Problemset 2

Albert Ye

September 10, 2022

1 Airport

For $n = 1$, we must have a scalene triangle $\triangle ABC$. If BC is the shortest edge, then B will point to C and C will point to B. Then A will point to the closer of B and C, but nothing will point to A.

Given the claim holds for n, we seek to prove the claim for $n+1$. Let A be an airport such that there is no flight towards it. If we were to add two vertices such that none of them affected A, we would still have no flights destined to A. Thus, the claim still holds for such cases.

Now, let's add one vertex B close to A such that a flight from B would be destined to A. Then, B would be the closest vertex to A as well, so we would have a cycle of size 2 and a remaining network of size $2n$. Now, let's add the final vertex C.

If C is closer to a vertex on the cycle of size 2, we would be able to split the problem into a network of size 3 and a network of size $2n$, and the part of size 3 would necessarily have a vertex without inbound flights. Analogously, if C is closer to the network of size $2n$, then we would split the graph into a cycle of size 2 and a flight network of size $2n+1$, which we assume to have a vertex without inbound flights.

Therefore, regardless of how we add two airports, there will always still be an airport without inbound flights for $n+1$ if there is such an airport for n. □

2 Proving Inequality

For $n \in \mathbb{Z}^+$, let $a_n = \frac{1}{3^1} + \frac{1}{3^2} + \cdots + \frac{1}{3^n}$. We strengthen the inductive hypothesis, claiming that $a_n = \frac{3^0+3^1+\cdots+3^{n-1}}{3^n} < \frac{1}{2}$ for all n. To see why it is less, consider base 3. $3^0 + 3^1 + \cdots + 3^{n-1} = 111\ldots1_3$ with n ones. Doubling that would give $222\ldots2_3$ with n twos. This is strictly less than $100\ldots0_3$ with n zeroes.

Plugging in $n = 1$, we see that $a_1 = \frac{3^0}{3^1} = \frac{1}{3}$ is true. Now, we can use induction to prove this over all positive integer n. Given $a_n = \frac{3^0+3^1+\cdots+3^{n-1}}{3^n}$, we add $\frac{1}{3^{n+1}}$ to get a_{n+1}. Thus, we have

$$\begin{aligned} a_{n+1} &= \frac{3^0 + 3^1 + \cdots + 3^{n-1}}{3^n} + \frac{1}{3^{n+1}} \\ &= \frac{3^1 + 3^2 + \cdots + 3^n}{3^{n+1}} + \frac{1}{3^{n+1}} \\ &= \frac{1 + 3^1 + 3^2 + \cdots + 3^n}{3^{n+1}} \\ &= \frac{3^0 + 3^1 + \cdots + 3^n}{3^{n+1}}. \end{aligned}$$

We see that the formula still holds for $n+1$ given n, so $a_n < \frac{1}{2}$ must be true for all $n \in \mathbb{Z}^+$. □

3 Inductive Charging

a. Assume for the sake of contradiction that $f_i < d_i$ for all i. Then, $\sum_{i=0}^{n} f_i < \sum_{i=0}^{n} d_i = D$, meaning that there is not enough total fuel for one car to circle the track. Since that is a given for this problem, we have a contradiction. Thus, at least one car must have enough fuel to reach the next. □

b. We once again use induction on n. Start with $n = 1$, which clearly works as $n = 1$ means $f_1 = D$ in order for the one car to have enough fuel to go around the track.

Now, given that n satisfies the claim, we try to prove the same for $n + 1$. From part (a), we know that one car has enough fuel to travel to the next car. Without loss of generality, let that car be the first one. Then $f_1 > d_1$.

Our first car can thus reach the second car, and then transfer the fuel from the second car. Thus, we need a total fuel count of $f_1 + f_2$ to go a distance of $d_1 + d_2$. We can combine the two into one segment: $(f_1 + f_2) + f_3 + \cdots + f_{n+1} = (d_1 + d_2) + \cdots + d_{n+1}$. Doing this essentially reduces the problem from $n + 1$ cars into n cars, which we can assume already satisfies this claim. □

4 A Coin Game

Clearly, if $n = 1$, then the resulting score must be 0 since there are no ways to split the coins. We seek to prove that if $n = a + b$, then $\frac{a(a-1)}{2} + \frac{b(b-1)}{2} + ab = \frac{n(n-1)}{2}$.

