ICPC Workshop 1

Graph Theory

Isaiah Iliffe and Angus Ritossa

Table of contents

Internet: Statement

There are N houses in a town, labelled from 1 to N.
M specified pairs of houses have a cable between them.
Which houses are connected by some sequence of cables to house 1?

Internet: Statement

There are N houses in a town, labelled from 1 to N.
M specified pairs of houses have a cable between them.
Which houses are connected by some sequence of cables to house 1?

Sample Input

```
9 8
1 2
1 5
2 5
5 4
4 3
2 3
4 6
7 8
```

Internet: Statement

There are N houses in a town, labelled from 1 to N.
M specified pairs of houses have a cable between them.
Which houses are connected by some sequence of cables to house 1?

Sample Input

```
9 8
1 2
1 5
2 5
5 4
4 3
2 3
4 6
7 8
```

Sample Output

```
1
2
3
4
5
6
```

Constraints

$N, M \leq 200000$

Internet: Statement

There are N houses in a town, labelled from 1 to N.
M specified pairs of houses have a cable between them.
Which houses are connected by some sequence of cables to house 1?

Sample Input

```
9 8
1 2
1 5
2 5
5 4
4 3
2 3
4 6
7 8
```

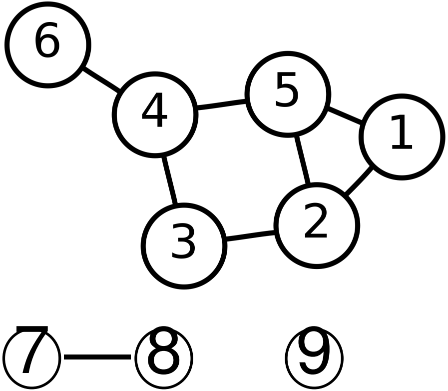
Sample Output

```
1
2
3
4
5
6
```

Constraints

$N, M \leq 200000$

Diagram



Representing graphs

- A graph is an abstraction of the town, as simply a set of objects in which some pairs of the objects are in some sense “related”

Representing graphs

- A graph is an abstraction of the town, as simply a set of objects in which some pairs of the objects are in some sense “related”
- Houses correspond to **nodes** and cables correspond to **edges**
- How can we represent a graph mathematically? Computationally?

Representing graphs

- A graph is an abstraction of the town, as simply a set of objects in which some pairs of the objects are in some sense “related”
- Houses correspond to **nodes** and cables correspond to **edges**
- How can we represent a graph mathematically? Computationally?
- Adjacency list
  - For each node, store a list (vector in C++) of adjacent nodes
  - Implementation: see code

Depth-first search

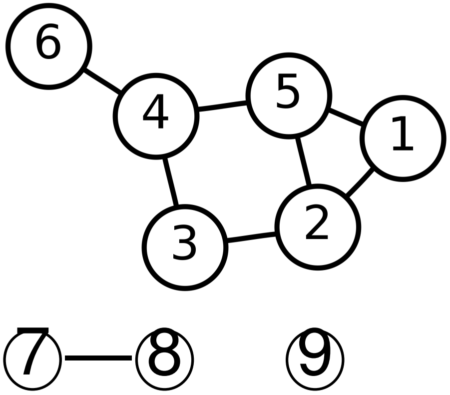
- To "process" a node, just "process" each of its neighbours
- But never "process" a node more than once
- Implementation: see code

Theorem (Fundamental Theorem of DFS)

A DFS initiated at a node u will process a node v exactly when there exists a path between u and v.

- So our problem can be solved by running a DFS from node 1, then checking which nodes have been processed. We'll come back to how exactly to code a solution up and submit it.

DFS Walkthrough



Two Colouring: Statement

You are given a graph with N nodes and M edges.

Two Colouring: Statement

You are given a graph with N nodes and M edges.
Can you colour each node either black or white, such that any two connected nodes are of different colour?

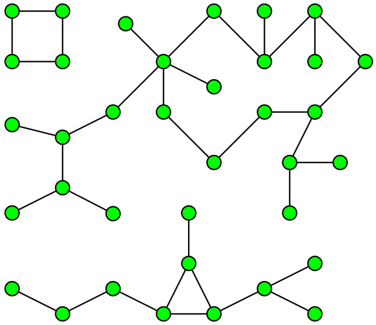
Two Colouring: Statement

You are given a graph with N nodes and M edges.
Can you colour each node either black or white, such that any two connected nodes are of different colour? (In other words, is the graph bipartite?)
If yes, output a possible allocation of colours for each node.

Two Colouring: Statement

You are given a graph with N nodes and M edges.
Can you colour each node either black or white, such that any two connected nodes are of different colour? (In other words, is the graph bipartite?)
If yes, output a possible allocation of colours for each node.

