CS 70 Discrete Mathematics and Probability Theory
Fall 2020 HW 2

1 Induction

Prove the following using induction:

(a) For all natural numbers $n > 2$, $2^n > 2n+1$.

(b) For all positive integers n, $1^2 + 2^2 + 3^2 + \cdots + n^2 = \frac{n(n+1)(2n+1)}{6}$.

(c) For all positive natural numbers n, $\frac{5}{4} \cdot 8^n + 3^{3n-1}$ is divisible by 19.

Solution:

(a) The inequality is true for $n = 3$ because $8 > 7$. Let the inequality be true for $n = k$, such that $2^k > 2k+1$. Then,

$$2^{k+1} = 2 \cdot 2^k > 2 \cdot (2k+1) = 4k+2$$

We know $2k > 1$ because k is a positive integer. Thus:

$$4k+2 = 2k+2k+2 > 2k+1+2 = 2k+3 = 2(k+1)+1$$

We've shown that $2^{k+1} > 2(k+1)+1$, which completes the inductive step.

(b) We can verify that the statement is true for $n = 1$. Assume the statement holds for $n = k$, so that

$$\sum_{i=1}^{k} i^2 = \frac{k(k+1)(2k+1)}{6}.$$

Then we can write

$$\begin{aligned}
\sum_{i=1}^{k+1} i^2 &= \frac{k(k+1)(2k+1)}{6} + (k+1)^2 \\
&= (k+1)\left(\frac{k(2k+1)}{6} + (k+1)\right) \\
&= (k+1)\left(\frac{2k^2+k+6k+6}{6}\right) \\
&= (k+1)\left(\frac{2k^2+7k+6}{6}\right) \\
&= (k+1)\left(\frac{(2k+3)(k+2)}{6}\right) \\
&= \frac{(k+1)(2(k+1)+1)((k+1)+1)}{6},
\end{aligned}$$

as desired. Since we've shown that the statement holds for $n = k+1$, our proof is complete.

(c) For $n = 1$, the statement is "$10+9$ is divisible by 19", which is true. Assume that the statement holds for $n = k$, such that $\frac{5}{4} \cdot 8^k + 3^{3k-1}$ is divisible by 19. Then,

$$\begin{aligned}
\frac{5}{4} \cdot 8^{k+1} + 3^{3(k+1)-1} &= \frac{5}{4} \cdot 8 \cdot 8^k + 3^{3k+2} \\
&= 8 \cdot \frac{5}{4} \cdot 8^k + 3^3 \cdot 3^{3k-1} \\
&= 8 \cdot \frac{5}{4} \cdot 8^k + 8 \cdot 3^{3k-1} + 19 \cdot 3^{3k-1} \\
&= 8 \left(\frac{5}{4} \cdot 8^k + 3^{3k-1} \right) + 19 \cdot 3^{3k-1}
\end{aligned}$$

The first term is divisible by the inductive hypothesis, and the second term is clearly divisible by 19. This completes our proof, as we've shown the statement holds for $k+1$.

2 Negative pacman returns

Pacman has had a bit of a wild night, and wakes up feeling a bit under the weather. He starts at some location $(i, j) \in \mathbb{N}^2$ in the third quadrant, and is constrained walk on the infinite 2D grid and stay in the third quadrant (say, by walls along the negative x and negative y axes). Every second he does one of the following (if possible):

(i) Walk one step up, to $(i, j+1)$.

(ii) Walk one step right, to $(i+1, j)$.

For example, if he is at $(-5,0)$, his only option is to walk right 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?)

Solution:

Following the hint, we notice that it seems as though Pacman takes $i+j$ seconds to reach $(0,0)$ if he starts in position (i, j), regardless of 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. Thus, if we can prove this stronger statement, we'll also have proved that Pacman reaches $(0,0)$ in finite time. In order to simplify the induction, we will induct on the quantity $i+j$ rather than inducting on i and j separately.

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 n seconds 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 n further seconds 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 n more moves to get to $(0,0)$ no matter what path he takes. Thus, in either case, we have that Pacman will take a total of $n+1$ seconds (one for the first move and n for the remainder) in order to reach $(0,0)$, proving the claim for $n+1$.

One can also prove this statement without strengthening the inductive hypothesis. 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.

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.

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 up, we can immediately apply the inductive hypothesis, since Pacman will be in position (i,n). However, if Pacman moves to the right, 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 right forever: after i such moves, he'll hit the wall on the y-axis and be forced to move up. 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 right, one second making his first move up, 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 Losing Marbles

Two EECS70 GSIs have inexplicably run out of research topics to pursue, papers to read, or homeworks to create, and so they decide to play an incredibly boring game. (This is EECS after all.)

