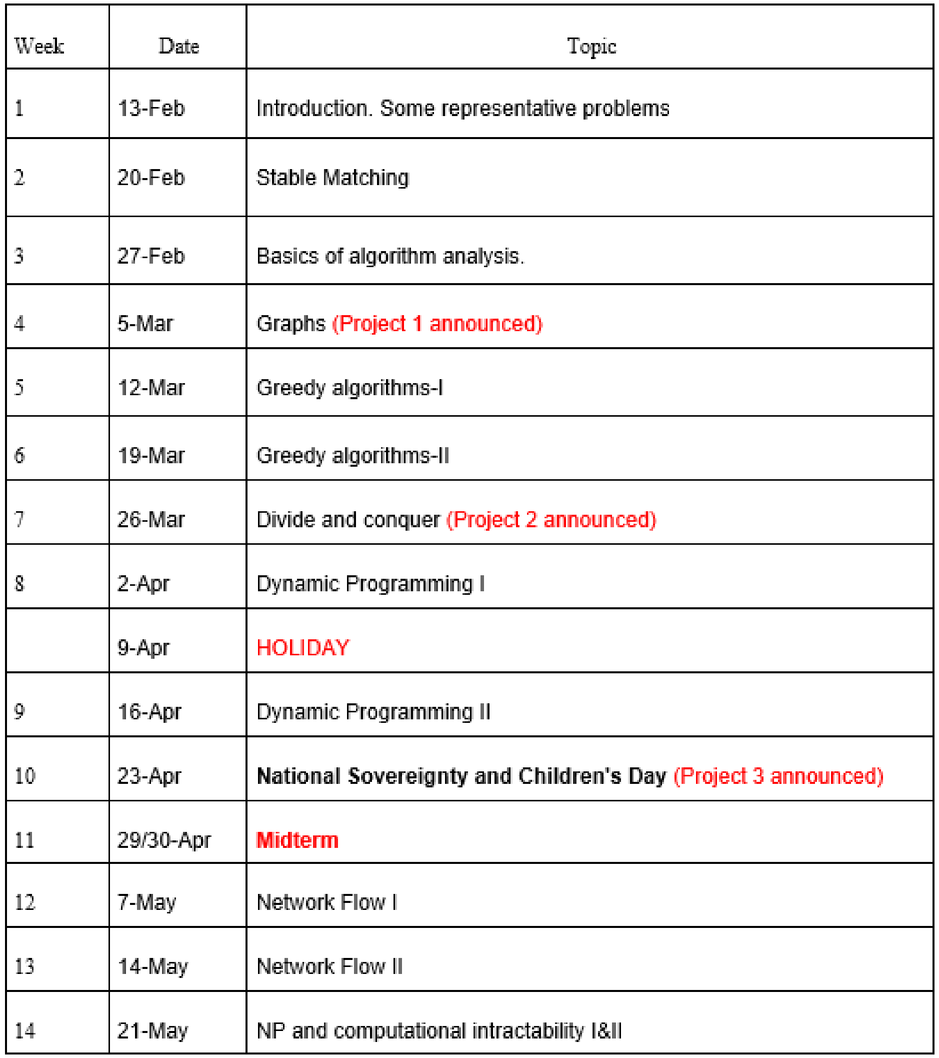
BLG 336E

Analysis of Algorithms II

Lecture 2:

Introduction, Stable Matching, and Gale-Shapley Algorithm

Syllabus and Grading



Week	Date	Topic
1	13-Feb	Introduction. Some representative problems
2	20-Feb	Stable Matching
3	27-Feb	Basics of algorithm analysis.
4	5-Mar	Graphs (Project 1 announced)
5	12-Mar	Greedy algorithms-I
6	19-Mar	Greedy algorithms-II
7	26-Mar	Divide and conquer (Project 2 announced)
8	2-Apr	Dynamic Programming I
	9-Apr	HOLIDAY
9	16-Apr	Dynamic Programming II
10	23-Apr	**National Sovereignty and Children's Day** (Project 3 announced)
11	29/30-Apr	**Midterm**
12	7-May	Network Flow I
13	14-May	Network Flow II
14	21-May	NP and computational intractability I&II

Grading

- 3 Programming Projects (30%)
- 1 Midterm (30%)
- Final (40%)

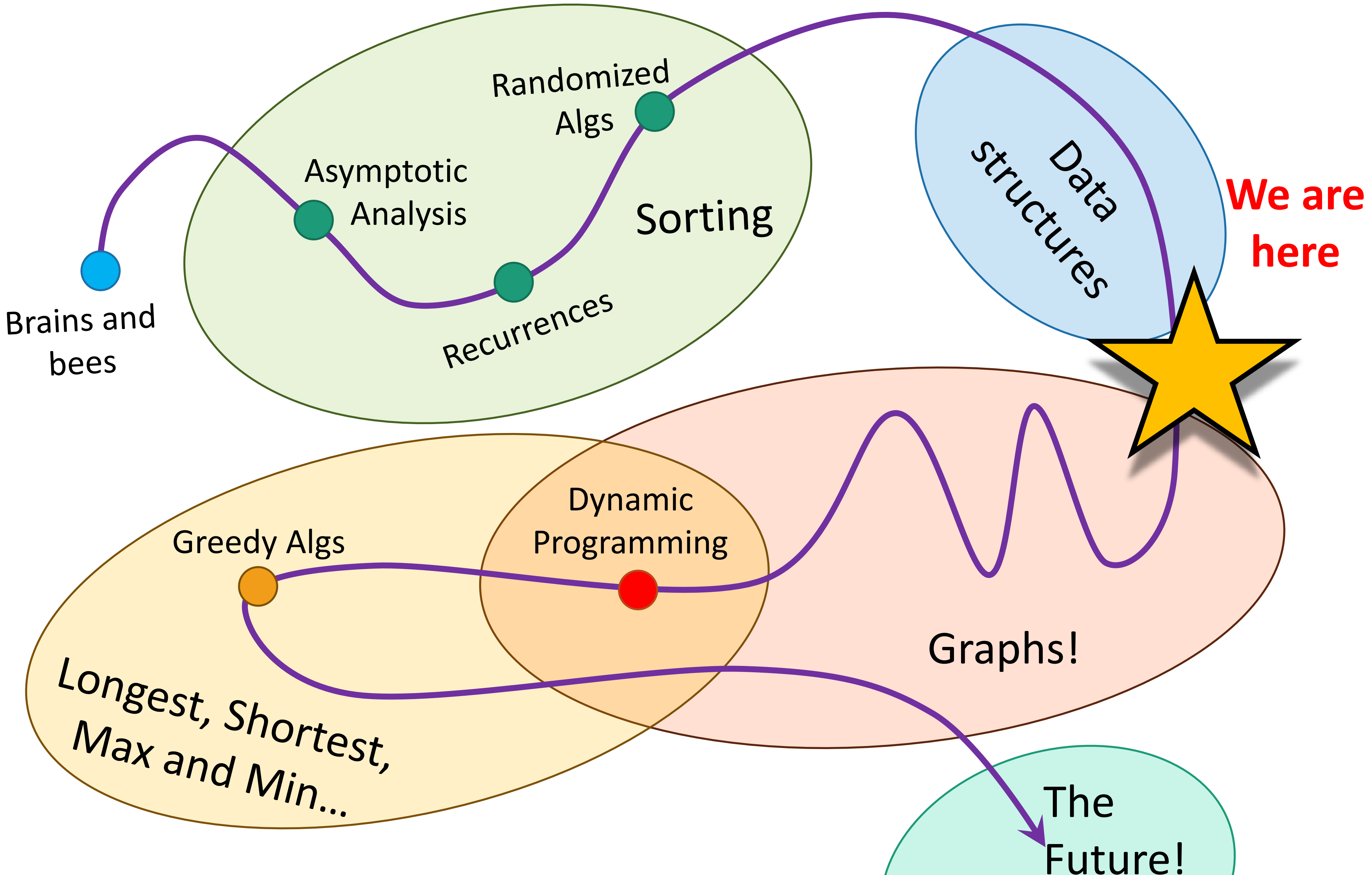
Please note:

- VF Condition **15/50 (First 2 Hws+Midterm)**

Homework!

- 3 Homeworks each %10 (Total %30)
- Use C++ and object oriented approach in your assignments.
- The goal is to practice your implementation skills, so copy-pasting is not encouraged.
- There will be explicit instructions on which files to submit, how it should be compiled, example cases etc.

Roadmap


Randomized Algs
Asymptotic Analysis
Sorting
Recurrences
Brains and bees
Data structures
We are here
Dynamic Programming
Greedy Algs
Graphs!
Longest, Shortest, Max and Min…
The Future!

Let’s start at the beginning

Sorting Problem

Input: A sequence of *n* numbers

$<a_1, a_2, \ldots, a_n>$

Output: A permutation (reordering)

$<a_1', a_2', \ldots, a_n'>$

such that

$a_1' <= a_2' <= \ldots <= a_n'$

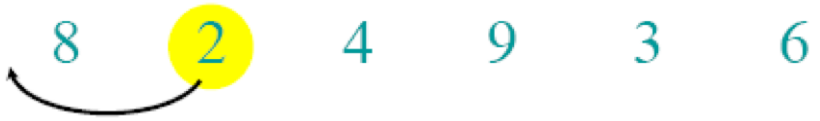
Insertion Sort

- Simple algorithm
- Basic idea:
  - Assume initial j-1 elements are sorted
  - Until you find place to insert j^{th} element, move array elements to right
  - Copy j^{th} element into its place
- Insertion Sort is an “in place” sorting algorithm. No extra storage is required.

