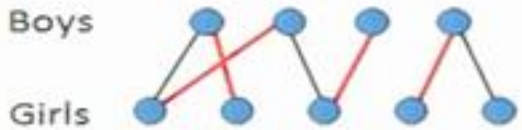
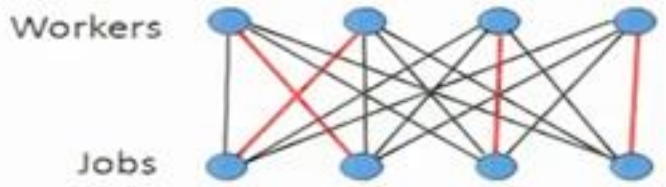
Bipartite graph, Euler circuits

Bipartite graph

- A graph G is **bipartite if** V(G) is the union of two disjoint independent sets, called partite sets of G
- The vertices V(G) can be partitioned into two disjoint sets, such that each set is independent
- Every edge connects one node of A with one node of B
- G(AUB, E)

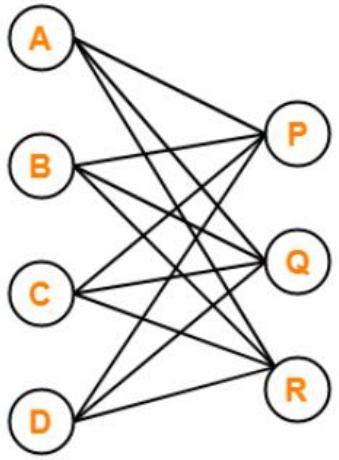
- Example:
  (i) Matching Problem, (ii) Job Assignment Problem





Bipartite graph

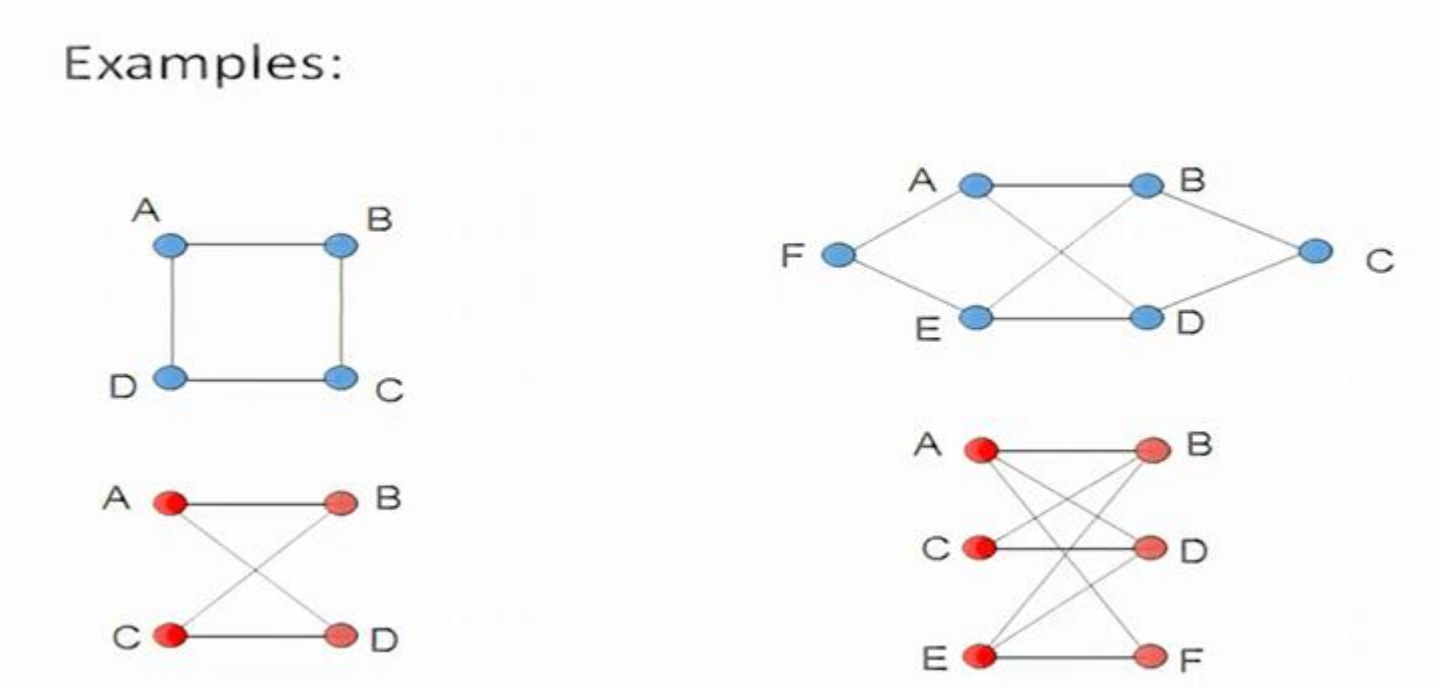
- A **complete bipartite graph** is a special kind of graph where vertices split into two disjoint independent sets A and B,
- No edges exist inside the same set A or B,
- Every vertex in the first set connects to every vertex in the second set.



