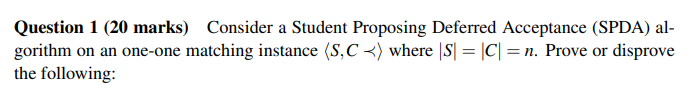
**Cooperative Game Theory and Matchings**

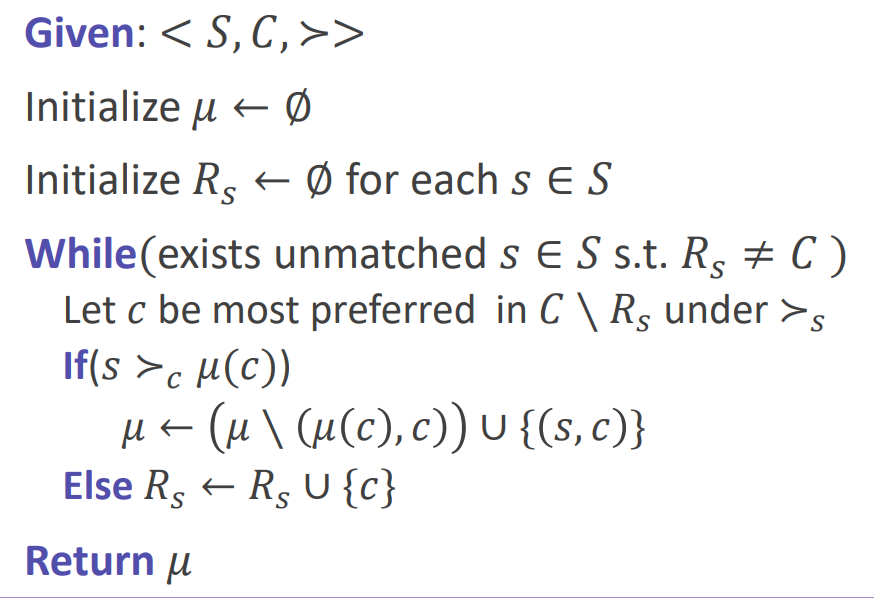
COMP 4418 – Assignment 2

Jiayang Jiang z5319476



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

According to the SPDA algorithm:



**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 students and colleges

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

**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 students and colleges Each student has the same preference list**(assume )**, ranking all colleges in the same order, and each college has the same preference list, ranking all students in the same order**(assume ).**

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

The rejected students then propose to their next preferred college . Again, the college accepts the highest-ranked student 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 makes 1 proposal.

Student makes 2 proposals (first to , then to ).

Student makes 3 proposals (first to , then to , then to ).

And so on, until student makes proposals.

The total number of proposals made is: Thus, there is an instance where the number of proposals made is exactly , 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 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**.

1. **(5 marks) For an instance n students and n colleges, the maximum number of proposals that can be made is .**

