Graphs

Lecture 14

Distance

Prove via contradiction

Hence ∃ walk → ∃ path

In many applications, the edges on the graph will have "lengths". For us, typically all edges are of length 1.

- Shortest walk between nodes u and v is always a path
- Shortest path is of great interest in many applications
  - e.g., nodes correspond to locations on a map and edges are roads, optic fibers etc.
  - Also, graph can be used to model probabilistic processes, with shortest path indicating the most likely outcome
- Length of the shortest path between u and v is called the distance between u and v (∞ if no path)

$$\min_{W:\ u\text{-}v\ \text{walk}} \text{Length}(W)$$

- Diameter is the largest distance in a graph (can be ∞)

$$\max_{u,v} \text{Distance}(u,v) = \max_{u,v} \min_{W:\ u\text{-}v\ \text{walk}} \text{Length}(W)$$

Graphs in action

Many Applications

- Graphs used to design networks of processors in a super-computer
- Used to keep data in an easy-to-search/manipulate fashion
  - Data structures: mainly, (balanced) trees of various kinds
- Want low degree (hardware cost; look at a few (neighbouring) pieces of data at a time), but good "connectivity" -- i.e., (possibly many) short paths between any two nodes (to route data; to reach the required piece of data quickly, by taking a path over the graph)
- Very efficient algorithms known for relevant graph problems
  - e.g., breadth/depth-first search, shortest path algorithm...
- But many other graph problems are known to be "NP-hard"
  - e.g., Traveling Salesperson Problem (TSP): visit all cities, by traveling the least distance

Graphs in action

Shortest Paths in Action

- Obvious example: nodes correspond to locations on a map and edges are roads, optic fibers etc.
  - Weighted edges: each edge has its own "length" (instead of 1)
- But also over more abstract graphs
  - e.g., Graph-based models in AI/machine-learning for modeling probabilistic systems
    - e.g., a graph, modeling speech production: nodes correspond to various "states" the vocal chords/lips etc. could be in while producing a given a sound sequence. Edges show transitions (next state) over time. Shortest path in this graph gives the "most likely" word that was spoken.

