Section 1: Solutions

1. Gale–Shapley review

Consider the following lists of preferences:

$$
\begin{aligned}
p_1 &: r_3 > r_1 > r_2 > r_4 \\
p_2 &: r_2 > r_1 > r_4 > r_3 \\
p_3 &: r_2 > r_3 > r_1 > r_4 \\
p_4 &: r_3 > r_4 > r_1 > r_2
\end{aligned}
$$

$$
\begin{aligned}
r_1 &: p_4 > p_1 > p_3 > p_2 \\
r_2 &: p_1 > p_3 > p_2 > p_4 \\
r_3 &: p_1 > p_3 > p_4 > p_2 \\
r_4 &: p_3 > p_1 > p_2 > p_4
\end{aligned}
$$

(a) Run the Gale–Shapley algorithm on the instance above, with p_i proposing. When multiple p_i are free to propose, choose the one with the smallest index (e.g., if p_1 and p_2 are both free, have p_1 propose).

Solution:

The steps of the Gale–Shapley Algorithm with the p_i with lowest index proposing first:

p_1 chooses r_3	(p_1, r_3)
p_2 chooses r_2	$(p_1, r_3), (p_2, r_2)$
p_3 chooses r_2	$(p_1, r_3), (p_3, r_2)$
p_2 chooses r_1	$(p_1, r_3), (p_2, r_1), (p_3, r_2)$
p_4 chooses r_3	$(p_1, r_3), (p_2, r_1), (p_3, r_2)$
p_4 chooses r_4	$(p_1, r_3), (p_2, r_1), (p_3, r_2), (p_4, r_4)$

(b) Run the Gale–Shapley algorithm again on the instance above, with p_i proposing. When multiple p_i are free to propose, now choose the one with the *largest* index. Do you get the same result?

Solution:

The steps of the Gale–Shapley Algorithm with the p_i with highest index proposing first:

p_4 chooses r_3	(p_4, r_3)
p_3 chooses r_2	$(p_3, r_2), (p_4, r_3)$
p_2 chooses r_2	$(p_3, r_2), (p_4, r_3)$
p_2 chooses r_1	$(p_2, r_1), (p_3, r_2), (p_4, r_3)$
p_1 chooses r_3	$(p_1, r_3), (p_2, r_1), (p_3, r_2)$
p_4 chooses r_4	$(p_1, r_3), (p_2, r_1), (p_3, r_2), (p_4, r_4)$

We ended up with the same result!

(c) Run the Gale–Shapley algorithm on the instance above, with r_i proposing. When multiple r_i are free to propose, choose the one with the smallest index. Do you get the same result?

Solution:

The steps of the Gale–Shapley Algorithm with r_i proposing:

r_1 chooses p_4	(p_4, r_1)
r_2 chooses p_1	$(p_1, r_2), (p_4, r_1)$
r_3 chooses p_1	$(p_1, r_3), (p_4, r_1)$
r_2 chooses p_3	$(p_1, r_3), (p_3, r_2), (p_4, r_1)$
r_4 chooses p_3	$(p_1, r_3), (p_3, r_2), (p_4, r_1)$
r_4 chooses p_1	$(p_1, r_3), (p_3, r_2), (p_4, r_1)$
r_4 chooses p_2	$(p_1, r_3), (p_2, r_4), (p_3, r_2), (p_4, r_1)$

No, the result is different when we have the r_i propose as opposed to the p_i.

2. The number of stable matchings

In the previous problem, we saw two distinct stable matchings for the same instance (depending on whether the p_i or r_i are the ones to propose). Is it possible to have an instance of the stable matching problem with more than 2 stable matchings? If so, give an instance with at least 3 stable matchings. If not, prove that every instance has at most 2 stable matchings.