Example



Two Colouring: Solution

- Modify the DFS algorithm so each node gets a colour, and all nodes seen so far satisfy the different colour criterion

Two Colouring: Solution

- Modify the DFS algorithm so each node gets a colour, and all nodes seen so far satisfy the different colour criterion
- If there is ever a contradiction in what colour a node should be, say NO

Two Colouring: Solution

- Modify the DFS algorithm so each node gets a colour, and all nodes seen so far satisfy the different colour criterion
- If there is ever a contradiction in what colour a node should be, say NO
- Otherwise, output the final colour of each node.

Two Colouring: Solution

- Modify the DFS algorithm so each node gets a colour, and all nodes seen so far satisfy the different colour criterion
- If there is ever a contradiction in what colour a node should be, say NO
- Otherwise, output the final colour of each node.
- What if not everything is connected, though?

Two Colouring: Solution

- Modify the DFS algorithm so each node gets a colour, and all nodes seen so far satisfy the different colour criterion
- If there is ever a contradiction in what colour a node should be, say NO
- Otherwise, output the final colour of each node.
- What if not everything is connected, though?
- Implementation: see code

Cards: Statement

There are N cards.
The i-th card has the colour $a[i]$ on one side, and the colour $b[i]$ on the other side.
What is the maximum number of colours you can see at once?

Cards: Statement

There are N cards.
The i-th card has the colour $a[i]$ on one side, and the colour $b[i]$ on the other side.
What is the maximum number of colours you can see at once?

Sample Input

```
4
1 2
1 3
4 2
2 3
```

Sample Output

```
4
```

Constraints

$N, M \leq 200000$

Cards: Statement

There are N cards.
The i-th card has the colour $a[i]$ on one side, and the colour $b[i]$ on the other side.
What is the maximum number of colours you can see at once?

Sample Input

```
4
1 2
1 3
4 2
2 3
```

Sample Output

```
4
```

Constraints

$N, M \leq 200000$

Thinking time

Cards: Solution

- Reframe in terms of a graph
  - You are given a graph
  - You must choose a node at one end of each edge to be activated
  - What is the maximum number of distinct activated nodes?

Cards: Solution

- Reframe in terms of a graph
  - You are given a graph
  - You must choose a node at one end of each edge to be activated
  - What is the maximum number of distinct activated nodes?
- Observations after trying stuff on paper
  - Each component (set of nodes reached by one DFS) can be dealt with independently

Cards: Solution

- Reframe in terms of a graph
  - You are given a graph
  - You must choose a node at one end of each edge to be activated
  - What is the maximum number of distinct activated nodes?
- Observations after trying stuff on paper
  - Each component (set of nodes reached by one DFS) can be dealt with independently
  - If there are no cycles in a component, we can activate every node but one. Why?

Cards: Solution

- Reframe in terms of a graph
  - You are given a graph
  - You must choose a node at one end of each edge to be activated
  - What is the maximum number of distinct activated nodes?
- Observations after trying stuff on paper
  - Each component (set of nodes reached by one DFS) can be dealt with independently
  - If there are no cycles in a component, we can activate every node but one. Why?
  - If there is a cycle, it allows us to activate everything. Why?

Cards: Solution

- Reframe in terms of a graph
  - You are given a graph
  - You must choose a node at one end of each edge to be activated
  - What is the maximum number of distinct activated nodes?
- Observations after trying stuff on paper
  - Each component (set of nodes reached by one DFS) can be dealt with independently
  - If there are no cycles in a component, we can activate every node but one. Why?
  - If there is a cycle, it allows us to activate everything. Why?
- So our problem can be solved by finding each component (using DFS), and for each component, determining whether or not there is a cycle (within our DFS, or otherwise).

Cards: Solution

- Reframe in terms of a graph
  - You are given a graph
  - You must choose a node at one end of each edge to be activated
  - What is the maximum number of distinct activated nodes?
- Observations after trying stuff on paper
  - Each component (set of nodes reached by one DFS) can be dealt with independently
  - If there are no cycles in a component, we can activate every node but one. Why?
  - If there is a cycle, it allows us to activate everything. Why?
- So our problem can be solved by finding each component (using DFS), and for each component, determining whether or not there is a cycle (within our DFS, or otherwise).
- Implementation

Lab

- Join the vjudge group: https://vjudge.net/group/unswicpc
- Go to the contest for this workshop
- If you need help, or don't know what to do, message me or Angus
- **A: A+B** — solve this first if you haven't used vjudge before
- **B: Internet** — implement the first problem from today
- **C: Cards** — implement the third problem from today
- **D: Paradox** — second problem from today (two colouring), but a bit harder
- **E: Maze** and **F: Graph** — harder problems
- Angus will go over Graph at 1:40