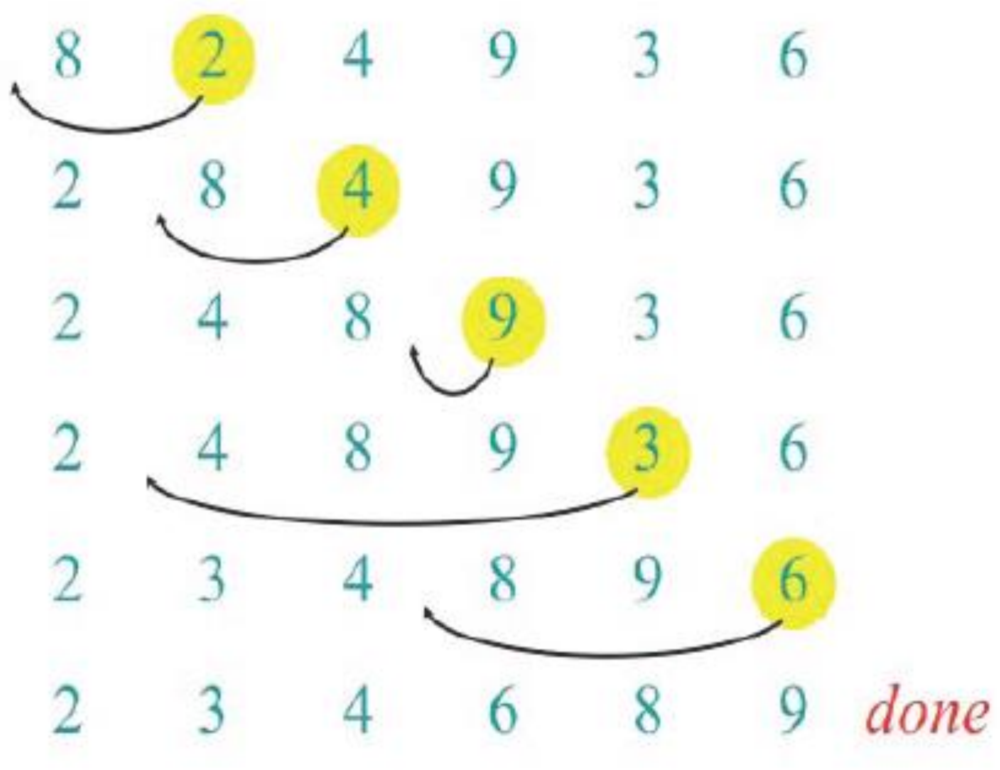
Insertion Sort Example

8 2 4 9 3 6

Insertion Sort Example



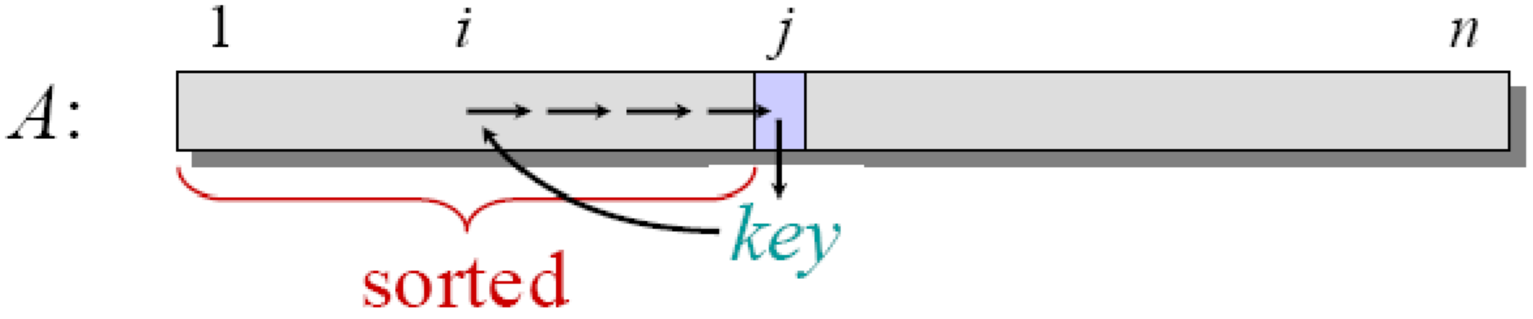
Insertion Sort Example



Insertion Sort

“pseudocode”

```
INSERTION-SORT $(A, n)$        $\triangleright A[1 \,.\,.\, n]$
    for $j \leftarrow 2$ to $n$
        do $key \leftarrow A[j]$
           $i \leftarrow j - 1$
           while $i > 0$ and $A[i] > key$
               do $A[i+1] \leftarrow A[i]$
                  $i \leftarrow i - 1$
           $A[i+1] = key$
```



Pseudocode Conventions

- Indentation
    - indicates block structure
    - saves space and writing time
- Looping constructs (**while, for, repeat**) and conditional constructs (**if, then, else**)
    - like in C, C++, and Java
    - we assume that loop variable in a **for** loop is still defined when loop exits
- Multiple assignment $i \leftarrow j \leftarrow e$ assigns to both variables i and j value of e *(==* $j \leftarrow e$*,* $i \leftarrow j$*)*
- Variables are local, unless otherwise specified

Pseudocode Conventions

- Array elements are accessed by specifying array name followed by index in square brackets
  - A[i] indicates ith element of array A
  - Notation ".." is used to indicate a range of values within an array (A[i..j] = A[1], A[2],…, A[j])
- We often use **objects**, which have **attributes** (equivalently, **fields**)
  - For an attribute *attr* of object x, we write *attr*[*x*]
  - Equivalent of *x.attr* in Java or *x*-> *attr* in C++
- Objects are treated as references, like in Java
  - If *x* and *y* denote objects, then assignment *y* ← *x* makes *x* and *y* reference same object
  - It does not cause attributes of one object to be copied to another

Pseudocode Conventions

- Parameters are passed **by value**, as in Java and C (and the default mechanism in C++).
    - When an object is passed by value, it is actually a reference (or pointer) that is passed
    - Changes to the reference itself are not seen by caller, but changes to the object's attributes are
- Boolean operators "and" and "or" are **short-circuiting**
    - If after evaluating left-hand operand, we know result of expression, then we do not evaluate right-hand operand
    - If x is FALSE in "x and y", then we do not evaluate y
    - If x is TRUE in "x or y", then we do not evaluate y

Efficiency

- Correctness alone is not sufficient
- **Brute-force** algorithms exist for most problems
- To sort *n* numbers, we can enumerate all permutations of these numbers and test which permutation has the correct order
    - Why cannot we do this?
    - Too slow!
    - By what standard?

How to measure complexity?

- Accurate running time is not a good measure
- It depends on **input**
- It depends on the **machine** you used and who implemented the algorithm

- We would like to have an analysis that **does not depend** on those factors

Machine-independent

- A generic uniprocessor random-access machine (RAM) model
  - No concurrent operations
  - Each simple operation (e.g. +, -, =, *, if, for) takes 1 step.
    - Loops and subroutine calls are *not* simple operations.
  - All memory equally expensive to access
    - Constant word size
    - Unless we are explicitly manipulating bits
    - No memory hierarch (caches, virtual mem) is modeled

Running Time

- **Running Time:T(n):** Number of primitive operations or steps executed for an input of size n.
- Running time depends on input
  - already sorted sequence is easier to sort
- Parameterize running time by size of input
  - short sequences are easier to sort than long ones
- Generally, we seek upper bounds on running time
  - everybody likes a guarantee

Kinds of Analysis

- **Worst-case:** (usually)
  - T(*n*) = maximum time of algorithm on any input of size *n*
- **Average-case:** (sometimes)
  - T(*n*) = expected time of algorithm over all inputs of size *n*
  - Need assumption about statistical distribution of inputs
- **Best-case:** (bogus)
  - Cheat with a slow algorithm that works fast on some input

Analyzing Insertion Sort

- $T(n) = c_1n + c_2(n-1) + c_3(n-1) + c_4S + c_5(S - (n-1)) + c_6(S - (n-1)) + c_7(n-1)$

  $= c_8S + c_9n + c_{10}$
- What can S be?
  - Best case -- inner loop body never executed
    - $t_j = 1$ ⍰ $S = n - 1$
    - $T(n) = an + b$ is a linear function
  - Worst case -- inner loop body executed for all previous elements
    - $t_j = j$ ⍰ $S = 2 + 3 + \ldots + n = n(n+1)/2 - 1$
    - $T(n) = an^2 + bn + c$ is a quadratic function
  - Average case
    - Can assume that in average, we have to insert A[j] into the middle of A[1..j-1], so $t_j = j/2$
    - $S \approx n(n+1)/4$
    - T(n) is still a quadratic function

Insertion Sort Running Time

Theta Notation, see next week.

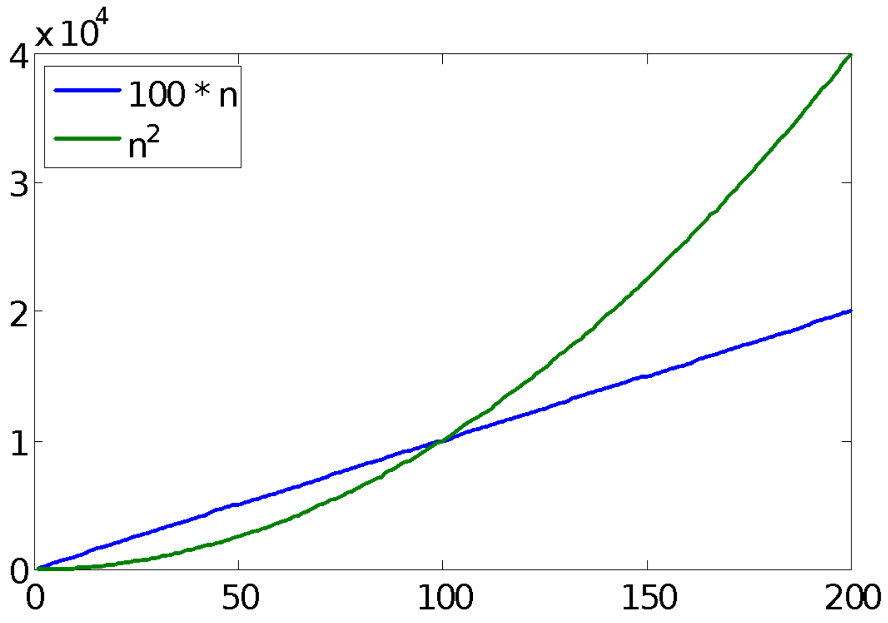
- **Best-case:**
  - $\Theta(n)$, inner loop not executed at all
- **Worst-case:** Input reverse sorted
  - $T(n) = \sum_{j=2}^{n} \Theta(j) = \Theta(n^2)$ [Arithmetic series]
- **Average-case:** All permutations equally likely
  - $T(n) = \sum_{j=2}^{n} \Theta(j/2) = \Theta(n^2)$

Is Insertion Sort a fast sorting algorithm?

- Moderately so, for small *n*
- Not at all, for large *n*

Asymptotic Analysis

- Ignore actual and abstract statement costs
- *Order of growth* is the interesting measure:
    - Highest-order term is what counts
        - As the input size grows larger it is the high order term that dominates



1.1 A First Problem: Stable Matching

Matching Residents to Hospitals

Goal. Given a set of preferences among hospitals and medical school students, design a self-reinforcing admissions process.

Unstable pair: applicant x and hospital y are unstable if:

- x prefers y to its assigned hospital.
- y prefers x to one of its admitted students.

Stable assignment. Assignment with no unstable pairs.

- Natural and desirable condition.
- Individual self-interest will prevent any applicant/hospital deal from being made.

Stable Matching Problem

Perfect matching: everyone is matched monogamously.

- Each man gets exactly one woman.
- Each woman gets exactly one man.

