GB13604 - Maths for Computer Science

Lecture 5 – Graphs Part II

Claus Aranha
caranha@cs.tsukuba.ac.jp

College of Information Science

2018-11-07

Last updated November 7, 2018

This course is based on Mathematics for Computer Science, Spring 2015, by Albert Meyer and Adam Chlipala, Massachusetts Institute of Technology OpenCourseWare.

Exercise Discussion, Weeks 3 and 4

Week 4 and 5 summary

Graphs

Lecture I: Chapter 9

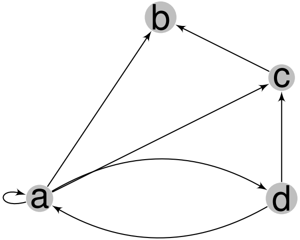
- Walks and Paths
- Scheduling and Partial Orders
- Equivalence Relations

Lecture II: Chapter 11

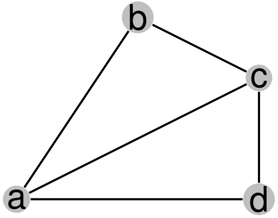
- Isomorphism
- Coloring and Connectivity
- Spanning Trees
- Matching

Simple Graphs and Directed Graphs

Directed Graph:



Simple Graph:



- No double edges allowed;
- No self-loop allowed;

Simple Graphs: Formal definitions

A Simple Graph G consists of:

- A non-empty set V of vertices;
- A set E of edges so that:
    - Each edge has two endpoints in V

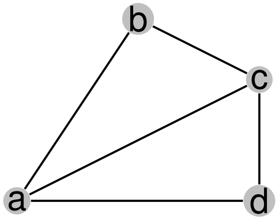
      (not an **start** and an **end**)
    - The order of the vertices do not matter

      $e_1 = \{v_1, v_2\} = \{v_2, v_1\}$
    - Two vertices with an edge between them are adjacent
    - An edge that connects two vertices is incident to them.

      (e_1 is incident to v_1 and v_2)

The Degree of a Vertex

The degree of a vertex is the number of incident edges.



deg(a) = 3 deg(d) = 2

Quiz: Is there a graph with degrees:

- 2, 2, 1?
- 3, 2, 2, 1?

Degree Properties

The Handshaking Lemma:

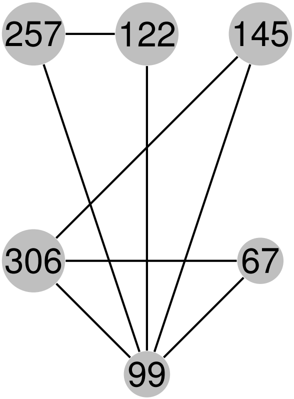
The sum of degrees must be 2x the number of edges

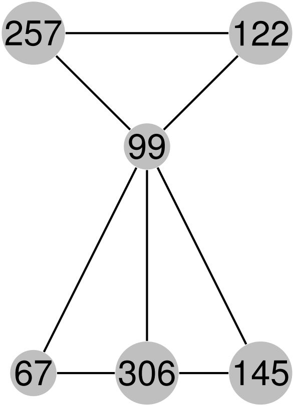
$$2|E| = \sum_{v \in V} \deg(v) \tag{1}$$

Proof: Each edge contribute 2 to LHS of (1)

So "2 + 2 + 1 = odd" is impossible!

