Algorithms

Aleks Ignjatović, ignjat@c
office: 504 (CSE buildi
Course Admin: Anahita Namvar, cs312

School of Computer Science and Engineering
University of New South Wales Sydney

1. INTRODUCTION

What is this course about?

It is about **designing algorithms** for solving practical problems.

What is this course about?

It is about **designing algorithms** for solving practical problems.

What is an algorithm?

An algori

executa

Introduction

What is this course about?

It is about **designing algorithms** for solving practical problems.

What is an algorithm?

An algori

executa

By "mechanical" we mean the methods that do not involve any creativity, intuition or even intelligence. Thus, al

by detailed, easily repeatable "recipes".

Introduction

What is this course about?

It is about **designing algorithms** for solving practical problems.

What is an algorithm?

An algori
executa

By "mechanical" we mean the methods that do not involve any creativity, intuition or even intelligence. Thus, al
by detailed easily repeatable "recipes".

The word "algorithm" comes by corruption of the name of *Muhammad ibn Musa al-Khwarizmi*, a Persian scientist 780-850, who wrote an important book on algebra, *"Al-kitab al-mukhtasar fi hisab al-gabr wal-muqabala"*. You are encouraged to read about him in Wikipedia.

In this course we will deal only with sequential deterministic algorith

- the ste

- the action of each step gives the same result whe executed for the same input

Why should you study algorithms design?

Can you find every algorithm you might need using Google?

Why should you study algorithms design?

Can you find every algorithm you might need using Google?

Our goal:

To learn iliar problem

Why should you study algorithms design?

Can you find every algorithm you might need using Google?

Our goal:

To learn **iliar**
problem

Course co

- a survey of algorithm **design techni**

Why should you study algorithms design?

Can you find every algorithm you might need using Google?

Our goal:

To learn **iliar**
problem

Course co

- a survey of algorithm **design techni**
- particular algorithms will be mostly used to ill
  techniques

Why should you study algorithms design?

Can you find every algorithm you might need using Google?

Our goal:

To learn **iliar**
problem

Course co

- a survey of algorithm **design techni**
- particular algorithms will be mostly used to ill techniques
- emphasis on development of your algorithm design **skills**

Textbook:

Kleinberg and Tardos: *Algorithm Design*
paperback edition available at the UNSW book store

Textbooks

Textbook:

Kleinberg and Tardos: *Algorithm Design*
paperback edition available at the UNSW book store

- exc

Textbooks

Textbook:

Kleinberg and Tardos: *Algorithm Design*
paperback edition available at the UNSW book store

- exc
- not s

Textbooks

Textbook:

Kleinberg and Tardos: *Algorithm Design*
paperback edition available at the UNSW book store

- exc
- not s

An alternative textbook:

Cormen, Leiserson, Rivest and Stein: *In*
preferably the third edition, should also be availab

Textbooks

Textbook:

Kleinberg and Tardos: *Algorithm Design*
paperback edition available at the UNSW book store

- exc
- not s

An alternative textbook:

Cormen, Leiserson, Rivest and Stein: *In*
preferably the third edition, should also be availab

- excellent: to be used later as a reference manual

Textbooks

Textbook:

Kleinberg and Tardos: *Algorithm Design*
paperback edition available at the UNSW book store

- exc
- not s

An alternative textbook:

Cormen, Leiserson, Rivest and Stein: *In*
preferably the third edition, should also be availab

- excellent: to be used later as a reference manual
- not so good: somewhat formalistic and written in a rather dry style.

Problem:

Two thieves have robbed a warehouse and have to split a pile of items without price tags on them. Design an algorithm for doing this in a way that ensures that each thief **believes** that he has got at least one half of the lo

Problem:

Two thieves have robbed a warehouse and have to split a pile of items without price tags on them. Design an algorithm for doing this in a way that ensures that each thief **believes** that he has got at least one half of the lo

The solu

One of the t

that both parts are of equal value. The other thief then c

part that he believes is no worse than the other.

Problem:

Two thieves have robbed a warehouse and have to split a pile of items without price tags on them. Design an algorithm for doing this in a way that ensures that each thief **believes** that he has got at least one half of the lo

The solu

One of the t

that both parts are of equal value. The other thief then c

part that he believes is no worse than the other.

The hard part: how can a thief split the pile into two equal parts? Remarkably, this turns out that, most likely, there is no more efficient algorithm than the brute force: we consider all partitions of the pile and see if there is one which results in two equal parts.

Problem:

Three thieves have robbed a warehouse and have to split a pile of items without price tags on them. How do they do this in a way that ensures that each thief believes that he has got at least one third of the loot?