Solution:

Consider the following "hexagon preference cycle" instance:

$$\begin{aligned} A &: 1 > 2 > 3 \\ B &: 2 > 3 > 1 \\ C &: 3 > 1 > 2 \end{aligned}$$

$$\begin{aligned} 1 &: B > C > A \\ 2 &: C > A > B \\ 3 &: A > B > C \end{aligned}$$

This instance has three stable matchings:

$$(A, 1), (B, 2), (C, 3)$$
$$(A, 2), (B, 3), (C, 1)$$
$$(A, 3), (B, 1), (C, 2)$$

Review of graph concepts

- **Degree:** The number of edges connected to a vertex.
- **Path[1]:** A list of vertices $v_1, v_2, \ldots, v_k$ such that each $\{v_i, v_{i+1}\}$ is an edge. ((v_i, v_{i+1}) in a directed graph)
- **Cycle[2]:** A path $v_1, v_2, \ldots, v_k$ with $v_1 = v_k$.
- **Simple path[3]:** A path with all distinct vertices

[1] Also known as a *walk* by other sources.

[2] Also known as a *closed walk* by other sources.

[3] Also known as a *path* by other sources.

- **Simple cycle[4]:** A cycle with all distinct vertices, except the first/last.
- **Connected:** There is a path between any two vertices in the graph.
- **Tree:** A connected, acyclic (no cycles) graph.
- **Rooted tree:** A tree with a designated vertex called the *root*. (Note: Words like "parent" and "child" require a root. For non-rooted trees, say "neighbor" to refer to vertices connected by a single edge to the current one.)

3. Proof-writing workshop

Attached as an appendix to this handout, there are 4 sample proofs of the following statement:

Every tree with at least 2 vertices has at least 2 vertices of degree 1.

(a) Take a minute to think about the problem yourself. (It's okay if you don't have a proof.)

(b) Read each sample proof. Discuss with people around you:

  (i) Is it correct? (Are there false statements?)

  (ii) Is it complete? (Are there unjustified claims, unused hypotheses, or undefined notation?)

  (iii) Is it concise? (Are there excessive details, unnecessary notations, or irrelevant arguments?)

  (iv) Is it clear? (Are the main ideas obvious or buried? Could stylistic choices like paragraph breaks, diagrams, bullets, etc. be improved? Are there spelling, grammar, or formatting errors?)

  (v) What do you like about the proof? How would you improve this proof?

Solution:

Many of these criteria are subjective, so it is perfectly valid to disagree. We will not take points off for bad style (unless it is excessively egregious), but we really appreciate thoughtful editing!

(a) Sample Solution? 1

- Correctness: No issues.
- Completeness: Does not explicitly use the fact that there are at least 2 vertices. (Need this to say "the rest have degree 2," since when there is 1 vertex it has degree 0.) Also missing "Let n be the number of vertices in the graph," though this is a minor point.
- Conciseness: No issues. In fact, fantastic.
- Clarity: The main idea is clear, mainly because it is so concise. Font size is a bit small. This is the default 10pt font. Prefer 11pt or 12pt, or increase the margins, though this is a minor point.

(b) Sample Solution? 2

- Correctness: The second to last line, "By IH, each tree has at least 2 vertices of degree 1," is not correct, because the resulting trees from the previous line may not have at least 2 vertices each. The author needs to treat the case of getting single-vertex trees.
- Completeness: No issues.
- Conciseness: No issues.
- Clarity: Prefer to separate out IH explicitly. Diagrams are great and helped point out the main ideas, but they obscured the correctness issue. A more generic image that shows the case of getting a single-vertex tree is better. Easy-to-read formatting with underlines and indenting.

(c) Sample Solution? 3

[4] Also known as a *cycle* by other sources.

- Correctness: One can argue whether this is a correctness or completeness error, but it is generally not advised to do induction by building up graphs from smaller graphs—instead, remove a vertex from the larger graph to get the smaller graph, as in the previous sample solution. One can ostensibly fix this by adding a proof that all trees can be constructed by adding a new vertex to a smaller tree, but this fact is not obvious and would probably double the length of this proof, so it is not advised. Without this extra part, the proof is not correct. Generally, avoid this whole situation by always starting with the bigger graph and removing a vertex to apply the IH.

