Cooperative Game Theory and Matchings

COMP 4418 – Assignment 2

Jiayang Jiang z5319476

Question 1 (20 marks) Consider a Student Proposing Deferred Acceptance (SPDA) algorithm on an one-one matching instance $\langle S, C \prec \rangle$ where $|S| = |C| = n$. Prove or disprove the following:

1. (5 marks) For any instance, at least one student always makes multiple proposals.

According to the SPDA algorithm:

Given: $< S, C, >>$

Initialize $\mu \leftarrow \emptyset$

Initialize $R_s \leftarrow \emptyset$ for each $s \in S$

While(exists unmatched $s \in S$ s.t. $R_s \neq C$)

Let c be most preferred in $C \setminus R_s$ under $\succ_s$

If($s \succ_c \mu(c)$)

$\mu \leftarrow \left(\mu \setminus (\mu(c), c)\right) \cup \{(s, c)\}$

Else $R_s \leftarrow R_s \cup \{c\}$

Return μ

Answer: The statement is **false**. To disprove the statement, we provide an example where no student makes multiple proposals. Consider the case where the preferences are perfectly aligned such that every student ranks a different college as their top choice, and every college also ranks a different student as their top choice. Specifically,

We have n students $(S = \{s_1, s_2, \ldots, s_n\})$ and n colleges $(C = \{c_1, c_2, \ldots, c_n\})$.

Each student s_i ranks college c_i as their top choice, and each college c_i ranks student s_i as their top choice.

In this scenario, the SPDA algorithm proceeds as follows:

In the first round, every student proposes to their top-choice college (every student's top-choice college is different). Since each college receives a proposal from the student it ranks highest and hasn't made a pair, all proposals are accepted immediately.

As a result, no student is rejected, and no student needs to make a second proposal. Thus, in this specific instance, **no student makes multiple proposals**. This counterexample shows that it is not always true that at least one student must make multiple proposals.

Therefore, the statement is **disproved**: there exists at least one instance in which no student makes multiple proposals.

2. (5 marks) There is an instance where the number of proposals made is $\frac{n(n+1)}{2}$.

Answer: The statement is **true**. To prove this statement, consider the scenario where every student proposes to every college until they are matched. Specifically, we have n students $(S = \{s_1, s_2, \ldots, s_n\})$ and n colleges $(C = \{c_1, c_2, \ldots, c_n\})$. Each student has the same preference list**(assume $c_1 \succ c_2 \succ \ldots \succ c_n$)**, ranking all colleges in the same order, and each college has the same preference list, ranking all students in the same order**(assume $s_1 \succ s_2 \succ \ldots \succ s_n$).**

In this scenario: In the first round, all students propose to their top choice c_1. c_1 will accept the highest-ranked student s_1 and reject the rest.

The rejected students then propose to their next preferred college c_2. Again, the college accepts the highest-ranked student s_2 and rejects the rest.

This process continues until all students are matched.

The number of proposals made by the students can be calculated as follows:

Student s_1 makes 1 proposal.

Student s_2 makes 2 proposals (first to $\boldsymbol{c_1}$, then to $\boldsymbol{c_2}$).

Student s_3 makes 3 proposals (first to $\boldsymbol{c_1}$, then to $\boldsymbol{c_2}$, then to $\boldsymbol{c_3}$).

And so on, until student s_n makes n proposals.

The total number of proposals made is: $\boldsymbol{1 + 2 + 3 + \ldots + n} = \frac{\boldsymbol{n(n+1)}}{\boldsymbol{2}}$. Thus, there is an instance where the number of proposals made is exactly $\frac{\boldsymbol{n(n+1)}}{\boldsymbol{2}}$, which proves the statement to be **true**.

3. (5 marks) For any instance, there is always one college that receives exactly one proposal.

Answer: The statement is **true**. Since in the SPDA algorithm, each student proposes to colleges in order of their preferences, and each college temporarily holds the best proposal it receives while rejecting others:

- Each student eventually gets matched to a college. Since $|S| = |C| = n$ and all agents would rather be matched, unmatched agents would form a blocking pair.
- **The last college** to accept a proposal does so without rejecting any other student at that point.

- During the **final round**, a student makes a proposal to a college based on preference list.
- This proposal is accepted by **the last college,** which completes stable matching. At this point, this college does not need to reject any other student, because this is the only proposal it holds in the final step.

Therefore, in any instance, there will always be at least one college that receives exactly one proposal, which proves the statement to be **true**.

4. (5 marks) For an instance n students and n colleges, the maximum number of proposals that can be made is $\boldsymbol{n(n-1)+1}$.

Answer: The statement is **true**.

The maximum number of proposals happens when the colleges are highly selective, causing many students to propose multiple times before being accepted.

