CS 70 Discrete Mathematics and Probability Theory
Fall 2024 Rao, Hug

HW 02

1 Airport

Note 3

Suppose that there are $2n+1$ airports, where n is a positive integer. The distances between any two airports are all different. For each airport, exactly one airplane departs from it and is destined for the closest airport. Prove by induction that there is an airport which has no airplanes destined for it.

Solution: We proceed by induction on n. For $n = 1$, let the 3 airports be A, B, C and without loss of generality suppose B, C is the closest pair of airports (which is well defined since all distances are different). Then the airplanes departing from B and C are flying towards each other. Since the airplane from A must fly to somewhere else, no airplanes are destined for airport A.

Now suppose the statement holds for $n = k$, i.e. when there are $2k+1$ airports. For $n = k+1$, i.e. when there are $2k+3$ airports, the airplanes departing from the closest two airports (say X and Y) must be destined for each other's starting airports. Removing these two airports reduce the problem to $2k+1$ airports.

From the inductive hypothesis, we know that among the $2k+1$ airports remaining, there is an airport with no incoming flights which we call airport Z. When we add back the two airports that we removed, there are two scenarios:

- Some of the flights get remapped to X or Y.
- None of the flights get remapped.

In either scenario, we conclude that the airport Z will continue to have no incoming flights when we add back the two airports, and so the statement holds for $n = k+1$. By induction, the claim holds for all $n \geq 1$.

2 Grid Induction

Note 3

Pacman is walking on an infinite 2D grid. He starts at some location $(i, j) \in \mathbb{N}^2$ in the first quadrant, and is constrained to stay in the first quadrant (say, by walls along the x and y axes).

Every second he does one of the following (if possible):

(i) Walk one step down, to $(i, j-1)$.

(ii) Walk one step left, to $(i-1, j)$.

For example, if he is at $(5,0)$, his only option is to walk left to $(4,0)$; if Pacman is instead at $(3,2)$, he could walk either to $(2,2)$ or $(3,1)$.

Prove by induction that no matter how he walks, he will always reach $(0,0)$ in finite time.

(*Hint*: Try starting Pacman at a few small points like $(2,1)$ and looking all the different paths he could take to reach $(0,0)$. Do you notice a pattern in the number of steps he takes? Try to use this to strengthen the inductive hypothesis.)

Solution: On first glance, this problem seems quite tricky, since we'd want to induct on *two* variables (i and j) rather than just one variable (as we've seen most commonly). However, following the hint, if we try out some smaller cases, we can notice that it takes Pacman $i+j$ seconds to reach $(0,0)$ if he starts in position (i,j), regardless what path he takes. This would imply that he reaches $(0,0)$ in a finite amount of time, since $i+j$ is a finite number.

This means that the quantity $i+j$ is something we could instead focus on, rather than the coordinate (i,j). In particular, we can try to induct on $i+j$ (essentially inducting on the amount of time it takes for Pacman to reach $(0,0)$), rather than inducting on i and j separately.

Proof. **Base Case**: If $i+j=0$, we know that $i=j=0$, since i and j must be non-negative. Hence, we have that Pacman is already at position $(0,0)$ and so will take $0=i+j$ steps to get there.

Inductive Hypothesis: Suppose that if Pacman starts at position (i,j) such that $i+j=n$, he will reach $(0,0)$ in finite time regardless of his path.

Inductive Step: Now suppose Pacman starts at position (i,j) such that $i+j=n+1$. If Pacman's first move is to position $(i-1,j)$, the sum of his x and y positions will be $i-1+j=(i+j)-1=n$. Thus, our inductive hypothesis tells us that it will take him a finite amount of time to get to $(0,0)$ no matter what path he takes. If Pacman's first move isn't to $(i-1,j)$, then it must be to $(i,j-1)$. Again in this case, the inductive hypothesis will tell us that Pacman will use a finite amount of time to get to $(0,0)$ no matter what path he takes. Thus, in either case, we have that Pacman will take a finite amount of time (one second for the first move and some additional finite time for the remainder) in order to reach $(0,0)$, proving the claim for $n+1$. □

Note that once we had observed that it seems to take exactly $i+j$ seconds for Pacman to reach $(0,0)$ from (i,j), we could have tried to prove this stronger claim. This is equivalent to the above proof, with the only difference being the more specific length of time used in the inductive hypothesis; all other steps are identical.