  A second point, the sentence in Case 3, "In this case, the graph would be left with no vertices of degree 1," is just incorrect—there may have been vertices of degree 1 beyond just u and v, so you cannot conclusively say that attaching w to u and v results in no degree 1 vertices. Luckily, this is also just a completely unnecessary statement and can be deleted.

  And a minor point, edges of undirected graphs are sets (use $\{\}$), not ordered pairs as written in Case 3.

- Completeness: See above.

- Conciseness: There are a several superfluous phrases: "undirected" in the base case, "Since we are interested in connected trees" in the first paragraph of inductive step, "In this case, …luckily" in Case 3, arguably the entire last paragraph, etc. It is not egregious, but editing these out would substantially shorten the proof.

- Clarity: Google Docs without the symbol browser/equation editor/italicized math is a little harder to read, compared to proper LaTeX/handwriting, but it's okay. Grammatically, the frequent use of "u, v" when it is more proper to write "u or v," makes parsing the text more difficult. Paragraph breaks and use of bold are good. Main ideas are sufficiently understandable.

(d) Sample Solution? 4

- Correctness: No issues.

- Completeness: No issues.

- Conciseness: In the first paragraph there is no need to define maximal in line 2. The entire second paragraph can be deleted and replaced with "The other end of the maximal path is similar and also has degree 1."

  In the first paragraph, there are too many variables that are not relevant or only marginally relevant to the proof, and can be safely removed to improve ease of reading. One could just say, "Suppose $P = x_1, \ldots, x_n$, and suppose $\deg(x_n) \geq 2$. So there exists $y \in V$ with $y \neq x_{n-1}$ such that $\{x_n, y\} \in E$. If $y \in P$, then the edge $\{x_n, y\}$ along with the subpath of P between y and x_n form a cycle…"

- Clarity: Severe LaTeX errors make this proof difficult to read. If you do choose to use LaTeX, make sure that you are using math mode (`$`) whenever appropriate, subscripts are surrounded by grouping braces (`{}`), proper symbols are used (`\neq`, `\le`), the phrases "there exists" and "for all" are spelled out instead of in symbols, actual braces are escaped (`\{\}`), and functions whose names are text are displayed in text font (`\text{deg}`, or in this case `\deg` is a built-in command).

  It would have improved the clarity to separate the various cases of the first paragraph into different paragraphs or bullet points.

  The main ideas of this proof are obscured by by difficult-to-parse formatting and wordy language.

The following problems will not be covered in section, but may be useful to think about.
We recommend trying them by yourself first. Solutions will be posted in the evening.

4. Find the bug: failed induction

In this problem, you will fix an incorrect induction proof.

Problem: Suppose you have a stable matching instance with n people in P and n people in R. Of the n members of R, 5 are *popular*. That is, every person in P has those 5 members of R as their first 5 choices (in some order, not necessarily the same for each person in P). Similarly, you have 5 popular members of P, such that every person in R has those 5 as their top choices. Prove that in every stable matching of such an instance, every popular person is matched with another popular person.

Spoof. Let $P(n)$ be "In every stable matching of an instance with two groups of size n and 5 popular people per group, every popular person is matched with another popular person." We will show $P(n)$ holds for all $n \geq 5$ by induction on n.

Base case $(n = 5)$**:** With both sets having size 5, every person is popular. Since every stable matching pairs every person, every person is matched to a popular person.

Inductive hypothesis: Suppose $P(n)$ holds for $n = 5, \ldots, k$ for an arbitrary integer $k \geq 5$.

