Adam Novak
CSCI 270: Algorithm Design and Analysis
Shaddin, Fall 2021

Homework #1

Problem 1.

We will consider what happens in stable matching when we are guaranteed additional structure on the preferences.

Problem 1a: Suppose every woman prefers m to m'. Prove that for every stable matching, m prefers his assigned partner to the assigned parter of m'

PROOF
Observation: We are looking at every stable matching, meaning when there is no pair of m and w who likes each other better than their matched partner

Proof by contradiction: show that it's unstable for m to get assigned w/ someone he likes less than who m' is paired with

1. Suppose that in a given stable matching, m and w' are matched, and m' and w are matched. m prefers w to his current partner, w'
2. Since every woman prefers m to m', w also prefers m to m'
3. If w prefers m to m', and m prefers w to w', the matching is unstable.
4. CONTRADICTION ▮

Problem 1b: Prove there is a unique stable matching when all women have the exact same pref ranking

The unique stable matching is a set of matches where the most preferred man is paired with his best available pick, the next most preferred man is paired with his best available pick, and so forth.

ALGORITHM:
```
For each $m_i$ in w's preference ranking from most to least preferred
    For each w   in $m_i$'s pref list
        If w   is single
            Match w   and $m_i$
            Break
    End for
End for
```

PROOF OF MATCHING EXISTS (ie no polygamy)
Proof by induction: on each iteration i of outer for loop

1. Base case: at i=0, no matches exist
2. N+1 inductive step: at i=n+1, the inner for loop was just broken out of and w and m_i were matched
3. Both w and m_i were single before and now they are matched.
4. Number of matches increased by 1 and number of single people decreased by 2. ∎

PROOF OF PERFECT MATCHING EXISTS (ie no one is left beind)
Proof by contradiction

1. Assume there is a man who was not matched
2. Then there was also a f who was not matched, since we know that we have an even number of people and an equal number of men/women
3. That means the man checked every woman in his list and found no woman who was single
4. W is single
5. CONTRADICTION ∎

PROOF OF STABLE MATCHING EXISTS (ie no instabilities are present)
Proof by contradiction

1. Assume there is an instability where m + w' and m' + w are matched yet m prefers w to w' and both w and w' prefer m to m'
2. W and m prefer each other but are paired with someone else
3. If w prefers m to m', then every woman does too
4. M must have chosen before m' according to our algorithm above
5. W and m' were not paired up until m' chose, according to our algorithm
6. So w must have been single during m's selection
7. M would have chosen w over w' since w is higher on his preference list and is single
8. CONTRADICTION ∎

PROOF OF UNIQUENESS (ie only one solution)
Proof by induction, where t is the rank of men in the women's preference lis
Define "best possible pick" as the most preferred woman on a man's preference list where their pair would contribute to a stable matching

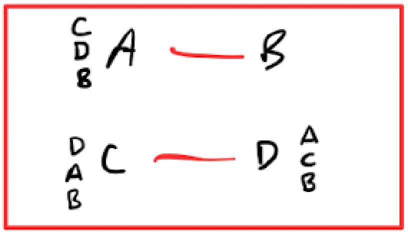
1. Base case: at t=1, where m_1 is the most preferred man
    a. If m_1 is not matched with his best possible pick, then he will be in a match where he prefers another woman over his current match
    b. That woman also prefers m_1 because all women prefer m_1 to the remaining men
    c. This would be unstable, so m_1 must get his best possible pick
2. Inductive case: t+1

   a. If m_{+1} does not get his best possible pick, then some man m where h>t+1 will be paired with m_{+1}'s best possible pick
   b. m_{+1} 's best possible pick will prefer m_{+1} over m because m_{+1} is preferred by all women to m
   c. This would be unstable, so m_{+1} must get his best possible pick.
3. There is only one woman whom each m can be paired with for a stable matching
4. Thus, the entire matching is unique ▌

Problem 2.