- Re
  pro

Problem:

Three thieves have robbed a warehouse and have to split a pile of items without price tags on them. How do they do this in a way that ensures that each thief believes that he has got at least one third of the loot?

- Re pro

- Let u first thief splits the loot into three piles which he t equal value; then the remaining two thieves c want to take.

Problem:

Three thieves have robbed a warehouse and have to split a pile of items without price tags on them. How do they do this in a way that ensures that each thief believes that he has got at least one third of the loot?

- Re
  pro

- Let u
  first thief splits the loot into three piles which he t
  equal value; then the remaining two thieves c
  want to take.

- If they choose different piles, they can each take the piles they have chosen and the first thief gets the remaining pile; in this case clearly each thief thinks that he got at least one third of the loot.

Examples of Algorithms

Problem:

Three thieves have robbed a warehouse and have to split a pile of items without price tags on them. How do they do this in a way that ensures that each thief believes that he has got at least one third of the loot?

- Re
  pro

- Let u
  first thief splits the loot into three piles which he t
  equal value; then the remaining two thieves c
  want to take.

- If they choose different piles, they can each take the piles they have chosen and the first thief gets the remaining pile; in this case clearly each thief thinks that he got at least one third of the loot.
- But what if the remaining two thieves choose the same pile?

- One might think that in this case the first thief can pick any of the two piles that the second and the third thief did not choose; the remaining two piles are put together and the two remaining thieves split them as in Problem 1 with only two thieves.

- One might think that in this case the first thief can pick any of the two piles that the second and the third thief did not choose; the remaining two piles are put together and the two remaining thieves split them as in Problem 1 with only two thieves.
- Unfortunately this does not work:

- One might think that in this case the first thief can pick any of the two piles that the second and the third thief did not choose; the remaining two piles are put together and the two remaining thieves split them as in Problem 1 with only two thieves.
- Unfortunately this does not work:
- after the first thief splits the loot into three piles A, B, C, it might happen, for example, that the second thief thinks that

of th

$$A = 50\%, B = 10\%, C = 40\%.$$

- One might think that in this case the first thief can pick any of the two piles that the second and the third thief did not choose; the remaining two piles are put together and the two remaining thieves split them as in Problem 1 with only two thieves.
- Unfortunately this does not work:

- after the first thief splits the loot into three piles A, B, C, it might happen, for example, that the second thief thinks that

of th

$$A = 50\%, B = 10\%, C = 40\%.$$

- Thus, if the first thief picks pile B, t object that the first thief is getting 40% while he w only $60\%/2 = 30\%$.

- One might think that in this case the first thief can pick any of the two piles that the second and the third thief did not choose; the remaining two piles are put together and the two remaining thieves split them as in Problem 1 with only two thieves.
- Unfortunately this does not work:

- after the first thief splits the loot into three piles A, B, C, it might happen, for example, that the second thief thinks that

of th

$$A = 50\%, B = 10\%, C = 40\%.$$

- Thus, if the first thief picks pile B, t

object that the first thief is getting 40% while he w only $60\%/2 = 30\%$.
- If the first thief picks pile C then the third thief will object for the same reason.

- One might think that in this case the first thief can pick any of the two piles that the second and the third thief did not choose; the remaining two piles are put together and the two remaining thieves split them as in Problem 1 with only two thieves.
- Unfortunately this does not work:

- after the first thief splits the loot into three piles A, B, C, it might happen, for example, that the second thief thinks that

of th

$$A = 50\%, B = 10\%, C = 40\%.$$

- Thus, if the first thief picks pile B, t
object that the first thief is getting 40% while he w
only $60\%/2 = 30\%$.

- If the first thief picks pile C then the third thief will object for the same reason.
- What would be a correct algorithm?

- One might think that in this case the first thief can pick any of the two piles that the second and the third thief did not choose; the remaining two piles are put together and the two remaining thieves split them as in Problem 1 with only two thieves.
- Unfortunately this does not work:

- after the first thief splits the loot into three piles A, B, C, it might happen, for example, that the second thief thinks that

of th

$$A = 50\%, B = 10\%, C = 40\%.$$

- Thus, if the first thief picks pile B, t object that the first thief is getting 40% while he w only $60\%/2 = 30\%$.