Inductive step: Let $r_1, \ldots, r_k$, $p_1, \ldots, p_k$ be k people in each group, with $r_1, \ldots, r_5, p_1, \ldots, p_5$ being the popular ones. We add r_{k+1} and p_{k+1}. By popularity, r_{k+1} has $p_1, \ldots, p_5$ (in some order) as their 5 favorite people and p_{k+1} has $r_1, \ldots, r_5$ (in some order) as their 5 favorite people. Further, let p_{k+1} and r_{k+1} be each other's 6th choices (i.e. top choice outside the popular people).

Now, consider any stable matching in the old (size k) instance. We create a stable matching for the new instance by pairing r_{k+1} with p_{k+1}. We now show that this matching is stable for the new instance.

Since it was stable for the small instance, the only possible unstable pairs must involve r_{k+1} or p_{k+1}. By IH, every popular person is matched to another popular person. Regardless of where r_{k+1} and p_{k+1} was added to the popular person's list, they fall after the popular ones, so r_{k+1} and p_{k+1} cannot form an unstable pair with the popular people. And since they have each other as their next choices, they cannot form an unstable pair with anyone else. Thus we have that there are no unstable pairs. The popular people remain matched to each other, as required. □

(a) There are at least two correctness errors in this proof. Describe them.

Solution:

The first mistake is in the setup of the inductive step. We need to show a claim for every instance of size $k+1$. Instead we build a particular instance of size $k+1$. In this example, the mistake is quite fundamental – there is no reason in the problem that p_{k+1} and r_{k+1} should have each other as their 6th choices. That is, if we introduced a recursive definition of stable matching instances and changed this to structural induction, we would have more steps to do beyond the one here (in contrast to the tree problem where all the cases are actually handled).

The second bug is again a mistake with handling a for-all. This time, the quantifier on the "every stable matching" part of the statement. We don't check every stable matching! We check every matching we built by starting with a stable matching on the small instance — how do we know there aren't stable matchings where p_{k+1} is matched to r_4 (for example)? We need to start with an arbitrary stable matching and argue whether the popular agents are matched or not.

(b) Write a correct proof of this claim. Do NOT use induction. Use a proof by contradiction instead. **Solution:**

Suppose, for the sake of contradiction, that there is a stable matching instance with 5 popular members of each group, and there is a stable matching M for this instance so that some popular person p from the first group is not matched to any popular member of the second group. Since there are the same number of popular people in the two groups, there is a popular member r of the second group that is also not matched to a popular agent. We claim that p and r form an unstable pair. Indeed, since each is popular, they are each in the top 5 of each other's lists, but each is matched to a non-popular agent, which must be 6th or lower on both lists. Thus p and r would rather be with each other than with their matches, so they form an unstable pair. But M was supposed to be a stable matching. A contradiction! So every popular person in the first group must be matched to a popular person in the second group.

5. Practice a reduction

A *reduction* from problem A to problem B is a solution to A in which you can call a library function that solves B. Typically, that library function does the bulk of the work, and your solution just consists of some preprocessing of the inputs to A in order to match what B expects, and postprocessing of the output of B to match what A requires. Note that you have no control over how the library function works internally—you only know what input it takes and what output it is guaranteed to give you.

In this question, you will solve a problem by reducing it to the basic stable matching problem.

Problem: Suppose that is a set of r riders and h horses with many more riders than horses; in particular, $2h < r < 3h$. You wish to set up a set of 3 rounds of rides which will give each rider exactly one chance to ride a horse. To keep things fair among the horses, you wish for each to have exactly 2 or 3 rides.

Because it's winter, by the time the third ride starts it will be very dark, so every rider would prefer *any* horse on the first two rides over being on the third ride. Between the first two rides, each rider doesn't have a preference over time of day, and have the same preference over horses. If a rider must be on the third ride, it has the same preference list for that ride as well.

Each horse has a single list over riders, which doesn't change by ride. Since horses love their jobs, they prefer to being one of the horses on the third ride to one of the ones left home.

Design an algorithm which calls the following library **exactly once** and ensures there are no pairs r, h which would both prefer to change the matching and get a better result for themselves.