One can also prove this statement without this trick inducting on $i+j$. The proof isn't quite as elegant, but is included here anyways for reference.

We first prove by induction on i that if Pacman starts from position $(i,0)$, he will reach $(0,0)$ in finite time.

Proof. **Base Case**: If $i=0$, Pacman starts at position $(0,0)$, so he doesn't need any more steps. Thus, it takes Pacman 0 steps to reach the origin, where 0 is a finite number.

Inductive Hypothesis: Suppose that if $i = n$ (that is, if Pacman starts at position $(n,0)$), he will reach $(0,0)$ in finite time.

Inductive Step: Now say Pacman starts at position $(n+1,0)$. Since he is on the x-axis, he has only one move: he has to move to $(n,0)$. From the inductive hypothesis, we know he will only take finite time to get to $(0,0)$ once he's gotten to $(n,0)$, so he'll only take a finite amount of time plus one second to get there from $(n+1,0)$. A finite amount of time plus one second is still a finite amount of time, so we've proved the claim for $i = n+1$. □

We can now use this statement as the base case to prove our original claim by induction on j.

Proof. **Base Case**: If $j = 0$, Pacman starts at position $(i,0)$ for some $i \in \mathbb{N}$. We proved above that Pacman must reach $(0,0)$ in finite time starting from here.

Inductive Hypothesis: Suppose that if Pacman starts in position (i,n), he'll reach $(0,0)$ in finite time no matter what i is.

Inductive Step: We now consider what happens if Pacman starts from position $(i,n+1)$, where i can be any natural number. If Pacman starts by moving down, we can immediately apply the inductive hypothesis, since Pacman will be in position (i,n). However, if Pacman moves to the left, he'll be in position $(i-1,n+1)$, so we can't yet apply the inductive hypothesis. But note that Pacman can't keep moving left forever: after i such moves, he'll hit the wall on the y-axis and be forced to move down. Thus, Pacman must make a vertical move after only finitely many horizontal moves–and once he makes that vertical move, he'll be in position (k,n) for some $0 \leq k \leq i$, so the inductive hypothesis tells us that it will only take him a finite amount of time to reach $(0,0)$ from there. This means that Pacman can only take a finite amount of time moving to the left, one second making his first move down, then a finite amount of additional time after his first vertical move. Since a finite number plus one plus another finite number is still finite, this gives us our desired claim: Pacman must reach $(0,0)$ in finite time if he starts from position $(i,n+1)$ for any $i \in \mathbb{N}$. □

3 Universal Preference

Note 4

Suppose that preferences in a stable matching instance are universal: all n jobs share the preferences $C_1 > C_2 > \cdots > C_n$ and all candidates share the preferences $J_1 > J_2 > \cdots > J_n$.

(a) What pairing do we get from running the algorithm with jobs proposing? Can you prove this happens for all n?

(b) What pairing do we get from running the algorithm with candidates proposing?

(c) What does this tell us about the number of stable pairings?

Solution:

(a) The pairing results in (C_i, J_i) for each $i \in \{1, 2, ..., n\}$. This result can be proved by induction:

Our base case is when $n = 1$, so the only pairing is (C_1, J_1), and thus the base case is trivially true.

Now assume this is true for some $n \in \mathbb{N}$. On the first day with $n+1$ jobs and $n+1$ candidates, all $n+1$ jobs will propose to C_1. C_1 prefers J_1 the most, and the rest of the jobs will be rejected. This leaves a set of n unpaired jobs and n unpaired candidates who all have the same preferences (after the pairing of (C_1, J_1)). By the process of induction, this means that every i^{th} preferred candidate will be paired with the i^{th} preferred job.

(b) The pairings will again result in (J_i, C_i) for each $i \in \{1, 2, ..., n\}$. This can be proved by induction in the same as above, but replacing "job" with "candidate" and vice-versa.