After graduating from USC, you go on to be the CEO of the successful software engineering firm WorkTogether. Over time, you realize that your employees are most productive when they work in pairs, and when the teams satisfy a certain “stability” condition. Formally, you have 2n employees (i.e., an even number), and each employee comes to you with an ordered list of the 2n−1 remaining employees in decreasing order of preference. You would like to pair up your employees to form a set M of n disjoint teams, with two employees per team, so that M is stable in the following sense: for every pair i and j of employees who are not partners in M, it is not the case that both i prefers j to his/her partner in M and j prefers i to his/her partner in M. This reminds you of the stable matching problem we described in class, and you wonder whether you can similarly compute a stable team assignment M.

Problem 2a: Exhibit an example in which a stable team assignment does not exist.



Here is an instance that’s unstable. Regardless of what B’s preference is, the assignment will always be unstable.
A and D prefer each other over C and B.
A and C prefer each other over D and B.
C and D prefer each other over A and B.

Problem 2b: Prove in this special case of ranking by reciprocal work similarity, a stable team assignment always exists and is unique

ALGORITHM
a_i, is the reciprocal work similarity between employee i and j

```
Sort $a_i$, from greatest to least
For each $a_i$, st i != j
    If neither employee i nor j are in relationships
        Pair i and j
End for loop
```

PROOF OF CORRECTNESS

Proof by contradiction

Unstable matching:

I --- L

J -- M

1. Suppose the result of our alg is *not* a stable team assignment
2. Then there exists a matching (pictured above) I-L and a matching J-M where $a_{i,}$ > a , and $a_{i,}$ > $a_{i,}$ (ie I and J would be a better match than both I and L and J and M)
3. According to our algorithm, potential pairs are checked in order of decreasing similarity, so $a_{i,}$ was checked before a , and $a_{i,}$
4. I and J were both paired at a , and $a_{i,}$, so I and J were both single when $a_{i,}$ was checked earlier
5. I and J must have been paired with each other
6. CONTRADICTION ∎

PROOF OF UNIQUENESS

Proof by contradiction

1. Suppose there is at least one other solution with a different matching than the one outputted by our algorithm
2. BASE CASE: $a_{i,}$ where $a_{i,}$ is the largest a value
    a. Suppose that our pairing of I and J from the greatest $a_{i,}$ does not exist in this alternate solution
    b. This means that later on the employees I and J will be paired with employees with whom they have a lower similarity than $a_{i,}$
    c. I and J will prefer each other to their pairings
    d. CONTRADICTION ∎
3. INDUCTIVE STEP: $a_{i,}$ where $a_{i,}$ is the next largest a value excluding a values containing employees which we have proved only have one unique matching through step 2 above
    a. Suppose that our pairing of I and J from the next largest $a_{i,}$ does not exist in this alternate solution
    b. This means that later on the employees I and J will be paired with employees with whom they have a lower similarity than $a_{i,}$
    c. I and J will prefer each other to their pairings
    d. CONTRADICTION ∎

Problem 2c: Show that you can compute the unique stable team assignment in O(n log n) when the instance is represented as a set of 2n such points, one per employee.

ALGORITHM FOR UNIQUE STABLE TEAM ASSIGNMENT

P is a list containing employees p_1, …, p_2 where p_i represents their comment average O(n)

Sort P from least to greatest O(nlogn)