> `BasicStableMatching`
> **Input:** A set of $2k$ people in two groups of k people each. Each person has an ordered preference list of all k members of the other group.
> **Output:** A stable matching among the $2k$ agents.

(a) Give a 1–2 sentence summary of your idea.

(b) Give the algorithm you're going to run.

(c) Give a 1–2 sentence summary of the idea of your proof.

(d) Write a proof of correctness.

(e) Give the running time of your algorithm, and briefly justify (1–3 sentences).

Solution:

(a) We will create a `BasicStableMatching` instance with $3h$ agents representing horses and $3h$ agents representing riders.

(b) For each horse h_i in the original instance, create three agents h_i^1,h_i^2, and h_i^3 representing three potential rides with horse h_i. We will specify their preference lists later.

For each rider r_j, their preference list will be the following: from r_j's original list, replace each h_i with h_i^1 followed by h_i^2. Then at the end, add another copy of the original list with each h_i replaced by h_i^3.

To make the total number of riders equal to $3h$, add "dummy" riders $d_1, \ldots, d_\ell$ until the number of riders and horses is equal. The preference list for each dummy will be all of the h_i^3, followed by the all of the h_i^2, the lastly all of the $h_i^{(1)}$ (in any order within each group).

Lastly, each h_i^j will have preference list starting identically to h_i's original list, then listing the dummy riders in any order.

Now, run the `BasicStableMatching` algorithm, then delete the dummy riders, and in every round, leave any horse whose partner was deleted unmatched.

(c) The `BasicStableMatching` algorithm doesn't produce unstable pairs, so we won't either (once we delete the dummies).

(d) We claim the result is a correct assignment. First, observe that each (real) rider is matched, and no horse is free on the first two rides. Since each horse prefers the real riders to the dummies and each rider prefers any of the first two rides to the third, a dummy rider matched with a horse on the first two rides would have created an unstable pair (the horse on the first two rides with any rider assigned to the third ride). Thus no horse is free on the first two rides.

It remains to show there is no unstable pair among matched agents. Suppose, for contradiction, there is a pair r, h_i where r and h_i would both prefer to be paired on ride j (over their current state). Then, by construction of the lists, r prefers h_i^j on its preference list and h_i^j prefers r on its preference list. This would have been an unstable pair for the `BasicStableMatching` instance. But the algorithm produces a stable matching, which by definition has no such unstable pairs, a contradiction!

(e) $\Theta(h^2)$. We have $3h$ agents on each side, so the guarantee on `BasicStableMatching` gives a $\Theta(h^2)$ guarantee for that call. All the other operations (copying lists, creating agents, etc.) can be done in time linear in the size of the final instance (since it's just copy-pasting) which is also $\Theta(h^2)$ ($\Theta(h)$ agents, each with lists of length $\Theta(h)$).

The corrections are SUGGESTIONS ONLY. We will not be so stylistically strict when grading your solutions.

Appendix — Problem 3 — Sample Solution? 1

Every tree with at least 2 vertices has at least 2 vertices of degree 1.

Proof. Suppose for contradiction that at most 1 vertex has degree 1, so the rest have degree at least 2. Then the sum of the degrees is at least $2n - 1$. However, recall that a tree has $n - 1$ edges, so the sum of degrees should be $2n - 2$, contradiction. □

- Correctness: No issues.
- Completeness: Does not explicitly use the fact that there are at least 2 vertices. (Need this to say "the rest have degree 2," since when there is 1 vertex it has degree 0.) Also missing "Let n be the number of vertices in the graph," though this is a minor point.
- Conciseness: No issues. In fact, fantastic.
- Clarity: The main idea is clear, mainly because it is so concise. Font size is a bit small. This is the default 10pt font. Prefer 11pt or 12pt, or increase the margins, though this is a minor point.

The corrections are SUGGESTIONS ONLY. We will not be so stylistically strict when grading your solutions.