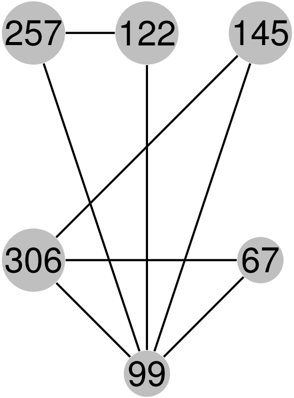
Isomorphism: The Graph Abstraction

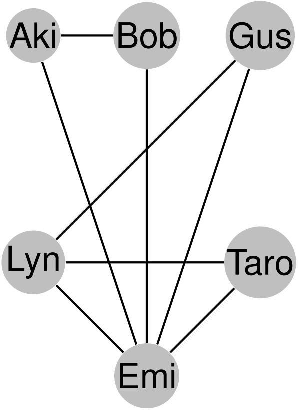




Same Graph, Different Layouts

Isomorphism: The Graph Abstraction





Same Graph, Different Labels

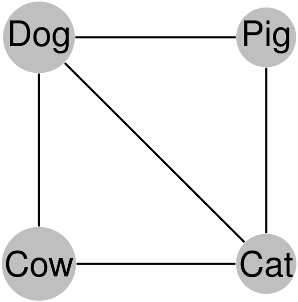
Isomorphic Graphs

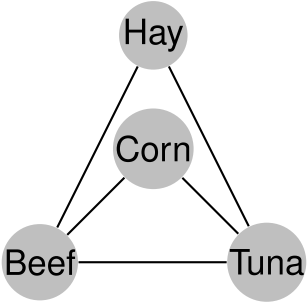
Isomorphic Graphs

- To determine isomorphism, all that matters is connections;
- Graphs with the same connections are isomorphic
- Two graphs are isomorphic if there is a Edge Preserving Matching of their vertices.

  ...In other words, a **bijection** between the vertices.

Are these Graphs Isomorphic?





f(dog) = Beef; f(cow) = Hay
f(cat) = Tuna; f(pig) = Corn

Is this a Bijection? Are the Edges preserved?

Formal Definition of Graph Isomorphism

G_1 isomorphic to G_2 means that $\exists$ Edge Preserving Vertex Matching:

$$\exists f : V_1 \rightarrow V_2, (u, v) \in E_1 \iff (f(u), f(v)) \in E_2 \tag{2}$$

Two graphs are nonisomorphic if:

- Not the same number of vertices;
- Not the same number of edges;
- Not the same degree distribution;
- Differenes in Paths, Distances, etc...

Finding Isomorphism

- Small Graphs: Check properties by hand;
- Large Graphs: Search for a **matching** $f : V_1 \rightarrow V_2$ that Preserve Isomorphic Properties:
    - Check vertices with same Degree. (Degree 4 must match with degree 4)
    - Check degrees of adjacent vertices. (Degree 4 adjacent to degree 3 must be matched with degree 4 adjacent with degree 3)

Finding Isomorphism

Even then, finding an isomorphism is a very expensive problem. In theory, we cannot guarantee that any algorithm to detect isomorphism is better than checking each bijection.

Graph Coloring

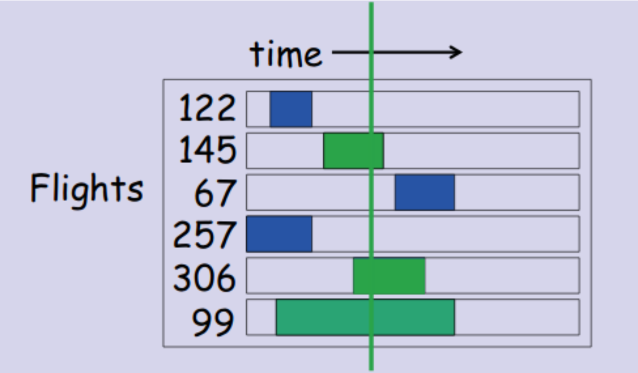
Planes and Gates

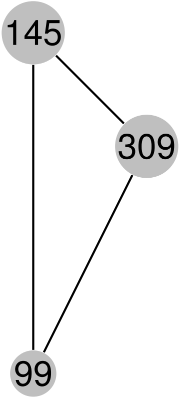
Graph Coloring is closely related to Scheduling Problems

Example:

- Every flight requires a gate for embark/disembark
- Sometimes flight times overlap.
- How many gates do we need?

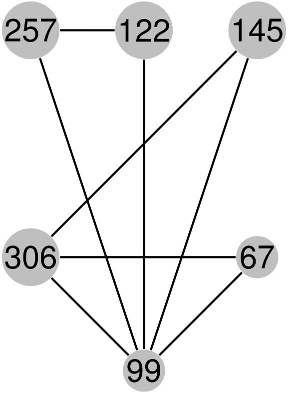
Gate Usage Graph





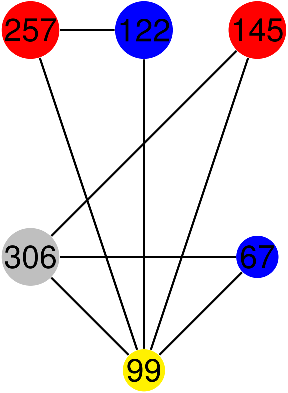
We define a Gate Usage Graph where two flights are neighbors if they are on the ground at the same time.