Stability: no incentive for some pair of participants to undermine assignment by joint action.

- In matching M, an unmatched pair m-w is unstable if man m and woman w prefer each other to current partners.
- Unstable pair m-w could each improve by eloping.

Stable matching: perfect matching with no unstable pairs.

Stable matching problem. Given the preference lists of n men and n women, find a stable matching if one exists.

Stable Matching Problem

Goal. Given n men and n women, find a "suitable" matching.

- Participants rate members of opposite sex.
- Each man lists women in order of preference from best to worst.
- Each woman lists men in order of preference from best to worst.

	favorite ↓		least favorite ↓
	1st	2nd	3rd
Xavier	Amy	Bertha	Clare
Yancey	Bertha	Amy	Clare
Zeus	Amy	Bertha	Clare

Men's Preference Profile

	favorite ↓		least favorite ↓
	1st	2nd	3rd
Amy	Yancey	Xavier	Zeus
Bertha	Xavier	Yancey	Zeus
Clare	Xavier	Yancey	Zeus

Women's Preference Profile

Stable Matching Problem

Q. Is assignment X-C, Y-B, Z-A stable?

	favorite ↓		least favorite ↓
	1st	2nd	3rd
Xavier	Amy	Bertha	Clare
Yancey	Bertha	Amy	Clare
Zeus	Amy	Bertha	Clare

Men's Preference Profile

	favorite ↓		least favorite ↓
	1st	2nd	3rd
Amy	Yancey	Xavier	Zeus
Bertha	Xavier	Yancey	Zeus
Clare	Xavier	Yancey	Zeus

Women's Preference Profile

Stable Matching Problem

Q. Is assignment X-C, Y-B, Z-A stable?

A. No. Bertha and Xavier will hook up.

favorite ↓ … least favorite ↓

	1st	2nd	3rd
Xavier	Amy	Bertha	Clare
Yancey	Bertha	Amy	Clare
Zeus	Amy	Bertha	Clare

Men's Preference Profile

favorite ↓ … least favorite ↓

	1st	2nd	3rd
Amy	Yancey	Xavier	Zeus
Bertha	Xavier	Yancey	Zeus
Clare	Xavier	Yancey	Zeus

Women's Preference Profile

Stable Matching Problem

Q. Is assignment X-A, Y-B, Z-C stable?

A. Yes.

favorite ↓ least favorite ↓

	1st	2nd	3rd
Xavier	Amy	Bertha	Clare
Yancey	Bertha	Amy	Clare
Zeus	Amy	Bertha	Clare

Men's Preference Profile

favorite ↓ least favorite ↓

	1st	2nd	3rd
Amy	Yancey	Xavier	Zeus
Bertha	Xavier	Yancey	Zeus
Clare	Xavier	Yancey	Zeus

Women's Preference Profile

Stable Roommate Problem

Q. Do stable matchings always exist?

A. Not obvious a priori.

$\nwarrow$ *is core of market (a housing term) nonempty?*

Stable roommate problem.

- 2n people; each person ranks others from 1 to 2n-1.
- Assign roommate pairs so that no unstable pairs.

	1st	2nd	3rd
Adam	B	C	D
Bob	C	A	D
Chris	A	B	D
Doofus	A	B	C

A-B, C-D	$\Rightarrow$	B-C unstable
A-C, B-D	$\Rightarrow$	A-B unstable
A-D, B-C	$\Rightarrow$	A-C unstable

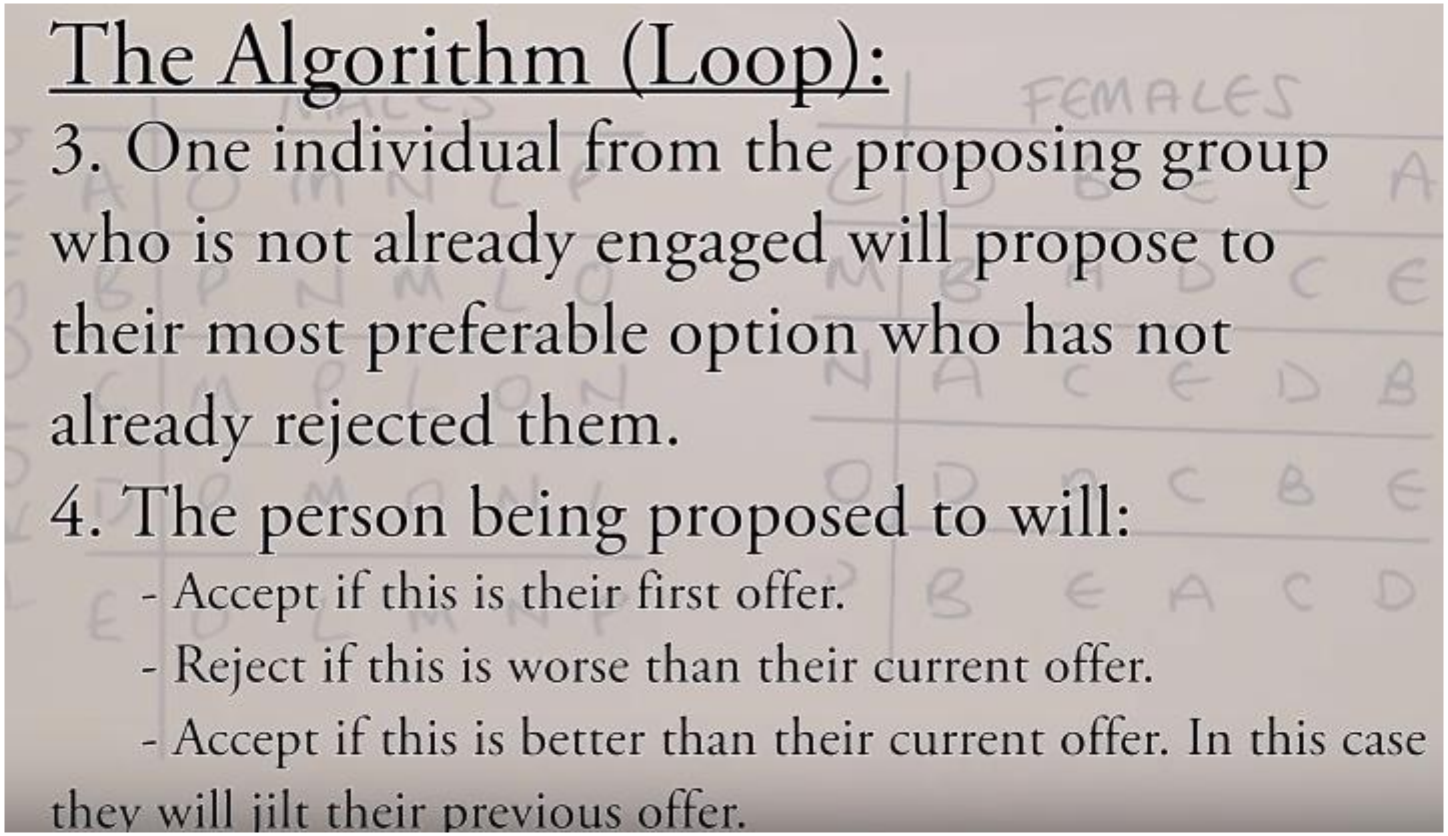
Observation. Stable matchings do not always exist for stable roommate problem.

Propose-And-Reject Algorithm

Propose-and-reject algorithm. [Gale-Shapley 1962] Intuitive method that guarantees to find a stable matching.

```
Initialize each person to be free.
while (some man is free and hasn't proposed to every woman) {
    Choose such a man m
    w = 1st woman on m's list to whom m has not yet proposed
    if (w is free)
        assign m and w to be engaged
    else if (w prefers m to her fiancé m')
        assign m and w to be engaged, and m' to be free
    else
        w rejects m
}
```

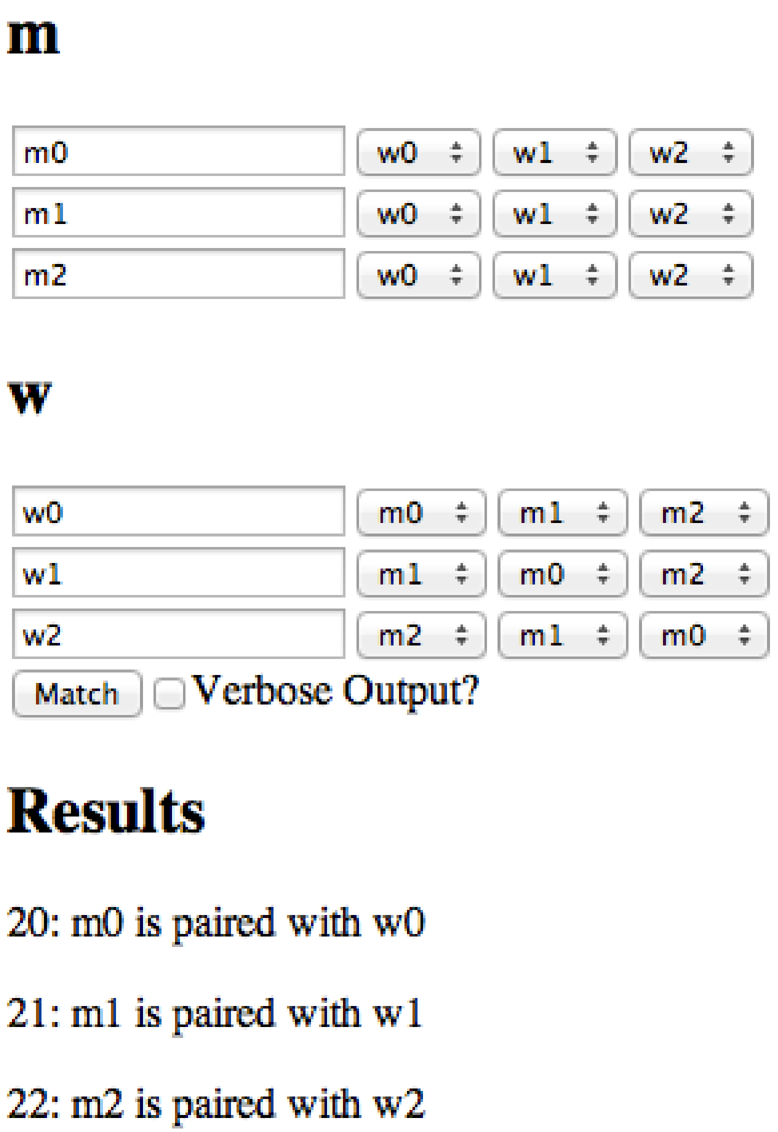
Algorithm Flow



The Algorithm (Loop):

3. One individual from the proposing group who is not already engaged will propose to their most preferable option who has not already rejected them.
4. The person being proposed to will:
   - Accept if this is their first offer.
   - Reject if this is worse than their current offer.
   - Accept if this is better than their current offer. In this case they will jilt their previous offer.