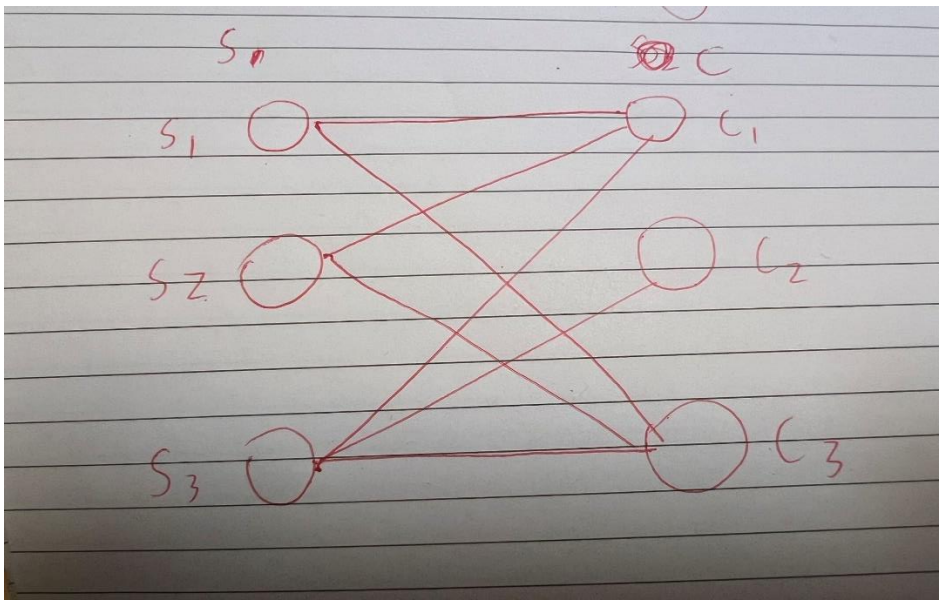
Specifically, there will always be at least one college that receives **exactly one proposal** in the end, while the other $\boldsymbol{n-1}$ colleges receive **proposals from all students**.

Consider the case where $n = 3$:

In this scenario, the worst-case preference strategy will make $\boldsymbol{n-1=2}$ **colleges receive proposals from all students**.

In this situation, **each of the first two colleges** receives **3 proposals** (one from each student), while the **last college receives exactly one proposal**.

In the general case with n students and n colleges, this behavior continues, where $\boldsymbol{n-1}$ **colleges** receive multiple proposals, and **the last college** receives exactly one proposal.



So that the last college will exactly receive one proposal and each $\boldsymbol{n-1}$ college will receive maximum n proposals, the maximum number of **proposals that can be made is $\boldsymbol{n(n-1)+1}$.**

Question 2 (10 marks) Consider the following one-one matching instance with $n = 5$.

$s_1 : c_2 \succ c_1 \succ c_3 \succ c_4 \succ c_5$
$s_2 : c_1 \succ c_2 \succ c_5 \succ c_3 \succ c_4$
$s_3 : c_3 \succ c_4 \succ c_5 \succ c_2 \succ c_1$
$s_4 : c_4 \succ c_5 \succ c_3 \succ c_1 \succ c_2$
$s_5 : c_1 \succ c_2 \succ c_5 \succ c_3 \succ c_4$

$c_1 : s_1 \succ s_2 \succ s_3 \succ s_4 \succ s_5$
$c_2 : s_2 \succ s_1 \succ s_4 \succ s_5 \succ s_3$
$c_3 : s_4 \succ s_5 \succ s_3 \succ s_2 \succ s_1$
$c_4 : s_5 \succ s_3 \succ s_4 \succ s_1 \succ s_2$
$c_5 : s_3 \succ s_4 \succ s_5 \succ s_1 \succ s_2$

Identify all colleges who can manipulate under the SPDA and explain why. Give an inconspicuous optimal manipulation for each, if any. Analogously identify all students who can manipulate under the CPDA and give an inconspicuous optimal manipulation for each, if any.

Computing Optimal Manipulations

Algorithm: $(S, C, \succ, c_j)$

Run SPDA with $\succ_{c_j}$, let μ be returned

Let S_j be the set of students who propose to c_j under $\succ_{c_j}$

Potential partners $P \leftarrow \{(s, \succ_{c_j}) | s \in S_j\}$, exhausted $E \leftarrow \{(\mu(c_j), \succ_{c_j})\}$

While$(P \setminus E \neq \emptyset)$
- Choose any $(s, \succ') \in P \setminus E$. Let $\succ^s$ be $\succ'$with s as most preferred student
- Run SPDA with $\succ^s$, let S_s be students who propose to c_j under $\succ_s$
- $P \leftarrow P \cup \{(s', \succ^s) | s' \in S_s \setminus P\}$, $E \leftarrow E \cup \{(s, \succ_s)\}$

Return $\succ^s$ for best $(s, \succ') \in P$

SPDA Matching Outcome:

1. **First Round Proposals:**
   - s_1 proposes to c_2.
   - s_2 proposes to c_1.
   - s_3 proposes to c_3.
   - s_4 proposes to c_4.
   - s_5 proposes to c_1.

Thus, $\boldsymbol{c_1}$ **received two proposals and can manipulate under SPDA**. Since c_1 prefers

s_2 to s_5, c_1 will accept s_2 and refuse s_5.

According to the algorithm:

$$\boldsymbol{P} = \{(\boldsymbol{s_2}, \succ_{\boldsymbol{c_1}})(\boldsymbol{s_5}, \succ_{\boldsymbol{c_1}})\}$$

Then s_5 will apply for c_2, and c_2 prefers s_1 to s_5, s_5 will be refused.

Then s_5 will apply for c_5, and c_2 will accept it.

$$\boldsymbol{E} = \{(\boldsymbol{s_2}, \succ_{\boldsymbol{c_1}})\}$$

Since $\boldsymbol{P \backslash E} \neq \emptyset$, let $\boldsymbol{s_5}$ be the most preferred student for $\boldsymbol{c_1}$.

$$\succ^{\boldsymbol{s}}: \boldsymbol{s_5} \succ \boldsymbol{s_1} \succ \boldsymbol{s_2} \succ \boldsymbol{s_3} \succ \boldsymbol{s_4}$$

Run SPDA again with $\succ^{\boldsymbol{s}}$, *we* can obtain

$$\boldsymbol{P} = \{(\boldsymbol{s_2}, \succ_{\boldsymbol{c_1}})(\boldsymbol{s_5}, \succ_{\boldsymbol{c_1}})(\boldsymbol{s_1}, \succ_{\boldsymbol{s_5}})\} \textbf{ and } \boldsymbol{E} = \{(\boldsymbol{s_2}, \succ_{\boldsymbol{c_1}})(\boldsymbol{s_5}, \succ_{\boldsymbol{c_1}})\}$$

according to SPDA.($s_1$also made a proposal to $\boldsymbol{c_1}$).