Appendix — Problem 3 — Sample Solution? 2

Every tree with at least 2 vertices has at least 2 vertices of degree 1.

By strong induction on the number of vertices n.

BC: $n=2$. The only tree is •—•, it satisfies the claim.

IS: Assume that every tree with n vertices has 2 vertices of degree 1, for $n = 2, \dots, k$. We prove for $k+1$.

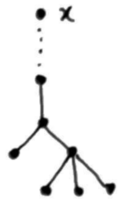
Let T be a tree with $k+1$ vertices and remove a vertex x.

Case 1: $\deg(x) = 1$

Then removing x cannot disconnect the graph, and removing any vertex will not introduce cycles, so the remaining graph is a tree.

By IH, it has at least 2 vertices of degree 1.

Attaching x may increase the degree of one of them, but $\deg(x) = 1$, so there are still at least 2

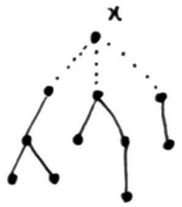


Case 2: $\deg(x) \geq 2$

Removing x disconnects the graph into a forest of ≥ 2 trees.

By IH, each tree has at least 2 vertices of degree 1.

Attaching x may increase the degree of one vertex per tree, but there are ≥ 2 trees and hence ≥ 2 remaining vertices of degree 1.



Not correct, because the resulting trees from the previous line may not have at least 2 vertices each. Need to treat the case of getting single-vertex trees.

- Correctness: See above.
- Completeness: No issues.
- Conciseness: No issues.
- Clarity: Prefer to separate out IH explicitly. Diagrams are great and helped point out the main ideas, but they obscured the correctness issue. A more generic image that shows the case of getting a single- vertex tree is better. Easy-to-read formatting with underlines and indenting.

The corrections are SUGGESTIONS ONLY. We will not be so stylistically strict when grading your solutions.

Appendix — Problem 3 — Sample Solution? 3

Every tree with at least 2 vertices has at least 2 vertices of degree 1.

Let P(n) be the statement, “Every tree on n vertices has at least 2 vertices of degree 1.” We will prove P(n) by induction for n >= 2.

Base Case: n=2. There is only one ~~undirected~~ tree with exactly 2 nodes, and it has 2 vertices that are both degree 1.

Inductive Hypothesis: Suppose P(n) is true for n = 2, …, k for an arbitrary k >= 2.

Inductive Step: Let T be an arbitrary tree with k nodes. By inductive hypothesis, T has at least two nodes of degree one. Call them u and v, and create a new node w. ~~Since we are interested in connected trees,~~ we must attach w; we break into cases depending on what it is adjacent to.

Case 1: w is attached to neither u nor v. If w is adjacent to a node other than u, v then u and v still have degree one, so the claim holds on T’.

Case 2: w is attached to one of u, v but not the other. If w is adjacent to u or v, then the other of u, v, and w will both be degree one.

There may have been vertices of degree 1 beyond just u and v, so you cannot conclusively say that attaching w to u and v results in no degree 1 vertices.

Case 3: w is attached to both u, v. ~~In this case, the graph would be left with no vertices of degree 1, but luckily~~ this case is impossible! If w were connected to both u and v, then the path in T between u and v (which exists because T was connected) along with (u, w) and (v, w) form a cycle, which is not allowed in a tree.

Edges of undirected graphs are sets (use {}), not ordered pairs

~~In all (allowed) cases, T’ has the required degree one vertices. Since we constructed T’ to have k + 1 vertices, we have shown P(k+1).~~

One can argue whether this is a correctness or completeness error, but it is generally not advised to do induction by building up graphs from smaller graphs—instead, remove a vertex from the larger graph to get the smaller graph, as in the previous sample solution. One can ostensibly fix this by adding a proof that all trees can be constructed by adding a new vertex to a smaller tree, but this fact is not obvious and would probably double the length of this proof, so it is not advised. Without this extra part, the proof is not correct. Generally, avoid this whole situation by always starting with the bigger graph and removing a vertex to apply the IH.

