CS 70 Discrete Mathematics and Probability Theory
Spring 2024 Seshia, Sinclair

HW 02

Due: Saturday, 2/3, 4:00 PM
Grace period until Saturday, 2/3, 6:00 PM

Sundry

Before you start writing your final homework submission, state briefly how you worked on it. Who else did you work with? List names and email addresses. (In case of homework party, you can just describe the group.)

1 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?

2 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.

3 Upper Bound

Note 4

(a) In the notes, we show that the stable matching algorithm terminates in at most n^2 days. Prove the following stronger result: the stable matching algorithm will always terminate in at most $(n-1)^2 + 1 = n^2 - 2n + 2$ days.

(b) Provide a set of preference lists for 4 jobs and 4 candidates that will result in the upper bound from part (a) when running the Propose-and-Reject algorithm. Verify this by running the Propose-and-Reject algorithm on your preference lists.

1 (a) (J_1, C_1) (J_2, C_2) ... (J_n, C_n)

(b) same

(c) This tells us that # stable matching ≥ 1
As job propose is candidate pessimal while candidate propose is candidate optimal
But they are exactly the same, thus
for each candidate m_1, S_1(job propose) $= S_2$(candidate propose)
So, there does not exist an intermediate matching $\Rightarrow$ This is the only stable matching

2 for A 1 2 | 1 BA
B 2 1 | 2 AB

exist 2 pair (1, A) (2, B) & (4, B) (2, A)

Having another pair exactly like the above example but with (3, 4) on top preference of (C, D) and vice versa we would create $2 \times 2 = 4$ new stable matchs, As matchs A, 1, B, 2's matchs would still be stable and same for C, D, 3, 4 Thus every $n+2$, we have $2 \cdot 2^{\frac{n}{2}} = 2^{\frac{n+2}{2}}$

3 can use discussion (b), at least one candidate only had 1 proposal
If every other candidate took at most $n-1$ days of rejection
Termination date $\leq (n-1)^2 + 1 = n^2 - 2n + 2$

(b) $4^2 - 8 + 2 = 10$

	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10
1	A	AD	A	AB	A	A	AC	C	C	C
2	B	B	BD	D	D	DC	D	AD	D	D
3	CD	C	C	C	CB	B	B	B	AB	B
4										A

	A	B	C	D
A	1	2	3	4
B	2	1	3	4
C	3	2	1	4
D	3	1	2	4

	A	B	C	D
1	C	A	D	B
2	D	B	C	A
3	B	A	C	D
4	D	A	C	B

4 Build-Up Error?

Note 5

What is wrong with the following "proof"? In addition to finding a counterexample, you should explain what is fundamentally wrong with this approach, and why it demonstrates the danger of build-up error.

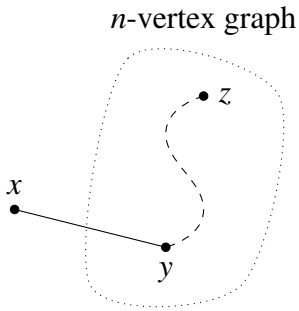
False Claim: If every vertex in an undirected graph has degree at least 1, then the graph is connected.

Proof? We use induction on the number of vertices $n \geq 1$.

Base case: There is only one graph with a single vertex and it has degree 0. Therefore, the base case is vacuously true, since the if-part is false.

Inductive hypothesis: Assume the claim is true for some $n \geq 1$.

Inductive step: We prove the claim is also true for $n+1$. Consider an undirected graph on n vertices in which every vertex has degree at least 1. By the inductive hypothesis, this graph is connected. Now add one more vertex x to obtain a graph on $(n+1)$ vertices, as shown below.



All that remains is to check that there is a path from x to every other vertex z. Since x has degree at least 1, there is an edge from x to some other vertex; call it y. Thus, we can obtain a path from x to z by adjoining the edge $\{x, y\}$ to the path from y to z. This proves the claim for $n+1$. □

5 Proofs in Graphs

Note 5

(a) On the axis from San Francisco traffic habits to Los Angeles traffic habits, Old California is more towards San Francisco: that is, civilized. In Old California, all roads were one way streets. Suppose Old California had n cities ($n \geq 2$) such that for every pair of cities X and Y, either X had a road to Y or Y had a road to X.

Prove that there existed a city which was reachable from every other city by traveling through at most 2 roads.