$K_{m,n}$

Theorem

Theorem: A graph is bipartite if and only if it has no odd cycles



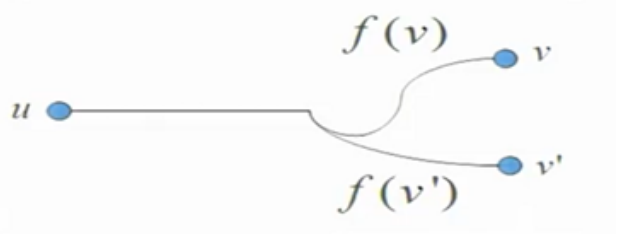
Theorem

Theorem: A graph is bipartite if and only if it has no odd cycles

Sufficiency condition: Assume a graph has no odd cycles -> Show the graph is bipartite

Prove by construction: Let G be a graph with no odd cycle → construct the bipartition

- We prove that G is bipartite by constructing a bipartition of **each nontrivial component H**.
- **Consider u be a node** in the connected component H
- For each **v∈V(H), let f(v) be the minimum length of a u-v path**. Since H is connected, f(v) is defined for every v∈V(H).



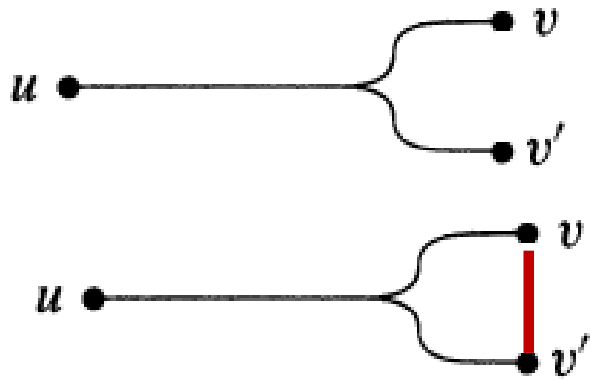
Theorem

Sufficiency condition: A graph is bipartite if and only if it has no odd cycles

Let $X=\{v \in V(H): f(v) \text{ is even}\}$ and $Y=\{v \in V(H): f(v) \text{ is odd}\}$

- We need to show that **X and Y are independent sets** – bipartitions
- Assume, we have an edge (v,v') within X (or Y)
- An edge (v,v') within (X) (or (Y)) would create a **closed walk** using:

1. a shortest (u)-(v) path,
2. the edge (vv') within (X) (or (Y)), and
3. the reverse of a shortest (u)-(v') path.



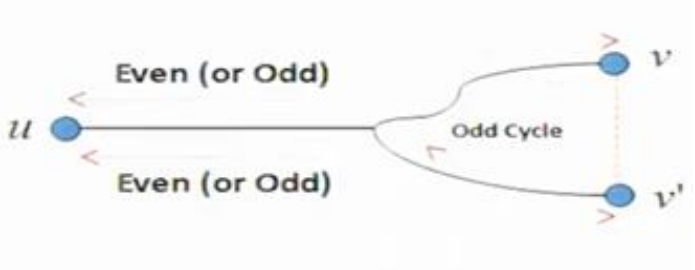
X

Theorem

Note, the edge (v,v') within (X) (or (Y)) would create a **closed odd walk**

even (or odd) + even (or odd) = even

even + 1 = odd



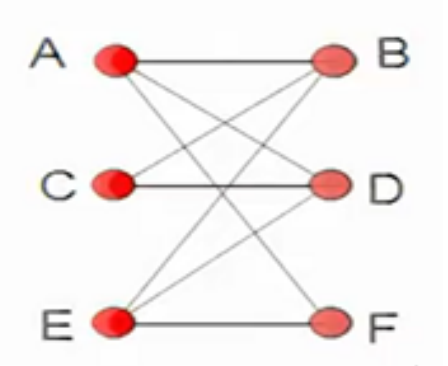
X

We know that such a **odd closed walk must contain an odd cycle**

Since there are no odd cycles, the edge (vv') cannot exist within (X) or within (Y).

Therefore, (X) and (Y) are independent sets – bipartitions

Theorem



Necessary condition

Assume a graph is bipartite -> it has no odd cycles

- Let G be a bipartite graph.
- Every walk alternates between the two partite sets of the bipartition.
- Therefore, every return to the original partite set occurs after an even number of steps.
- Hence, every cycle in G has even length.
- Therefore, (G) has no odd cycle.

Theorem

Theorem: If (k>0), then a k-regular bipartite graph has the same number of vertices in each partite set.

Proof:

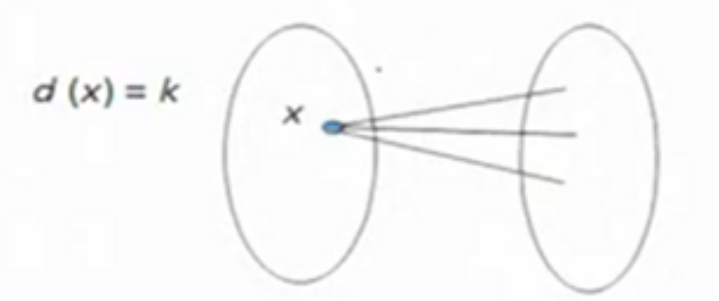
Let G be a k-regular X,Y-bipartite graph.