- If the first thief picks pile C then the third thief will object for the same reason.
- What would be a correct algorithm?
- Let the thieves be T_1, T_2, T_3;

Assignment Project Exam Help

https://eduassistpro.github.i

Add WeChat edu_assist_pr

Algorithm:

T_1 makes a pile P_1 which he believes is $1/3$ of the whole loot;
T_1 proceeds to ask T_2 if T_2 agrees that $P_1 \leq 1/3$;
If T_2 says YES, **then** T_1 asks T_3 if T_3 agrees that $P_1 \leq 1/3$;

  If T_3 says YES, **then** T_1 takes P_1;
    T_2 and T_3 split the rest as in Problem 1.
  Else if T_3 says NO, **then** T_3 takes P_1;

Else if ... $_1$ such
that T_2 ...
  T_2 ... $_3$ ... $_2$

   If T_3 says YES **then** T_2 takes P_2;
    T_1 and T_3 split the rest as in Problem 1.
   Else if T_3 says NO then T_3 takes
    T_1 and T_2 split the rest as in Problem 1.

Algorithm:

T_1 makes a pile P_1 which he believes is $1/3$ of the whole loot;
T_1 proceeds to ask T_2 if T_2 agrees that $P_1 \leq 1/3$;
If T_2 says YES, **then** T_1 asks T_3 if T_3 agrees that $P_1 \leq 1/3$;

  If T_3 says YES, **then** T_1 takes P_1;
   T_2 and T_3 split the rest as in Problem 1.
  Else if T_3 says NO, **then** T_3 takes P_1;

Else if ... $_1$ such
that T_2 ...
  T_2 ... $_3$ ... $_2$

  If T_3 says YES **then** T_2 takes P_2;
   T_1 and T_3 split the rest as in Problem 1.
  Else if T_3 says NO then T_3 takes
   T_1 and T_2 split the rest as in Problem 1.

Homework: Try generalising this to n thieves! (a bit harder than with three thieves!)

Algorithm:

T_1 makes a pile P_1 which he believes is $1/3$ of the whole loot;
T_1 proceeds to ask T_2 if T_2 agrees that $P_1 \leq 1/3$;
If T_2 says YES, **then** T_1 asks T_3 if T_3 agrees that $P_1 \leq 1/3$;

  If T_3 says YES, **then** T_1 takes P_1;
  T_2 and T_3 split the rest as in Problem 1.
  Else if T_3 says NO, **then** T_3 takes P_1;

Else if ... $_1$ such
that T_2 ...
T_2 ... $_3$ ... $_2$

  If T_3 says YES **then** T_2 takes P_2;
  T_1 and T_3 split the rest as in Problem 1.
  Else if T_3 says NO then T_3 takes
  T_1 and T_2 split the rest as in Problem 1.

Homework: Try generalising this to n thieves! (a bit harder than with three thieves!)
Hint: there is a *nested recursion* happening even with 3 thieves!

The role of proofs in algorithm design

When do we need to give a **mathematical proof** that an algorithm we have just designed terminates and returns a solution to the problem at hand?

The role of proofs in algorithm design

When do we need to give a **mathematical proof** that an algorithm we have just designed terminates and returns a solution to the problem at hand?

When thi

sense!

When do we need to give a **mathematical proof** that an algorithm we have just designed terminates and returns a solution to the problem at hand?

When thi

sense!

Mathematical proofs are **NOT** academi
to justify things which are not obvious to common se

Example: MergeSort

Merge-Sort(A,p,r) *sorting A[p..r]*

1. **if** $p < r$
2. **then** $q \leftarrow \lfloor \frac{p+r}{2} \rfloor$
3. Merge-Sort(A, p, q)
4. Merge-Sort$(A, q+1, r)$
5.

Example: MERGESORT

Merge-Sort(A,p,r) *sorting A[p..r]*

1. **if** $p < r$
2. **then** $q \leftarrow \lfloor \frac{p+r}{2} \rfloor$
3. MERGE-SORT(A, p, q)
4. MERGE-SORT$(A, q+1, r)$
5.

1. Th

Example: MERGESORT

Merge-Sort(A,p,r)   *sorting A[p..r]*

1. **if** $p < r$
2. **then** $q \leftarrow \lfloor \frac{p+r}{2} \rfloor$
3. MERGE-SORT(A, p, q)
4. MERGE-SORT$(A, q+1, r)$
5.

1. Th
2. On each level of recursion merging intermedi $O(n)$ steps.

Example: MergeSort

Merge-Sort(A,p,r) *sorting A[p..r]*

1. **if** $p < r$
2. **then** $q \leftarrow \lfloor \frac{p+r}{2} \rfloor$
3. $\textsc{Merge-Sort}(A, p, q)$
4. $\textsc{Merge-Sort}(A, q+1, r)$
5.

1. Th
2. On each level of recursion merging intermedi $O(n)$ steps.
3. Thus, MergeSort always terminate in $O(n \log_2 n)$ many steps.