**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.

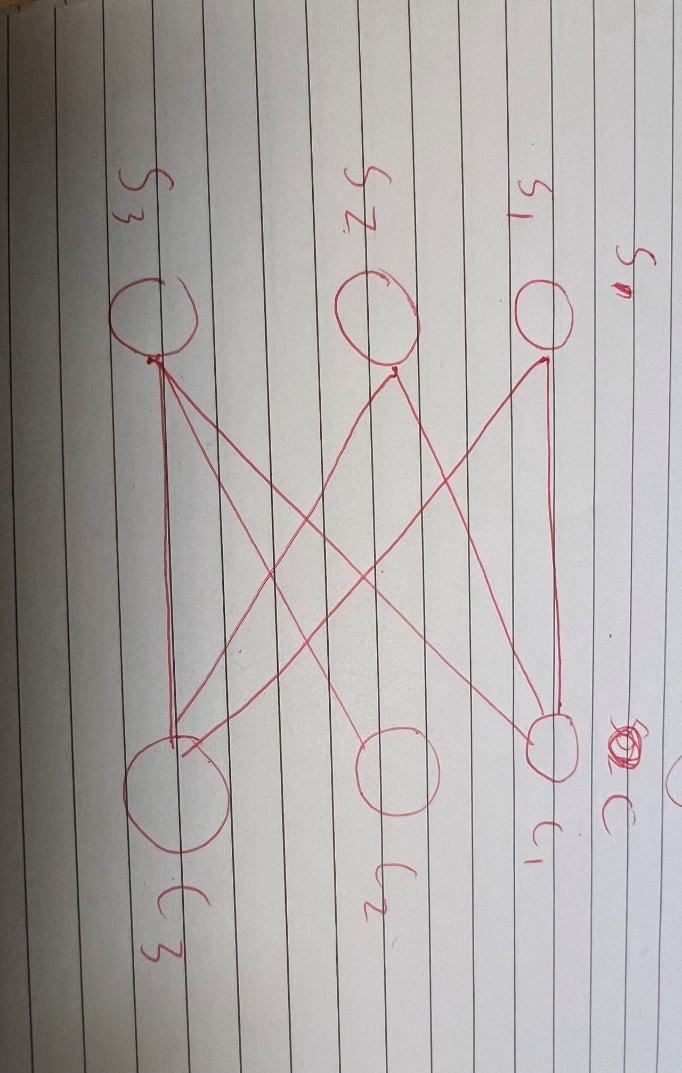
Specifically, there will always be at least one college that receives **exactly one proposal** in the end, while the other colleges receive **proposals from all students**.

Consider the case where :

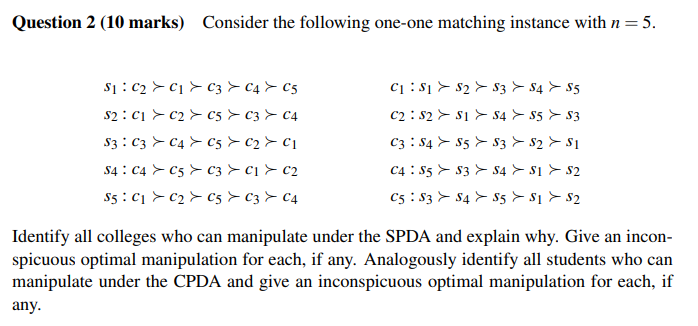
In this scenario, the worst-case preference strategy will make  **colleges receive proposals from all students**.