Since $\boldsymbol{P \backslash E} \neq \emptyset$, let s_1 be the most preferred student for c_1.

$$\succ^{\boldsymbol{s}}: \boldsymbol{s_1} \succ \boldsymbol{s_5} \succ \boldsymbol{s_2} \succ \boldsymbol{s_3} \succ \boldsymbol{s_4}$$

Run SPDA again with $\succ^{\boldsymbol{s}}$, *we* found that no other students made a proposal to c_1,

$$\boldsymbol{P} = \{(\boldsymbol{s_2}, \succ_{\boldsymbol{c_1}})(\boldsymbol{s_5}, \succ_{\boldsymbol{c_1}})(\boldsymbol{s_1}, \succ_{\boldsymbol{s_5}})\} \textbf{ and } \boldsymbol{E} = \{(\boldsymbol{s_2}, \succ_{\boldsymbol{c_1}})(\boldsymbol{s_5}, \succ_{\boldsymbol{c_1}})(\boldsymbol{s_1}, \succ_{\boldsymbol{s_5}})\}.$$

Now $\boldsymbol{P \backslash E} = \emptyset$, so the Optimal Manipulations for $\boldsymbol{c_1}$ is $\boldsymbol{s_1} \succ \boldsymbol{s_5} \succ \boldsymbol{s_2} \succ \boldsymbol{s_3} \succ \boldsymbol{s_4}$.

Inconspicuous Optimal Manipulation

Vaish and Garg:

Find optimal manipulation $s, \succ^s$.

Let s' be the second best student to propose to c_j under $\succ^s$.

Let $\succ'$ be the same as $\succ_{c_j}$ with s' moved to the right of s.

According to the definition, s' here is $\boldsymbol{s_5}$, then move s_5 to the right of s_1.

Therefore, **IOM for** $\boldsymbol{c_1}$: $\boldsymbol{s_1} \succ \boldsymbol{s_5} \succ \boldsymbol{s_2} \succ \boldsymbol{s_3} \succ \boldsymbol{s_4}$.

There are also two students making a proposal to c_2, $\boldsymbol{c_2}$ can manipulate under SPDA and using the same method as c_1 so the Optimal Manipulations for $\boldsymbol{c_2}$ is

$$\boldsymbol{s_2 \succ s_5 \succ s_1 \succ s_4 \succ s_3}$$

According to the definition, s' here is $\boldsymbol{s_5}$, then move s_5 to the right of s_2.

Therefore, **IOM for $\boldsymbol{c_2}$:**.

$$\boldsymbol{s_2 \succ s_5 \succ s_1 \succ s_4 \succ s_3}$$

For colleges, $\boldsymbol{c_3\ c_4\ c_5}$ there is only one student making proposals to them during SPDA, so they can't manipulate under SPDA.

CPDA Matching Outcome:

Under CPDA, colleges propose to students:

1. **First Round Proposals:**
    - c_1 proposes to s_1.
    - c_2 proposes to s_2.
    - c_3 proposes to s_4.
    - c_4 proposes to s_5.
    - c_5 proposes to s_3.

Thus, each student holds their only proposal.

CPDA Final Matching: (s_1, c_1), (s_2, c_2), $(s_4, c_3), (s_5, c_4), (s_3, c_5)$.

No student receives the proposal from multiple colleges.

Therefore, Under CPDA, students cannot force colleges to propose to them. Misrepresenting their preferences cannot change which colleges propose to them. Since they cannot receive proposals from their more-preferred colleges, manipulation is ineffective.

The time complexity of SPDA is $O(n^2)$, Each time a preference list is changed, the **SPDA** algorithm must be run again to determine the new matching result.

The algorithm tries different possible manipulations of the preference list to determine the best one.

$\boldsymbol{O(n)}$ adjustments need to be evaluated before identifying the optimal manipulation that gives the best result for the college.

Therefore, the total time complexity for optimal manipulation $\boldsymbol{O(n) \times O(n^2) = O(n^3)}$

Question 3 (10 marks) Give a polynomial time algorithm to check if a given one-one matching instance has a unique stable matching. Prove its correctness and running time.

To check for a unique stable matching, run the Gale-Shapley algorithm twice—once with workers proposing, once with firms. If both runs yield the same matching, it's unique; otherwise, there are multiple stable matchings.

Algorithm:

1. **Run the SPDA Algorithm:**
   - Input: The preference lists of all students and colleges.
   - Output: The **student-optimal stable matching** M_s .
2. **Run the CPDA Algorithm:**
   - Input: The same preference lists.
   - Output: The **college-optimal stable matching** M_c.
3. **Compare the Two Matchings:**
   - If $M_s = \mathrm{M_c}$, then the instance has a **unique stable matching**.
   - Else, the instance has **multiple stable matchings**.

Proof of Correctness:

To prove the correctness of the algorithm, we need to show that:

- If $\mathrm{M_s} = \mathrm{M_c}$, then there is exactly one stable matching.
- If $\mathrm{M_s} \neq \mathrm{M_c}$, then there are multiple stable matchings.