Demo (from: http://sephlietz.com/gale-shapley/)



m

m0	w0	w1	w2
m1	w0	w1	w2
m2	w0	w1	w2

w

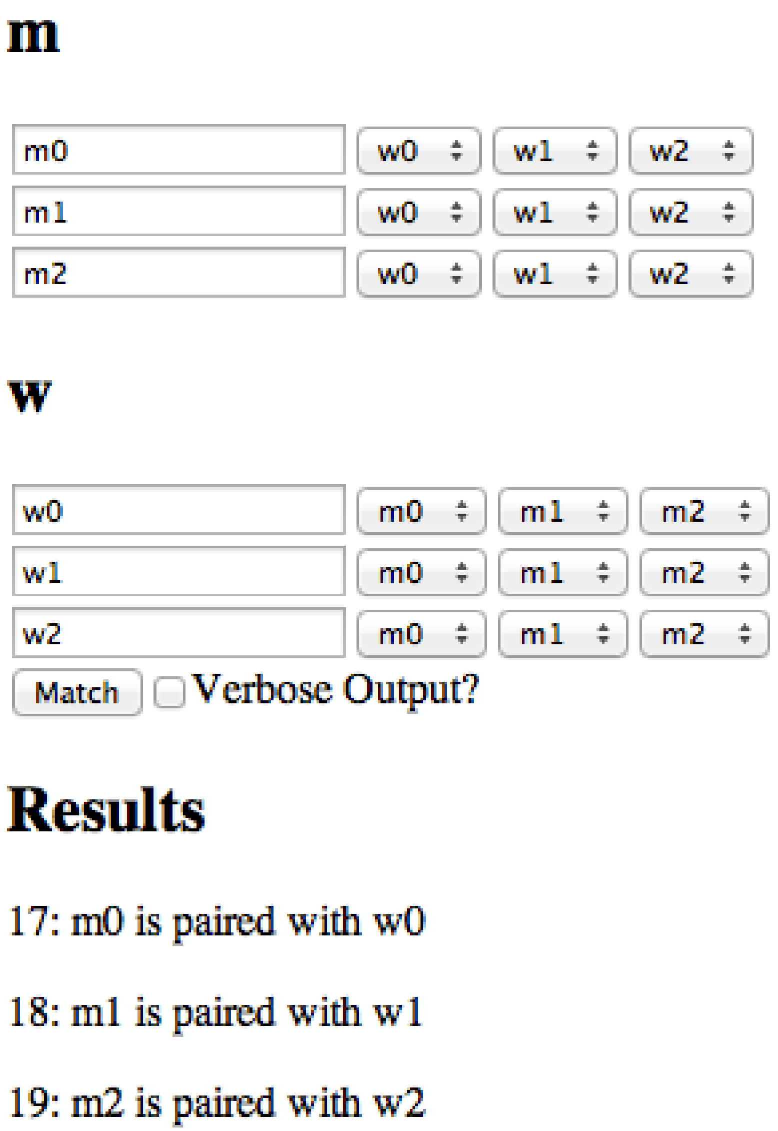
w0	m0	m1	m2
w1	m1	m0	m2
w2	m2	m1	m0

Results

```
20: m0 is paired with w0

21: m1 is paired with w1

22: m2 is paired with w2
```



m

m0	w0	w1	w2
m1	w0	w1	w2
m2	w0	w1	w2

w

w0	m0	m1	m2
w1	m0	m1	m2
w2	m0	m1	m2

Results

```
17: m0 is paired with w0

18: m1 is paired with w1

19: m2 is paired with w2
```

Proof of Correctness: Termination

Observation 1. Men propose to women in decreasing order of preference.

Observation 2. Once a woman is matched, she never becomes unmatched; she only "trades up."

Claim. Algorithm terminates after at most n^2 iterations of while loop.
Pf. Each time through the while loop a man proposes to a new woman. There are only n^2 possible proposals. ▪

	1st	2nd	3rd	4th	5th
Victor	A	B	C	D	E
Wyatt	B	C	D	A	E
Xavier	C	D	A	B	E
Yancey	D	A	B	C	E
Zeus	A	B	C	D	E

	1st	2nd	3rd	4th	5th
Amy	W	X	Y	Z	V
Bertha	X	Y	Z	V	W
Clare	Y	Z	V	W	X
Diane	Z	V	W	X	Y
Erika	V	W	X	Y	Z

$n(n-1) + 1$ proposals required

Proof of Correctness: Perfection

Claim. All men and women get matched.

Pf. (by contradiction)

- Suppose, for sake of contradiction, that Zeus is not matched upon termination of algorithm.
- Then some woman, say Amy, is not matched upon termination.
- By Observation 2, Amy was never proposed to.
- But, Zeus proposes to everyone, since he ends up unmatched. ▪

	1st	2nd	3rd	4th	5th
Victor	A	B	C	D	E
Wyatt	B	C	D	A	E
Xavier	C	D	A	B	E
Yancey	D	A	B	C	E
Zeus	A	B	C	D	E

	1st	2nd	3rd	4th	5th
Amy	W	X	Y	Z	V
Bertha	X	Y	Z	V	W
Clare	Y	Z	V	W	X
Diane	Z	V	W	X	Y
Erika	V	W	X	Y	Z

- Note: Please read the following to review proof of contradiction:
- http://zimmer.csufresno.edu/~larryc/proofs/proofs.contradict.html
- Proofs in general: http://zimmer.csufresno.edu/~larryc/proofs/proofs.html

Proof of Correctness: Stability

Claim. No unstable pairs.

Pf. (by contradiction)

- Suppose A-Z is an unstable pair: each prefers each other to partner in Gale-Shapley matching S*.

- Case 1: Z never proposed to A.
  - $\Rightarrow$ Z prefers his GS partner to A. ← men propose in decreasing order of preference
  - $\Rightarrow$ A-Z is stable.

S*
Amy-Yancey
Bertha-Zeus
. . .

- Case 2: Z proposed to A.
  - $\Rightarrow$ A rejected Z (right away or later)
  - $\Rightarrow$ A prefers her GS partner to Z. ← women only trade up
  - $\Rightarrow$ A-Z is stable.

- In either case A-Z is stable, a contradiction. ▪

Summary

Stable matching problem. Given n men and n women, and their preferences, find a stable matching if one exists.

Gale-Shapley algorithm. Guarantees to find a stable matching for any problem instance.

Q. How to implement GS algorithm efficiently?

Q. If there are multiple stable matchings, which one does GS find?

Efficient Implementation

Efficient implementation. We describe O(n^2) time implementation.

Representing men and women.

- Assume men are named 1, …, n.
- Assume women are named 1', …, n'.

Engagements.

- Maintain a list of free men, e.g., in a queue.
- Maintain two arrays `wife[m]`, and `husband[w]`.
    - set entry to `0` if unmatched
    - if m matched to w then `wife[m]=w` and `husband[w]=m`

Men proposing.

- For each man, maintain a list of women, ordered by preference.
- Maintain an array `count[m]` that counts the number of proposals made by man `m`.

Efficient Implementation

Women rejecting/accepting.

- Does woman `w` prefer man `m` to man `m'`?
- For each woman, create inverse of preference list of men.
- Constant time access for each query after O(n) preprocessing.

Amy	1st	2nd	3rd	4th	5th	6th	7th	8th
Pref	8	3	7	1	4	5	6	2

Amy	1	2	3	4	5	6	7	8
Inverse	4th	8th	2nd	5th	6th	7th	3rd	1st

Amy prefers man 3 to 6
since `inverse[3]` < `inverse[6]`

2 7

```
for i = 1 to n
   inverse[pref[i]] = i
```

Understanding the Solution

Q. For a given problem instance, there may be several stable matchings. Do all executions of Gale-Shapley yield the same stable matching? If so, which one?

	1st	2nd	3rd
Xavier	A	B	C
Yancey	B	A	C
Zeus	A	B	C

	1st	2nd	3rd
Amy	Y	X	Z
Bertha	X	Y	Z
Clare	X	Y	Z

Understanding the Solution

Q. For a given problem instance, there may be several stable matchings. Do all executions of Gale-Shapley yield the same stable matching? If so, which one?

An instance with two stable matchings.

- $\boxed{\text{X-A, Y-B, Z-C.}}$
- $\boxed{\text{Y-A, X-B, Z-C.}}$

	1st	2nd	3rd
Xavier	A	B	C
Yancey	B	A	C
Zeus	A	B	C

	1st	2nd	3rd
Amy	Y	X	Z
Bertha	X	Y	Z
Clare	X	Y	Z

Understanding the Solution

Q. For a given problem instance, there may be several stable matchings. Do all executions of Gale-Shapley yield the same stable matching? If so, which one?

Def. Man m is a valid partner of woman w if there exists some stable matching in which they are matched.

Man-optimal assignment. Each man receives best valid partner.

Claim. All executions of GS yield man-optimal assignment, which is a stable matching!

- No reason a priori to believe that man-optimal assignment is perfect, let alone stable.
- Simultaneously best for each and every man.

Define: S* ={(m,best(m)): m in M} where best(m) is the best valid partner of m

Examples for S* = {(m,best(m)): m in M}

An instance with two stable matchings.

- X-A, Y-B, Z-C.
- Y-A, X-B, Z-C.

S*={(X,A),(Y,B),(Z,C)}

	1st	2nd	3rd
Xavier	A	B	C
Yancey	B	A	C
Zeus	A	B	C

	1st	2nd	3rd
Amy	Y	X	Z
Bertha	X	Y	Z
Clare	X	Y	Z

- X-A, Y-B, Z-C

S*={(X,A),(Y,B),(Z,C)}

	1st	2nd	3rd
Xavier	A	B	C
Yancey	A	B	C
Zeus	A	B	C

	1st	2nd	3rd
Amy	X	Y	Z
Bertha	X	Y	Z
Clare	X	Y	Z

□ man optimal matching

⬚ woman optimal matching

Man Optimality

Claim. Every execution of the GS algorithm results in the (man-optimal) set S* !

Pf. (by contradiction)

- Suppose an execution E of GS resulted in some man paired with someone who is not his best valid partner.
- Since men propose in decreasing order of preference, then there must be some man who is rejected by a valid partner during E.
- Consider the first moment during E in which some man (m) is rejected by a valid partner (w).
- men propose in decreasing order of preference AND this is the first time such a rejection occurred
  - Therefore it must be that best(m)=w
- w may have rejected m,
  - either because m proposed and w turned it down because she was already engaged with someone she prefers more, or
  - w broke her engagement to m in favor of a better proposal.
  - Let m’ be the man whom w prefers to compared to m.