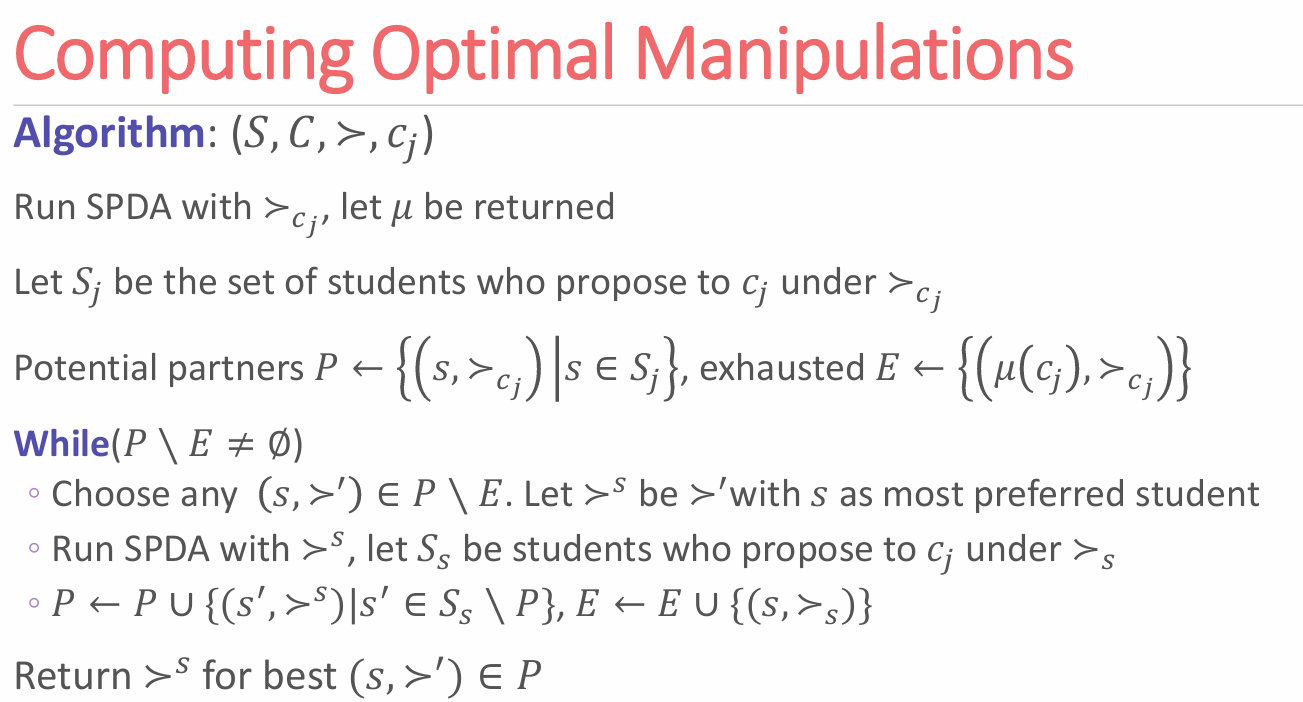
In this situation, **each of the first two colleges** receives **3 proposals** (one from each student), while the **last college receives exactly one proposal**.

In the general case with students and colleges, this behavior continues, where  **colleges** receive multiple proposals, and **the last college** receives exactly one proposal.



So that the last college will exactly receive one proposal and each college will receive maximum proposals, the maximum number of **proposals that can be made is .**





**SPDA Matching Outcome:**

1. **First Round Proposals:**
   * proposes to .
   * proposes to .
   * proposes to .
   * proposes to .
   * proposes to .

Thus,  **received two proposals and can manipulate under SPDA**. Since prefers to , will accept and refuse .

According to the algorithm:

Then will apply for , and prefers to , will be refused.

Then will apply for , and will accept it.

, let be the most preferred student for .

Run SPDA again with **,**  can obtain

**and**

according to SPDA.also made a proposal to .

, let be the most preferred student for .

Run SPDA again with **,**  found that no other students made a proposal to

**and**

Now **,** so the Optimal Manipulations for is .

A screenshot of a computer

Description automatically generated

According to the definition, 𝑠′ here is **,** then move to the right of .

Therefore, **IOM for** : ***.***

There are also two students making a proposal to , can manipulate under SPDA and using the same method as so the Optimal Manipulations for is

According to the definition, 𝑠′ here is **,** then move to the right of .

Therefore, **IOM for** :***.***

For colleges, 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:**
   * proposes to .
   * proposes to .
   * proposes to .
   * proposes to .
   * proposes to .

Thus, each student holds their only proposal.

**CPDA Final Matching:** ,

No student receives the proposal from multiple colleges.

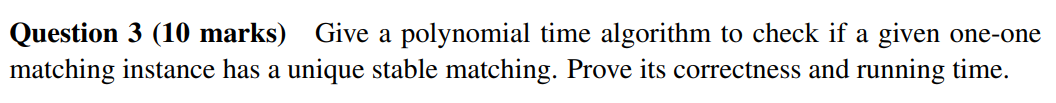
Therefore, Under CPDA, students cannot force colleges to propose to them. Misrepresenting their preferences cannot change which colleges propose to them. Since they cannot receive proposals from their more-preferred colleges, manipulation is ineffective.

The time complexity of SPDA is , 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.