Full Conflict Graph and the Coloring Problem



- Color vertices so that adjacent vertices don't have the same color.
- If Edges = conflict, then min # colors = min # of gates.

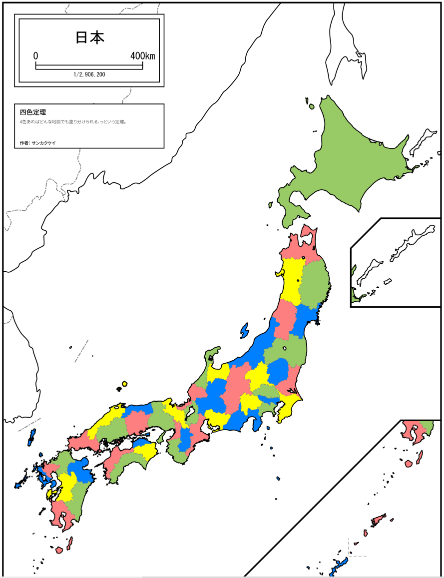
Full Conflict Graph and the Coloring Problem



- We select colors for each vertex so that no adjacent vertex has the same color.
- Each color = One new Gate
- Final gate assignment:
  - Blue Gate: Flight 122 and 67
  - Red Gate: Flight 145 and 257
  - Yellow Gate: Flight 99
  - Gray Gate: Flight 306
- Can you find a better coloring using only 3 gates?

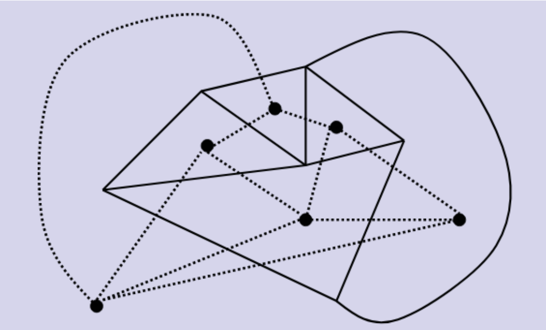
Conflict Allocation Problems

- Allocate classrooms for courses, when courses can be at the same time.
- Allocate cages for animals, when some animals can not be at the same cage.
- Different Frequencies for radio stations, when the radio stations interfere with each other
- Map Coloring!



Vertex Coloring and Face Coloring

Maps to Graphs



Theorem: Maps can always be colored with 4 colors

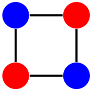
- 1970: “Proof” with computers (checks 1000’s of maps)
- 1990: Better Mathematical Proof. (still need program)

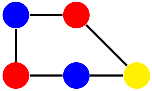
Chromatic Number

The Chromatic Number $\chi(G)$ is the minimum number of colors neecessary to color G.

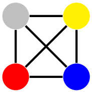
Examples:

- Cycle Graphs: $\chi(C_{\text{even}}) = 2$, $\chi(C_{\text{odd}}) = 3$

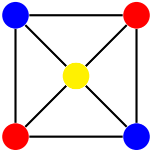




- Complete Graph with n vertices: $\chi(K_n) = n$



- Wheel Graph: $\chi(W_{\text{odd}}) = 4, \chi(W_{\text{even}} = 3)$

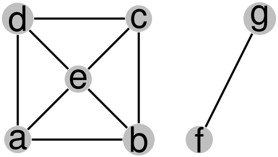


Bounding Chromatic Numbers

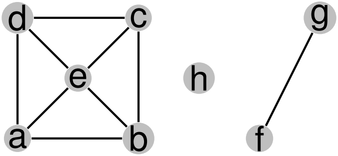
- All degrees $\leq k \implies \chi(G) \leq k + 1$
  (Proof by Greedy coloring algorithm).
- Is the graph 2-colorable?
  (**easy** to check: e.g.: BFS)
- Is the graph 3-colorable?
  (**hard** to check)
- Is $\chi(G) = k$?
  (in theory, not harder than 3 color, harder in practice).

Connectivity Definition

- Two vertices are connected **iff** there is a path between the two.
- Every vertice is connected to itself.
- A whole graph is connected if every vertice is connected to each other.