Counting the edges according to their endpoints in X partition yields

$e(G)=k|X|$

Counting the edges according to their endpoints in Y partition yields

$e(G)=k|Y|$

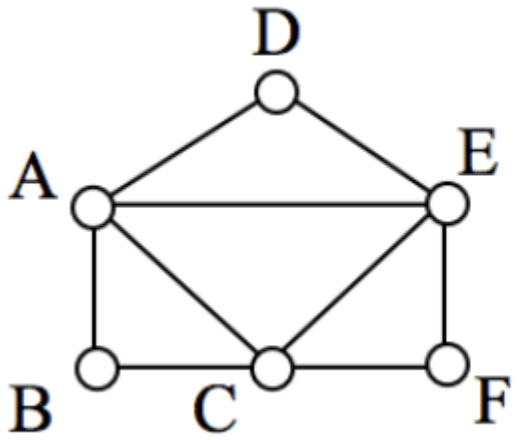


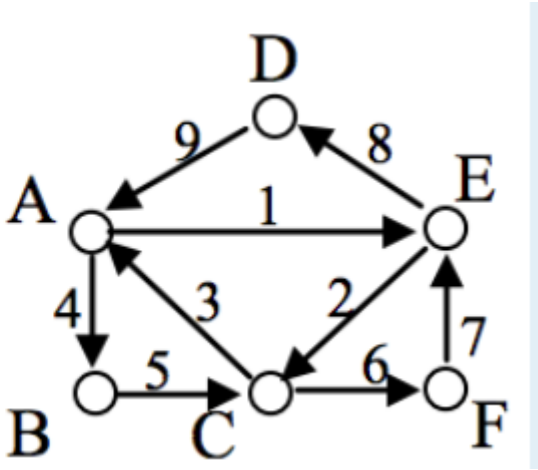
Eulerian Circuit

- We call a **closed trail** **a circuit**
  - We do not specify the first vertex but keep the list in cyclic order.
- **Eulerian circuit** is a **closed trail** that traverses **every edge of a graph exactly once**.
  - Hence Eulerian circuit in a graph is a trail containing all the edges.
- A graph is Eulerian if it has a Eulerian circuit (**closed trail containing all edges)**.
- An **Euler trail** is a trail that **uses every edge** in a graph with **no repeats**.
  - Being a trail, it does not have to return to the starting vertex.

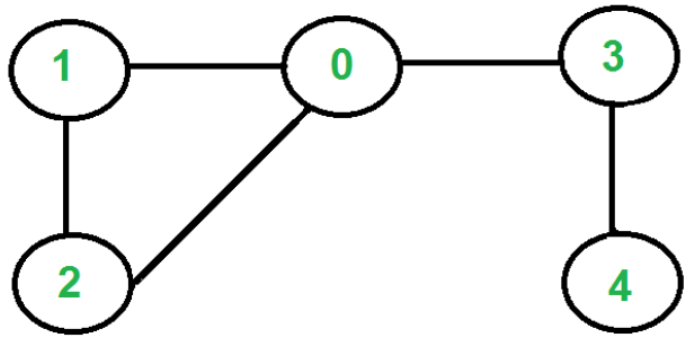
(called Euler path also)

Eulerian Circuit





AEDABCFECA



The graph has Eulerian Paths, for example "4 3 0 1 2 0", but no Eulerian Cycle. Note that there are two vertices with odd degree (4 and 0)

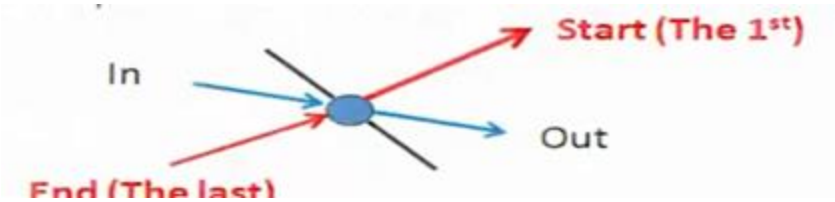
Euler Theorem

Theorem: A graph G is Eulerian if and only if it has at most one nontrivial component and all its vertices have even degree

Proof (Necessity)

Consider graph G is Eulerian => show that it has at most one nontrivial component and every vertex has even degree.

- **Suppose that graph G has an Eulerian circuit (C).**
- Each passage of (C) through **a vertex** uses **two** incident edges,
- **At the starting vertex** the first edge is paired with the last edge. Hence every vertex has **even degree.**
- Also, **two edges can belong to the same trail** only if they lie in the **same connected component**.
- Therefore, an Eulerian circuit can exist in at most one nontrivial component of G.



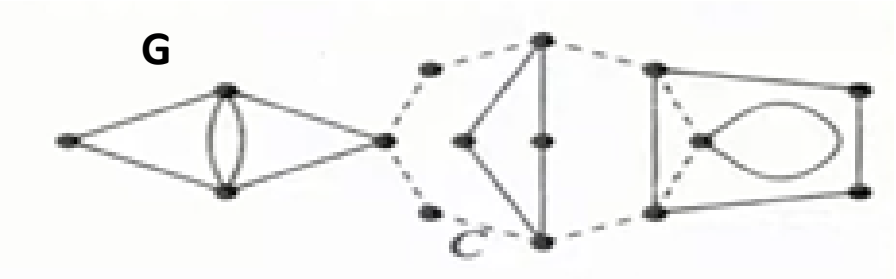
Theorem

Theorem: A graph (G) is Eulerian if and only if it has at most one nontrivial component and all its vertices have even degree.

Proof (Sufficiency)

- To show: **If G has at most one nontrivial component and every vertex has even degree**, then (G) is **Eulerian**.
- Assume that the given conditions hold.
- We prove that (G) has an Eulerian circuit by **induction on the number of edges, m.**
- **Basis step**: **m=0**. A closed trail consisting of a single vertex is an Eulerian circuit.

Theorem: A graph G is Eulerian if and only if it has at most one nontrivial component and its vertices all have even degree.



- **Induction step: m > 0**.