In the game, there is an urn that contains some number of red marbles (R), green marbles (G), and blue marbles (B). There is also an infinite supply of marbles outside the urn.

When it is a player's turn, the player may either:

(i) Remove one red marble from the urn, and add 3 green marbles.

(ii) Remove two green marbles from the urn, and add 7 blue marbles.

(iii) Remove one blue marble from the urn.

These are the only legal moves. The last player that can make a legal move wins. We play optimally, of course – meaning we always play one of the best possible legal moves.

(a) If the urn contains (R, G, B) red, green, and blue marbles initially, then determine the conditions on R, G, B for the first player to win the game. Prove it. In this case, does it matter what strategy the players use?

Hint: Assign each marble a weight, and argue that at every step, the combined weight will go down by exactly 1.

(b) Prove by induction that, if the urn initially contains a finite number of marbles at the start of the game, then the game will end after a finite number of moves.

Solution:

(a) We assign a weight to each marble such that after each step, the combined weight of all the marbles goes down by exactly 1.

We start with blue marbles and move (3), and so we assign each blue marble to have a weight of 1.

From move (2), we conclude that two green marbles must weigh $7+1$, and so each green marble must weight 4.

From move (1), we conclude that each red marble must weigh $3 \cdot 4 + 1 = 13$.

Therefore the combined weight of the marbles at the very start is $13R + 4G + B$. By construction, this weight will decrease by 1 every step (except when the weight is 0, in which case game over), so whenever $13R + 4G + B$ is odd, the first player will win.

(b) Let Φ_n be the combined weight after taking n moves. We will prove using induction, n moves, the combined weight Φ_n will be $13R + 4G + B - n$.

Proof. Proof by induction over n, the number of legal moves made.

Base case: Let $n = 0$, then $\Phi_0 = 13R + 4G + B - 0$, trivially.

Inductive hypothesis: Assume that for some $k \in \mathbb{N}, \Phi_k = \Phi_0 - k$.

Inductive step: Now consider the $k+1$-th turn. We know that any move will decrease the value of Φ_k by exactly 1, so we conclude that: $\Phi_{k+1} = \Phi_k - 1$. Subbing in our inductive hypothesis, we get $\Phi_{k+1} = (\Phi_0 - k) - 1 = \Phi_0 - (k+1)$, as desired. The claim follows by induction. □

Since R, G, B are all finite, then $13R + 4G + B$ is also finite, and we know from induction that the weight achieves 0 after $13R + 4G + B$ moves. We also know that since R, G, B are all natural numbers, the weight must be non-negative, and equals to 0 if and only if there are no more marbles.

Hence, after a finite number of moves, the game will end.

4 Nothing Can Be Better Than Something

In the stable matching problem, suppose that some jobs and candidates have hard requirements and might not be able to just settle for anything. In other words, in addition to the preference orderings they have, they prefer being unmatched to being matched with some of the lower-ranked entities (in their own preference list). We will use the term entity to refer to a candidate/job. A matching could ultimately have to be partial, i.e., some entities would and should remain unmatched.

Consequently, the notion of stability here should be adjusted a little bit to capture the autonomy of both jobs to unilaterally fire employees and employees to just walk away. A matching is stable if

- there is no matched entity who prefers being unmatched over being with their current partner;
- there is no matched/filled job and unmatched candidate that would both prefer to be matched with each other over their current status;
- similarly, there is no unmatched job and matched candidate that would both prefer to be matched with each other over their current status;
- there is no matched job and matched candidate that would both prefer to be matched with each other over their current partners; and
- there is no unmatched job and unmatched candidate that would both prefer to be with each other over being unmatched.

(a) Prove that a stable pairing still exists in the case where we allow unmatched entities.

(HINT: You can approach this by introducing imaginary/virtual entities that jobs/candidates "match" if they are unmatched. How should you adjust the preference lists of jobs/candidates, including those of the newly introduced imaginary ones for this to work?)

(b) As you saw in the lecture, we may have different stable matchings. But interestingly, if an entity remains unmatched in one stable matching, it/she must remain unmatched in any other stable matching as well. Prove this fact by contradiction.

Solution:

(a) Following the hint, we introduce an imaginary mate (let's call it a robot) for each entity. Note that we introduce one robot for each entity, i.e. there are as many robots as there are candidates+jobs. For simplicity let us say each robot is owned by the entity we introduce it for.