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



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** ​.
2. **Run the CPDA Algorithm:**
   * Input: The same preference lists.
   * Output: The **college-optimal stable matching** ​.
3. **Compare the Two Matchings:**
   * If ​, 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 , then there is exactly one stable matching.
* If , 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:**

* **Proof:** is the best stable matching for students and ​ 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:**

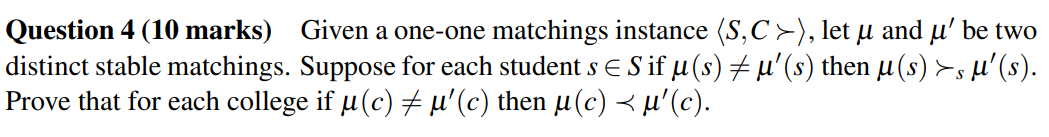
* **Proof:** Since and are different, and both are stable, there must be at least these two stable matchings. The lattice structure allows for other stable matchings between ​ and ​. Therefore, the instance has **multiple stable matchings**.

Every studentcan be rejected by at most colleges. Thus, DA runs in time .

**Total Running Time of the Algorithm:**

1. Run SPDA:
2. Run CPDA:
3. Compare Matchings: (checking pairs)

Total Running Time: .



We will prove this by contradiction.

**Assume the** **contrary:** There exists a college such that:

1. .
2. ( prefers over *μ*′(c)).

Let . Since , we have:

* In matching : Student is matched with college .
* In matching ′: College is matched with student and student is matched with some college .

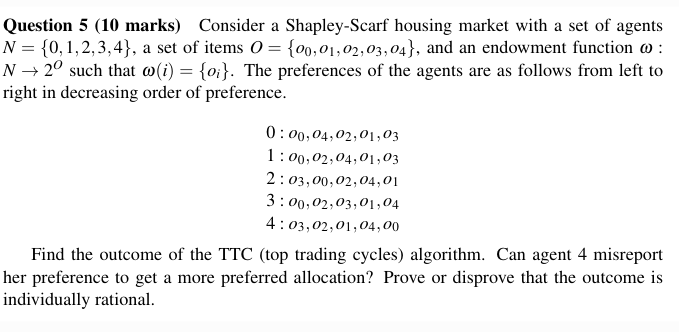
Since , it follows that , thus, prefers college over college .

Now, consider the pair in matching :

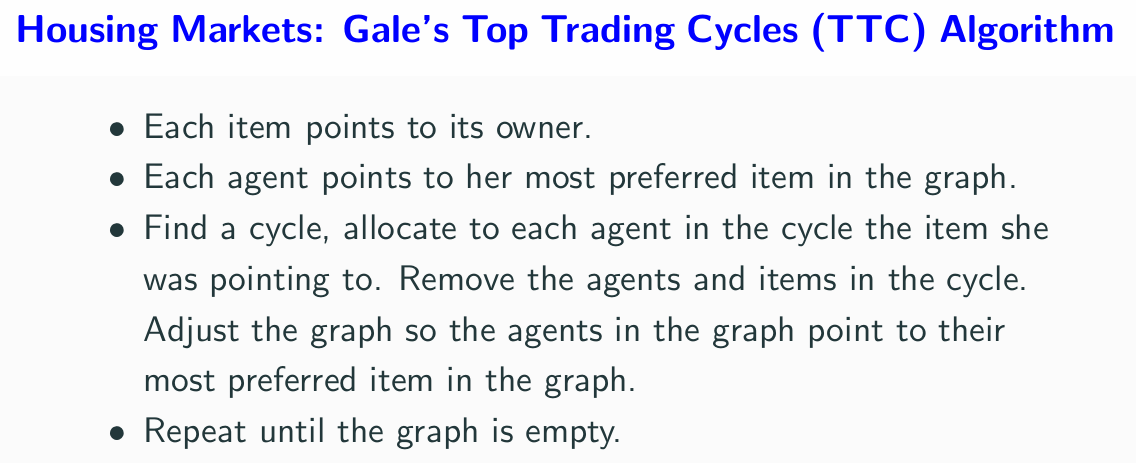
* Student Prefers over .
* College Prefers over (by our assumption).