Example: MERGESORT

Merge-Sort(A,p,r) *sorting A[p..r]*

1. **if** $p < r$
2. **then** $q \leftarrow \lfloor \frac{p+r}{2} \rfloor$
3. MERGE-SORT(A, p, q)
4. MERGE-SORT$(A, q + 1, r)$
5.

1. Th
2. On each level of recursion merging intermedi $O(n)$ steps.
3. Thus, MERGESORT always terminate in $O(n \log_2 n)$ many steps.
4. Merging two sorted arrays always produces a sorted array, thus, the output of MERGESORT will be a sorted array.

Example: MergeSort

Merge-Sort(A,p,r) *sorting A[p..r]*

1. **if** $p < r$
2. **then** $q \leftarrow \lfloor \frac{p+r}{2} \rfloor$
3. Merge-Sort(A, p, q)
4. Merge-Sort$(A, q+1, r)$
5.

1. Th
2. On each level of recursion merging intermedi $O(n)$ steps.
3. Thus, MergeSort always terminate in $O(n \log_2 n)$ many steps.
4. Merging two sorted arrays always produces a sorted array, thus, the output of MergeSort will be a sorted array.
5. The above is essentially a proof by induction, but we will never bother formalising proofs of (essentially) obvious facts.

The role of proofs in algorithm design

- However, sometimes it is **NOT** clear from a description of an algorithm that such an algorithm will not enter an infinite loop and fail to terminate.

- However, sometimes it is **NOT** clear from a description of an algorithm that such an algorithm will not enter an infinite loop and fail to terminate.
- Sometimes it is **NOT** clear that an algorithm will not run in exponentially many steps (in the size of the input), which is usu

- However, sometimes it is **NOT** clear from a description of an algorithm that such an algorithm will not enter an infinite loop and fail to terminate.

- Sometimes it is **NOT** clear that an algorithm will not run in exponentially many steps (in the size of the input), which is usu

- So y suc

The role of proofs in algorithm design

- However, sometimes it is **NOT** clear from a description of an algorithm that such an algorithm will not enter an infinite loop and fail to terminate.

- Sometimes it is **NOT** clear that an algorithm will not run in exponentially many steps (in the size of the input), which is usu

- So y suc

- Proofs are needed for such circumstances; in a l **the only way** to know that the algorith

The role of proofs in algorithm design

- However, sometimes it is **NOT** clear from a description of an algorithm that such an algorithm will not enter an infinite loop and fail to terminate.

- Sometimes it is **NOT** clear that an algorithm will not run in exponentially many steps (in the size of the input), which is usu
- So y suc

- Proofs are needed for such circumstances; in a l **the only way** to know that the algorith

- For that reason we will **NEVER** p textbook sometimes does just that, by sometimes formulating and proving trivial little lemmas, being too pedantic!). We will prove only what is genuinely nontrivial.

The role of proofs in algorithm design

- However, sometimes it is **NOT** clear from a description of an algorithm that such an algorithm will not enter an infinite loop and fail to terminate.

- Sometimes it is **NOT** clear that an algorithm will not run in exponentially many steps (in the size of the input), which is usu

- So y suc
- Proofs are needed for such circumstances; in a l **the only way** to know that the algorith

- For that reason we will **NEVER** p textbook sometimes does just that, by sometimes formulating and proving trivial little lemmas, being too pedantic!). We will prove only what is genuinely nontrivial.
- **However, <u>BE VERY CAREFUL</u> what you call trivial!!**

The Stable Matching Problem

The Stable Matching Problem

- Assume that you are running a dating agency and have n men and n women as customers;

The Stable Matching Problem

- Assume that you are running a dating agency and have n men and n women as customers;
- Th
  -
  -

The Stable Matching Problem

- Assume that you are running a dating agency and have n men and n women as customers;
- Th
  -
  -
- Design an algorithm which produces a is:

The Stable Matching Problem

- Assume that you are running a dating agency and have n men and n women as customers;
- Th
  -
  -
- Design an algorithm which produces a is: a set of n pairs $p = (m, w)$ of a man at the following situation never happens:

The Stable Matching Problem

- Assume that you are running a dating agency and have n men and n women as customers;
- Th
    -
    -
- Design an algorithm which produces a is: a set of n pairs $p = (m, w)$ of a man at the following situation never happens:

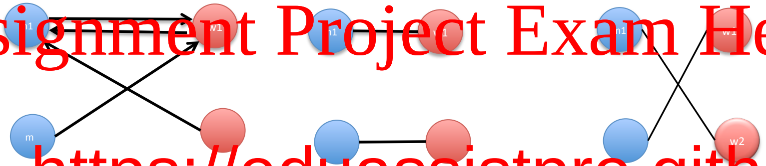
for two pairs $p = (m, w)$ and $p' = (m', w')$:

- man m prefers woman w' to woman w, **and**
- woman w' prefers man m to man m'.

