Lecture 14: Graphs and ~~Distances~~

COSC 311 *Algorithms*, Fall 2022

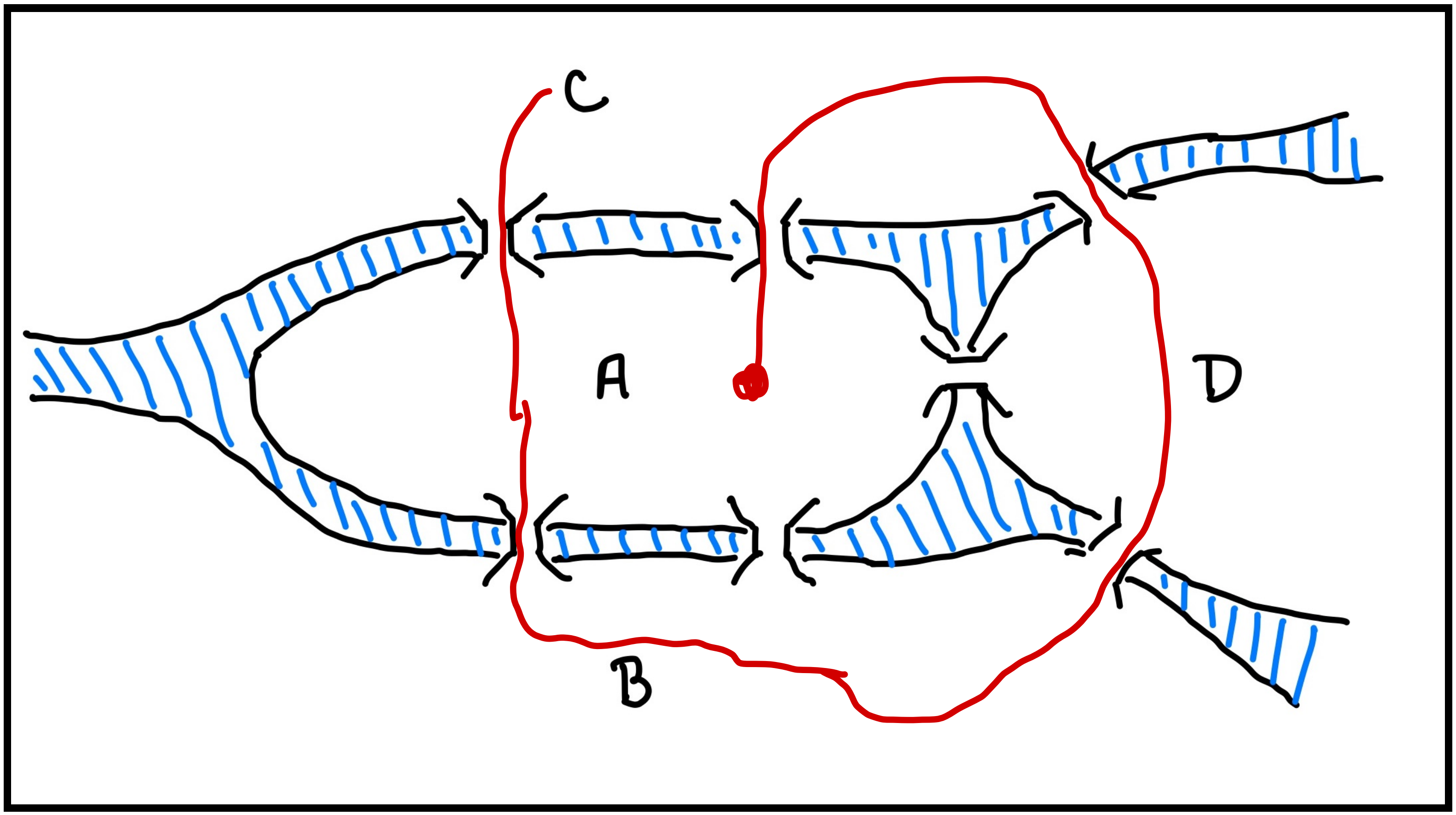
Announcements

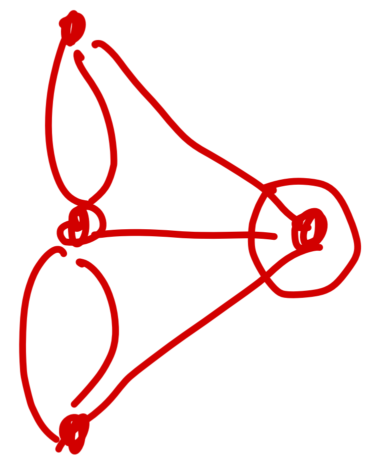
1. Midterm on Friday
   - Makeup date following week ← Thursday
   - Accommodations
2. Study guide posted
   - Solutions to come (tomorrow?)