Stable Matching is A matching where there is no blocking pair—a student and a college who prefer each other over their assigned partners.

Lattice Structure of Stable Matchings: The set of all stable matchings forms a distributive lattice with respect to students' preferences. The student-optimal matching is the greatest element, and the college-optimal matching is the least element.

Case 1: $\mathrm{M_s} = \mathrm{M_c}$

- **Proof:** $\mathrm{M_s}$ is the best stable matching for students and $\mathrm{M_c}$ is the worst stable matching for students (best for colleges), If both matchings are identical, no

other stable matching exists between these two extremes. Thus, the instance has a **unique stable matching**.

Case 2: $M_S \neq M_C$

- **Proof:** Since M_S and M_C are different, and both are stable, there must be at least these two stable matchings. The lattice structure allows for other stable matchings between M_S and M_C. Therefore, the instance has **multiple stable matchings**.

Every student $s \in S$ can be rejected by at most n colleges. Thus, DA runs in time $O(n^2)$.

Total Running Time of the Algorithm:

1. Run SPDA: $O(n^2)$
2. Run CPDA: $O(n^2)$
3. Compare Matchings: $O(n)$ (checking n pairs)

Total Running Time: $O(n^2) + O(n^2) + O(n) = O(n^2)$.

Question 4 (10 marks) Given a one-one matchings instance $\langle S, C \succ \rangle$, let μ and μ' be two distinct stable matchings. Suppose for each student $s \in S$ if $\mu(s) \neq \mu'(s)$ then $\mu(s) \succ_s \mu'(s)$. Prove that for each college if $\mu(c) \neq \mu'(c)$ then $\mu(c) \prec \mu'(c)$.

We will prove this by contradiction.

Assume the contrary: There exists a college $c \in C$ such that:

1. $\mu(c) \neq \mu'(c)$.
2. $\mu(c) \succ \mu'(c)$ (c prefers $\mu(c)$ over μ'(c)).

Let $s = \mu(c)$. Since $\mu(c) \neq \mu'(c)$, we have:

- In matching μ: Student s is matched with college c.
- In matching μ': College c is matched with student $\mu'(c) \neq s$ and student s is matched with some college $c' = \mu'(s) \neq c$.

Since $\mu(s) \neq \mu'(s)$, it follows that $\mu(s) \succ_s \mu'(s)$, thus, s prefers college c over college c'.

Now, consider the pair (s, c) in matching μ':

- Student s Prefers c over $\mu'(s) = c'$.
- College c Prefers $\mu(c) = s$ over $\mu'(c)$ (by our assumption).

In matching μ': Both s and c prefer each other over their current matches in μ'.The pair (s, c) is a **blocking pair** for μ', it blocks μ' . A stable matching cannot have any blocking pairs. Since μ' given is distinct stable matchings, our assumption must be false.

Therefore, for every college c where $\mu(c) \neq \mu'(c)$, we have $\mu(c) \prec \mu'(c)$.

Question 5 (10 marks) Consider a Shapley-Scarf housing market with a set of agents $N = \{0,1,2,3,4\}$, a set of items $O = \{o_0, o_1, o_2, o_3, o_4\}$, and an endowment function $\omega : N \to 2^O$ such that $\omega(i) = \{o_i\}$. The preferences of the agents are as follows from left to right in decreasing order of preference.

$$0 : o_0, o_4, o_2, o_1, o_3$$
$$1 : o_0, o_2, o_4, o_1, o_3$$
$$2 : o_3, o_0, o_2, o_4, o_1$$
$$3 : o_0, o_2, o_3, o_1, o_4$$
$$4 : o_3, o_2, o_1, o_4, o_0$$

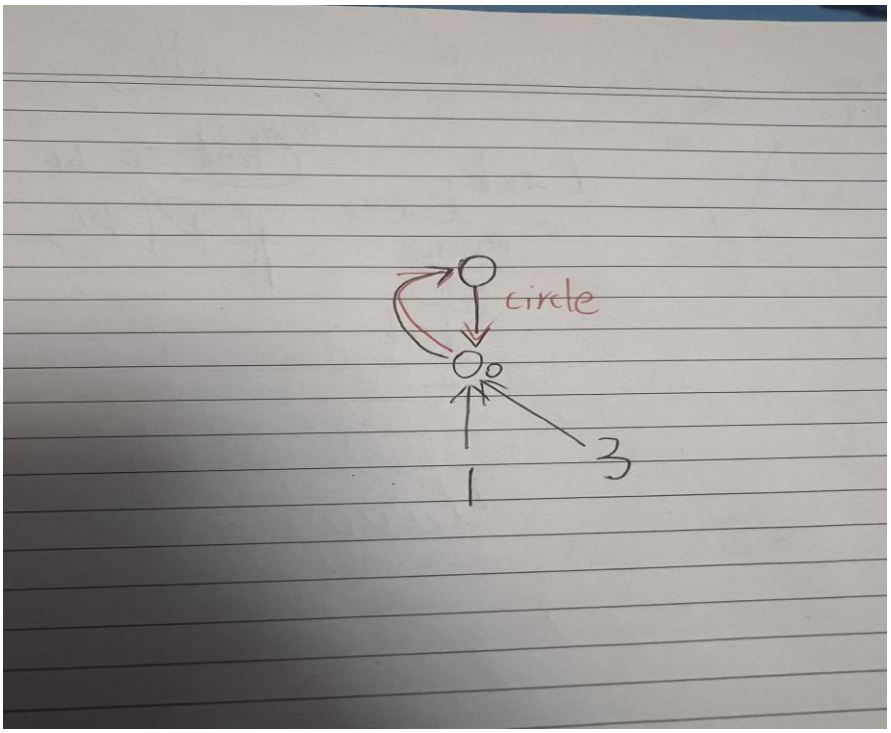
Find the outcome of the TTC (top trading cycles) algorithm. Can agent 4 misreport her preference to get a more preferred allocation? Prove or disprove that the outcome is individually rational.