Stable Matching Problem: examples

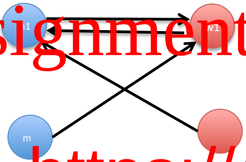
Case1:

Preferences: | stable matching | not stable

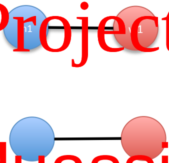


Stable Matching Problem: examples

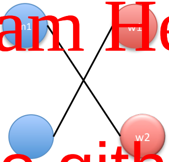
Preferences:



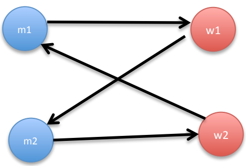
Case1:
stable matching



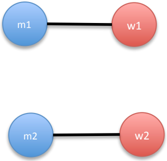
not stable

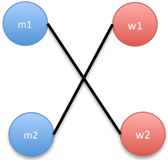


Preferences:



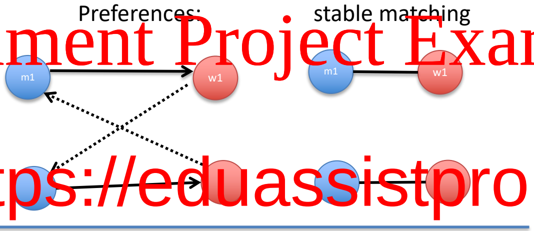
Case2:
stable matchin





Is there always a stable matching for any preferences of two pairs?

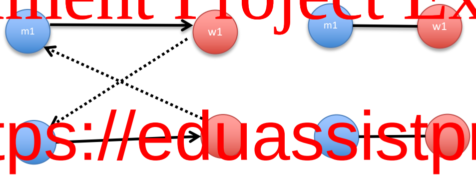
Case1: two men like two different women (or vice versa)



Is there always a stable matching for any preferences of two pairs?

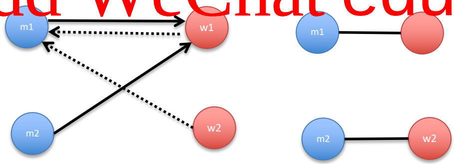
Case1: two men like two different women (or vice versa)

Preferences: stable matching



Case2: men like the same woman and

Preferences: stab



Question: Is it true that for every possible collection of n lists of preferences provided by all men, and n lists of preferences provided by all women, a stable matching always exists?

Question: Is it true that for every possible collection of n lists of preferences provided by all men, and n lists of preferences provided by all women, a stable matching always exists?

Answer:

Question: Is it true that for every possible collection of n lists of preferences provided by all men, and n lists of preferences provided by all women, a stable matching always exists?

Answer:

Given n
i.e., just to f

Stable Matching Problem: Gale - Shapley algorithm

Question: Is it true that for every possible collection of n lists of preferences provided by all men, and n lists of preferences provided by all women, a stable matching always exists?

Answer:

Given n
i.e., just to f

Answer: n!

Question: Is it true that for every possible collection of n lists of preferences provided by all men, and n lists of preferences provided by all women, a stable matching always exists?

Answer:

Given n
i.e., just to f

Answer: $n! \approx (n/e)^n$ - more than exponentia 2.71);

Question: Is it true that for every possible collection of n lists of preferences provided by all men, and n lists of preferences provided by all women, a stable matching always exists?

Answer:

Given n
i.e., just to f

Answer: $n! \approx (n/e)^n$ - more than exponentia 2.71);

Can we find a stable matching in a reasonable amoun

Stable Matching Problem: Gale - Shapley algorithm

Question: Is it true that for every possible collection of n lists of preferences provided by all men, and n lists of preferences provided by all women, a stable matching always exists?

Answer:

Given n
i.e., just to f

Answer: $n! \approx (n/e)^n$ - more than exponentia 2.71);

Can we find a stable matching in a reasonable amoun

Answer: **YES**, using the **Gale - Shapley algorithm**.

Stable Matching Problem: Gale - Shapley algorithm

Question: Is it true that for every possible collection of n lists of preferences provided by all men, and n lists of preferences provided by all women, a stable matching always exists?

Answer:

Given n
i.e., just to f

Answer: $n! \approx (n/e)^n$ - more than exponentia 2.71);

Can we find a stable matching in a reasonable amoun

Answer: **YES**, using the **Gale - Shapley algorithm**.