Question

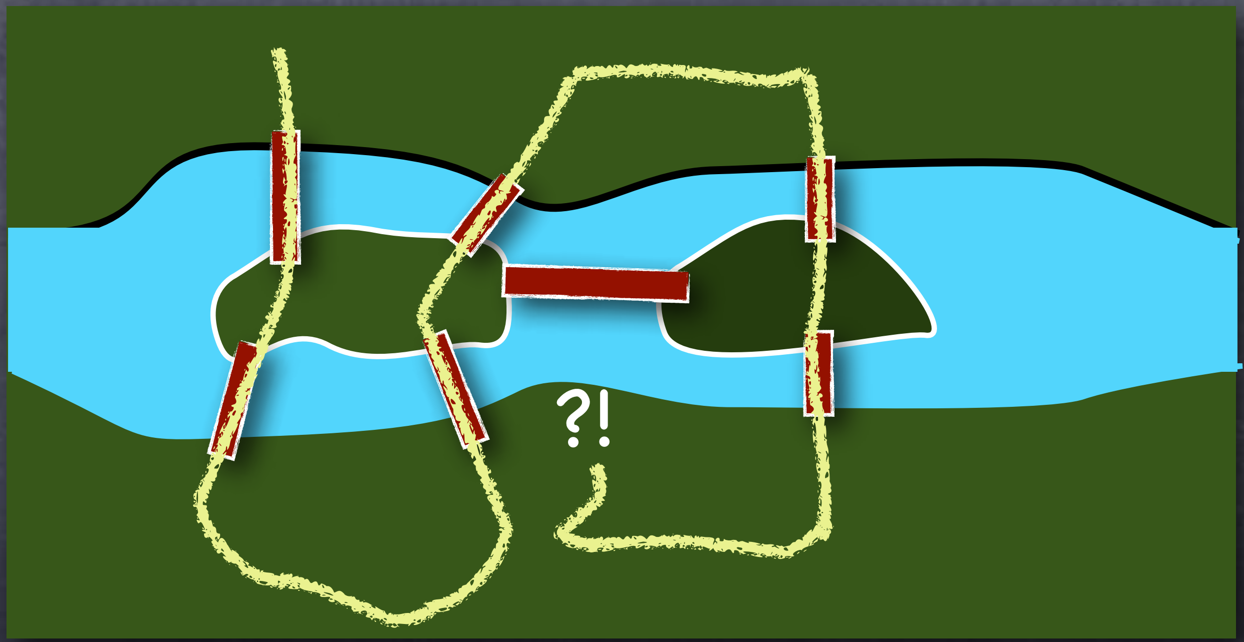
02:00

- What is the diameter of C_n
  A. n
  B. $\lceil n/2 \rceil$
  C. $\lfloor n/2 \rfloor$
  D. $n-1$
  E. 1

e.g., C_3 has diameter 1

Bridges of Königsberg

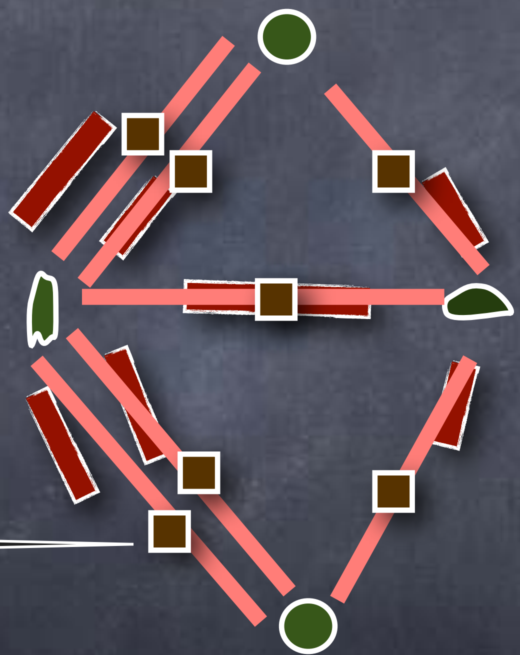
- Cross each bridge exactly once



- Impossible! But how do we know for sure?

Bridges of Königsberg

- Cross each bridge exactly once



Add a node for each bridge too, if we want it to be a simple graph

- Impossible! But how do we know for sure?
- If there is a walk that takes each edge exactly once, then only the end nodes of the walk can have odd degree (why?)

Eulerian Trail & Circuit

- Eulerian trail: a walk visiting every edge exactly once
  - Eulerian trail exists → at most 2 odd degree nodes
- Eulerian circuit: a closed walk visiting every edge exactly once
  - Eulerian circuit exists → no odd degree nodes
- If no odd degree nodes and all edges in one connected component, then must have an Eulerian circuit!
  - Informal argument: find and remove one cycle at a time (take a walk until repeated node), so that no odd degree node ever. Finally stitch them all together into one Eulerian circuit (possible since connected).

Question

02:00

- Suppose G_1, G_2, G_3 are simple graphs with the following degree sequences: (2,2,2), (2,2,2,2,2,2), (0,0,2,2,2). Then which ones **must** have Eulerian circuits?

Only possibility is K_3

Two possibilities: C_6 or the disjoint union of two K_3's.

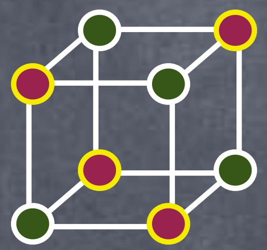
All edges within K_3

A. G_1 alone
B. G_2 alone
C. G_1 and G_2
D. G_1 and G_3
E. G_1, G_2 and G_3

Hamiltonian Cycle

- Eulerian circuit: a closed walk visiting every edge exactly once
  - Eulerian circuit exists $\longleftrightarrow$ all edges in the same connected component and no odd degree nodes
  - Can efficiently find one if they exist
- Hamiltonian Cycle: a cycle that contains all the nodes in the graph
  - No efficient algorithm known to check if a graph has a Hamiltonian cycle!
    - An "NP-hard" problem. Widely believed that no efficient algorithm exists!
      - (cf. Graph Isomorphism: It is believed to be hard, but also believed to be not NP-hard)