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

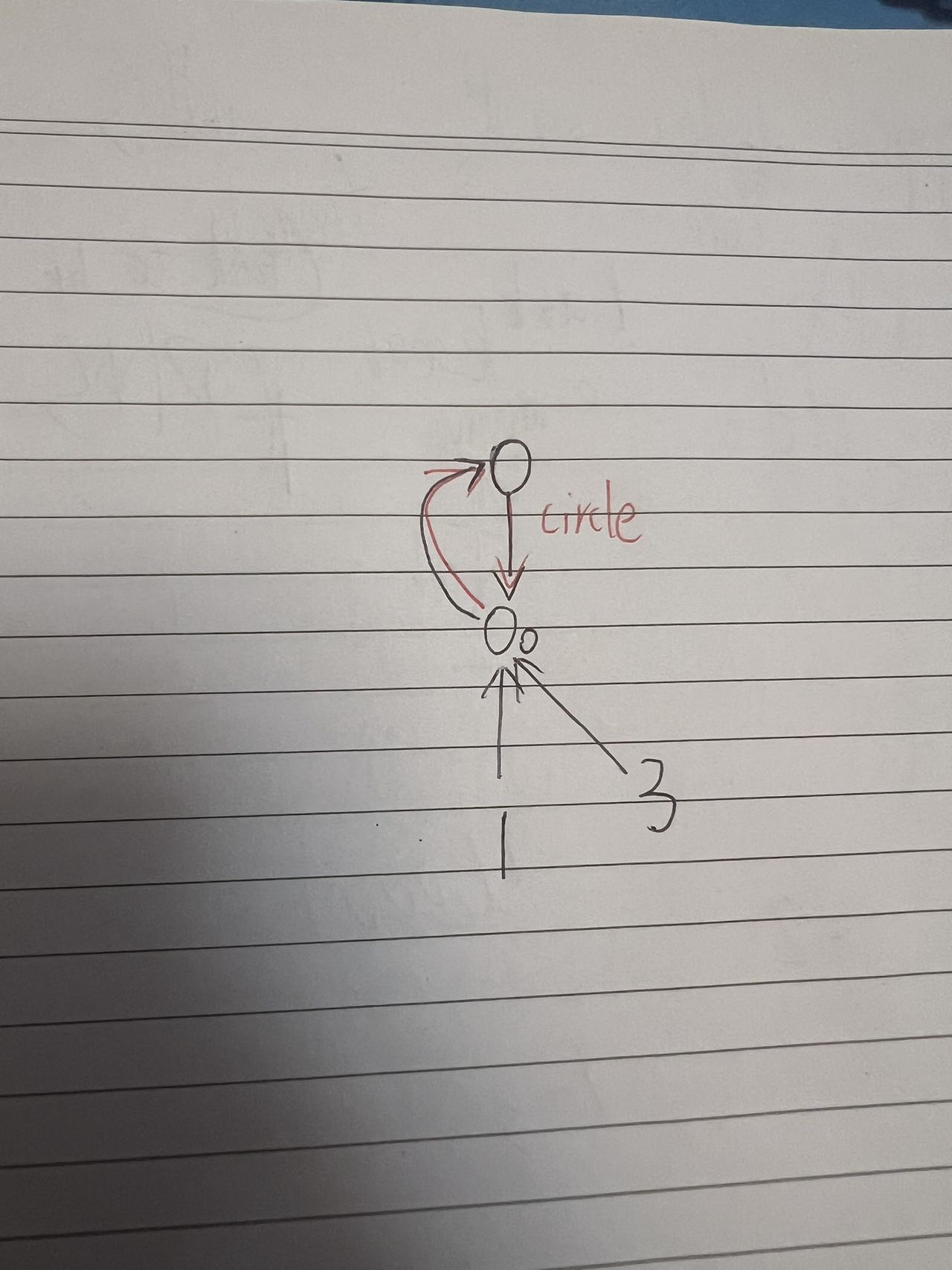
Therefore, for every college where , we have .