Originally invented to pair newly graduated physicians with US hospitals for residency training.

- Produces pairs in stages, with possible revisions;

Stable Matching Problem: Gale - Shapley algorithm

- Produces pairs in stages, with possible revisions;
- A man who has not been paired with a woman will be called *free*.

Stable Matching Problem: Gale - Shapley algorithm

- Produces pairs in stages, with possible revisions;
- A man who has not been paired with a woman will be called *free*.
- Men will be proposing to women. Women will decide if they accept a proposal or not.

Stable Matching Problem: Gale - Shapley algorithm

- Produces pairs in stages, with possible revisions;
- A man who has not been paired with a woman will be called *free*.
- Men will be proposing to women. Women will decide if they accept a proposal or not.
- Start with all men free;

Stable Matching Problem: Gale - Shapley algorithm

- Produces pairs in stages, with possible revisions;
- A man who has not been paired with a woman will be called *free*.
- Men will be proposing to women. Women will decide if they accept a proposal or not.
- Start with all men free;

While

Stable Matching Problem: Gale - Shapley algorithm

- Produces pairs in stages, with possible revisions;
- A man who has not been paired with a woman will be called *free*.
- Men will be proposing to women. Women will decide if they accept a proposal or not.
- Start with all men free;

While

p

r

Stable Matching Problem: Gale - Shapley algorithm

- Produces pairs in stages, with possible revisions;
- A man who has not been paired with a woman will be called *free*.
- Men will be proposing to women. Women will decide if they accept a proposal or not.

- Start with all men free;

While

p
r

If no one has proposed to w yet

Stable Matching Problem: Gale - Shapley algorithm

- Produces pairs in stages, with possible revisions;
- A man who has not been paired with a woman will be called *free*.
- Men will be proposing to women. Women will decide if they accept a proposal or not.
- Start with all men free;

While

p

r

If no one has proposed to w yet

she always accepts and a pair

Stable Matching Problem: Gale - Shapley algorithm

- Produces pairs in stages, with possible revisions;
- A man who has not been paired with a woman will be called *free*.
- Men will be proposing to women. Women will decide if they accept a proposal or not.

- Start with all men free;

While

p

r

If no one has proposed to w yet

she always accepts and a pair

Else she is already in a pair p'

Stable Matching Problem: Gale - Shapley algorithm

- Produces pairs in stages, with possible revisions;
- A man who has not been paired with a woman will be called *free*.
- Men will be proposing to women. Women will decide if they accept a proposal or not.

- Start with all men free;

While

p

r

If no one has proposed to w yet

she always accepts and a pair

Else she is already in a pair p'

If m is higher on her prefere

- Produces pairs in stages, with possible revisions;
- A man who has not been paired with a woman will be called *free*.
- Men will be proposing to women. Women will decide if they accept a proposal or not.

- Start with all men free;

While

p

r

If no one has proposed to w yet
she always accepts and a pair

Else she is already in a pair p'

If m is higher on her prefere
the pair $p' = (m', w)$ is deleted;

- Produces pairs in stages, with possible revisions;
- A man who has not been paired with a woman will be called *free*.
- Men will be proposing to women. Women will decide if they accept a proposal or not.

- Start with all men free;

While

p

r

If no one has proposed to w yet

she always accepts and a pair

Else she is already in a pair p'

If m is higher on her prefere

the pair $p' = (m', w)$ is deleted;

m' becomes a free man;

- Produces pairs in stages, with possible revisions;
- A man who has not been paired with a woman will be called *free*.
- Men will be proposing to women. Women will decide if they accept a proposal or not.

- Start with all men free;

While

p

r

If no one has proposed to w yet

she always accepts and a pair

Else she is already in a pair p'

If m is higher on her prefere

the pair $p' = (m', w)$ is deleted;

m' becomes a free man;

a new pair $p = (m, w)$ is formed;

- Produces pairs in stages, with possible revisions;
- A man who has not been paired with a woman will be called *free*.
- Men will be proposing to women. Women will decide if they accept a proposal or not.

- Start with all men free;

While

p

r

If no one has proposed to w yet

she always accepts and a pair

Else she is already in a pair p'

If m is higher on her prefere

the pair $p' = (m', w)$ is deleted;

m' becomes a free man;

a new pair $p = (m, w)$ is formed;

Else m is lower on her preference list than m';

the proposal is rejected and m remains free.