Induction hypothesis: The claim is valid for graph with number of edge < m

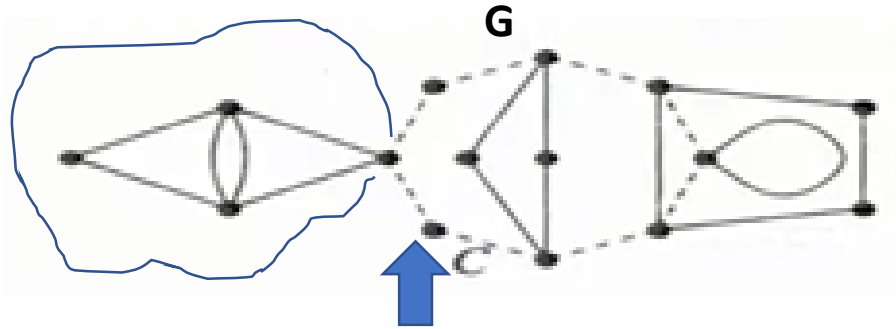
We need to **show that it's also valid for number of edges m** → prove by **construction**

- **With even degrees**, each vertex in the nontrivial component of G has **degree at least 2.**

[Lemma : If every vertex of graph G has degree at least 2, then G contains a cycle]

- Hence, the nontrivial component has a cycle C.
- Let G' be the graph obtained from G by **deleting E(C).**
- Since C has 0 or 2 edges at each vertex, each component of G' is also an even graph.
- Since **each component** is also connected and has **fewer than m edges**,
  - we can apply the induction hypothesis to conclude that **each component of G' has an Eulerian circuit**.

Theorem: A graph G is Eulerian if and only if it has at most one nontrivial component and its vertices all have even degree.



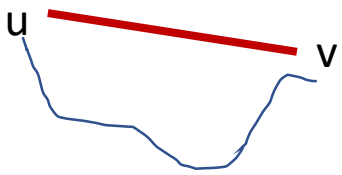
- To construct the Eulerian circuit of G,
- We traverse C,
  - when a **component of G'** is entered for the **first time**
  - we detour along an Eulerian circuit of that component.
- **This circuit ends** at the vertex where we **began the detour**.
- When we complete the traversal of C, we have completed an Eulerian circuit of G.

Lemma

- Lemma: A graph will contain an **Euler path** if it contains **at most two vertices of odd degree**.
- G has (i) 0 odd degree vertex, (ii) one odd degree vertex, (iii) two odd degree vertex
- G has Euler path

Lemma: A graph will contain an Euler path if it contains at most two vertices of odd degree.

- 0 odd degree vertex, **one odd degree vertex**, two odd degree vertex
- We consider two cases:
- **No odd-degree vertex**
  - The graph is Eulerian.
  - Hence, it has an Euler circuit, which is also an Euler path.
- **Exactly two odd-degree vertices**
  - Convert the graph into an even graph by adding one extra edge.
  - Find an Euler circuit.
  - Remove the extra edge to obtain an Euler path.



Eulerian circuits with loops

Our discussion of Eulerian circuits also applies to graphs with loops.

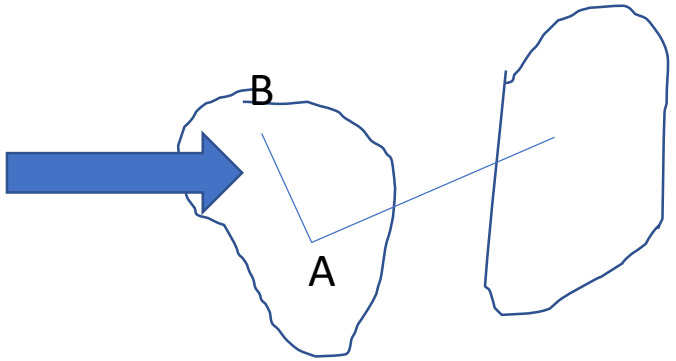
We extend the notion of vertex degree to graphs with loops by counting each loop as contributing 2 to the degree of its incident vertex.

This does not change the parity of a vertex degree.

Characterization of graphs

- Characterization class of graph **G** by a condition **P** means proving the equivalence – "**H€G** if and only if **H satisfies** the **condition P**"
- Hence, **P is the both necessary and sufficient conditions** for the membership in **G**
- H€G => H satisfies P
- H satisfies P => H€G
- If the graph H has **no odd cycles** (P) => The graph is **bipartite** (in **G**)
- if H has at most one nontrivial component and its vertices all have even degree (P) => The graph H is **Eulerian** (in **G**)
- **if edge** e belongs to **no cycle => the** edge e **cut edge**

Fleury's Algorithm



Step 1

Choose the starting vertex.

- Start at any vertex (say, A).

Step 2

Among all edges incident on the current vertex (A),

- Prefer an edge that **is not a bridge** – doesn't disconnect G to two components
- Use a bridge only when it is the only available edge.

Step 3

Traverse the selected edge (AB).