[*Hint:* Induction]

(b) Consider a connected graph G with n vertices which has exactly $2m$ vertices of odd degree, where $m > 0$. Prove that there are m walks that *together* cover all the edges of G (i.e., each

4 Inductive step inducted from n to n+1
In the inductive step, one should only start from n+1 to reduce to n, not the other way around. As one can't by such extension is sufficient to encapsulate all size n+1 cases.

5 (a) Base $n=2$ ⋮ obviously true

Inductive Step: For vertices of size n, exist a vertex such that all other vertices reach this vertex in less than 2 steps.

Suppose size n+1, Randomly delete a node, say X, so the size is reduced to n
Apply inductive hypothesis, the target vertex as Y (name)
Now classify the remaining vertices into 2 groups: 1 step group & 2 steps group
Now put X back, we see that if X has a directed edge pointed to any of 1 step group or Y the proposition is automatically satisfied.
So, we consider the remaining case that X is pointed by group 1 & Y. We note that X at last needs to point to one of group 2, or else X is the desired vertex.
Consider the set of vertex containing only the group 2 vertices that X points to. Apply the inductive hypothesis as the size has to be smaller than n+1. The resulting vertex call it Z. Z is the desired vertex for the n+1 problem. Why? As X reaches Z in one step while group 1 & Y reaches Z in 2 step via X, and group 2 reaches Z within 2 steps. Thus Q.E.D

(b) To turn G even we simply partition the set of odd degree vertices into two groups, and pair each member from one set with an unique member from the other set, and add an edge between each pair.
Thus, we would have a connected even graph G', which has an Eulerian tour.
Now, we can partition the Eulerian tour into m walks of the original graph G by deleting each added edge from the Eulerian tour and treat each segment as a walk with the final return to start deleted. Thus we would have 1+m-1=m walks that cover each edge exactly once.

(c) only if: If G has an odd length tour. The after coloring each vertex in the tour as blue or red, we would end up with the final vertex having same color with the starting vertex. Thus contradiction. So there mustn't be any odd length tour.

If: prove by induction no odd length → bipartite
Base: G has only one vertex → bipartite
Inductive Step: G has n+1 vertex
Delete one vertex say X, doesn't affect the non-odd-tourney of G.
Thus new G' of size n satisfy induction hypothesis and color each vertex as red or blue.
Add back X, say connected to m other vertex. If the color doesn't unite change the color all toned one at a time and switch neighboring colors. This would be okay as switching color of m_1 doesn't affect color of $m_2 \ldots m_m$. As if it did, then there exist an odd length path connecting m_1 or m_2, but via X, producing a tour of odd length of m. Thus the switching scheme works. and we color X as blue, making graph G bipartite

edge of G occurs in exactly one of the m walks, and each of the walks should not contain any particular edge more than once).

[*Hint:* In lecture, we have shown that a connected undirected graph has an Eulerian tour if and only if every vertex has even degree. This fact may be useful in the proof.]

(c) Prove that any graph G is bipartite if and only if it has no tours of odd length.

[*Hint:* In one of the directions, consider the lengths of paths starting from a given vertex.]

6 (Optional) Nothing Can Be Better Than Something

Note 4

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, each job/candidate prefers being unmatched rather than be matched with those below a certain point in their preference list. Let the term "entity" 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/or 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;
- there is no matched job and matched candidate that would both prefer to be matched with each other over their current partners; and
- 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 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, they must remain unmatched in any other stable matching as well. Prove this fact by contradiction.

6 (a) For each job/candidate list there are number of partners that they prefer to be unmatched. say m. add in their pref-list a nonexisting entity P_i and insert P_i infront of the m entities. Thus creating $2n$ candidates & jobs. In the list of P_i s, make their top priority their corresponding entities, add the rest priorities in order of P_i s and then real entities.
Applying stable matching algorithm produces a valid result, as, in the case of employer proposal – rejection, the candidates first get take jobs and then if they receive real job offers (and no other P_i could get there), than their respective P_i leaves to take candidates. and they would always have real job. Similarly for jobs, after being rejected by the $n-m$ employees (if not they have real employees), then they get paired with empty entities as empty ones have them as top priority. Thus the result is a legal stable matching.

(b) Suppose the contrary that exist stable matching of (C_i, P_i) say A while another instance (C_i, J_j) say B. In order for $C(i, J_i)$ to be stable C_i must prefer J_j to P_{C_i} while J_j prefer C_i to P_j. Thus in A. (J_j, C_j) must exist.