Run the Top Trading Cycles (TTC) algorithm step by step.

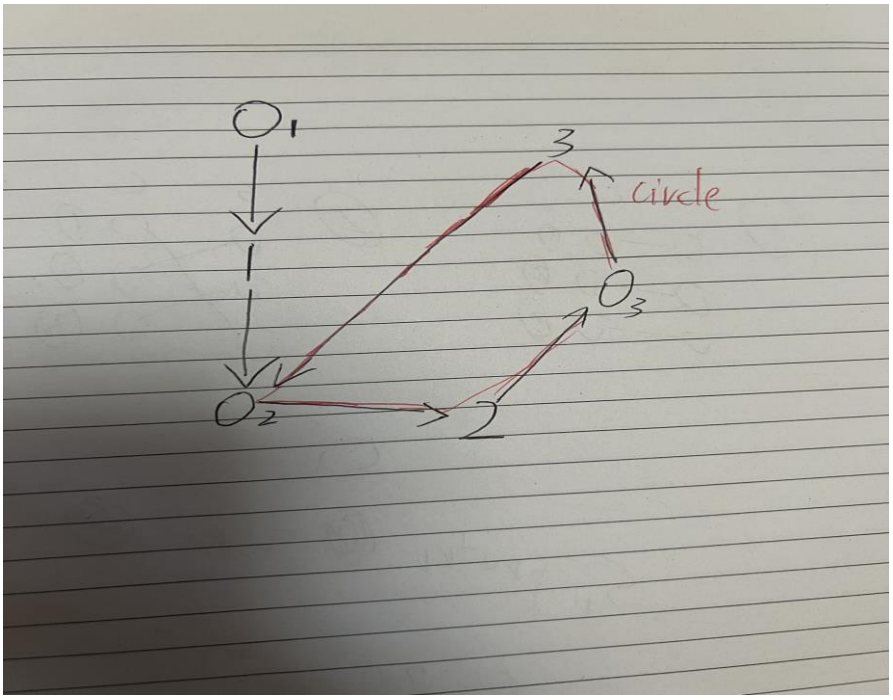
Housing Markets: Gale's Top Trading Cycles (TTC) Algorithm

- Each item points to its owner.
- Each agent points to her most preferred item in the graph.
- Find a cycle, allocate to each agent in the cycle the item she was pointing to. Remove the agents and items in the cycle. Adjust the graph so the agents in the graph point to their most preferred item in the graph.
- Repeat until the graph is empty.

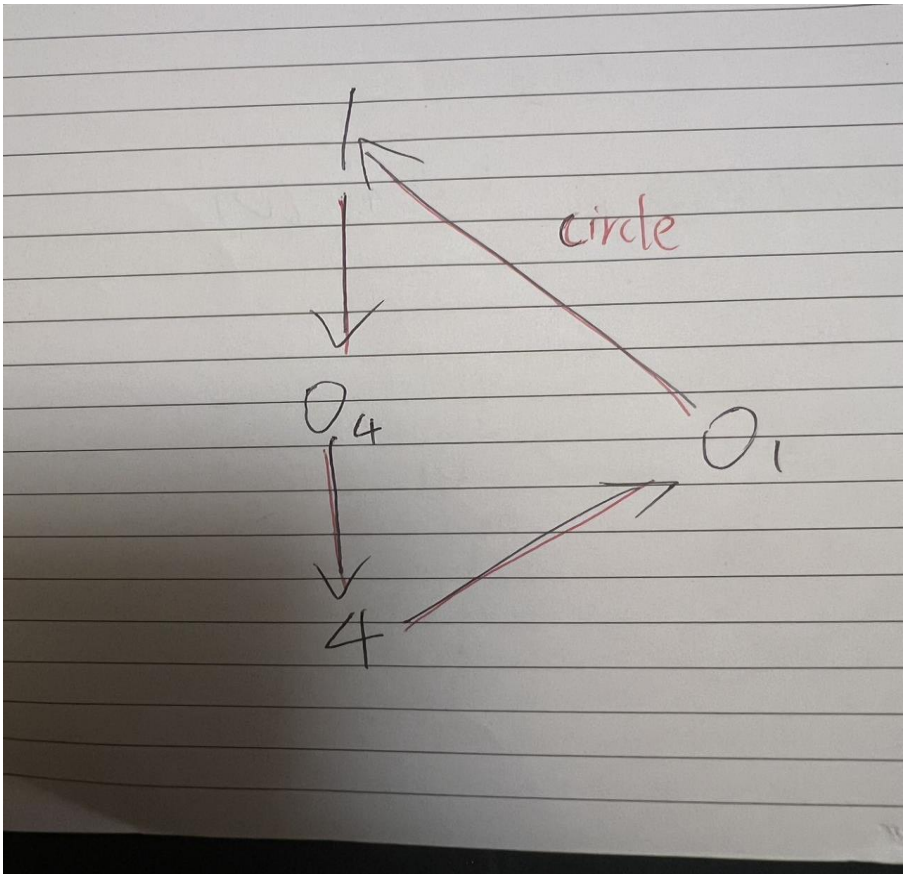
Given by an endowment function w: $w(i) = \{o_i\}$, Each agent i owns house o_i.



Then **Agent 0** gets O_0 and we removed them in the cycle.