Connected Components

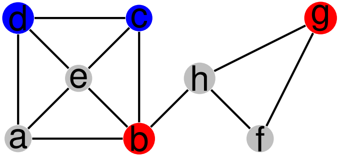


- Every Graph is composed of connected subgraphs called connected components
- connected component of v $::=$ {w | w connected to v}.
- connected component of $v = E^*(v)$ (walk relation of v)

- A graph is connected **iff** it has exactly 1 connected component.

Edge connectivity

- vertices v, w are k-edge connected if they remain connected whenever fewer than k edges are deleted.



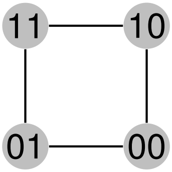
- blue vertices are 3-edge connected, red vertices are 1-edge connected;
- A **Graph** is k-edge connected if all pairs of vertices are at least k-edge connected.

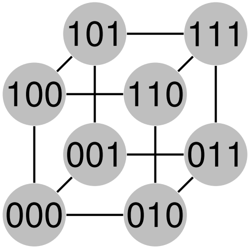
Edge Connectivity

- **Edge Connectivity** represents the degree of **fault tolerance** in a graph.
- **Example:** In a communication network, how many channels can fail before communication is disrupted?

- Related Concept: k-vertice connectivity
  - k-vertice connected graph $\implies$ k-edge connected;
  - BUT! k-edge connected $\not\Longrightarrow$ k-vertice connected.
- The complete graph K_n is n-1 connected.

Connectivity and Hypercubes





- Consider the n-dimensional hypercube H_n
- $V(H_n) ::= \{0, 1\}^n$
- $E(H_n) ::= \{(u, v)$ **iff** u and v differ in 1 bit $\}$
- H_n is n vertex connected. (H_n has n^2 vertices)

Trees

Tree Definitions

- Trees are connected Graphs with no cycles.
- Has 1-edge connectivity, 1-vertex connectivity.
- Chromatic Number = 2

- Trees come up all the time:

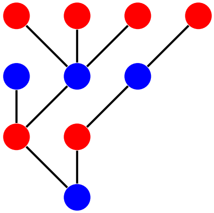
- Family Trees;
- Search Trees;
- Game Trees;
- Parse Trees;
- Spanning Trees;
- Rooted Trees;
- Ordered Trees;
- Binary Trees;
- etc...

A few more definitions

- Cut Edge: An edge is a cut edge if removing it makes two edges disconnected.
- **Lemma:** An edge is not a cut edge if it is on a cycle.
- A tree is a connected graph where every edge is a cut edge
- This implies that a tree is a connected graph which is **Edge Minimal**
    - A tree has the minimum number of edges necessary to connect a set of vertices.

Tree Coloring

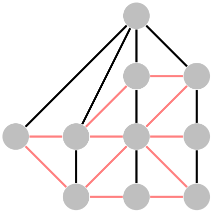
- A tree is a graph with a unique path between every pair of vertices.
- As a consequence, $\chi(\text{tree}) = 2$
- **Constructive Demonstration**



- Pick any node in the tree to be the **root**, color it "blue".
- Color nodes "odd" length from the root as "red"
- Color nodes "even" length from the root as "blue"
- This is the algorithm for 2-coloring on general graphs

Spanning Trees

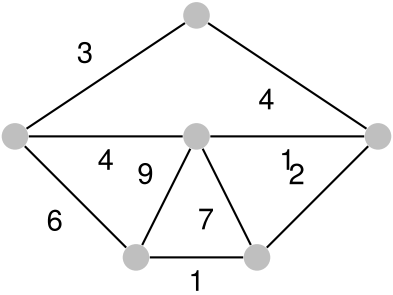
- A Spanning Subgraph of G is a subgraph of G that has all vertices of G (and some of the edges).
- A Spanning Tree of G is a spanning graph of G that is also a tree.



- One graph can have multiple spanning trees.
- Every connected graph has a spanning tree.

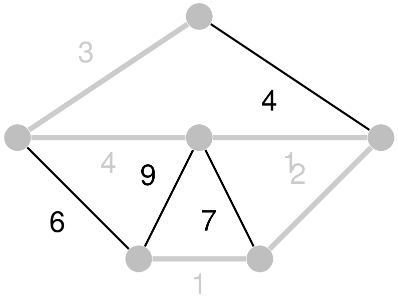
Weighted Spanning Trees

The Spanning Tree problem becomes more interesting when we consider weighted edges.