	1st	2nd	3rd	4th	5th
m	w				

	1st	2nd	3rd	4th	5th
w			m′		m

Man Optimality

- Since w is a valid partner of m, there exists a stable matching S’ containing the pair (m,w).
- Let m’ be matched with some w’ ≠w in that matching S’.
  - S’={(m,w),(m’,w’),...}
- Rejection of m by w was the first rejection in THEREFORE m’ had not been rejected by any valid partner at the point in E when he became engaged to w.
- Since m’ proposed in decreasing order of preference AND w’ is a valid partner of m’ THEREFORE m’ prefers w to w’.
- But we have already seen that w prefers m’ to m, because in E she rejected m in favor of m’.
- Since (m’,w) is not in S’, then (m’,w) is an unstable pair in S’ (because both m’ and w are willing to leave their current partners and get engaged, see below)!
- This contradicts our claim that S’ is stable, hence it contradicts our initial assumption.

□ Matching S′ in which (m,w) happens

	1st	2nd	3rd	4th	5th
m	w				
m′			w	w′	

	1st	2nd	3rd	4th	5th
w			m′		m
w′					

Stable Matching Summary

Stable matching problem. Given preference profiles of n men and n women, find a stable matching.

↖ no man and woman prefer to be with each other than assigned partner

Gale-Shapley algorithm. Finds a stable matching in $O(n^2)$ time.

Man-optimality. In version of GS where men propose, each man receives best valid partner.

↖ w is a valid partner of m if there exist some stable matching where m and w are paired

Q. Does man-optimality come at the expense of the women?

Woman Pessimality

Woman-pessimal assignment. Each woman receives worst valid partner.

Claim. GS finds woman-pessimal stable matching S*.

Pf.

- Suppose A-Z matched in S*, but Z is not worst valid partner for As
- There exists stable matching S in which A is paired with a man, say Y, whom she likes less than Z. $\leftarrow$ man-optimality
- Let B be Z's partner in S.
- Z prefers A to B.
- Thus, Z-A is an unstable pair in S. ▪

Yancey-Amy
Zeus-Bertha
. . .

Extensions: Matching Residents to Hospitals

Ex: Men ≈ hospitals, Women ≈ med school residents.

Variant 1. Some participants declare others as unacceptable.

$\nwarrow$ resident A unwilling to work in Cleveland

Variant 2. Unequal number of men and women.

Variant 3. Limited polygamy.

$\nwarrow$ hospital X wants to hire 3 residents

Def. Matching S unstable if there is a hospital h and resident r such that:

- h and r are acceptable to each other; and
- either r is unmatched, or r prefers h to her assigned hospital; and
- either h does not have all its places filled, or h prefers r to at least one of its assigned residents.

Application: Matching Residents to Hospitals

NRMP. (National Resident Matching Program)

- Original use just after WWII. $\longleftarrow$ predates computer usage
- Ides of March, 23,000+ residents.

Rural hospital dilemma.

- Certain hospitals (mainly in rural areas) were unpopular and declared unacceptable by many residents.
- Rural hospitals were under-subscribed in NRMP matching.
- How can we find stable matching that benefits "rural hospitals"?

Rural Hospital Theorem. Rural hospitals get exactly same residents in every stable matching!

Stable Marriage Interesting Notes

other stable marriages possible?
-can be many

More questions, rich theory

do better by lying? boys -No! girls -Yes!

CC Huang, [How Hard is it to Cheat in the Gale-Shapley ...]
To our knowledge, ours is the first attempt in proposing men-*lying*

Stable Matching Publications:

- **Local search algorithms on the stable marriage problem: Experimental studies**
- Gelain, Pini, Rossi, Venable… - 2010

- **Stable marriage with ties and bounded length preference lists**
- Irving, Manlove… - Journal of Discrete Algorithms, 2009

- **Approximation algorithms for hard variants of the stable marriage and hospitals/residents problems**
- RW Irving… - Journal of Combinatorial Optimization, 2008

- **A 1.875: approximation algorithm for the stable marriage problem**
- Iwama, S Miyazaki… - Proceedings of the eighteenth …, 2007

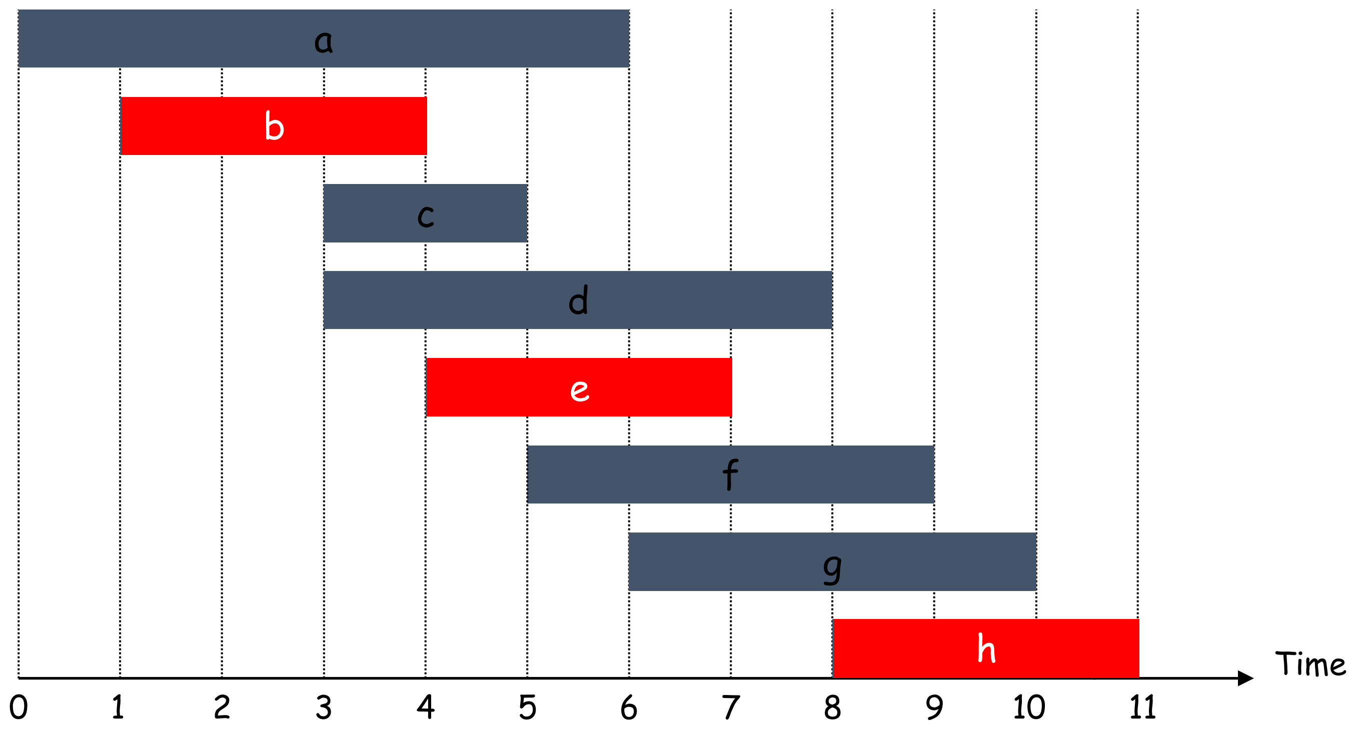
1.2 Five Representative Problems

1. Interval Scheduling

Input. Set of jobs with start times and finish times.

Goal. Find maximum cardinality subset of mutually compatible jobs.

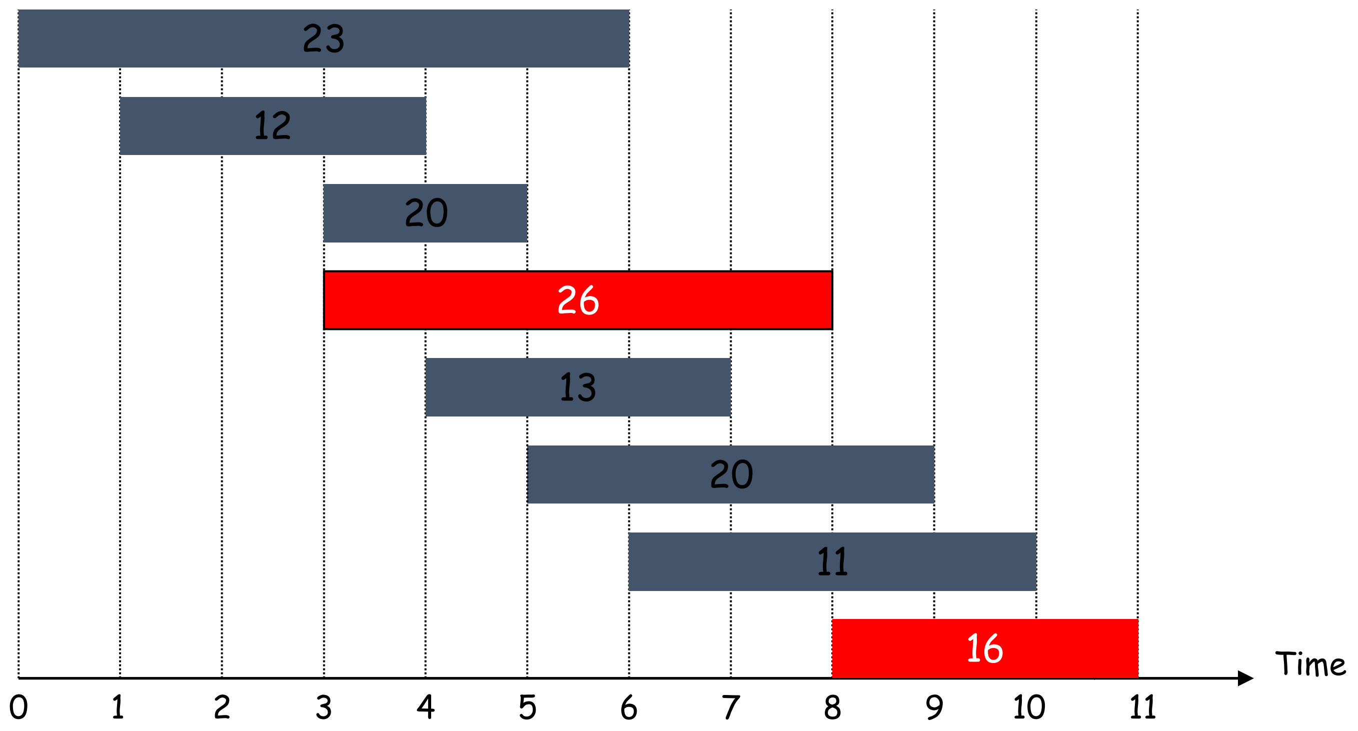
jobs don't overlap



2. Weighted Interval Scheduling

Input. Set of jobs with start times, finish times, and weights.