Each robot prefers its owner, i.e. it puts its owner at the top of its preference list. The rest of its preference list can be arbitrary. The owner of a robot puts it in their preference list exactly after the last entity they are willing to match with. i.e. owners like their robots more than entities they are not willing to match, but less than entities they like to match. The ordering of entities who someone does not like to match as well as robots they do not own is irrelevant as long as they all come after their robot.

To illustrate, consider this simple example: there are three jobs $1,2,3$ and three candidates A,B,C. The preference lists for jobs is given below:

Job	Preference List
1	$A > B$
2	$B > A > C$
3	C

The following depicts the preference lists for candidates:

Candidate	Preference List
A	1
B	$3 > 2 > 1$
C	$2 > 3 > 1$

In this example, 1 is willing to match A and B and it likes A better than B, but it'd rather be single than to be with C. On the other side B has a low standard and does not like being single at all. She likes 3 first, then 2, then 1 and if there is no option left she is willing to be forced into singleness. On the other hand, A has pretty high standards. She either matches 1 or remains single.

According to our explanation we should introduce a robot for each entity. Let's name the robot owned by entity X as R_X. So we introduce job robots R_A, R_B, R_C and candidate robots R_1, R_2, R_3. Now we should modify the existing preference lists and also introduce the preference lists for robots.

According to our method, 1's preference list should begin with its original preference list, i.e. $A > B$. Then comes the robot owned by 1, i.e. R_1. The rest of the ordering, which should include C and R_2, R_3 does not matter, and can be arbitrary.

For B, the preference list should begin with $3 > 2 > 1$ and continue with R_B, but the ordering between the remaining robots (R_A and R_C) does not matter.

What about robots' preference lists? They should begin with their owners and the rest does not matter. So for example R_A's list should begin with A, but the rest of the entities/robots (B, C, R_1, R_2, and R_3) can come in any arbitrary order.

So the following is a list of preference lists that adhere to our method. There are arbitrary choices which are shown in bold (everything in bold can be reordered within the bold elements).

Job	Preference List
1	$A > B > R_1 > \mathbf{3} > \mathbf{R_3} > \mathbf{R_2}$
2	$B > A > C > R_2 > \mathbf{R_1} > \mathbf{R_3}$
3	$C > R_3 > \mathbf{R_1} > \mathbf{R_3} > \mathbf{A} > \mathbf{B}$
R_A	$A > \mathbf{B} > \mathbf{C} > \mathbf{R_1} > \mathbf{R_2} > \mathbf{R_3}$
R_B	$B > \mathbf{R_1} > \mathbf{R_2} > \mathbf{R_3} > \mathbf{A} > \mathbf{C}$
R_C	$C > \mathbf{A} > \mathbf{R_2} > \mathbf{B} > \mathbf{R_1} > \mathbf{R_3}$

and the following depicts the preference lists for candidates and job robots:

Candidate	Preference List
A	$1 > R_A > \mathbf{3} > \mathbf{R_B} > \mathbf{2} > \mathbf{R_C}$
B	$3 > 2 > 1 > R_B > \mathbf{R_C} > \mathbf{R_A}$
C	$2 > 3 > 1 > R_C > \mathbf{R_A} > \mathbf{R_B}$
R_1	$1 > \mathbf{R_B} > \mathbf{2} > \mathbf{R_C} > \mathbf{3} > \mathbf{R_A}$
R_2	$2 > \mathbf{R_A} > \mathbf{R_C} > \mathbf{1} > \mathbf{3} > \mathbf{R_B}$
R_3	$3 > \mathbf{2} > \mathbf{1} > \mathbf{R_A} > \mathbf{R_C} > \mathbf{R_B}$

Now let us prove that a stable pairing between robots and owners actually corresponds to a stable pairing (with singleness as an option). This will finish the proof, since we know that in the robots and owners case, the propose and reject algorithm will give us a stable matching.

It is obvious that to extract a pairing without robots, we should simply remove all pairs in which there is at least one robot (two robots can match each other, yes). Then each entity which is not matched is declared to be single. It remains to check that this is a stable matching (in the new, modified sense). Before we do that, notice that an entity will never be matched with another entity's robot, because if that were so it and its robot would form a rogue couple (the robot prefers its owner, and the owner actually prefers their robot more than other robots).

(a) No one who is paired would rather break out of their pairing and be single. This is because if that were so, that entity along with its robot would have formed a rogue couple in the original pairing. Remember, the robot prefers its owner more than anything, so if the owner likes it more than their mate too, they would be a rogue couple.

