Homework 2

Keshav Ramamurthy

February 9, 2026

Contents

1 Airport

Notice this holds for $n = 1$, and we can do casework to verify.
For the case $n = 1$, let's say the airports are A, B, C such that A and B have the smallest distance between any two airports. Then, we have that A flies to B, B to A, and C to either A or B, where we'd have that no one flew to C and the inductive hypothesis is true.
Now, suppose via our inductive hypothesis that this condition holds for $2n + 1$ airports assuming that the distance between any two airports is distinct.
Now, when we have an arbitrary $2n + 3$ airports, suppose A and B are again the pair of airports with the smallest distance between any 2 airports.
Notice that A flies to B, B flies to A and that we have two cases that result.

- If no outside airport flies to A or B, notice that this condition reduces the problem to $2n + 1$ airports, and thus it is true via our inductive hypothesis
- The second case is at least one airport flies to A or B or both,, which we'll solve using the pigeonhole principle.

Notice that both A and B have flights leading towards them. Thus, out of the remaining $2n + 1$ airports, in order for our hypothesis to be false, they must each have a flight "towards" them.
However, in this case we have at least one flight towards either A or B and thus we only have $2n$ flights remaining for $2n + 1$ airports, meaning at least one has no flights towards them and it holds for $2n + 3$ and we are done.

2 Universal Preference

2.1 Part A

The propose-and reject algorithm would result in the pairing

$$\{C_1, J_1\}, \{C_2, J_2\}, ..., \{C_n, J_n\}$$

This is because on the first day, the best candidate would be offered every job, in which they'd put the best job on a string, saying no to the rest.
This results in the pairing $\{C_n, J_n\}$
The next day, the second best candidate is offered jobs $\{C_{n-1}, C_{n-2}, ..., C_1\}$, where the second best candidate would put the second best on their string, saying no, etc.
This eventually results in $\{C_{n-1}, J_{n-1}\}, \{C_{n-2}, J_{n-2}\}, ..., \{C_1, J_1\}$.

2.2 Part B

If the candidates propose, the pairings are the same, as the best person will still pick up the best job, the second best person picking up the second best, etc... The pairing stays $\{J_1, C_1\}, \{J_2, C_2\}, ..., \{J_n, C_n\}$

2.3 Part C

In this situation, because of the fact that if the candidates or employers propose, since they yield the same exact pairing, it shows that there is only 1 stable pairing in this scenario, as otherwise any pair $\{J_x, C_a\}, \{J_y, C_b\}$ where $a > b$, and $x < y$, would result in an unhappy Candidate A, meaning there can't be "mismatched" pairings.

3 Pairing Up

We can construct jobs in the following manner to meet this condition:
Suppose $n = 2$. Then, we can make jobs in the following manner

$$J_1 : C_1, C_2, J_2 : C_2, C_1, C_1 : J_2, J_1 : C_2, J_1, J_2$$

Where the order from left to right denotes best to worst. Notice that if jobs propose, there is 1 set of jobs, and if the candidates get first choice, there's another for $2^{\frac{2}{2}} = 2$ ways.
Let's proceed with induction. Suppose this holds for the first $2n - 2$ numbers. Then when considering $2n$, let's build it in the following manner:
Suppose we pair adjacent jobs up, into a total of n groups, and then pair adjacent candidates for n groups. Notice that arbitrarily ordering the groups $1, 2, 3, ..., n$, we can make

$$J_{2i-1} : C_{2i-1}, C_{2i}, J_{2i}, C_{2i}, C_{2i-1}, C_{2i-1}, J_{2i}, J_{2i-1}, C_{2i}, J_{2i-1}, J_{2i}$$

The rest of jobs / candidates in these preferences lists can just be written chronologically. Then notice for each group, we'd have a matching pairing either through "happy" employers or happy candidates, yielding

$$\prod_{i=1}^{n} 2 = 2^n = 2^{\frac{2n}{2}} \quad \square$$

4 Short Tree Proofs

4.1 Part A

Every connected component in an acyclic graph would be connected by definition, and notice the fact that since G is acyclic, and that every node in a connected component is reachable from any other nose, it satisfies the conditions for being a tree.

4.2 part B

We showed that every connected component in an acyclic graph is a tree, and a tree with V vertices has $V-1$ edges so if there were k connected components, we'd have

$$\sum_{i=1}^{k} E_i = \sum_{i=1}^{K} V_i - 1 \rightarrow |E| = |V| - k \quad \square$$

4.3 Part C

Suppose by contradition, this graph, where $|E| = |V|$ exists yet it is acyclic. Then, notice that an acyclic graph with k($k \geq 1$) connected components would have

$$|E| = |V| - k$$

Thus, we can bound

$$|E| = |V| - k < |V|$$

, which is a contradiction.

5 Proofs in Graphs

5.1 Part A

Let's show this by building m pairs, and then showing that these together cover all the edges of G.
Suppose we pair up random edges of G, creating m pairs. Construction of an edge(could be repeated) between them would yield adding 1 edge to every odd vertex, making them all have even degree, and thus, it contains an Eulerian Tour.
Notice that each created edge "removed" splits the tour into one more walk. For example, removal of the first edge splits the tour into 1 walk. Removal of our second created edge splits the tour into 2. This continues into the removal of the mth edge that splits it into m walks that contain all the edges. $\square$

5.2 Part B

Notice that by definition of a bipartite graph,
we can label vertices such that each vertex labelled zero is only connected via an edge to a vertex labelled 1 and vice versa.
From there, Notice that any tour on this bipartite graph must alternate: even, odd, even, ... or
odd, even, odd, ...
Notice that if this tour had odd length, there would be a contradiction a path alternates parities and the parity must be the same as the starting vertex.
Thus, if the tour had odd length, the graph can't be bipartite, and if the graph is bipartite, every tour must have even length because of the parity condition, proving both sides $\square$.