What is the minimal cost structure that allows me to connect everything?

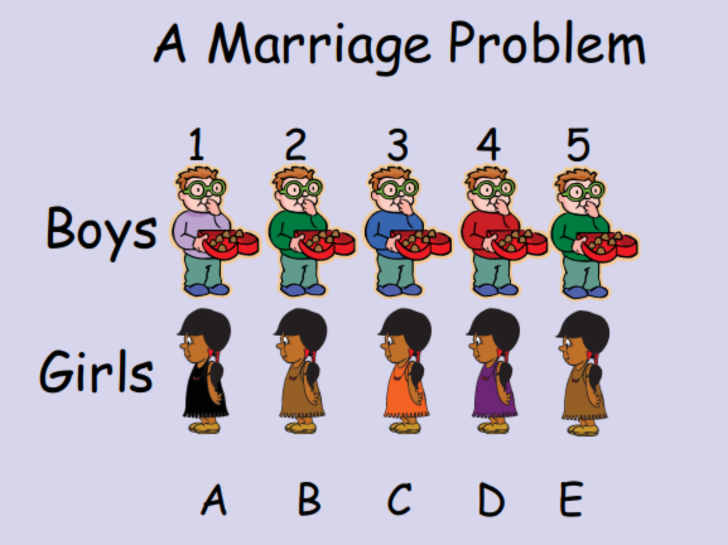
Minimum Spanning Tree Algorithm



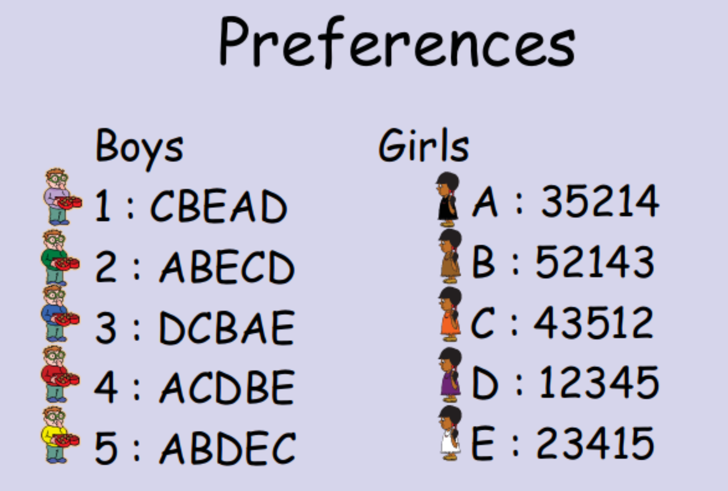
1. Start with one arbitrary vertex and add it to the MST.
2. From all edges connected with the MST, select one with minimum weight;
3. Add the edge, and vertex, to the MST;
4. Return to (2)