(b) There is no rogue couple. If a rogue couple j and c existed, they would also be a rogue couple in the pairing which includes robots. If neither j nor c is single, this is fairly obvious. If one or both of them are single, they prefer the other entity over being single, which in the robots scenario means they prefer being with each other over being with their robot(s) which is their actual match.

This shows that each stable pairing in the robots and entities setup gives us a stable pairing in the entities-only setup. It is noteworthy that the reverse direction also works. If there is a stable pairing in the entities-only setup, one can extend it to a pairing for robots and entities setup by first creating pairs of owners who are single and their robots, and then finding an arbitrary stable matching between the unmatched robots (i.e. we exclude everything other than the unmatched robots and find a stable pairing between them). To show why this works, we have to refute the possibility of a rogue pair. There are three cases:

(a) A entity-entity rogue pair. This would also be rogue pair in the entities-only setup. The entities prefer each other over their current matches. If their matches are robots, that translates to them preferring each other over being single in the entities-only setup.

(b) A entity-robot rogue pair. If the entity is matched to their robot, our pair won't be a rogue pair since a entity likes their robot more than any other robot. On the other hand if the entity is matched to another entity, they prefer being with that entity over being single which places that entity higher than any robot. Again this refutes the entity-robot pair being rogue.

(c) A robot-robot rogue pair. If both robots are matched to other robots, then by our construction, this won't be a rogue couple (we explicitly selected a stable matching between left-alone robots). On the other hand, if either robot is matched to a entity, that entity is its owner, and obviously a robot prefers its owner more than anything, including other robots. So again this cannot be a rogue pair.

This completes the proof.

(b) We will perform proof by contradiction. Assume that there exists some job j_1 who is paired with a candidate c_1 in stable pairing S and unpaired in stable pairing T. Note that this means j_1 and c_1 both prefer to be with each other over being single. Since T is a stable pairing and j_1 is unpaired, c_1 must be paired in T with a job j_2 whom she prefers over j_1. (If c_1 were unpaired or paired with a job she does not prefer over j_1, then (j_1, c_1) would be a rogue couple in T, which is a contradiction.)

Since j_2 is paired with c_1 in T, it must be paired in S with some candidate c_2 whom j_2 prefers over c_1. This process continues (c_2 must be paired with some j_3 in T, j_3 must be paired with some c_3 in S, etc.) until all entitys are paired. Indeed, the last candidate c_n needs a partner in T and cannot be single (for the same reasons that c_1, c_2, and all the candidates before her need partners in T who they like better than their partners in S, to maintain stability). At this point, j_1 will be the only unpaired job, but to maintain the stability of T, we require j_1 to be paired in T with c_n. Yet we assumed j_1 was single, so we have reached a contradiction. Therefore,

our assumption must be false, and there cannot exist some job who is paired in a stable pairing S and unpaired in a stable pairing T. A similar argument can be used for candidates.

Since no job or candidate can be paired in one stable pairing and unpaired in another, every job or candidate must be either paired in all stable pairings or unpaired in all stable pairings.

Here is another possible proof:

We know that some job-optimal stable pairing exists. Call this pairing M. We first establish two lemmas.

Lemma 1. If a job is single in job-optimal pairing M, then it is single in all other stable pairings.

Proof. Assume there exists a job that is single in M but not single in some other stable pairing M'. Then M would not be a job-optimal pairing, so this is a contradiction.

Lemma 2. If a candidate is paired in job-optimal pairing M, she is paired in all other stable pairings.

Proof. Assume there exists a candidate that is paired in M but single in some other stable pairing M'. Then M would not be candidate-pessimal, so this is a contradiction.

Let there be k single jobs in M. Let M' be some other stable pairing. Then by Lemma 1, we know single jobs in M' will be greater than or equal to k. We also know that there are $n-k$ paired jobs and candidates in M. Then by Lemma 2, we know that the number of paired candidates in M' will be greater than or equal to $n-k$.

Now, we want to prove that if a job is paired in M, then it is paired in every other stable pairing. We prove this by contradiction. Assume that there exists a job m that is paired in M but is single in some other stable pairing M'. Then there must be strictly greater than k single jobs in M', and thus strictly greater than k single candidates in M'. Since there are strictly greater than k single candidates in M', there must be strictly less than $n-k$ paired candidates in M'. But this contradicts that the number of paired candidates in M' will be greater than or equal to $n-k$.

We also have to prove that if a candidate is single in M, then she must be single every other stable pairing. We again prove this by contradiction. Assume that there exists a candidate w that is single in M and paired in some other stable pairing M'. Then there are strictly greater than $n-k$ paired candidates in M', which means there are strictly greater than $n-k$ paired jobs in M'. This means there must be strictly less than k single jobs in M'. But this contradicts that the number of single jobs in M' will be greater than or equal to k.

