, in that it justifies why there has to be an additional offer. But what's missing here is: why does this mean there have to be more than n offers? (Helpful tip: it's a little hard to completely prove that some quantity – in this case, the number of iterations – is greater than n without even **mentioning** n.)

   Now, for the second one: this proof states that when two employers have the same preference list, we need at least one more offer. But it doesn't say why this is the case.

   Note that the two proofs together give us everything we need: the second proof gives us the little bit of background knowledge that was missing from the first proof – namely, that Gale-Shapley only completes in n iterations if every employer makes exactly one offer; and the first proof provides the justification of the need for an extra offer that was missing from the second one. So here's how a correct solution could look:

   Proof: To complete in n iterations, we need each employer to make exactly one offer and be accepted. Since two employers share the same preference list, they will both make an offer to the same applicant first. The applicant can only accept one of them, so one of the employers will have to make an offer to someone else, and we will therefore have at least $n + 1$ iterations.

2. We don't have the LaTeX source for the sample solutions for this question, and we don't want to have to rewrite it all out. So we've appended the two pages of the 2016W2 midterm sample solution after this page. The solution includes the answers to all parts of this question (including the bonus), plus one additional part to this question that wasn't included in the worksheet.

5.1 Sample Solution

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

   5 dollars in the account. Charges of 3 dollars, 4 dollars, and 5 dollars. The optimal solution is 3, 4, and then 5 (although 4, 3, and then 5 results in the same answer), with a fee of $5^2 + 2^2 = 25 + 4 = 29$ cents.

2. Give pseudocode for a very simple greedy algorithm that solves the problem optimally in $O(n \lg n)$ time for n debits.

   Sort the debits in increasing order.

3. Complete the following proof of the 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 $\mathcal{G}$ optimal. Otherwise, let d_i and d_{i+1} be a pair of neighboring 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 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) Since they're out of order with respect to the greedy solution, $d_i > d_{i+1}$ so $T + d_{i+1} < T + d_i$. There are then two interesting cases.

   (1) When $T < B < T + d_{i+1}$, debit d_{i+1} incurs some fees after the swap and d_i is entirely in overdraft after the swap. Before the swap, the total fees on these two debits are $d_{i+1}^2 + (T + d_i - B)^2 = d_{i+1}^2 + T^2 + d_i^2 + B^2 + 2Td_i - 2TB - 2d_iB$. After the swap, the fees are $d_i^2 + (T + d_{i+1} - B)^2 = d_i^2 + T^2 + d_{i+1}^2 + B^2 + 2Td_{i+1} - 2TB - 2d_{i+1}B$. Subtracting the pre-swap fees from the post-swap fees:

$$\begin{aligned}
\text{post-swap} - \text{pre-swap} &= d_i^2 + T^2 + d_{i+1}^2 + B^2 + 2Td_{i+1} - 2TB - 2d_{i+1}B - \\
&\qquad (d_{i+1}^2 + T^2 + d_i^2 + B^2 + 2Td_i - 2TB - 2d_iB) \\
&= 2Td_{i+1} - 2Td_i - 2d_{i+1}B + 2d_iB \\
&= 2T(d_{i+1} - d_i) - 2B(d_{i+1} - d_i) \\
&= 2(T - B)(d_{i+1} - d_i) \\
&= 2(B - T)(d_i - d_{i+1})
\end{aligned}$$

   We know $B - T > 0$ because $T < B$. We also know $d_i - d_{i+1} > 0$ because $d_i > d_{i+1}$. Thus, the post-swap fees are greater than the pre-swap fees. (Technically, that's a contradiction with this case being possible, since we're improving the optimal solution. However, for our purposes, the point is that the step doesn't reduce the value of the solution.)

   (2) When $T + d_{i+1} \leq B < T + d_i$, the second debit is entirely in overdraft before the swap and not at all in overdraft afterward, while the first debit is partly in overdraft before the swap and partly or entirely in overdraft afterward. For simplicity, we name the quantity $o = T + d_i + d_{i+1} - B$, which is the total amount of overdraft between the two debits.

   So, the pre-swap fees are $d_{i+1}^2 + (o - d_{i+1})^2 = d_{i+1}^2 + o^2 - 2od_{i+1} + d_{i+1}^2 = 2d_{i+1}^2 + o^2 - 2od_{i+1}$. The post-swap fees are just o^2, since the entire amount of the overdraft on these two debits is within d_i. Subtracting the pre-swap fees from the post-swap fees gives us: $2od_{i+1} - 2d_{i+1}^2 = 2d_{i+1}(o - d_{i+1})$,

which is greater than 0 since the total overdraft amount (o) was larger than d_{i+1}. Thus, again, this swap results in a solution that collects no less in fees than it used to.

(Note: clearly there is no change in the fees due to any other debit in the sequence when we make this swap.)

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)** Since $d_i > d_{i+1}$, when we swap these two debits, d_i will now be the debit that takes the account into overdraft. Since the total of the debits up to and including these two (now-swapped) debits remains the same, the amount of overdraft remains the same, and there is no change to the fees collected. (For some math to throw at this, note that $T + d_{i+1} < T + d_i \leq B$.)

(Note: clearly there is no change in the fees due to any other debit in the sequence when we make this swap.)

Case 4, $T + d_i + d_{i+1} \leq B$ **(i.e., neither debit is in overdraft)** When we swap the two debits, they're still both not in overdraft, and the fees collected remain the same.

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

4. Imagine the fee were $\frac{k}{100}$ instead instead of $\frac{k^2}{100}$. A friend proposes to you an algorithm that resequences the debits. Without knowing the details of the proposal, what can you say about how close that algorithm's result is to the optimal result? **Briefly and clearly justify your answer.**

   The total of the k values for all the debits is exactly the size of the negative balance. (Or, to put it another way, the total of all the debits is always the same, regardless of their order, and the total k values will be the total of the debits minus the balance, or zero of there are no overdrafts.) Thus, since all strategies produce the same negative balance, all strategies produce the same overdraft charge and are optimal.

3. a. The trick is to notice that people who know less than five other people can not be part of the subset Alice wants, nor can people who know at least $n-5$ other people. So the algorithm is the following:

```
Algorithm PickSubset(S)

while an element x of S knows fewer than 5 or more than |S| - 5 people
    remove x from S
endwhile

return S
```

Note that when we write "knowing fewer than 5 or more than $|S|-5$ people", we mean with respect to the current set S, not the set the algorithm started with.

b. The base case is the case $i = 0$, before the first iteration of the `while` loop. At that point, S_0 contains every person Alice has to choose from, and hence contains every member of the optimal solution.

Let us now consider the induction step. Suppose the statement was true after the i^{th} iteration of the loop, and consider now the set S_{i+1}. The person x removed from S_i to obtain S_{i+1} either knows fewer than 5 other people in S_i, which means he/she does not know enough people in the optimal subset to be invited, or he/she knows more than $|S_i| - 5$ people in S_i, which means there are not enough people in the optimal subset that he/she does not know. In both cases x can not be part of the optimal solution, and hence the optimal solution must also be a subset of S_{i+1}.

This completes the induction step. Now, when the `while` loop finally exits after t iterations, every person in S_t knows at least 5 other people, and there are at least 5 other people that he/she does not know. Hence S_t is a possible solution to Alice's problem. Because we proved that the optimal solution is a subset of S_t, this means that S_t is the optimal solution we were looking for.