Adding this together, we have

$$\begin{aligned}
\frac{a(a-1)}{2} + \frac{b(b-1)}{2} + ab &= \frac{a^2 - a + b^2 - b}{2} + ab \\
&= \frac{a^2 + b^2 + 2ab - a - b}{2} \\
&= \frac{(a+b)^2 - (a+b)}{2} \\
&= \frac{n^2 - n}{2} \\
&= \frac{n(n-1)}{2}.
\end{aligned}$$

Using this, we can divide a stack of n coins into stacks of smaller coins. Since we already know $n = 1$ must hold true, our claim that the score will always be $\frac{n(n-1)}{2}$ holds for all n from induction. □

5 Pairing Up

Let's construct the solution for just one pair.

Imagine the case where J_1 prefers $C_2 > C_1$ and J_2 prefers $J_2 > J_1$, while C_1 prefers $J_1 > J_2$ and C_2 prefers $J_2 > J_1$. Then the pairs $(J_1, C_1), (J_2, C_2)$ are a stable matching since the candidates both prefer the jobs they've received. But the pairs $(J_1, C_2), (J_2, C_1)$ are also a stable matching because the jobs both prefer the candidates they've received. Therefore, we can construct an instance of 2 jobs and 2 candidates with $2^{2/2} = 2$ stable matchings.

For a larger n jobs and n candidates, we can divide them into $\frac{n}{2}$ pairs $(i, i+1)$ and run the same preference scheme between J_i, J_{i+1} and C_i, C_{i+1}. To avoid some other job J_k or candidate C_k causing the matchings between J_i, J_{i+1} and C_i, C_{i+1} to be rogue, we can put J_i and J_{i+1} as the top preferences for C_i and C_{i+1}, and vice versa.

Assume there are $2^{n/2}$ stable matchings for n jobs and n candidates. When we add a pair $(n+1, n+2)$, we get two possible stable matchings: $(J_{n+1}, C_{n+1}), (J_{n+2}, C_{n+2})$ and $(J_{n+1}, C_{n+2}), (J_{n+2}, C_{n+1})$. For each of the $2^{n/2}$ original stable matchings, we can choose one of the possible stable matchings between $n + 1$ and $n + 2$ without affecting any of the other pairs. This gives us $2^{(n+2)/2}$ stable matchings for $n + 2$ jobs and candidates.

6 Nothing Can Be Better Than Something

a. Let the jobs be $J_1, J_2, \ldots, J_n$ and let the matching candidates by $C_1, C_2, \ldots, C_n$. Let's introduce a "phantom" set of jobs $J'_1, J'_2, \ldots, J'_n$ where J'_i is the job C_i takes if it does not like any of the jobs it gets. We introduce another phantom set of candidates $C'_1, C'_2, \ldots, C'_n$ where C'_i is the candidate job J_i takes if it does not like any of its candidates.

Assume there is a stable matching where k candidates C_i that do not have a preferred job. Then, obviously, these candidates would match to J'_i. But when these candidates unmatch from their jobs, there will be k jobs that will have to match to phantom candidates C'_i. We now show that this arrangement meets all five criteria for a stable matching.

In this case, there will be no matched entity who prefers being unmatched over being with their current partner. All candidates are the top preference of their matching phantom jobs (and vice versa), so by the improvement lemma no candidate or job will ever be matched with a candidate they do not want to be matched with.

Assume there'll be a matched job J and unmatched candidate C that prefer each other. Then, J would have to be higher on the candidate's list than the phantom job J' that candidate did choose. If J prefers C over a candidate it matched with, then C would have earlier gotten an offer from J, contradicting the claim that C is unmatched. This logic applies similarly for a matched candidate and unmatched job.

Finally, assume there is an unmatched job J and unmatched candidate C that prefer to be with each other over their phantom matches C' and J' respectively. Then $J > J'$ for C and $C > C'$ for J. Then, J would have at some point extended an offer to C before settling for C', and C would have accepted it over J' because J is preferred. This contradicts the claim that J, C are unmatched.

Then, there are $n - k$ "phantom" jobs and candidates that still need to be matched, but we can necessarily create a stable matching out of all of them since all pairs of x jobs and x candidates can be stably matched. □

b. Let's assume for the sake of contradiction that we have a candidate C that gets no match and has to go for J' in one stable matching, but C does get a match J_1 in another stable matching.

This means that the original match to J_1, which we can call C_1, has to be matched to another job J_2. Whenever we rematch a candidate C_i to another job J_{i+1}, we will have to displace a candidate C_{i+1} to another job J_{i+2}. But as the set of candidates and jobs are finite, there will be a candidate C_n that cannot be matched to a better job, creating a rogue couple between C_n and J_n. □

7 Sundry

I worked on this alone, but I discussed the approaches towards some problems with my roommate Saathvik Selvan.