****

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

****

Given by an endowment function: , Each agent owns house ​.



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

A paper with lines and a diagram

Description automatically generated

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

A diagram of a circle

Description automatically generated with medium confidence

Then **Agent 1** gets , **Agent 4** gets , we removed them in the cycle.

**Final Allocation:**

* **Agent 0** gets
* **Agent 1** gets
* **Agent 2** gets
* **Agent 3** gets **​**
* **Agent 4** gets **​**

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

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

Consider the path in the current graph from house to agent , which forms a cycle. This path will continue to exist until agent 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 to agent always exists, agent could have deferred pointing to house until there were no better houses available to point to. This is exactly what the TTC algorithm does on behalf of agent when they report their preferences truthfully.

Therefore, misreporting does not provide any benefit to agent . By being honest, agent 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:** ​
  + **Allocated:** ​
  + **Preference:** ​ is top choice.
* **Agent 1:**
  + **Endowment:** ​
  + **Allocated:** ​
  + **Preference Order:**
  + **Comparison:** (allocated house better than endowment)
* **Agent 2:**
  + **Endowment:** ​
  + **Allocated:** ​
  + **Preference Order:** ​
  + **Comparison:** (allocated house better than endowment)
* **Agent 3:**
  + **Endowment:** ​
  + **Allocated:**
  + **Preference Order:**
  + **Comparison:** (allocated house better than endowment)
* **Agent 4:**
  + **Endowment:**
  + **Allocated:**
  + **Preference Order:** ​
  + **Comparison:** (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.**

**A close-up of a text

Description automatically generated**

**(a) Algorithm Steps:**

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

**(Improvement in  preference by receiving house )+ (Improvement in j’s preference by receiving house )**

* + The improvement is measured by the difference in preference rankings between the agent's current house and the potential new house.

1. **Create a Weighted Graph:**
   * Nodes represent agents.
   * Edges represent potential exchanges between agents, with assigned weights .
2. **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.
3. **Assign Houses Based on the Matching:**
   * Agents in Matched Pairs:
     + Agents and in each matched pair 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 and only if:
    - Agent prefers ' s house over their own.
    - Agent prefers ’ 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 is Pareto optimal if there exists no other allocation such that for all and for some .

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 agents, there are at most pairs to consider.
  + Calculating weights for all potential exchanges takes time.
* Creating the Weighted Graph:
  + The graph has nodes and up to edges.
* Finding the Maximum Weight Matching:
  + Edmonds' Algorithm for general graphs runs in time.
* Assigning Houses Based on the Matching:
  + Assigning exchanges according to the matching takes time.
* Conclusion:
  + The algorithm runs in polynomial time .

**(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 .
   * 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 to :
       - Agent 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 time if preferences are represented appropriately.
* Total Time Complexity: .

**A white text with black text

Description automatically generated with medium confidence**

**A math equations and formulas

Description automatically generated with medium confidence**

Need to find an allocation that minimizes the lexicographical order of the sorted excesses.

Compute the excesses for all coalitions :

1. Singleton Coalitions:
2. Two-Player Coalitions:

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

1. **Set the excesses of two-player coalitions equal:**

Let be the common excess for two-player coalitions.

1. **Express in terms of****:**
   * From (1) and (2):

Subtract (1) from (2):

* + From (1) and (3):

Subtract (3) from (1):