Since we have proved both 1) If a job is single in M then it is single in every other stable pairing and 2) If a job is paired in M then it is paired in every other stable pairing (note that the contrapositive of this is if a job is single in any other stable pairing, then this job is single in M), we know that a job is single in M if and only if it is single in every other stable pairing.

Similarly, since we have proved both 1) If a candidate is single in M then she is single in every other stable pairing and 2) If a candidate is paired in M then she is paired in every other stable pairing, we know that a candidate is single in M if and only if she is single in every stable pairing. Thus we have proved that if a entity is single in one stable pairing, it is single in every stable pairing.

5 The Ranking List

Let's study the stable matching problem a little bit quantitavely. Here we define the following notation: on day j, let $P_j(M)$ be the rank of the job that applicant M proposes to (where the first application on her list has rank 1 and the last has rank n). Also, let $R_j(W)$ be the total number of applicants that job W has rejected up through day $j-1$ (i.e. not including the proposals on day j). Answer the following questions using the notation above.

(a) Prove or disprove the following claim: $\sum_M P_j(M) - \sum_W R_j(W)$ is independent of j. If it is true, also give the value of $\sum_M P_j(M) - \sum_W R_j(W)$. The notation, $\sum_M$ and $\sum_W$, simply means that we are summing over all applicants and all jobs.

(b) Prove or disprove the following claim: one of the **applicants or jobs** must be matched to something that is ranked in the top half of their preference list. You may assume that n is even.

Solution:

(a) **True.** On day $j = 1$, each applicant proposes to the first job on her list so $\sum_M P_1(M) = n$, and no job rejected any applicant through day 0, and therefore $\sum_W R_1(W) = 0$. Hence, $\sum_M P_1(M) - \sum_W R_1(W) = n$. In general, each time a job rejects an applicant on day $j-1$, $\sum_W R_j(W)$ increases by exactly 1. Similarly $\sum_M P_j(M)$ increases by exactly 1, since the rejected applicant proposes to the next job on her list on day j. Therefore $\sum_M P_j(M) - \sum_W R_j(W)$ stays constant and is independent of j. □

More formally, we can prove this by induction on j, with $j = 1$ as base case.
Induction Hypothesis: Assume $\sum_M P_j(M) - \sum_W R_j(W) = n$.
Induction Step: The quantity $\sum_W R_{j+1}(W) - \sum_W R_j(W)$ is exactly the number of men rejected by women on day j. But each of the rejected applicants proposes to the next job on their list on day $j+1$, and so this quantity is also equal to $\sum_M P_{j+1}(M) - \sum_M P_j(M)$. Equating the two, we get

$$\sum_W R_{j+1}(W) - \sum_W R_j(W) = \sum_M P_{j+1}(M) - \sum_M P_j(M).$$

Therefore,

$$\sum_M P_{j+1}(M) - \sum_W R_{j+1}(W) = \sum_M P_j(M) - \sum_W R_j(W)$$

and the right hand side is equal to n by the induction hypothesis. □

(b) **True.** Assume that no applicant is matched with a job in the top half of her preference list. Then each applicant must have been rejected at least $n/2$ times, for a total of at least $n^2/2$ rejections. This implies that at least one job must have rejected at least $n/2$ applicants (because if not, then the total number of rejections must be less than $(n/2) \cdot n$, contradiction). But now, by the improvement lemma, this job must be matched with an applicant she likes more than the $n/2$ men she rejected, meaning that the applicant she is matched with is in the top half of her preference list. □

Alternative Proof:

Assume towards contradiction that every applicant and every job is matched to someone who is ranked in the bottom half of their preference list.

Observe that an applicant M is matched to someone in the top half of her preference list if and only if $P_m(M) \leq n/2$, where m is the last day of the algorithm. Therefore, if M is matched to someone in the bottom half of her preference list, then $P_m(M) > n/2$, i.e., $P_m(M) \geq n/2 + 1$. Summing over the men gives us $\sum_M P_m(M) \geq n^2/2 + n$. By part (a), it follows that $\sum_W R_m(W) = \sum_M P_m(M) - n \geq n^2/2$.

Observe also that if $R_m(W) \geq n/2$, then by the improvement lemma, W must be matched to a job in the top half of her preference list. Therefore, from our assumption that W is matched to someone in the bottom half of her preference list, we get $R_m(W) < n/2$. Summing over the women gives us $\sum_W R_m(W) < n^2/2$. But this contradicts our earlier result above! □