Graph Colouring



- Recall bi-partite graphs
  - We can "colour" the nodes using 2 colours (which part they are in) so that no edge between nodes of the same colour
- More generally, a colouring (using k colours) is valid if there is no edge between nodes of the same colour
  - k-colouring: a function $c: V \to \{1,..,k\}$ s.t. $\forall x,y \in V$ $\{x,y\} \in E \to c(x) \neq c(y)$
  - The least number of colours possible in a valid colouring of G is called the **Chromatic number** of G, $\chi(G)$
    - G has a k-colouring $\leftrightarrow$ $\chi(G) \leq k$

Upper-bounding $\chi(G)$

Graph Colouring

- Suppose H is a subgraph of G. Then:
  - G has a k-colouring $\longrightarrow$ H has a k-colouring
    - i.e., $\chi(G) \geq \chi(H)$

Lower-bounding $\chi(G)$

- e.g., G has K_n as a subgraph $\longrightarrow$ $\chi(G) > n-1$ (i.e., $\chi(G) \geq n$)
- e.g., G has C_n for odd n as a subgraph $\longrightarrow$ $\chi(G) > 2$
- Summary: One way to show $k_{lower} \leq \chi(G) \leq k_{upper}$

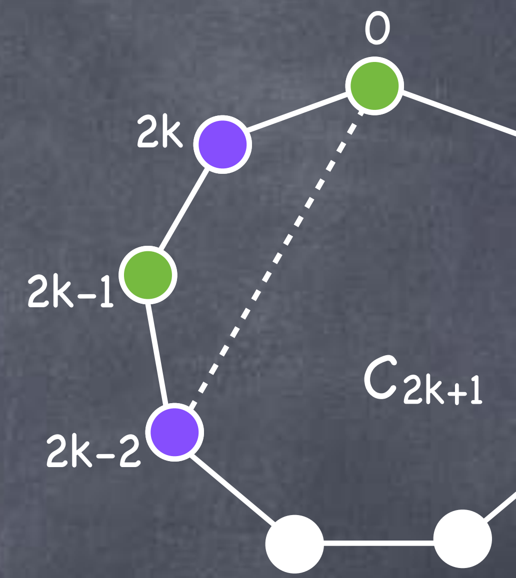
  Show a colouring $c: V \longrightarrow \{1, \ldots, k_{upper}\}$

  And show a subgraph H with $k_{lower} \leq \chi(H)$

Graph Colouring

- The origins: map-making
  - "Graph": one node for each country; an edge between countries which share a border
  - Neighbouring countries shouldn't have the same colour. Use as few colours as possible.
- Efficient algorithms known for colouring many special kinds of graphs with as few colours as possible
  - But computing chromatic number in general is believed to be "hard" (it is NP-hard)

Bi-partite Graph



- Claim: for all integers $n \geq 1$, C_{2n+1} is <u>not</u> bi-partite
- Base case: $n=1$. C_3 has chromatic number 3. ✔
- Induction step: For all integers $k \geq 2$:
  Induction hypothesis: C_{2k-1} is not bi-partite (corresponds to $n=k-1$)
  To prove: C_{2k+1} is not bi-partite (corresponds to $n=k$)
  - Will prove contrapositive: C_{2k+1} bi-partite $\rightarrow$ C_{2k-1} bi-partite
  - Suppose valid 2-colouring $c:\{0,..,2k\} \rightarrow \{1,2\}$ of C_{2k+1}.
  - Then, $c(0) \neq c(2k) \neq c(2k-1) \neq c(2k-2)$. i.e., $c(0)=c(2k-1) \neq c(2k-2)$.
  - Only edge in C_{2k-1} not in C_{2k+1} is $\{0,2k-2\}$.
  - So c respects all edges of C_{2k-1}.
  - So $c':\{0,..,2k-2\} \rightarrow \{1,2\}$ with $c'(u)=c(u)$ is a valid colouring of C_{2k-1}.