1. **Apply the efficiency condition:**
2. **Determine ​ and ​:**

Compute Excesses with Allocation

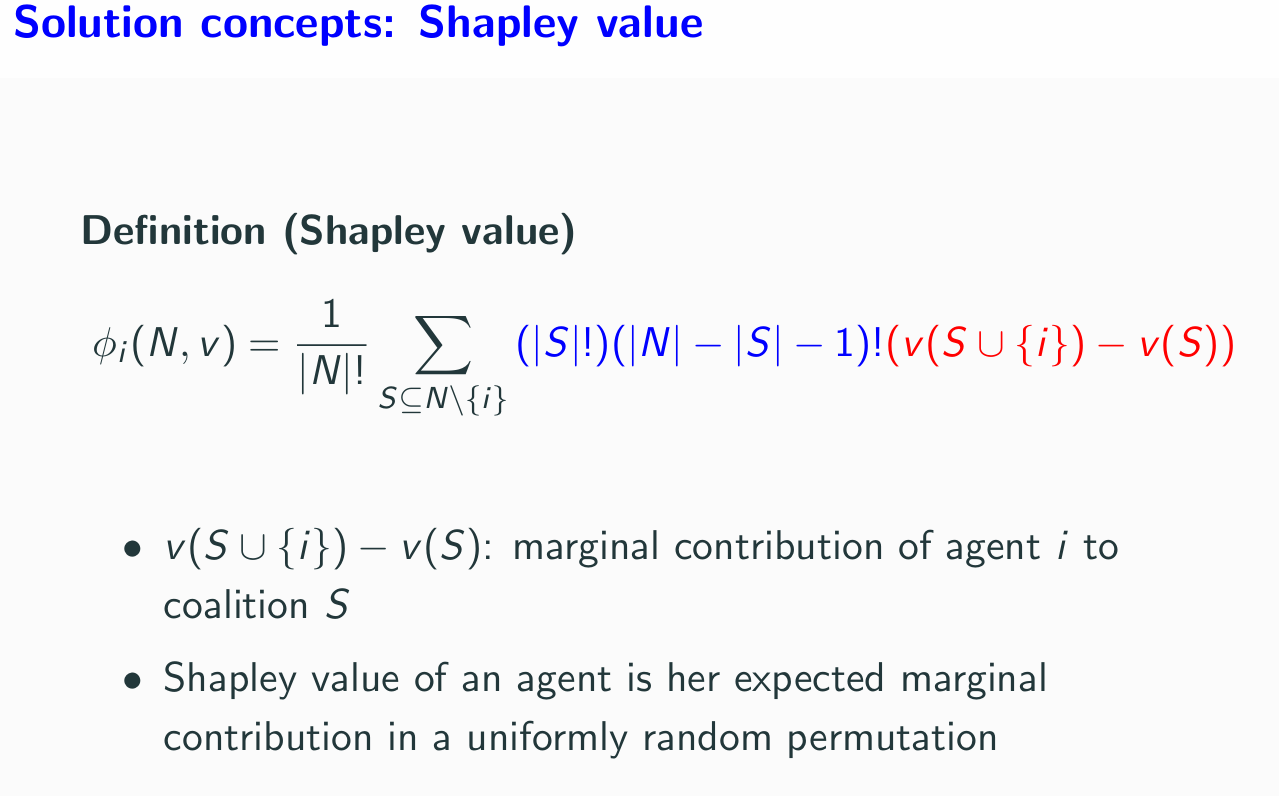
1. **Singleton Coalitions:**
2. **Two-Player Coalitions:**
3. **Excess Vector (sorted):**

**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:**
* **Efficiency:**

**Conclusion:**

* The allocation is nucleolus because it:
  + Minimizes the maximal excess among all coalitions.
  + Ensures individual rationality and efficiency.
  + Balances the excesses among critical coalitions.

****

**Shapley value of agent 1:**

* 321: ({3,2,1}) −({2,3}) = 12 – 2 = 10
* 231:(2,3,1}) −({2,3}) = 12 – 2 = 10

**Shapley value of agent 2:**

* 312: ({1,2,3}) −({1,3}) = 12 – 3 = 9
* 132:(1,2,3}) −({1,3}) = 12 – 3 = 9

**Shapley value of agent 3:**

* 123: ({1,2,3}) −({1,2}) = 12 – 4 = 8
* 213:(2,1,3}) −({1,2}) = 12 – 4 = 8

Therefore, Shapley value:  **.**