Claim 1: Algorithm terminates after $\leq n^2$ rounds of the *While* loop

Stable Matching Problem: Gale - Shapley algorithm

Claim 1: Algorithm terminates after $\leq n^2$ rounds of the *While* loop

Proof:

- In every round of the *While* loop one man proposes to one woman;

Stable Matching Problem: Gale - Shapley algorithm

Claim 1: Algorithm terminates after $\leq n^2$ rounds of the *While* loop

Proof:

- In every round of the *While* loop one man proposes to one woman;
- every man can propose to a woman at most once;

Claim 1: Algorithm terminates after $\leq n^2$ rounds of the *While* loop

Proof:

- In every round of the *While* loop one man proposes to one woman;
- every man can propose to a woman at most once;
- thus, every man can make at most n proposals;

Claim 1: Algorithm terminates after $\leq n^2$ rounds of the *While* loop

Proof:

- In every round of the *While* loop one man proposes to one woman;
- every man can propose to a woman at most once;
- thus, every man can make at most n proposals;
- there a ... ls.

Stable Matching Problem: Gale - Shapley algorithm

Claim 1: Algorithm terminates after $\leq n^2$ rounds of the *While* loop
Proof:

- In every round of the *While* loop one man proposes to one woman;
- every man can propose to a woman at most once;
- thus, every man can make at most n proposals;
- there a ... ls.

Thus the ... y times.

Claim 1: Algorithm terminates after $\leq n^2$ rounds of the *While* loop

Proof:

- In every round of the *While* loop one man proposes to one woman;
- every man can propose to a woman at most once;
- thus, every man can make at most n proposals;
- there a ... ls.

Thus the ... y times.

Claim 2: Algorithm produces a matching, i.e., every man is eventually paired with a woman (and thus also every woman is pa

Stable Matching Problem: Gale - Shapley algorithm

Claim 1: Algorithm terminates after $\leq n^2$ rounds of the *While* loop
Proof:

- In every round of the *While* loop one man proposes to one woman;
- every man can propose to a woman at most once;
- thus, every man can make at most n proposals;
- there a ls.

Thus the y times.

Claim 2: Algorithm produces a matching, i.e., every man is eventually paired with a woman (and thus also every woman is pa
Proof:

Claim 1: Algorithm terminates after $\leq n^2$ rounds of the *While* loop

Proof:

- In every round of the *While* loop one man proposes to one woman;
- every man can propose to a woman at most once;
- thus, every man can make at most n proposals;
- there a ls.

Thus the y times.

Claim 2: Algorithm produces a matching, i.e., every man is eventually paired with a woman (and thus also every woman is pa

Proof:

- Assume that the while *While* loop has te ll free.

Stable Matching Problem: Gale - Shapley algorithm

Claim 1: Algorithm terminates after $\leq n^2$ rounds of the *While* loop

Proof:

- In every round of the *While* loop one man proposes to one woman;
- every man can propose to a woman at most once;
- thus, every man can make at most n proposals;
- there a ls.

Thus the y times.

Claim 2: Algorithm produces a matching, i.e., every man is eventually paired with a woman (and thus also every woman is pa

Proof:

- Assume that the while *While* loop has te ll free.
- This means that m has already proposed to every woman.

Claim 1: Algorithm terminates after $\leq n^2$ rounds of the *While* loop

Proof:

- In every round of the *While* loop one man proposes to one woman;
- every man can propose to a woman at most once;
- thus, every man can make at most n proposals;
- there a ... ls.

Thus the ... y times.

Claim 2: Algorithm produces a matching, i.e., every man is eventually paired with a woman (and thus also every woman is pa

Proof:

- Assume that the while *While* loop has te ... ll free.
- This means that m has already proposed to every woman.
- Thus, every woman is paired with a man, because a woman is not paired with anyone only if no one has made a proposal to her.

Claim 1: Algorithm terminates after $\leq n^2$ rounds of the *While* loop

Proof:

- In every round of the *While* loop one man proposes to one woman;
- every man can propose to a woman at most once;
- thus, every man can make at most n proposals;
- there a ls.

Thus the y times.

Claim 2: Algorithm produces a matching, i.e., every man is eventually paired with a woman (and thus also every woman is pa

Proof:

- Assume that the while *While* loop has te ll free.
- This means that m has already proposed to every woman.
- Thus, every woman is paired with a man, because a woman is not paired with anyone only if no one has made a proposal to her.
- But this would mean that n women are paired with all of n men so m cannot be free.

Stable Matching Problem: Gale - Shapley algorithm

Claim 1: Algorithm terminates after $\leq n^2$ rounds of the *While* loop

Proof:

- In every round of the *While* loop one man proposes to one woman;

- every man can propose to a woman at most once;
- thus, every man can make at most n proposals;
- there a ... ls.

Thus the ... y times.