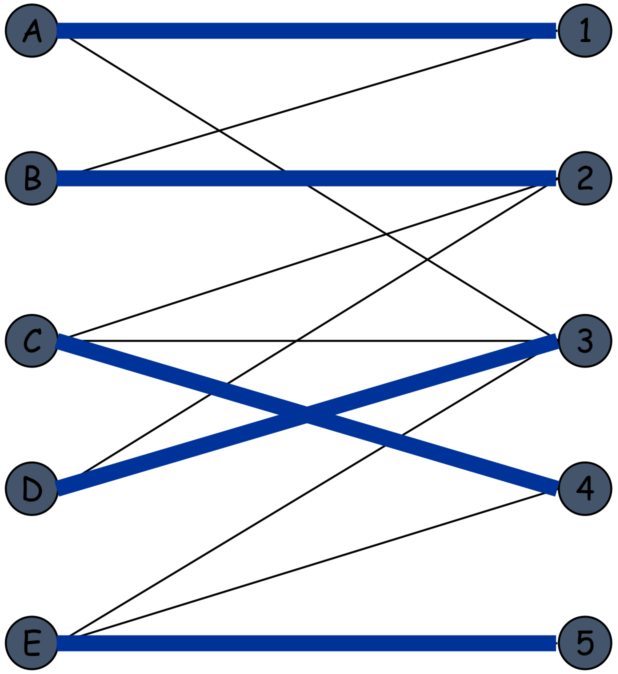
Goal. Find maximum weight subset of mutually compatible jobs.



3. Bipartite Matching

Input. Bipartite graph.

Goal. Find maximum cardinality matching.



Why do we care?

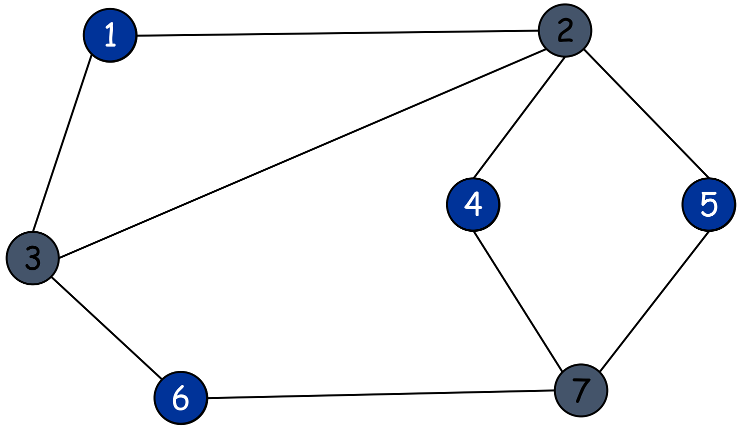
- *There are M job applicants and N jobs.*
- *Each applicant has a subset of jobs that he/she is interested in.*
- *Each job opening can only accept one applicant and a job applicant can be appointed for only one job.*
- *Find an assignment of jobs to applicants in such that as many applicants as possible get jobs.*

4. Independent Set

Input. Graph.

Goal. Find maximum cardinality independent set.

↑
subset of nodes such that no two joined by an edge

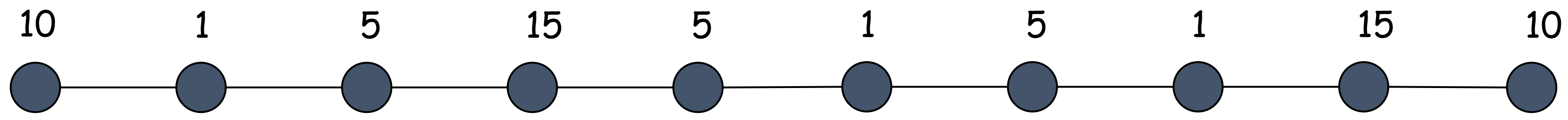


5. Competitive Facility Location

Input. Graph with weight on each node.

Game. Two competing players alternate in selecting nodes. Not allowed to select a node if any of its neighbors have been selected.

Goal. Select a maximum weight subset of nodes.



Second player can guarantee 20, but not 25.

Five Representative Problems

Variations on a theme: independent set.

Interval scheduling: n log n greedy algorithm.

Weighted interval scheduling: n log n dynamic programming algorithm.

Bipartite matching: n^k max-flow based algorithm.

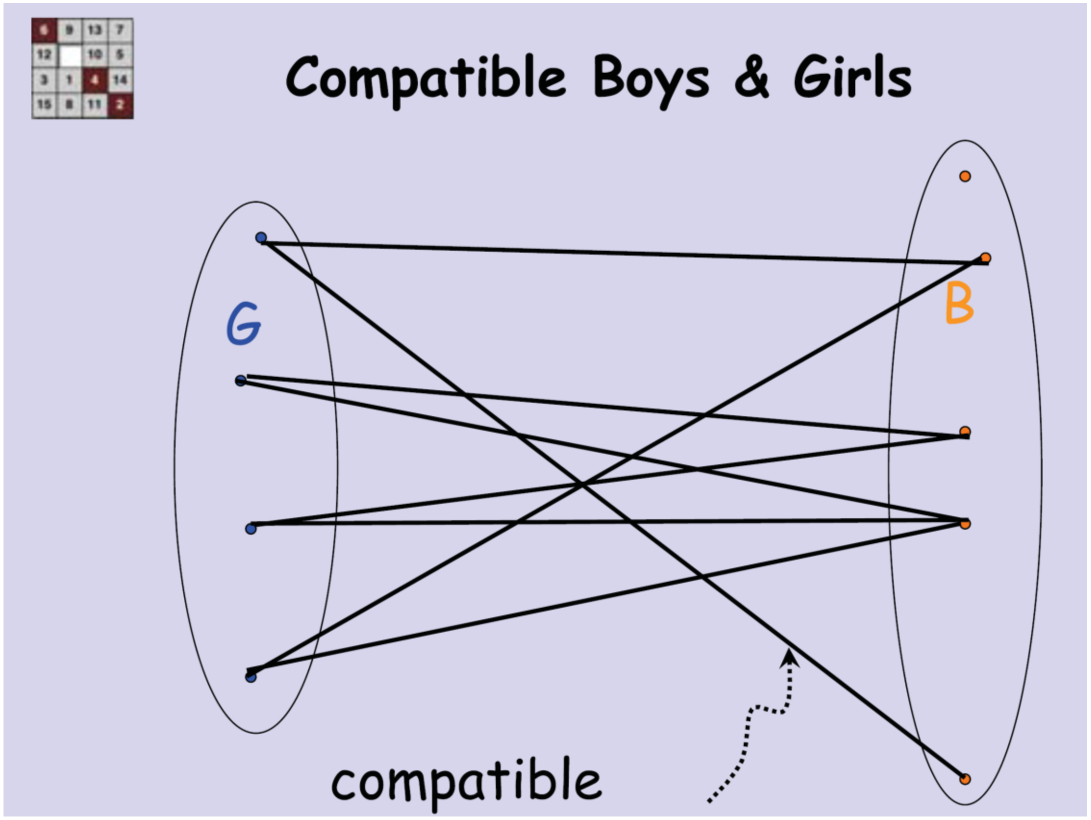
Independent set: NP-complete.

Competitive facility location: PSPACE-complete.

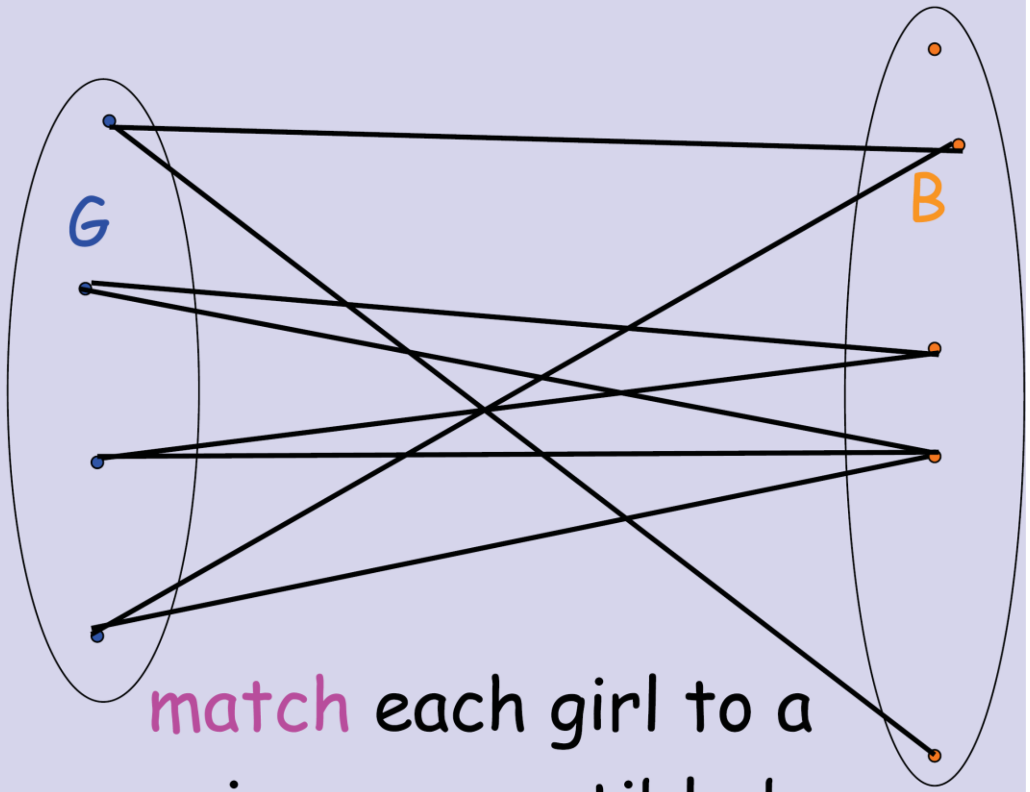
PSPACE: The set of all problems that can be solved by an algorithm with polynomial space complexity (Chapter 9).
P $\subseteq$ PSPACE (in poly time an algorithm can only consume poly space.)
NP $\subseteq$ PSPACE (There is an algorithm that can solve 3-SAT using only a polynomial amount of space.

Bipartite Matching

A matching in a Bipartite Graph is a set of the edges chosen in such a way that no two edges share an endpoint.