Stable Matching Problem

The Stable Marriage Problem

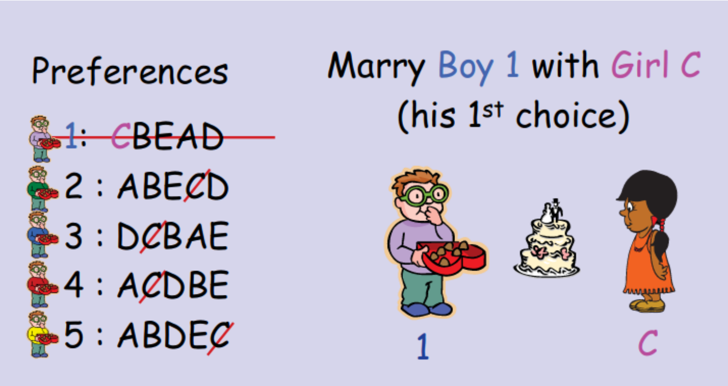


Each boy and girl has a preference list



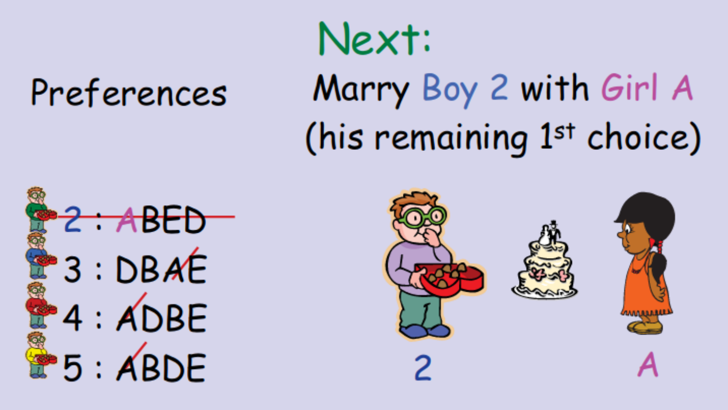
"Boy-greedy" Algorithm

Let's marry each boy to their favorite girl:



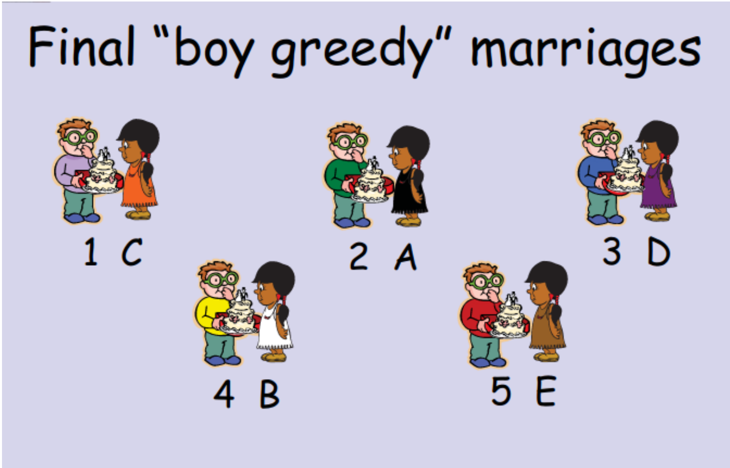
"Boy-greedy" Algorithm

Let's marry each boy to their favorite girl:

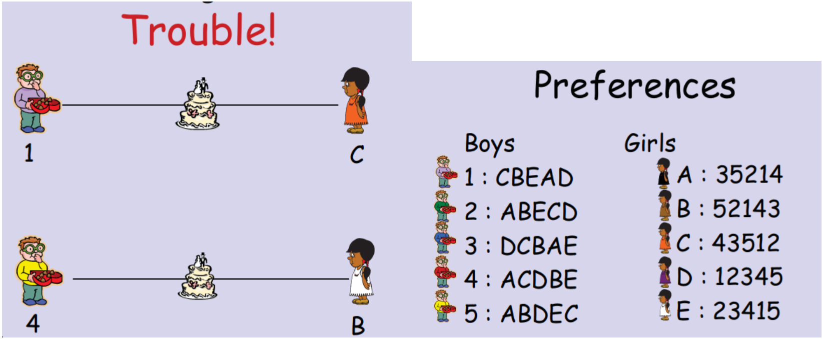


“Boy-greedy” Algorithm

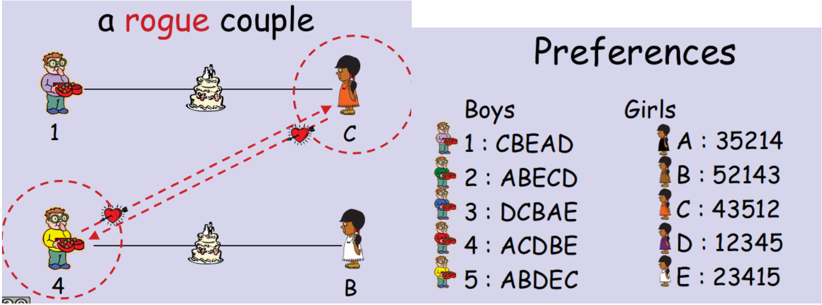
Final pairing:



Trouble with the boy Greedy algorithm!