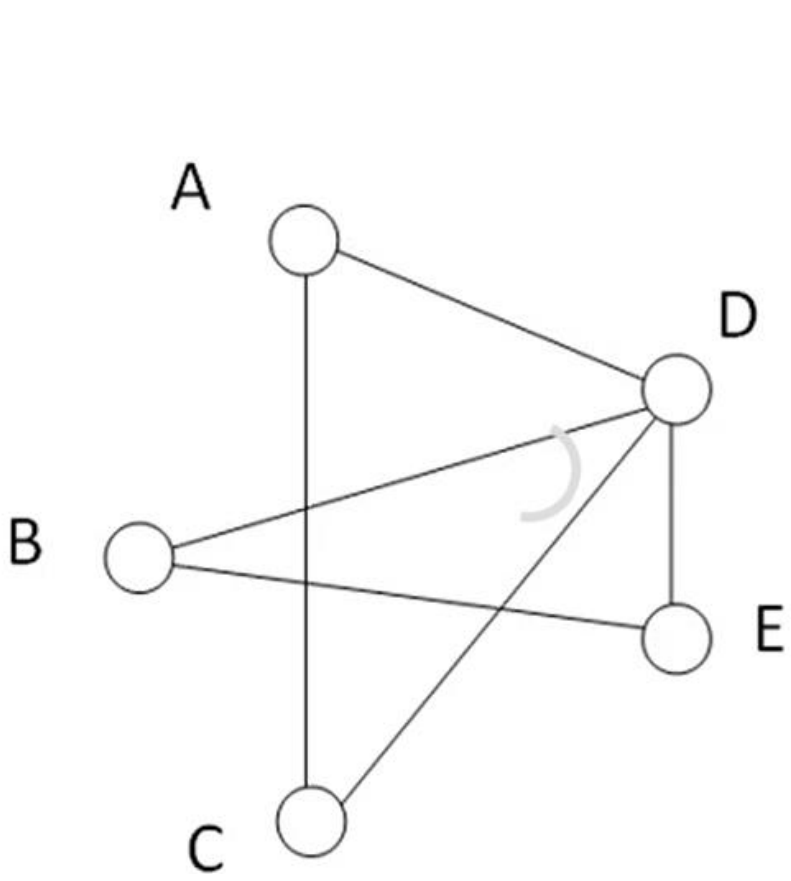
- Add it to the Euler tour.
- Delete the edge from the graph.

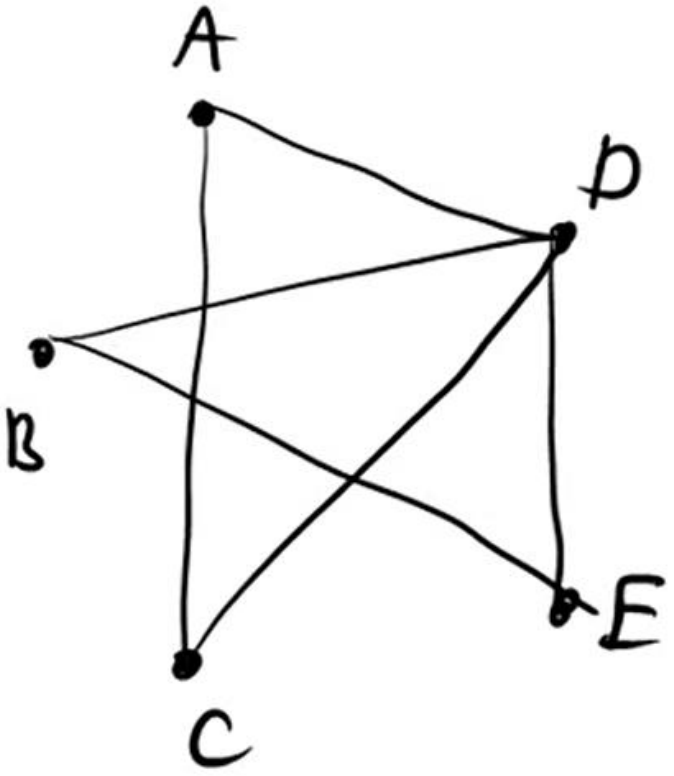
Step 4

Move to the adjacent vertex (B).

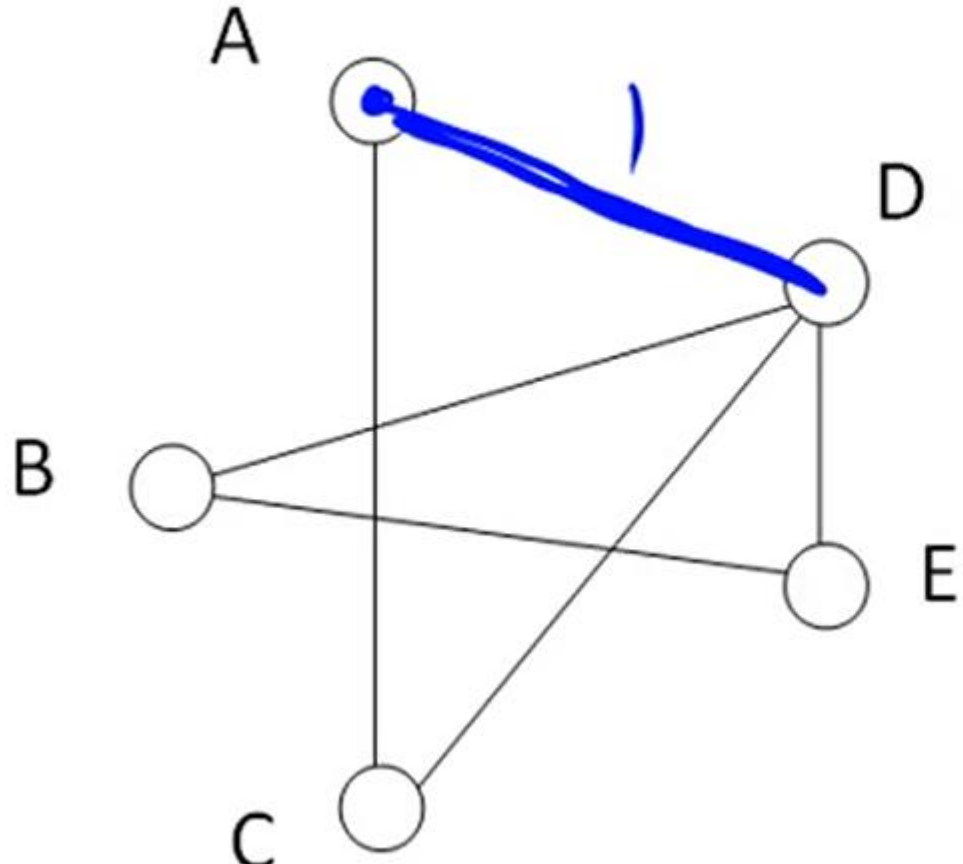
Repeat Steps 2–4 until all edges are used exactly once.