potentialPairings = empty, doubly linked list O(1)

```
isPaired = array of size 2n O(2n)
i=1, j=2 O(1)
For each $p_i$, p in P O(n-1)
    Add a potentialPairing $pot_i$, to potentialPairings containing $p_i$, p $p_{i-1}$, $p_{+1}$,and dif = |$p_i$ -
    p | O(1)
    i++, j++ O(1)
Paired = empty list O(1)
While potentialsPairings is not empty O(2n)
    Remove pairing $pot_i$, at the front of potentialPairings O(1)
    If (isPaired[i] == false && isPaired[j] == false)
        isPaired[i] = true; isPaired[j] = true; O(1)
        Add diff to paired O(1)
        If $pot_i$, was not one of the ends of potentialPairings (meaning there was a
        potentialPairing on both sides of the node)
            Create new potentialPairing pot , containing $p_{i-1}$, $p_{+1}$ (the two
            employees which were also in potentialPairs with the employees that *just
            got paired* on this iteration) and dif = |$p_{i-1}$ - $p_{+1}$| O(1)
                    (****this is because you have to account for the new
                    potentialPairing that arises between two employees which
                    were on the opposite sides of a recent pairing*****)
            Binary search potentialPairings with $pot_{i-1}$, $_{+1}$'s dif value O(logn)
            Insert pot , into potentialPairings at this location O(1)
End while
Output paired
```

PROOF OF RUNTIME IN O(nlogn) where there are 2n employees

The runtime is O(nlogn).
There are two times that O(nlon) can be reached
First, when P is sorted from least to greatest at the beginning.
Secondly, during the pairing loop.

In regards to the pairing loop:

It makes sense that *while potential pairings is not empty* would be O(2n), because checks every potential pairing that was originally inserted, and at each loop at most 1 additional potentialPairing can be added.

Furthermore, in the case when a new potentialPairing needs to be created, the two employees which were also in potentialPairs with the employees that just got paired can be found in O(1) time since references to them are stored within the potentialPair object pot being looked at in the current iteration

Binary search potentialPairings O(logn), which is already in sorted order by dif value, to find where the new potentialPairing should go. Because potentialPairings is a doubly linked list, once the node which has the next largest/smallest dif value is found, the new node can be added in O(1) time

So O(nlogn) + O(2nlogn) = O(nlogn)

PROOF OF MATCHING EXISTS (ie no polygamy)
Proof by induction: on each iteration i of while loop

1. Base case: at i=0, no matches exist
2. N+1 inductive step: at i=n+1, if i and j were not paired before, then they are now paired together
3. Number of matches increased by 1 and number of single people decreased by 2. ▮

PROOF OF PERFECT MATCHING EXISTS (ie no one is left behind)
Proof by contradiction

1. Assume there is an employee who was not matched
2. Then there must be another employee who was not matched since there are an even number of them
3. Observation: every employee starts off with at least one potential pairing
4. Whenever an employee loses one of their potential pairings because their the employee in that pairing was matched, they also gain a new potential pairing at the same time. Thus, every employee always either has a match or at least one potential pairing
5. Thus, at the end of the algorithm, since there are two employees who are not matched and are both single, they must be in a potential pairing with each other
6. Our algorithm checks every potential pairing and pairs single people together
7. These two employees would have been paired together by our algorithm
8. CONTRADICTION ▮

PROOF OF STABLE MATCHING EXISTS (ie no instabilities are present)
Proof by contradiction

1. Assume there is an instability where p1 and p1' and p2 and p2' are matched yet the dif value between p1 and p1' as well as p2 and p2' is actually higher than betw p1 and p2
2. Because potentialPairing(s) are added to list potentialPairings in order of increasing dif values, potentialPairings starts off sorted
3. Because new potentialPairing(s) created during the while loop are added to potentialPairings at a location sorted by their dif values, potentialPairings always remains sorted
4. Additionally, every new potentialPairing created will have a dif value greater than all dif values preceding it, since the two employees in the new pairing are at least the distance of the two employees which were just matched together plus some extra distance apart
5. potentialPairing(s) are checked from potentialPairings in order of increasing dif values

6. p1 and p2 must have been checked before p1 and p1’ and before p2 and p2’ since they have a lower dif value
7. Since p1 and p2 were not paired until they were checked with p1’ and p2’ later on, then they are single at this point in time
8. Since p1 and p2 were single when they were checked earlier, then they were paired
9. CONTRADICTION ▌