- Correctness: See comments.
- Completeness: See comments.
- Conciseness: See strikethroughs.
- Clarity: Google Docs without the symbol browser/equation editor/italicized math is a little harder to read, compared to proper LaTeX/handwriting, but it’s okay. Grammatically, the frequent use of “u, v” when it is more proper to write “u or v,” makes parsing the text more difficult. Paragraph breaks and use of bold are good. Main ideas are sufficiently understandable.

The corrections are SUGGESTIONS ONLY. We will not be so stylistically strict when grading your solutions.

Appendix — Problem 3 — Sample Solution? 4

Every tree with at least 2 vertices has at least 2 vertices of degree 1.

Proof. Let T = (V, E) be an arbitrary tree. Let P be a simple path of maximal length in the tree, ~~so P cannot be extended any longer by definition of maximal.~~ Let $x_1, ..., x_n$ be the vertices in the path, so $\{x_1, x_2\}, \{x2, x3\}, ..., \{x_n - 1, x_n\} \in E$. Suppose that $deg(x_n) >= 2$. So $\exists y \in V$ such that $y \neq x_n - 1$ and $x_n, y \in E$. If $\exists i = 1, ..., n - 2$ such that $y = x_i$, then $x_i, x_i + 1, ..., x_n, x_i = y$ is a cycle, which is a contradiction because trees are always acyclic. If $\forall i = 1, ..., n - 2$ we have $y \neq x_i$ then $x_1, \ldots, x_n, y$ is a longer path, which is a contradiction because we said P had maximal length. So now we've covered all the cases and we can conclude that $deg(x_n) < 2$. And $deg(x_n) \neq 0$ because $\{x_n - 1, x_n\}$ is an edge, according to P. So $deg(x_n) = 1$.

Too many variables that are not relevant or only marginally relevant to the proof, and can be safely removed to improve ease of reading. One could just say, "Suppose P = x_1, ... , x_n, and suppose deg(x_n) ≥ 2. So there exists y ∈ V with y != x_n−1 such that {x_n, y} ∈ E. If y ∈ P, then the edge {x_n,y} along with the subpath of P between y and x_n form a cycle..."

~~Next, suppose that $deg(x_1) >= 2$. So $\exists z \in V$ such that $z \neq x_2$ and $x_1, z \in E$. If $\exists i = 3, ..., n$ such that $z = x_i$, then $x_i = z, x_1, ...x_i$ is a cycle, which is a contradiction because trees are always acyclic. If $\forall i = 3, ..., n$ we have $z \neq x_i$ then $z, x_1, \ldots, x_n$ is a longer path, which is a contradiction because we said P had maximal length. So now we've covered all the cases again and we can conclude that $deg(x_1) < 2$. And $deg(x_1) \neq 0$ because $\{x_1, x_2\}$ is an edge, according to P. So $deg(x_1) = 1$.~~

Replace with "The other end of the maximal path is similar and also has degree 1."

Lastly, considering that every tree with at least two vertices contains at least one edge, and the longest simple path P contains at least two distinct vertices, it follows that x1 != xn. So x1 and xn are our two vertices that satisfy the claim, and we conclude that the claim holds. Q.E.D. □

- Correctness: No issues.
- Completeness: No issues.
- Conciseness: See comments
- Clarity: Severe LaTeX errors make this proof difficult to read. If you do choose to use LaTeX, make sure that you are using math mode ($) whenever appropriate, subscripts are surrounded by grouping braces ({}), proper symbols are used (\neq, \le), the phrases "there exists" and "for all" are spelled out instead of in symbols, actual braces are escaped (\{\}), and functions whose names are text are displayed in text font (\text{deg}, or in this case \deg is a built-in command).

It would have improved the clarity to separate the various cases of the first paragraph into different paragraphs or bullet points.

The main ideas of this proof are obscured by by difficult-to-parse formatting and wordy language.