Then **Agent 2** gets O_3, **Agent 3** gets O_2, we removed them in the cycle.



Then **Agent 1** gets $\boldsymbol{O_4}$, **Agent 4** gets $\boldsymbol{O_1}$, we removed them in the cycle.

Final Allocation:

- **Agent 0** gets $\boldsymbol{O_0}$
- **Agent 1** gets $\boldsymbol{O_4}$
- **Agent 2** gets $\boldsymbol{O_3}$
- **Agent 3** gets $\boldsymbol{O_2}$
- **Agent 4** gets $\boldsymbol{O_1}$

The Top Trading Cycles (TTC) algorithm is strategyproof. This means that an agent cannot gain a better outcome by misreporting their preferences; the best strategy is to report their true preferences honestly.

Consider an agent i who attempts to misreport their preferences to manipulate the outcome. For example, agent i may point to their $k-th$ favorite house (h_k) in the current preference graph, where $k > 1$ (meaning agent i is not choosing their most preferred house but instead a less preferred option). As a result, the algorithm may allocate house h_k to agent i.

When agent i is allocated house h_k, there must be a path from h_k to agent i that forms a cycle.

Consider the path P in the current graph from house h_k to agent i, which forms a cycle. This path will continue to exist until agent i is matched to a house because agents only change their outgoing edge when the endpoint of their current outgoing edge (i.e., the house) is removed.

Since the path from h_k to agent i always exists, agent i could have deferred pointing to house h_k until there were no better houses available to point to. This is exactly what the TTC algorithm does on behalf of agent i when they report their preferences truthfully.

Therefore, misreporting does not provide any benefit to agent i. By being honest, agent i will end up with the same or a better result compared to when they misreport. **Agent 4 can't misreport her preference to get a more preferred allocation.**

An allocation is **individually rational** if every agent receives a house that they prefer at least as much as their initial endowment (no agent minds participating in the allocation procedure).

Agents' Endowments and Allocations:

- **Agent 0:**
  - **Endowment:** $\boldsymbol{O_0}$
  - **Allocated:** $\boldsymbol{O_0}$
  - **Preference:** $\boldsymbol{O_0}$ is top choice.
- **Agent 1:**
  - **Endowment:** $\boldsymbol{O_1}$
  - **Allocated:** $\boldsymbol{O_4}$
  - **Preference Order:** $\boldsymbol{O_0 \succ O_2 \succ O_4 \succ O_1 \succ O_3}$
  - **Comparison:** $\boldsymbol{O_4 \succ O_1}$ (allocated house better than endowment)
- **Agent 2:**
  - **Endowment:** $\boldsymbol{O_2}$
  - **Allocated:** $\boldsymbol{O_3}$
  - **Preference Order:** $\boldsymbol{O_3 \succ O_0 \succ O_2 \succ O_4 \succ O_1}$
  - **Comparison:** $\boldsymbol{O_3 \succ O_2}$ (allocated house better than endowment)
- **Agent 3:**
  - **Endowment:** $\boldsymbol{O_3}$
  - **Allocated:** $\boldsymbol{O_2}$
  - **Preference Order:** $\boldsymbol{O_0 \succ O_2 \succ O_3 \succ O_1 \succ O_4}$
  - **Comparison:** $\boldsymbol{O_2 \succ O_3}$ (allocated house better than endowment)
- **Agent 4:**
  - **Endowment:** $\boldsymbol{O_4}$
  - **Allocated:** $\boldsymbol{O_1}$
  - **Preference Order:** $\boldsymbol{O_3 \succ O_2 \succ O_1 \succ O_4 \succ O_0}$
  - **Comparison:** $\boldsymbol{O_1 \succ O_4}$ (allocated house better than endowment)

All agents receive a house they prefer at least as much as their own. Therefore, **the outcome is individually rational.**

Question 6 (20 marks) Consider Shapley-Scarf housing markets in which we are only allowed to obtain allocations in which at most two agents are a part of a trading cycle and each agent can be a part of at most of one trading cycle. (a) Design a polynomial-time algorithm for the problem and that is individually rational and Pareto optimal among feasible outcomes. Prove its properties. (b) Design a polynomial-time algorithm for the problem and that is strategyproof and Pareto optimal among feasible outcomes. Prove its properties.

(a) Algorithm Steps:

1. **Construct Potential Exchange Pairs:**
   - For each agent i, identify all agents j such that:
     - Agent i prefers agent j's house over their own house.
     - Agent j prefers agent i's house over their own house.
   - These pairs (i, j) represent potential mutually beneficial exchanges and then i and j can't trade with other agents, **remove them in the circle.**
2. **Assign Weights to Potential Exchanges:**
   - For each potential exchange (i, j), calculate a weight w_{ij} representing the mutual benefit:

     $\boldsymbol{w_{ij}}$ **= (Improvement in** $\boldsymbol{i's}$ **preference by receiving house** $\boldsymbol{h_j}$**)+ (Improvement in j's preference by receiving house** $\boldsymbol{h_i}$**)**
   - The improvement is measured by the difference in preference rankings between the agent's current house and the potential new house.
3. **Create a Weighted Graph:**
   - Nodes represent agents.
   - Edges represent potential exchanges between agents, with assigned weights $\boldsymbol{w_{ij}}$.
4. **Find the Maximum Weight Matching:**
   - Use an appropriate algorithm to find the maximum weight matching in the graph.
   - This matching represents the set of exchanges that maximize the total mutual benefit under the constraints.
5. **Assign Houses Based on the Matching:**
   - Agents in Matched Pairs:
     - Agents i and j in each matched pair (i, j) exchange houses.
   - Agents Not in the Matching:
     - Agents not included in any matched pair retain their own houses.