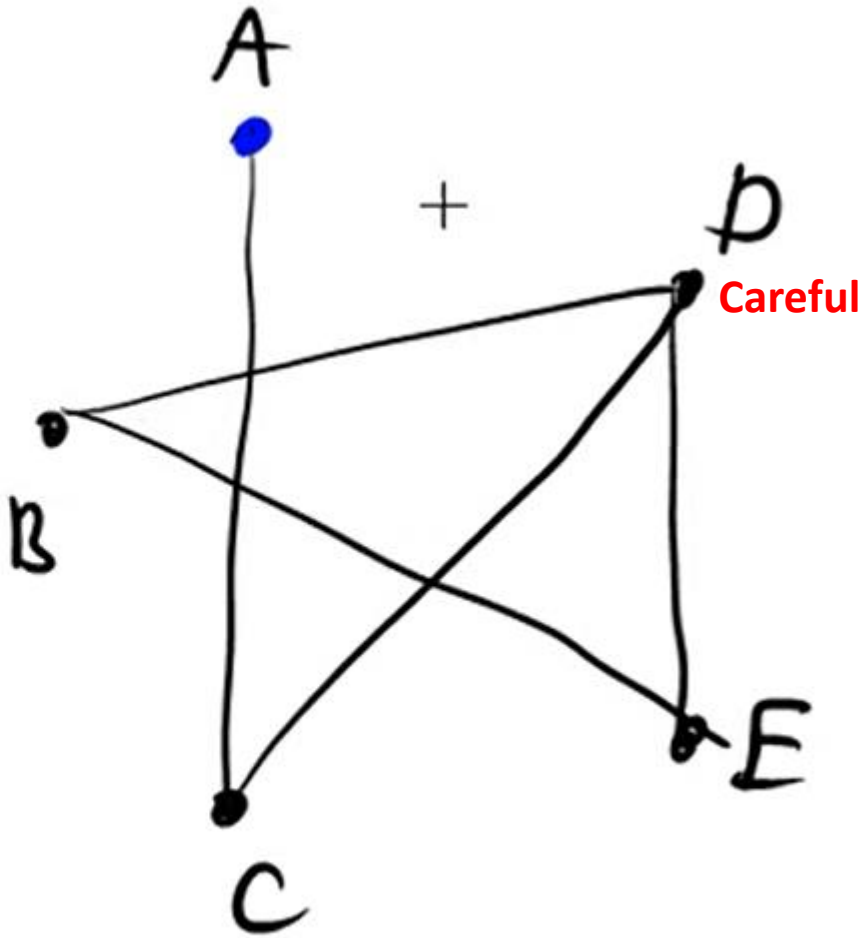
Fleury's Algorithm



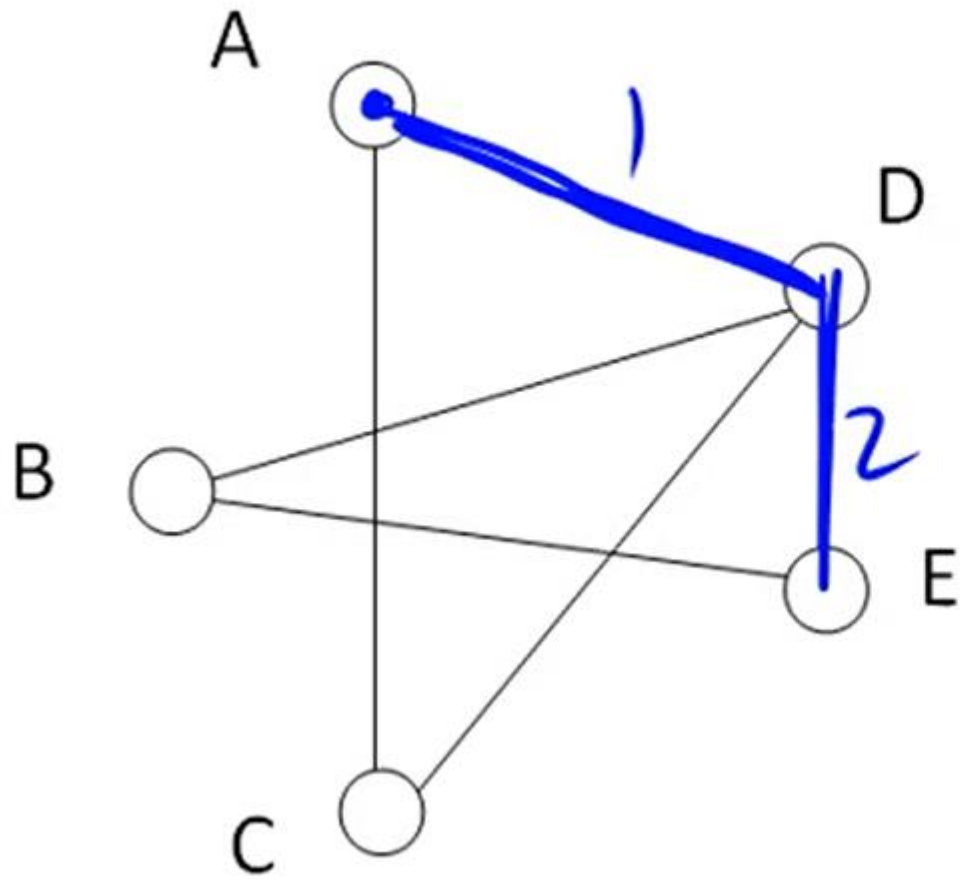