PROOF OF UNIQUENESS (ie only one solution)
Observation A: the two first potential pairings for each employee are with employees with the closest comment average to them, meaning they produce the lowest possible dif values with each other

Proof by contradiction

1. Suppose there is at least one other solution with a different matching than the one outputted by our algorithm
2. BASE CASE: potentialParing $pot_{i,}$ between p_i and p with the lowest dif value
    e. Suppose that our pairing between p_i and p does not exist in this alternate solution
    f. This means that later on the employees p_i and p will be paired with employees with whom they have a greater dif than with each other
    g. p_i and p will prefer each other to their pairings
    h. CONTRADICTION ▌

3. INDUCTIVE STEP: potentialParing $pot_{i,}$ between p_i and p with the next lowest dif value, excluding potentialPairings containing employees which we have proved only have one unique matching through step 2 above
    i. Suppose that our pairing between p_i and p does not exist in this alternate solution
    j. This means that later on the employees p_i and p will be paired with employees with whom they have a greater dif than with each other
    k. p_i and p will prefer each other to their pairings
    l. CONTRADICTION ▌

Problem 3.

You would like to introduce yourself to all of the CPs in CSCI 270, which requires you to visit all of their office hours, held in the SAL common area. Let's say that there are n CPs, and for each, you are given one interval [si,fi] when they hold office hours. DifferentCPsmayhaveoverlapping office hours. We assume that you only have time for very brief (instantaneous) visits. In other words, if you decide to visit the SAL common area at 10:00am then you meet exactly the CPs whose office hours include 10:00am. Since you leave immediately afterwards, you meet no other CPs during this visit, not even a CP whose office hours start at 10:01am.

You would like to find a smallest (minimum-cardinality) set of visit times that together hit every CP's office hour interval. Give (and of course analyze) an algorithm that runs in time O(n log n) and finds a smallest set of visit times to meet all the CPs.

3.1 Give and analyze an algorithm that finds a smallest set of visit times

Define OH: office hours

```
ALGORITHM:
earliestEndTimes = sorted queue of CPs by their OH end times (in order from earliest to latest)
O(nlogn)
earliestStartTimes = sorted queue of CPs by their OH start times (in order from earliest to latest)
O(nlogn)
Visit times = empty list

CP = dequeue from earliestEndTimes
Visit times += CP's end time
for each overlappingCP in earliestStartTimes  O(n)
    If overlappingCP.starttime <= CP.endtime
        CP.visited = true
    If overlappingCP.starttime > CP.endtime
        CP = dequeue from earliestEndTimes until CP.visited = false
        Visit times += CP's end time
Output visit times
```

PROOF OF RUNTIME IN O(nlogn)

Clearly it takes O(nlogn) to sort the two lists of CPs by OH start/end times using merge sort or quick sort.

Now, prove that the outer for loop only runs O(n) times in total

Outer for loop iterates over each CP in earliestStartTimes, which would be O(n) total
The loop towards the bottom of the algorithm will iterate through the entire earliestEndTimes list which will be O(n) total. The quantity of these iterations is completely independent from the iterations on the outer loop
Thus, the outer for loop will have a O(n) + O(n) = O(2n) runtime in total
This means the algorithm will have an amortized run time of O(2n + nlogn) = O(nlogn)

PROOF OF CORRECTNESS: that our algorithm outputs a smallest set of visit times to meet all CPs

Proof by induction using stays ahead method
Take an optimal solution

Inductive hypothesis: our algorithm visits the most possible number of CPs with the fewest number of visits up to a certain point in time