Complete Graph

- $\chi(G)=|V| \leftrightarrow$ G is isomorphic to $K_{|V|}$
  - $\leftarrow$: $\chi(K_n) = n$ (else, by pigeonhole principle, two nodes with same colour!), and isomorphism preserves χ (exercise!)
  - $\rightarrow$: We will prove the contrapositive: i.e., that if G not isomorphic to $K_{|V|}$, then $\chi(G) \neq |V|$.
    - Suppose G not isomorphic to $K_{|V|}$. So G should have at least two distinct nodes u, v s.t. $\{u,v\} \notin E$. Consider the colouring which assigns colours $\{1,\ldots, |V|-2\}$ to the nodes in $V-\{u,v\}$ and the colour $|V|-1$ to both u and v. This is a valid colouring (because $f(x)=f(y) \rightarrow \{x,y\} \notin E$). So $\chi(G) \leq |V|-1$

Proof describes a recursive algorithm for colouring with $\Delta(G)+1$ colours

olouring and Degree

- Claim: $\forall n \in \mathbb{Z}^+$ for every graph $G=(V,E)$ s.t. $|V|=n$, $\chi(G) \le \Delta(G)+1$
- Base case: $n=1$.
  There is only one graph with $|V|=1$, for which $\Delta(G)=0$, $\chi(G)=1$
- Induction step: For all integers $k \ge 1$:
  Induction hypothesis: for all $G=(V,E)$ with $|V|=k$, $\chi(G) \le \Delta(G)+1$
  To prove: for all graphs $G=(V,E)$ with $|V|=k+1$, $\chi(G) \le \Delta(G)+1$.
  - Let $G=(V,E)$ be an arbitrary graph with $|V|=k+1$.

    Important!
  - Let $G'=(V',E')$ be obtained from G by removing some $v \in V$ (i.e., $V'=V-\{v\}$) and all edges incident on it.
  - $|V'|=k$. So $\chi(G') \le \Delta(G')+1 \le \Delta(G)+1$. Colour G' with $\Delta(G)+1$ colours.
  - $\deg(v) \le \Delta(G)$. So colour v with a colour in $\{1,..,\Delta(G)+1\}$ that does not appear in its neighbourhood. Valid colouring.
    So $\chi(G) \le \Delta(G)+1$.

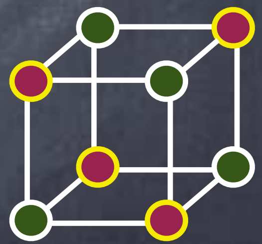
Graphs in action

Graph Colouring in Action

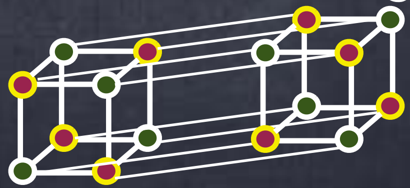
- Many problems can be modeled as a graph colouring problem
- Resource scheduling: allocate "resources" (e.g. time slots, radio frequencies) to "demands" (exams, radio stations) so that there are no "conflicts." Use as few resources as possible.
  - Create a "conflict graph": Demands are the nodes; connect them by an edge if they have a conflict (same student, inhabited area with signal overlap)
  - Colour the graph with as few colours as possible
  - Allocate one resource per colour. Then, no two demands satisfied by the same resource have a conflict

Another Example

- The hypercube graph Q_n
  - Nodes: all n-bit strings. e.g., {000,001,010,011,100,101,110,111}
  - Edges: x and y connected iff they differ in exactly one position
    - i.e., x & y neighbours if toggling a single bit changes x to y
  - e.g. Q_3 can be drawn like a "cube"



- 2^n nodes, but "diameter" (longest shortest path) is only n
- Q_n is an n-regular bi-partite graph
  - The two parts: nodes labeled with strings which have <u>even parity</u> (even# 1s) and those labeled with strings of <u>odd parity</u> (odd# 1s)
- Q_{n-1} is a subgraph of Q_n



Question

- In Q_5, what is the distance (length of a shortest path) between the nodes labeled 00100 and 10001?

  A. 6
  B. 5
  C. 4
  D. 3
  E. 2

“weight of x⊕y”

- Many shortest paths (How many?)