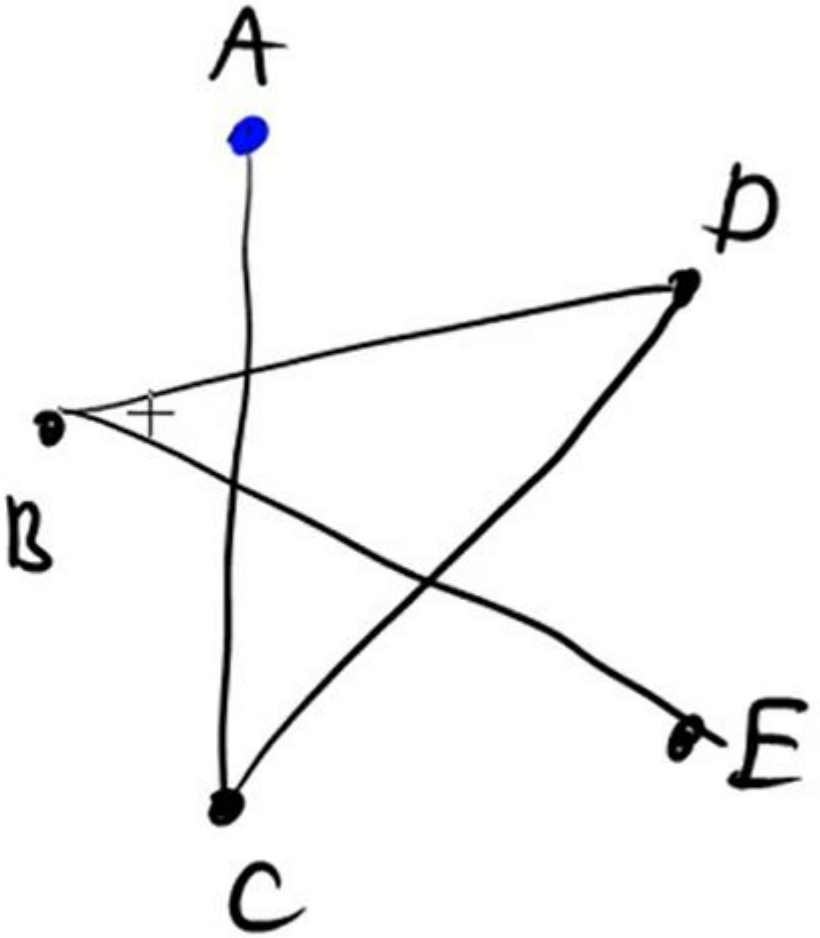
Fleury's Algorithm



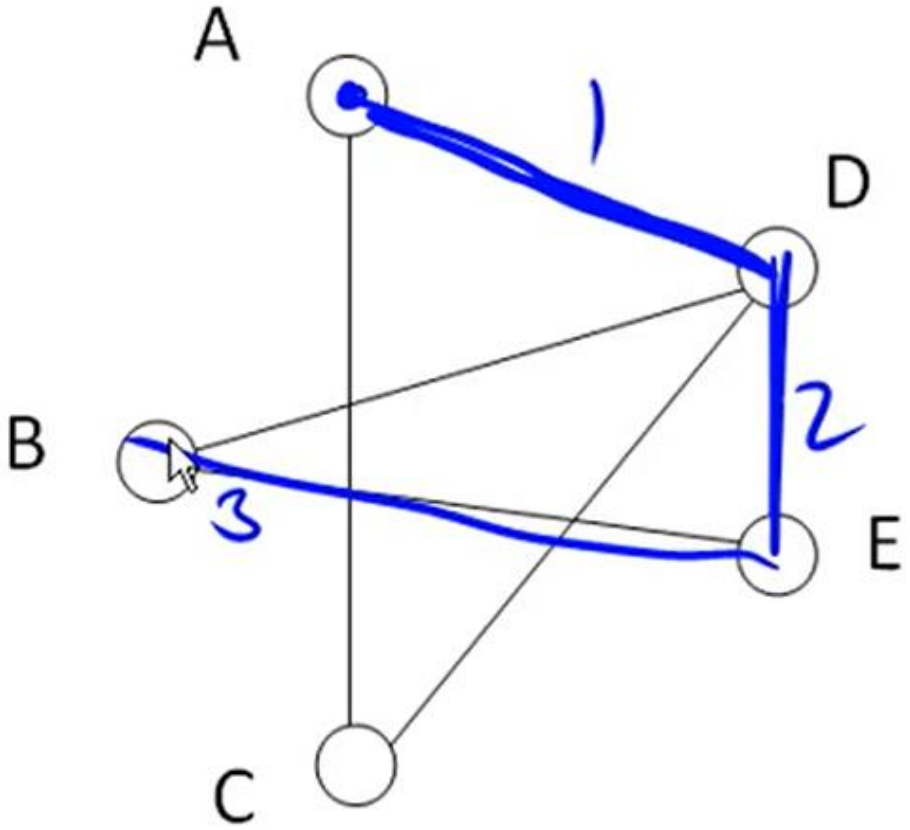