Proof of Properties:

1. Individual Rationality

An allocation is individually rational if no agent minds participating in the allocation procedure:

Proof:

- Agents in Matched Pairs:
  - By construction, an edge exists between agents i and j only if:
    - Agent i prefers j' s house over their own.
    - Agent j prefers i' s house over their own.

    - Therefore, both agents receive houses they strictly prefer over their own.
- Agents Not in the Matching:
    - These agents retain their own houses.
    - They are no worse off than initially.
- Conclusion:
    - No agent ends up with a house they prefer less than their own.
    - The allocation is individually rational.

2. Pareto Optimality Among Feasible Outcomes

Definition:

An allocation x is Pareto optimal if there exists no other allocation y such that $y(i) \succsim i\, x(i)$ for all $i \in N$ and $y(i) \succ i\, x(i)$ for some $i \in N$.

Proof:

- Maximum Total Mutual Benefit:
    - The algorithm finds a matching that maximizes the total weight, representing the highest possible sum of mutual benefits under the constraints.
- No Better Feasible Allocation Exists:
    - Suppose there exists another feasible allocation that makes some agents better off without making others worse off.
        - This would imply a higher total mutual benefit, contradicting the maximality of the matching found.
- Constraints Respected:
    - Cycle Length Constraint: Only exchanges between pairs of agents (cycles of length 2) are allowed.
    - Participation Constraint: Each agent is involved in at most one exchange.
- Conclusion:
    - Within the given constraints, the allocation is Pareto optimal among feasible outcomes.

3. Polynomial Time Complexity

Analysis:

- Constructing Potential Exchanges and Assigning Weights:
    - For n agents, there are at most $O(n\text{^}2)$ pairs to consider.
    - Calculating weights for all potential exchanges takes $O(n\text{^}2)$ time.
- Creating the Weighted Graph:
    - The graph has n nodes and up to $O(n\text{^}2)$ edges.
- Finding the Maximum Weight Matching:
    - Edmonds' Algorithm for general graphs runs in $O(n\text{^}3)$ time.
- Assigning Houses Based on the Matching:
    - Assigning exchanges according to the matching takes $O(n)$ time.
- Conclusion:
    - The algorithm runs in polynomial time $O(n\text{^}3)$.

(b) Algorithm Design:

To achieve strategyproofness, we need to design an algorithm where no agent can benefit from misreporting their preferences.

Given the constraints, we can adapt the **'Serial Dictator'** mechanism, which is known to be strategyproof and Pareto optimal.

Algorithm Steps:

1. Fix an Order of Agents:
    - Determine a priority ordering of agents $\{1,2,\dots,n\}$.
    - This ordering can be arbitrary or based on a pre-specified rule (e.g., lottery).
2. Each Agent Picks a House:
    - Step through Agents in Order:
        - For $i = 1$ to n:
            - Agent i chooses their most preferred available house.
            - Remove the chosen house from the list of available houses.
    - Constraints Handling:
        - In the Serial Dictatorship mechanism, each agent makes only one choice, and no trading cycles occur. Therefore, there are no cycles involving more than two agents, and no agent participates in multiple cycles. This allocation naturally meets the requirements.

Proof of Properties:

1. Strategyproofness:

- Definition: An algorithm is strategyproof if no agent can benefit by misreporting their preferences.
- Proof:
    - In Serial Dictatorship, each agent's choice does not affect the earlier agents' choices.
    - An agent's best strategy is to report their true preferences to get the best available house when it's their turn.
    - Misreporting can only result in receiving a less preferred house.
    - Therefore, the algorithm is strategyproof.

2. Pareto Optimality Among Feasible Outcomes:

- Proof:
    - In Serial Dictatorship, each agent receives the best available house at their turn.
    - There is no way to reassign houses to make any agent better off without making someone else worse off.
    - Given the constraints (no cycles of length two), this allocation is Pareto optimal among feasible outcomes.

3. Feasibility Constraints:

- At most one trading cycle per agent: Each agent makes a single selection; no trading cycles are formed.
- At most two agents in a trading cycle: Since no cycles are formed, this constraint is satisfied.

4. Polynomial Time Complexity:

- Analysis:
  - The algorithm iterates through each agent once.
  - At each step, the agent selects their most preferred available house.
    - This can be done in $O(n)$ time if preferences are represented appropriately.
- Total Time Complexity: $O(n\text{^}2)$.

Question 7 (20 points) Consider the coalitional game (N, v) such that $N = \{1,2,3\}$ and v is defined as follows:

S	$\emptyset$	$\{1\}$	$\{2\}$	$\{3\}$	$\{1,2\}$	$\{1,3\}$	$\{2,3\}$	$\{1,2,3\}$
$v(S)$	0	4	3	2	4	3	2	12

For this game, compute the nucleolus and explain why it is the nucleolus.
Also compute the Shapley value and explain how you computed it.

Given a coalitional game (N, v) and payoff vector $x = (x_1, \ldots, x_n)$, the **excess of a coalition S under x** is defined by

$$e(x, S) = x(S) - v(S).$$

Need to find an allocation $x = (x_1, x_2, x_3)$ that minimizes the lexicographical order of the sorted excesses.