1. Base case: at iteration i = 0
    a. Our solution has not visited any OHs and neither has the optimal solution, so both solutions are equally good
    b. Since every CP must be visited, we must also visit CP a, the CP with the earliest finishing OHs
    c. Since CP a has the earliest finishing OHs, there are no CPs which start and finish before it, but there might be some CPs which start during a's OH and finish later
    d. If we visit at the last moment of a's OHs, we are ensuring that we visit CP a and every CP with OHs that started before a's OHs
    e. Thus, for an optimal solution (which must also visit a), we visit as many or more CPs in an a-hitting visit as the optimal solution does
2. Inductive step: inductive hypothesis holds for n, prove true for iteration i = n + 1
    a. We know that after n visits, our solution has visited all the CPs with start times before our last visit time in as many or fewer visits than the optimal solution
    b. Prove by contradiction: suppose that at iteration n+1 (aka after n+1 visits), the optimal solution has visited the greatest possible number of CPs with the fewest visits up to a point in time
    c. The optimal solution visited either before or after when we visited
    d. Case 1: optimal solution visited before we visited
        i. Our algorithm chooses the CP whom we havent yet visited with the earliest possible finish time, so there is no CP we havent visited yet which finishes earlier than our n+1 visit
        ii. Then we must have also seen all the CPs whom the optimal solution saw, plus more possibly
        iii. CONTRADICTION
    e. Case 2: optimal solution visited after we visited
        i. Our algorithm chooses the CP with the earliest possible finish time

    ii. The OHs of the CP we are basing our visit off of will already be finished when optimal solution visits
    iii. Then the optimal solution is guaranteed to not have visited that CP we are basing our visit off of
    iv. CONTRADICTION

3. We have proven that our algorithm visits the most possible number of CPs up to a certain point in time with the fewest visits
4. If that “certain point in time” is the latest time that any CP has OHs, then we prove that our algorithm visits all CPs in the fewest visits ∎

Problem 4.

You are asked to help the captain of the USC tennis team to arrange a series of matches against UCLA's team. Each team has n players; the tennis rating (a positive number, where a higher number can be interpreted to mean a better player) of the ith member of USC's team is ai and the tennis rating for the kth member of UCLA's team is bk. To keep things simple, we will assume that all ai and all bk are distinct. You would like to set up a competition in which each person plays exactly one match, and that match is against a player from the opposite school. Because you get to select who plays against whom, your goal is to make sure that in as many matches as possible, the USC player has a higher tennis rating than his or her opponent.

4.1 Give and analyze an algorithm that finds a smallest set of visit times

```
ALGORITHM:
Sort USC players from best to worst
Sort UCLA players from best to worst
While we still have unmatched players
        If our best player can beat their best player, send our best player to face them
        If our best player cannot beat their best player, send our worst player to face them
```

PROOF OF RUNTIME IN O(nlogn)
Sorting players from best to worst takes O(nlogn)
We could implement the algorithm with two doubly linked lists and three iterators: one for going down our best players, one for going up our worst players, and one for going down their best players.
We match one of our players after each iteration, so the outer while loop will take total O(n) where n is the number of USC players
Each iteration of the while loop takes O(1) , or constant time, because we just iterate a few iterators to find the next item in the linked list

PROOF BY INDUCTION
Inductive hypothesis: as many of the faceups as possible result in USC wins

1. Base case:
    a. Case 1: If our best player can beat their best player, then they face each other and we have a win
    b. Case 2: If our best USC player cannot beat their best UCLA player, then there is no way we could have beat their best player, so there will be a loss no matter which USC player faces them
    c. In either of the two cases above, our solution is just as good as an optimal solution

2. Assume IH is true for k faceups, prove true for k+1 faceups
    a. PROOF BY CONTRADICTION: Assume after the k+1th player we send out, we have a worse solution with fewer than optimal wins
    b. Case 1: we sent our best player out and they COULD HAVE won a match but they didnt
        i. We only send out our best player if they can win a match
        ii. CONTRADICTION ▮
    c. Case 2: we sent out our worst player which resulted in a loss but could have resulted in a win later on if they had faced some UCLA player that was worse than them
        i. We can still match our best player to that UCLA player which is worse^ for a win, since our best player is better than our worst player
        ii. We have an equal number of wins as the optimal solution
        iii. CONTRADICTION ▮