Overview

1. Eulerian Graphs
2. ~~Graph Exploration~~

Last Time





Question. Is it possible to walk around Königsberg, cross every bridge *exactly* once, and return to where you started?

BoK as a Graph Problem

Original Question. Is it possible to walk around Königsberg, cross every bridge *exactly* once, and return to where you started?

Rephrasing as Graph Problem. Given a graph $G = (V, E)$, is there a circuit that contains every edge $e \in E$ exactly once?

- A graph with this property is called **Eulerian**.

Theorem (Euler 1736). G is Eulerian if and only if G is connected and every vertex has even degree.

- Showed " $\implies$ " direction last time

Finding Eulerian Circuits

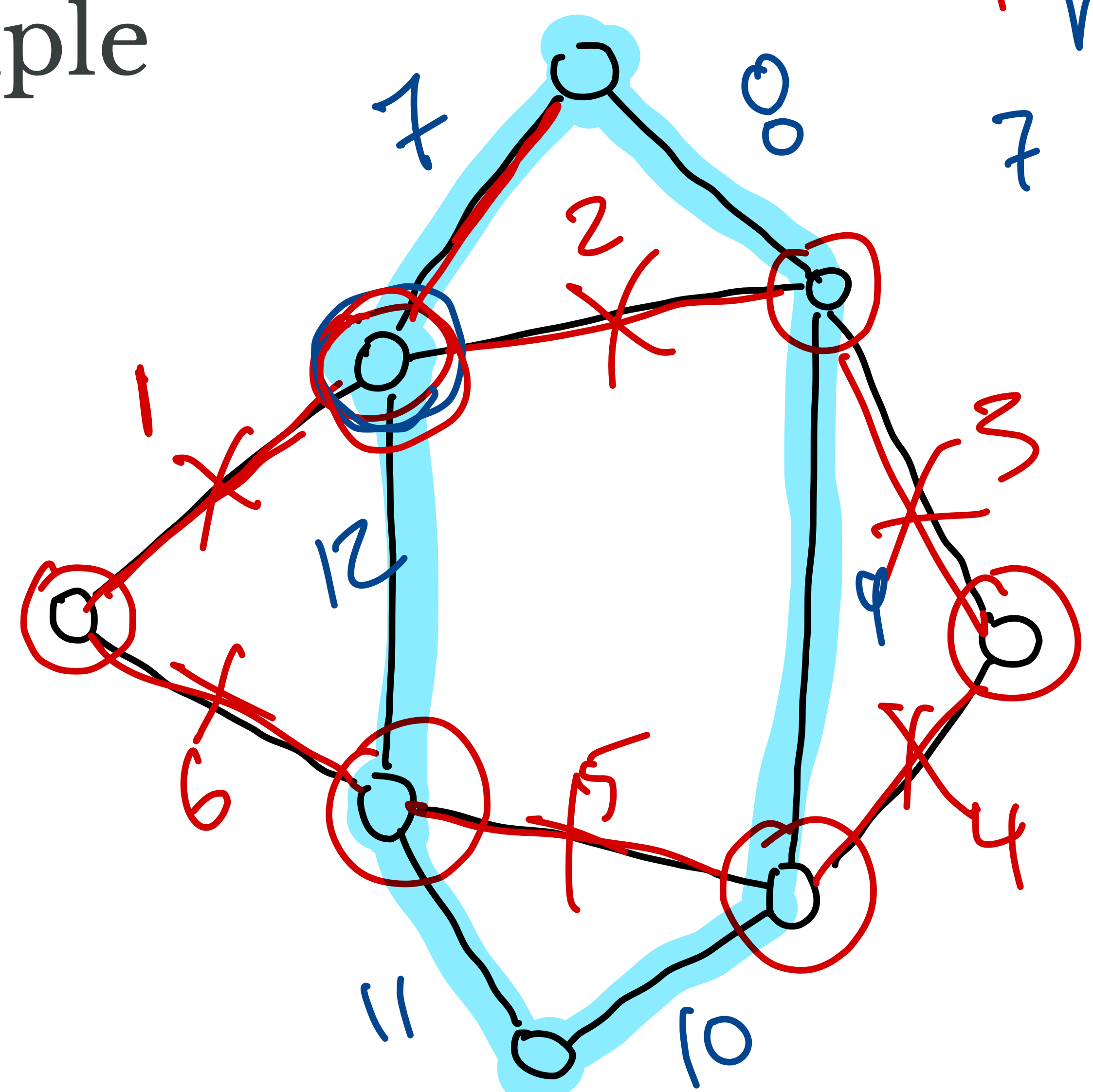
Assume:

- G is connected and all vertices have even degrees

Strategy:

- Starting from vertex v, walk in any direction
  - cross any edge from current location
  - remove edge from future consideration
  - repeat
- When stuck, reassess

Example



1 2 3 4 5 6

7 8 9 10 11 12

$\Downarrow$

1 7 8 9 10 11 12

2 3 4 5 6

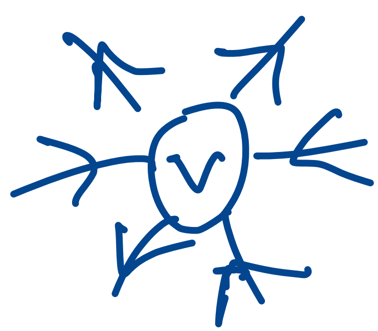
Finding a Circuit

Input:

- graph G
  - set V of vertices
  - set E of edges
- starting vertex $v \in V$
- *assume* all vertices have even degree (G is **even**)

Output:

- a circuit $P = v_0 e_1 v_1 e_2 v_2 \cdots e_k v_k$ with $v_0 = v_k = v$
- every edge e incident to v is contained in P



`FindCircuit` Subroutine

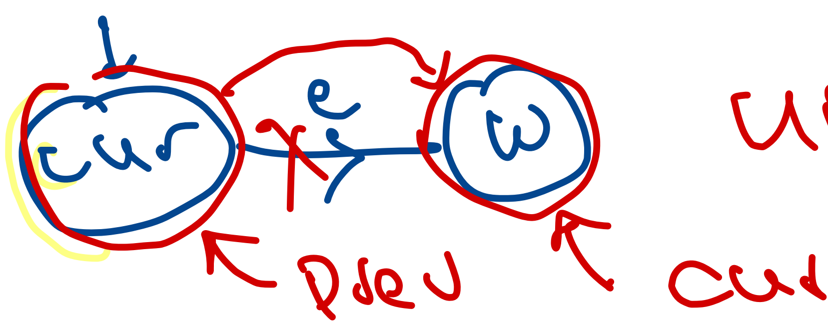
```
FindCircuit(V, E, v):
  cur <- v
  P <- v
  while deg(cur) > 0
    e <- any edge in E incident to cur
    (prev, cur) <- e
    append e, cur to P
    remove e from E
    if deg(prev) = 0 then remove prev from V
  endwhile
  remove cur from V
  return P
```

current location

Path found so far

Not stuck

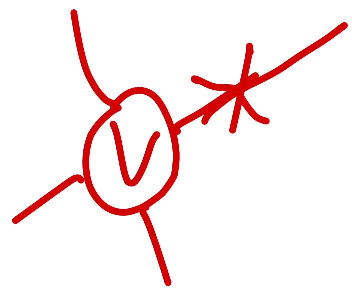
– update current node



update

Circuit Finding

Claim 1. If every vertex in $G = (V, E)$ has even degree, then `FindCircuit(V, E, v)` returns a circuit beginning and ending at v.



Every time leave v its degree is odd

$u \neq v$, when we leave u, u has even degree $\Rightarrow$ when visit u, u has odd deg $\Rightarrow \deg(u) > 1$.

Circuit Finding

Claim 1. If every vertex in $G = (V, E)$ has even degree, then `FindCircuit(V, E, v)` returns a circuit beginning and ending at v.

- *Loop invariant.* If cur $\neq v$, then cur and v have odd degress, while all other vertices have even degrees.

· v started w/ even, and we left

· cur started w/ even and we visited

· visit then leave ⇒ decrease degree by 2, so if start even, end even.

Circuit Finding

Claim 1. If every vertex in $G = (V, E)$ has even degree, then `FindCircuit(V, E, v)` returns a circuit beginning and ending at v.