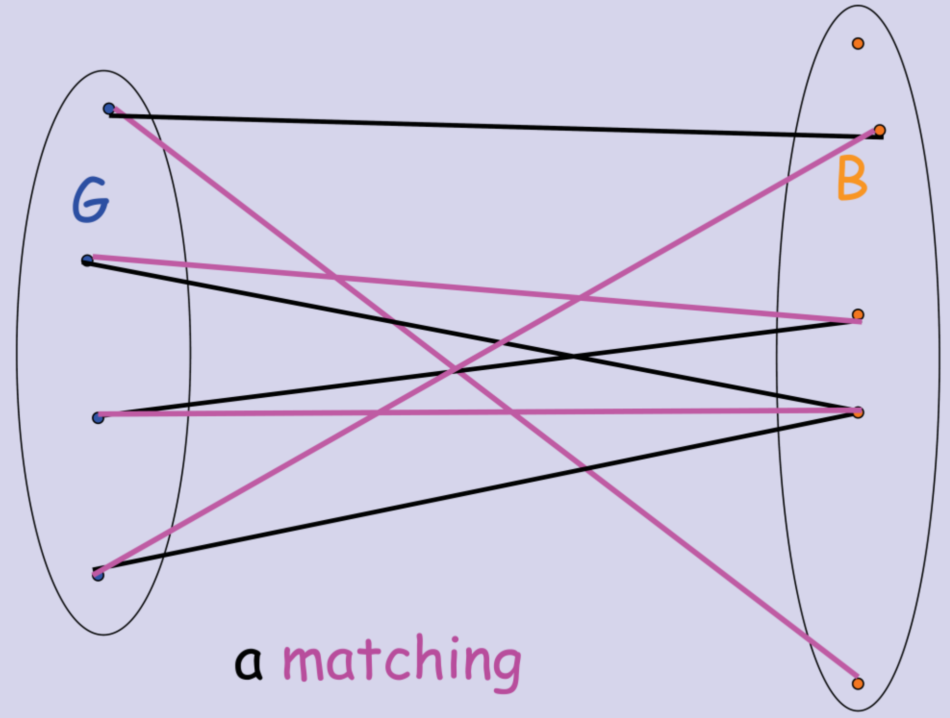


Compatible Boys & Girls

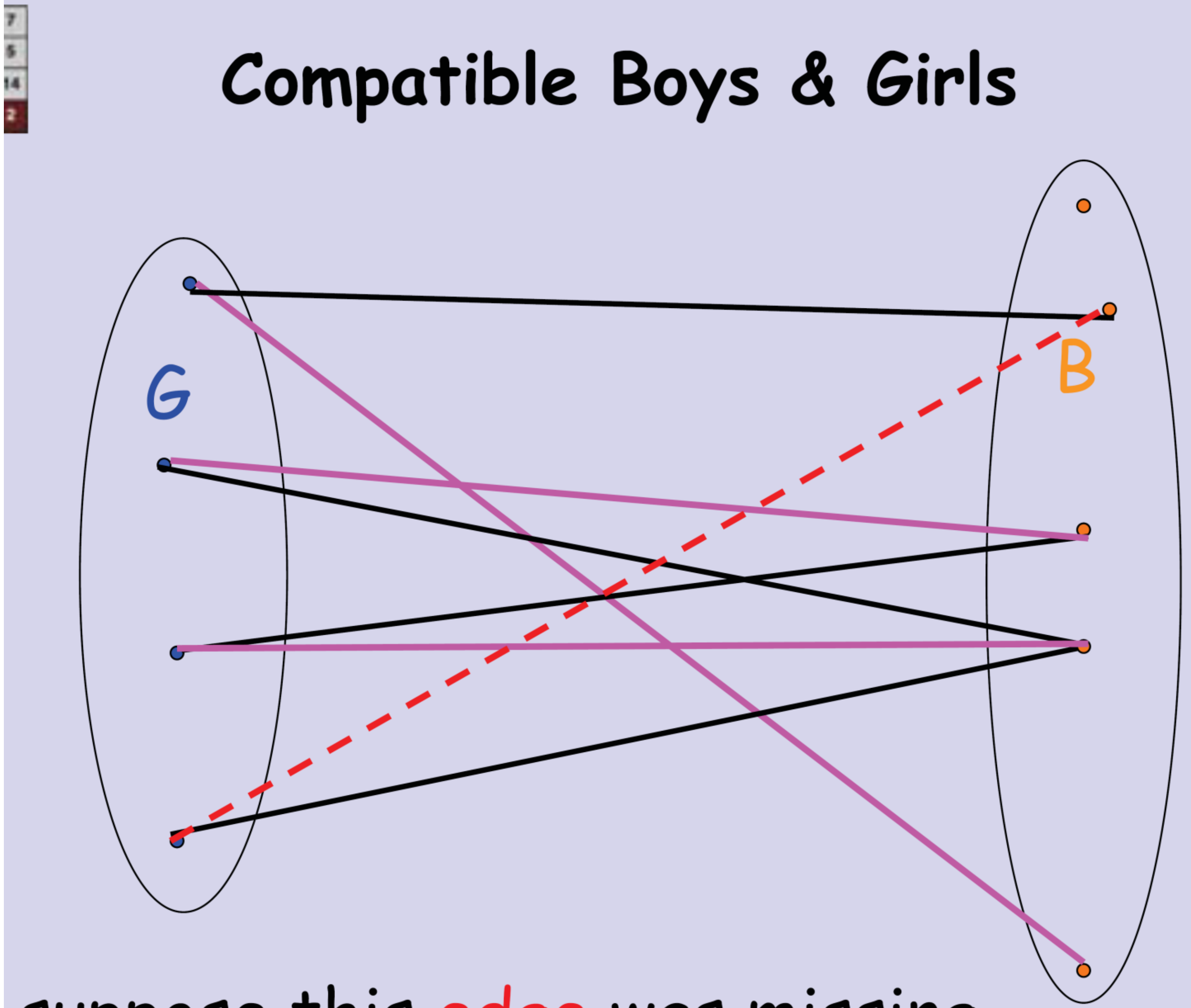


match each girl to a unique compatible boy

Compatible Boys & Girls

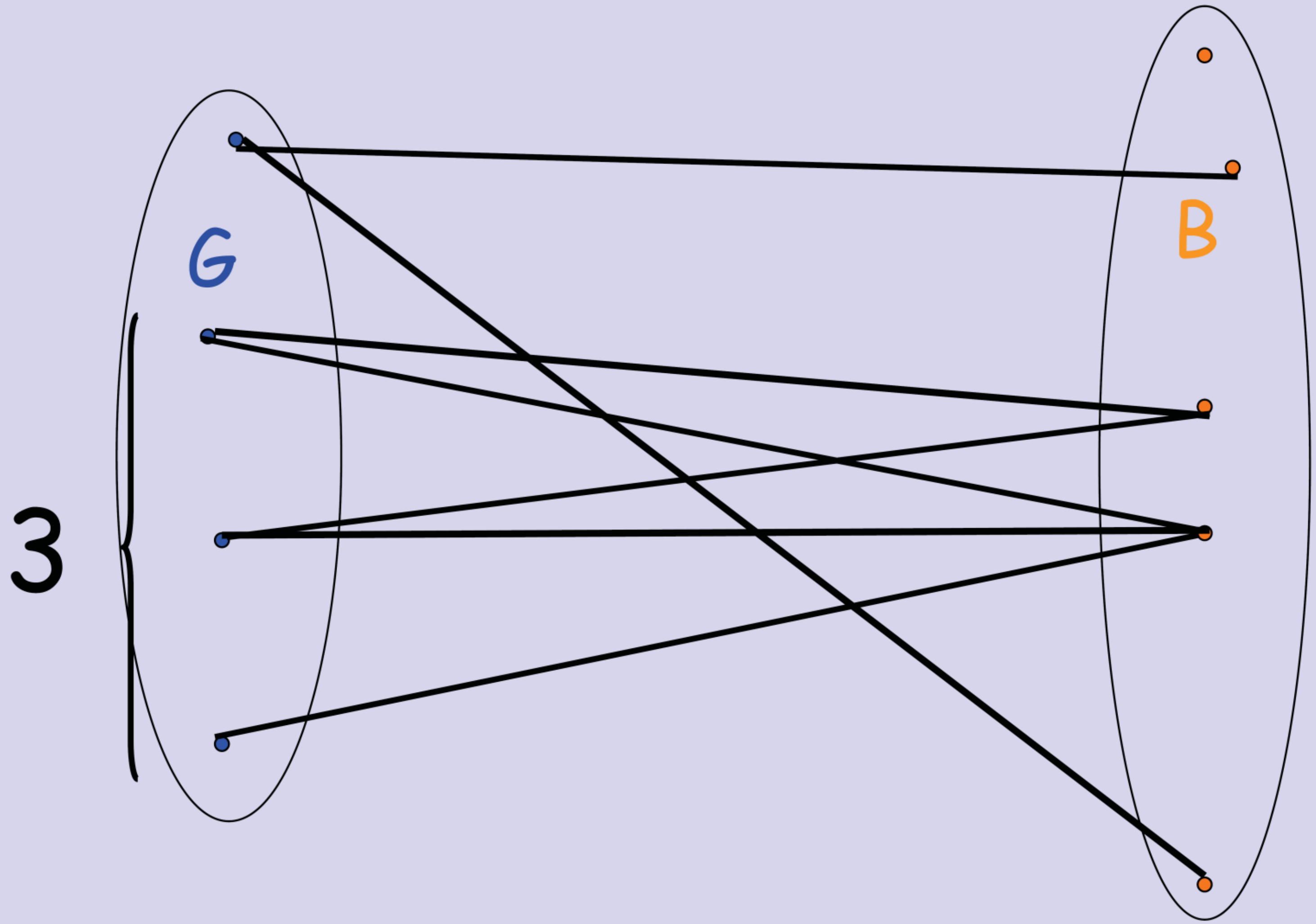


Compatible Boys & Girls

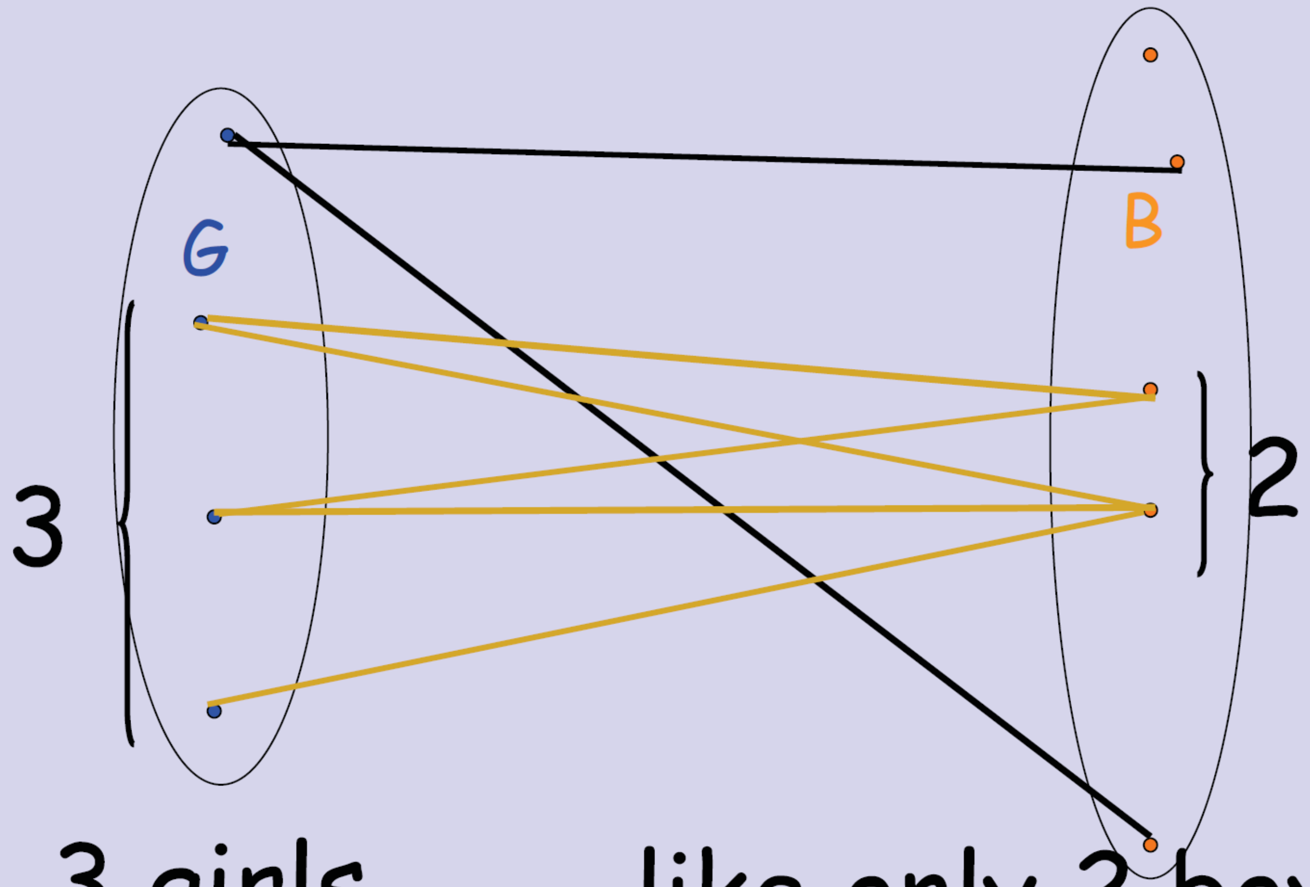


suppose this edge was missing

Compatible Boys & Girls

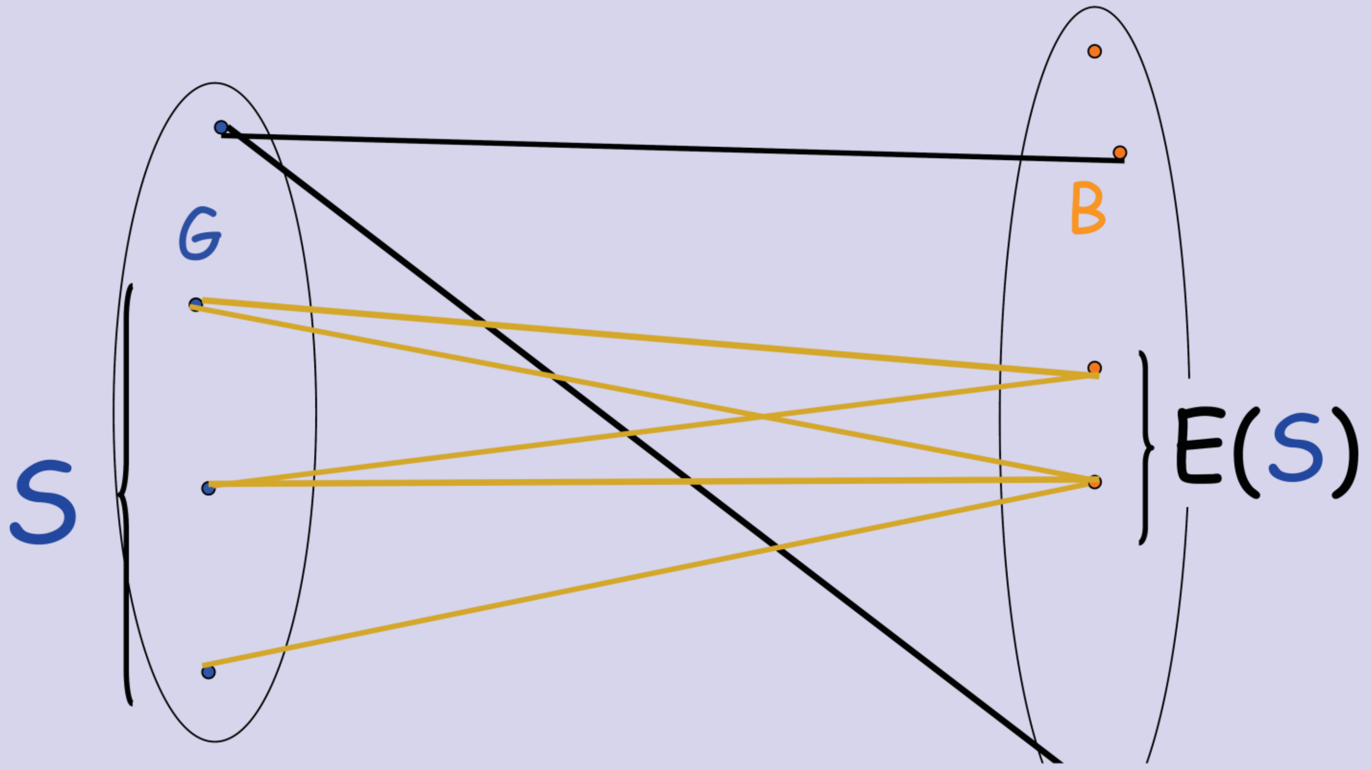


Not enough boys for these girls!



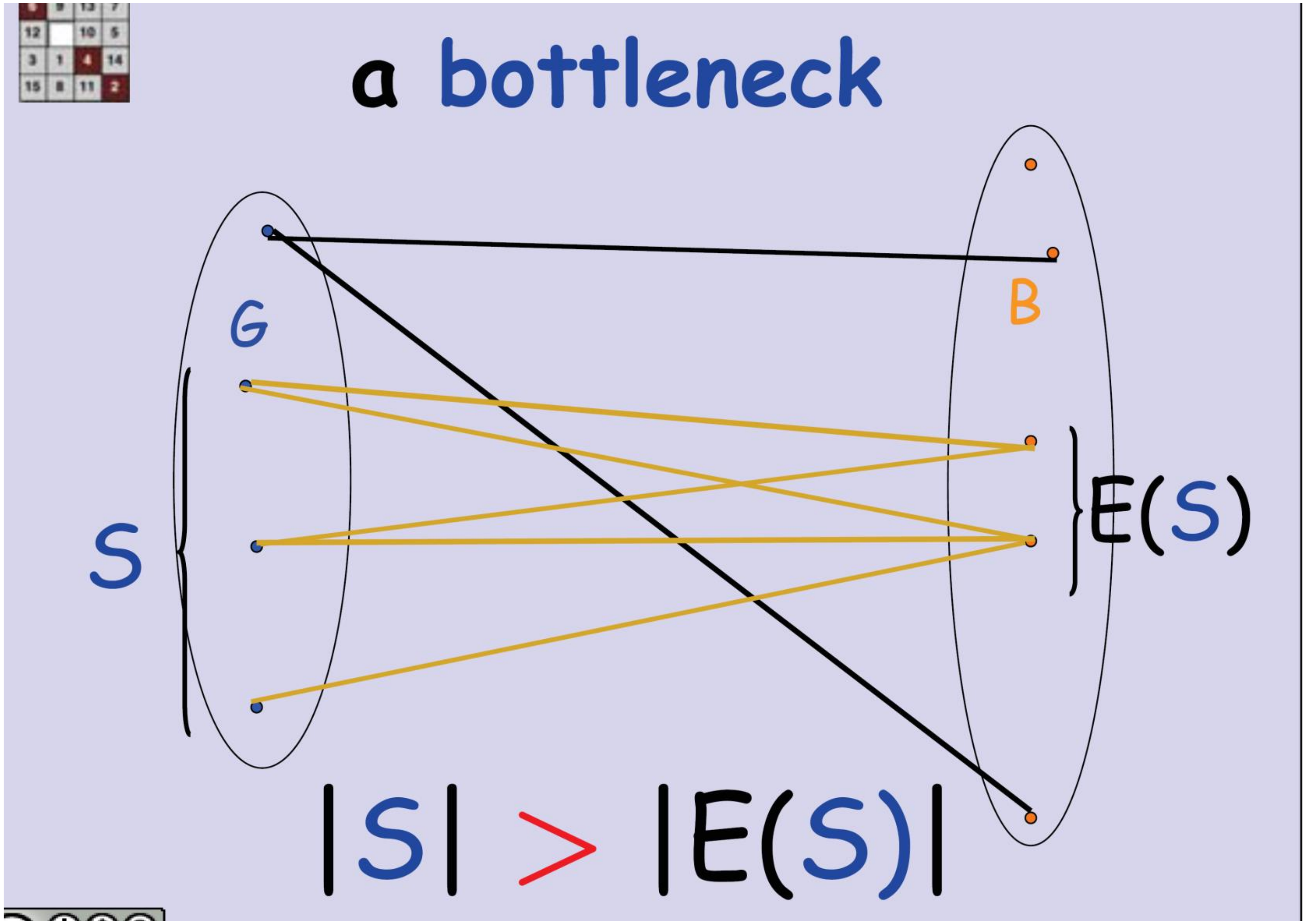
3 girls like only 2 boys

No match is possible!



$$|S| = 3 > 2 = |E(S)|$$

a bottleneck



$|S| > |E(S)|$

Bottleneck Lemma

Bottleneck: a set S of girls without enough boys.

E(S) ::= boys adjacent to at least one girl in S.

$$|S| > |E(S)|$$

Hall's Theorem

Conversely, if there are no bottlenecks, then there is a match.

If there is a bottleneck, then no match is possible obviously

Next Lecture

- Asymptotically Tight Bounds
- Big-O Notation
- Big Theta and Omega
- A Survey of runtimes