Compute the excesses for all coalitions $S \neq N$:

1. Singleton Coalitions:
   - $e(\{1\}, x) = x_1 - v(\{1\}) = x_1 - 4$
   - $e(\{2\}, x) = x_2 - v(\{2\}) = x_2 - 3$
   - $e(\{3\}, x) = x_3 - v(\{3\}) = x_3 - 2$
2. Two-Player Coalitions:
   - $e(\{1,2\}, x) = x_1 + x_2 - 4$
   - $e(\{1,3\}, x) = x_1 + x_3 - 3$
   - $e(\{2,3\}, x) = x_2 + x_3 - 2$

We aim to find x that minimizes the largest excess. To do this:

1. **Set the excesses of two-player coalitions equal:**
   Let θ be the common excess for two-player coalitions.
   - $x_1 + x_2 - 4 = \theta$
   - $x_1 + x_3 - 3 = \theta$
   - $x_2 + x_3 - 2 = \theta$
2. **Express x_i in terms of θ:**
   - From (1) and (2):
     Subtract (1) from (2):

$$(x_1 + x_3 - 3) - (x_1 + x_2 - 4) = 0$$
$$(x_3 - x_2) + 1 = 0$$
$$x_2 = x_3 + 1$$

- From (1) and (3):
  Subtract (3) from (1):

$$(x_1 + x_2 - 4) - (x_2 + x_3 - 2) = 0$$
$$(x_1 - x_3) - 2 = 0$$
$$x_1 = x_3 + 2$$

3. **Apply the efficiency condition:**

$$x_1 + x_2 + x_3 = 12$$
$$x_3 + 2 + x_3 + 1 + x_3 = 12$$
$$x_3 = 3$$

4. **Determine x_1 and x_2:**
   - $x_1 = 3 + 2 = 5$
   - $x_2 = 3 + 1 = 4$

Compute Excesses with Allocation $x = (5,4,3)$

1. **Singleton Coalitions:**
   - $e(\{1\}, x) = 5 - 4 = 1$
   - $e(\{2\}, x) = 4 - 3 = 1$
   - $e(\{3\}, x) = 3 - 2 = 1$

2. **Two-Player Coalitions:**
   - $e(\{1,2\}, x) = 5 + 4 - 4 = 5$
   - $e(\{1,3\}, x) = 5 + 3 - 3 = 5$
   - $e(\{2,3\}, x) = 3 + 4 - 2 = 5$
3. **Excess Vector (sorted):**

$$Excesses = [1,1,1,5,5,5]$$

Verify the Nucleolus

- **Lexicographical Minimization:**
  - The minimal excess is 111 (for singleton coalitions).
  - The maximal excess is 555 (for two-player coalitions).
  - There is no allocation that can reduce the maximal excess below 555 without violating individual rationality or efficiency.
- **Individual Rationality:**
  - $x_1 = 5 \geq 4$
  - $x_2 = 4 \geq 3$
  - $x_3 = 3 \geq 2$
- **Efficiency:**
  - $x_1 + x_2 + x_3 = 12$

Conclusion:

- The allocation $\boldsymbol{x = (5, 4, 3)}$ is nucleolus because it:

- Minimizes the maximal excess among all coalitions.
- Ensures individual rationality and efficiency.
- Balances the excesses among critical coalitions.

Solution concepts: Shapley value

Definition (Shapley value)

$$\phi_i(N,v) = \frac{1}{|N|!} \sum_{S \subseteq N \setminus \{i\}} (|S|!)(|N| - |S| - 1)!(v(S \cup \{i\}) - v(S))$$

- $v(S \cup \{i\}) - v(S)$: marginal contribution of agent i to coalition S
- Shapley value of an agent is her expected marginal contribution in a uniformly random permutation

Shapley value of agent 1:

- $123: v(\{1\}) - v(\emptyset) = 4$
- $132: v(\{1\}) - v(\emptyset) = 4$
- $213: v(\{2,1\}) - v(\{2\}) = 4 - 3 = 1$
- $312: v(\{3,1\}) - v(\{3\}) = 3 - 2 = 1$
- 321: $v(\{3,2,1\}) - v(\{2,3\}) = 12 - 2 = 10$
- 231: $v(2,3,1\}) - v(\{2,3\}) = 12 - 2 = 10$

$\phi 1 = (4 + 4 + 1 + 1 + 10 + 10)/6 = \mathbf{5}$

Shapley value of agent 2:

- $213: v(\{2\}) - v(\emptyset) = 3$
- $231: v(\{2\}) - v(\emptyset) = 3$
- $123: v(\{1,2\}) - v(\{1\}) = 4 - 4 = 0$
- $321: v(\{3,2\}) - v(\{3\}) = 2 - 2 = 0$
- 312: $v(\{1,2,3\}) - v(\{1,3\}) = 12 - 3 = 9$
- 132: $v(1,2,3\}) - v(\{1,3\}) = 12 - 3 = 9$

$\phi 2 = (3 + 3 + 0 + 0 + 9 + 9)/6 = \mathbf{4}$

Shapley value of agent 3:

- $312: v(\{3\}) - v(\emptyset) = 2$
- $321: v(\{3\}) - v(\emptyset) = 2$
- $132: v(\{3,1\}) - v(\{2\}) = 3 - 3 = 0$
- $231: v(\{3,2\}) - v(\{1\}) = 2 - 4 = -2$
- 123: $v(\{1,2,3\}) - v(\{1,2\}) = 12 - 4 = 8$
- 213: $v(2,1,3\}) - v(\{1,2\}) = 12 - 4 = 8$

$\phi 3 = (2 + 2 + 0 + -2 + 8 + 8)/6 = \mathbf{3}$

Therefore, Shapley value: $\phi 1 = \mathbf{5}$, $\phi 2 = \mathbf{4}$, $\phi 3 = \mathbf{3}$.