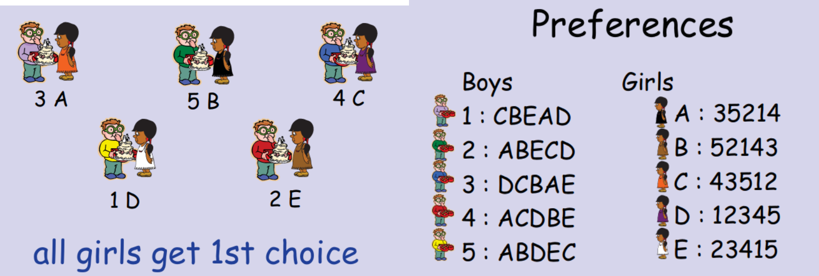


Trouble with the boy Greedy algorithm!



QUIZ: Find a better pairing!

One stable matching (Girl Greedy)



Why is the Stable Marriage Problem Important?

- School Admissions in the US
- Matching between Hospitals and Doctors
- Akamai Server/Request Matching
- Etc...

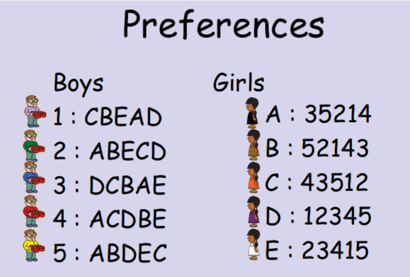
The “Mating Ritual” Algorithm

Let us describe an algorithm to **always** find a stable matching:

Start State: No boy is proposing to any girl;

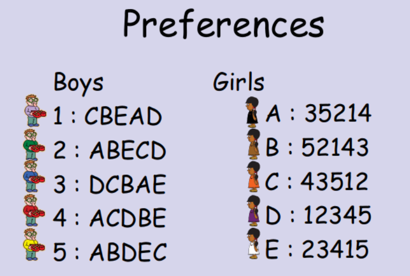
1. If all girls have ≤ 1 proposers (and it is not the first iteration), they marry and the algorithm stops.
2. Any girl that has > 1 proposers, they reject all except the favorite boy.
3. If any boy is not proposing to anyone, they propose to their favorite girl.
4. Return to (1).

Example Execution



- **iter 1**: No rejections. Proposals:
  - A: 2, 4, 5
  - B:
  - C: 1
  - D: 3
  - E:
- **iter 2**: A rejects 2 and 4. Proposals:
  - A: 5
  - B: 2
  - C: 1, 4
  - D: 3
  - E:

Example Execution



- **iter 3**: C rejects 1. Proposals:
  - A: 5
  - B: 1, 2
  - C: 4
  - D: 3
  - E:
- **iter 4**: B rejects 1. Proposals:
  - A: 5
  - B: 2
  - C: 4
  - D: 3
  - E: 1

Proof of Correctness

- The algorithm stops;
- The algorithm is correct when it stops;

Proof of Correctness: The algorithm stops

Every day, the total number of girls in the boy's lists is reduced.

- The number of girls in the boy's list is reduced when someone is rejected.
- If no one is rejected then the algorithm stops.
- **Therefore**, every iteration the list of girls reduces by at least one.

Because the total number of girls is strictly decreasing, the algorithm must stop.

Proof of Correctness: No rogue couples

- **Lemma 1**: The rank of a girl's favorite is **weakly increasing**
  Every iteration, the girl rejects a favorite **iff** she finds a better one.

- **Lemma 2**: The rank of a boy's favorite is **weakly decreasing**
  Every iteration, the boy stays with current favorite, or is rejected and goes to the next lower one.

Proof of Correctness: No rogue couples

Invariant: If G_i is not on B_j list, she has a better curent favorite.

- At the moment that G_i rejected B_i, she had a better favorite (definition of rejection rule)
- A girl's favorite never get worse (lemma 1)

Proof of Correctness: No rogue couples

Lemma: When a Boy B_i marries, he cannot form a rogue couple.

Proof by Cases:

- **Case 1:** B_i tries to form a rogue couple with someone not on his list. However, by Invariant, any girl not on his list has a better favorite, and no rogue couple is possible.
- **Case 2:** B_i tries to form a rogue couple with someone on his list. However, by Lemma 2, B_i always propose to the best girl in his list, and no rogue couple is possible.

Therefore, no rogue couple is possible.

Main Points for Today's Class

- Coloring Problems, and Chromatic Number;
- Trees, and Minimum Spanning Tree;
- Matching, and the Stable Marriage Problem;

Extra Topics

Check the class materials for “Hall’s Graphs”, for more information on matching.