(c) We know that job-proposing produces a candidate-pessimal stable pairing. We also know that candidate-proposing produces a candidate-optimal stable pairing. We found that candidate-optimal and candidate-pessimal pairings are the same. This means that there is only one stable pairing, since both the best and worst pairings (for candidates) are the same pairings.

4 Pairing Up

Note 4

Prove that for every even $n \geq 2$, there exists an instance of the stable matching problem with n jobs and n candidates such that the instance has at least $2^{n/2}$ distinct stable matchings.

Solution:

To prove that there exists such a stable matching instance for any even $n \geq 2$, it suffices to construct such an instance. But first, we look at the $n = 2$ case to generate some intuition. We can recognize that for the following preferences:

J_1	$C_1 > C_2$
J_2	$C_2 > C_1$

C_1	$J_2 > J_1$
C_2	$J_1 > J_2$

both $S = \{(J_1, C_1), (J_2, C_2)\}$ and $T = \{(C_1, J_2), (C_2, J_1)\}$ are stable pairings.

The $n/2$ in the exponent motivates us to consider pairing the n jobs into $n/2$ groups of 2 and likewise for the candidates. We pair up job $2k-1$ and $2k$ into a pair and candidate $2k-1$ and $2k$ into a pair, for $1 \leq k \leq n/2$.

From here, we recognize that for each pair (J_{2k-1}, J_{2k}) and (C_{2k-1}, C_{2k}), mirroring the preferences above would yield 2 stable matchings from the perspective of just these pairs. If we can extend this perspective to all $n/2$ pairs, this would be a total of $2^{n/2}$ stable matchings.

Our construction thus results in preference lists like follows:

J_1	$C_1 > C_2 > \dots$
J_2	$C_2 > C_1 > \dots$
$\vdots$	$\vdots$
J_{2k-1}	$C_{2k-1} > C_{2k} > \dots$
J_{2k}	$C_{2k} > C_{2k-1} > \dots$
$\vdots$	$\vdots$
J_{n-1}	$C_{n-1} > C_n > \dots$
J_n	$C_n > C_{n-1} > \dots$

C_1	$J_2 > J_1 > \dots$
C_2	$J_1 > J_2 > \dots$
$\vdots$	$\vdots$
C_{2k-1}	$J_{2k} > J_{2k-1} > \dots$
C_{2k}	$J_{2k-1} > J_{2k} > \dots$
$\vdots$	$\vdots$
C_{n-1}	$J_n > J_{n-1} > \dots$
C_n	$J_{n-1} > J_n > \dots$

Each match will have jobs in the kth pair paired to candidates in the kth pair for $1 \leq k \leq n/2$.

A job j in pair k will never form a rogue couple with any candidate c in pair $m \neq k$ since it always prefers the candidates in this pair over all candidates across other pairs. Since each job in pair k can be stably matched to either candidate in pair k, and there are $n/2$ total pairs, the number of stable matchings is $2^{n/2}$.

5 Optimal Candidates

Note 4

In the notes, we proved that the propose-and-reject algorithm always outputs the job-optimal pairing. However, we never explicitly showed why it is guaranteed that putting every job with its optimal candidate results in a pairing at all. Prove by contradiction that no two jobs can have the same optimal candidate. (Note: your proof should not rely on the fact that the propose-and-reject algorithm outputs the job-optimal pairing.)

Solution:

For the sake of contradiction, assume that we have some instance of the Stable Matching problem where both job J and job J' have candidate C as their optimal candidate. We further assume without loss of generality that C prefers J to J' (if this is not the case, we can just switch the names to make it so). This leads to preferences as follows:

J	$C > \dots$
J'	$C > \dots$

C	$J > J'$
C^*	$\dots$

Because C is J''s optimal partner, we know by definition that there must exist some stable pairing P in which J' is paired with C - i.e. $P = \{(J', C), (J, C^*), \dots\}$.

Since C is J's optimal partner, we know by definition that J must prefer C to any candidate it is ever paired with in any stable pairing–including C^*. Moreover, we previously said that C prefers J to J'. Thus, J and C would form a rogue couple in P, which is a contradiction because P is stable. So our initial assumption must be false: there must never exist two jobs who have the same optimal candidate.