Claim 2: Algorithm produces a matching, i.e., every man is eventually paired with a woman (and thus also every woman is pa

Proof:

- Assume that the while *While* loop has te ... ll free.
- This means that m has already proposed to every woman.
- Thus, every woman is paired with a man, because a woman is not paired with anyone only if no one has made a proposal to her.
- But this would mean that n women are paired with all of n men so m cannot be free. **Contradiction!**

Claim 3: The matching produced by the algorithm is stable.

Claim 3: The matching produced by the algorithm is stable.
Proof:

Claim 3: The matching produced by the algorithm is stable.

Proof: Note that during the *While* loop:

- a woman is paired with men of increasing ranks on her list;

Claim 3: The matching produced by the algorithm is stable.

Proof: Note that during the *While* loop:

- a woman is paired with men of increasing ranks on her list;
- a man is paired with women of decreasing ranks on his list.

Claim 3: The matching produced by the algorithm is stable.

Proof: Note that during the *While* loop:

- a woman is paired with men of increasing ranks on her list;
- a man is paired with women of decreasing ranks on his list.

Assume n

Claim 3: The matching produced by the algorithm is stable.

Proof: Note that during the *While* loop:

- a woman is paired with men of increasing ranks on her list;
- a man is paired with women of decreasing ranks on his list.

Assume n

thus, ther t:

Claim 3: The matching produced by the algorithm is stable.

Proof: Note that during the *While* loop:

- a woman is paired with men of increasing ranks on her list;
- a man is paired with women of decreasing ranks on his list.

Assume n

thus, ther t:

Claim 3: The matching produced by the algorithm is stable.

Proof: Note that during the *While* loop:

- a woman is paired with men of increasing ranks on her list;
- a man is paired with women of decreasing ranks on his list.

Assume n

thus, ther t:

- Since m prefers w' over w, he must hav ore proposing to w;

Stable Matching Problem: Gale - Shapley algorithm

Claim 3: The matching produced by the algorithm is stable.

Proof: Note that during the *While* loop:

- a woman is paired with men of increasing ranks on her list;
- a man is paired with women of decreasing ranks on his list.

Assume n

thus, ther t:

- Since m prefers w' over w, he must hav ore proposing to w;
- Since he is paired with w, woman w

Stable Matching Problem: Gale - Shapley algorithm

Claim 3: The matching produced by the algorithm is stable.

Proof: Note that during the *While* loop:

- a woman is paired with men of increasing ranks on her list;
- a man is paired with women of decreasing ranks on his list.

Assume n

thus, ther t:

- Since m prefers w' over w, he must hav ore proposing to w;
- Since he is paired with w, woman w
  - rejected him because she was already with someone whom she prefers, or

Claim 3: The matching produced by the algorithm is stable.

Proof: Note that during the *While* loop:

- a woman is paired with men of increasing ranks on her list;
- a man is paired with women of decreasing ranks on his list.

Assume n

thus, ther t:

- Since m prefers w' over w, he must hav ore proposing to w;
- Since he is paired with w, woman w
  - rejected him because she was already with someone whom she prefers, or
  - dropped him later after a proposal from someone whom she prefers;

Claim 3: The matching produced by the algorithm is stable.

Proof: Note that during the *While* loop:

- a woman is paired with men of increasing ranks on her list;
- a man is paired with women of decreasing ranks on his list.

Assume n

thus, ther t:

- Since m prefers w' over w, he must hav ore proposing to w;
- Since he is paired with w, woman w
  - rejected him because she was already with someone whom she prefers, or
  - dropped him later after a proposal from someone whom she prefers;
- In both cases she would now be with m' whom she prefers over m.

Claim 3: The matching produced by the algorithm is stable.

Proof: Note that during the *While* loop:

- a woman is paired with men of increasing ranks on her list;
- a man is paired with women of decreasing ranks on his list.

Assume n

thus, ther t:

- Since m prefers w' over w, he must hav ore proposing to w;
- Since he is paired with w, woman w
  - rejected him because she was already with someone whom she prefers, or
  - dropped him later after a proposal from someone whom she prefers;
- In both cases she would now be with m' whom she prefers over m.
- **Contradiction!**

A Puzzle!!!

Why puzzles? It is a fun way to practice problem solving!

Problem : Tom and his wife Mary went to a party where nine more couples were present.

- Not
  eac

- People who knew each other from before did no
- Later that evening Tom got bored, so he walke
  all other guests (including his wife) how many
  shaken that evening, and got 19 different answers.

- How many hands did Mary shake?
- How many hands did Tom shake?

Assignment Project Exam Help

https://eduassistpro.github.i

Add WeChat edu_assist_pr

That's All, Folks!!