- *Loop invariant.* If cur $\neq v$, then cur and v have odd degress, while all other vertices have even degrees.
- *Consequence.* If deg(cur) $= 0$, then cur $= v$.
  - $\implies$ can only get "stuck" at starting point!

Circuit Removal

C = circuit found, removed after `FindCircuit(V, E, v)`

Question. What can we say about remaining graph?

C is even (all even degrees)

$\Rightarrow$ so if G was even, then

G remains even after removing C.

Circuit Removal

C = circuit found, removed after `FindCircuit(V, E, v)`

Question. What can we say about remaining graph?

Claim 2. Remaining graph $G - C$ has all even degrees.

Why?

Circuit Removal

$C =$ circuit found, removed after `FindCircuit(V, E, v)`

Question. What can we say about remaining graph?

Claim 2. Remaining graph $G - C$ has all even degrees.

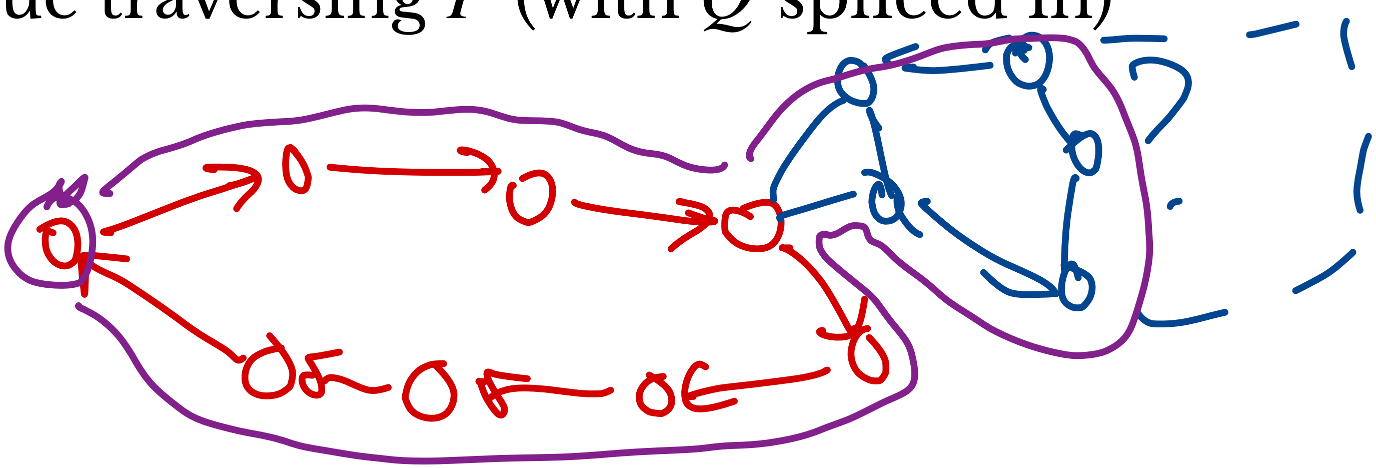
Why?

- vertices in G originally have even degree
- vertices in C have even degrees
- each vertex in G has even number of edges removed
- even - even = even

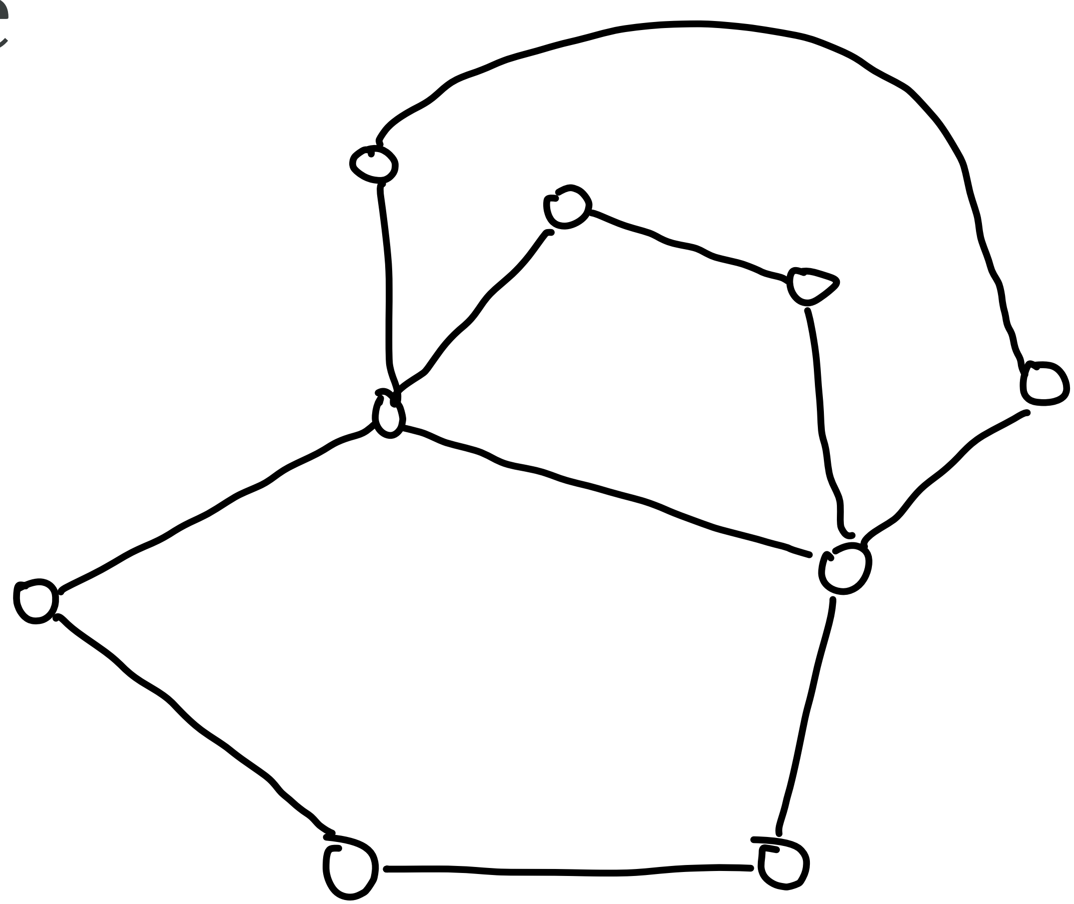
Finding Eulerian Circuits

Strategy.

1. Apply `FindCircuit` to find a circuit $P = v_0 e_1 v_1 \cdots v_k$
2. Traverse P
   - if a vertex v_i with $\deg(v_i) > 0$ is encountered,
     1. apply `FindCircuit` to v_i to get a circuit Q
     2. splice Q into P at v_i
   - continue traversing P (with Q spliced in)



Example



Eulerian Circuit Pseudocode

```
EulerCircuit(V, E, v):
  P <- FindCircuit(V, E, v)
  for each edge e = (u, w) in P do
      if deg(w) > 0 then
        Q <- EulerianCircuit(V, E, w)
        Splice(P, Q, w)
      endif
  endfor
```

Correctness

Claim. If G is even and connected, then `EulerCircuit` returns an Eulerian circuit.

Argue by induction on $\boxed{m}$ = number of edges in G.

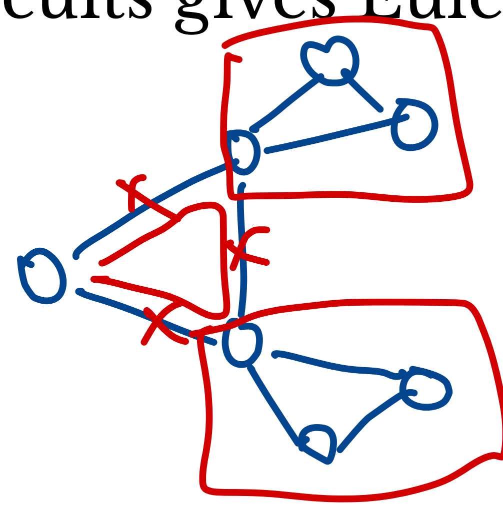
Base Case, $m = 0$. If G is connected and has no edges, then G has only one vertex, so `EulerCircuit` correctly outputs an Eulerian circuit (of length 0)

Inductive Step

Ind. Hyp.

Suppose `EulerCircuit` finds an Eulerian circuit on all connected, even graphs with fewer than m edges. Then:

1. After removing P from G, G has fewer than m edges
2. G is still even
3. G may be disconnected, but all components touch P
4. By inductive hypothesis, `EulerCircuit` finds Eulerian circuit in each component
5. Splicing together circuits gives Eulerian circuit for whole graph



Conclusion

G is Eulerian if and only if G is even and connected.

If G is Eulerian, then an Eulerian circuit can be found by *greedily* traversing the graph, and recursively finding Eulerian circuits on remaining components.