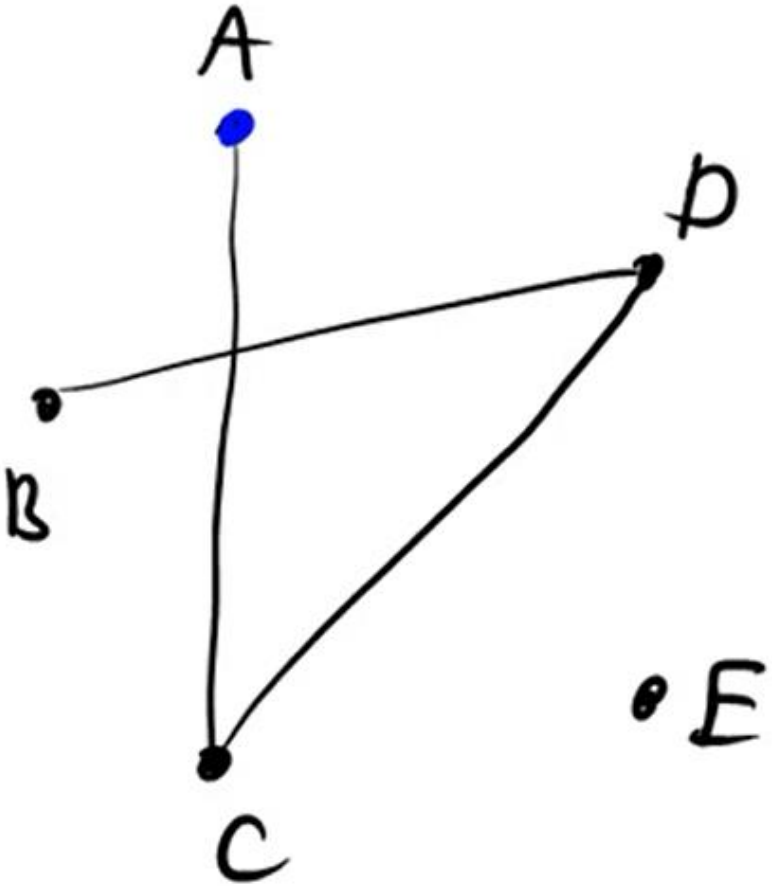
Fleury's Algorithm



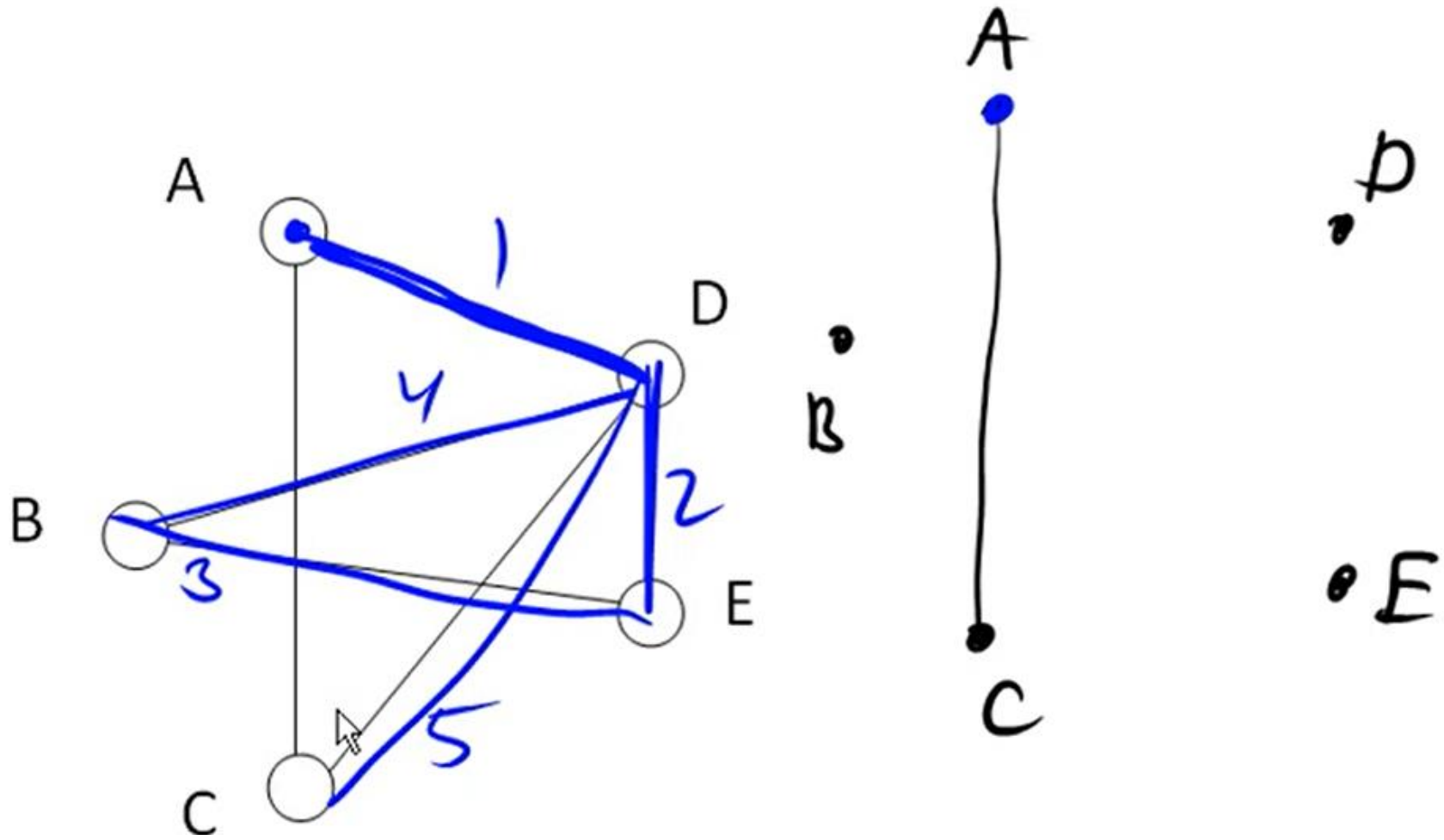


Fleury's Algorithm

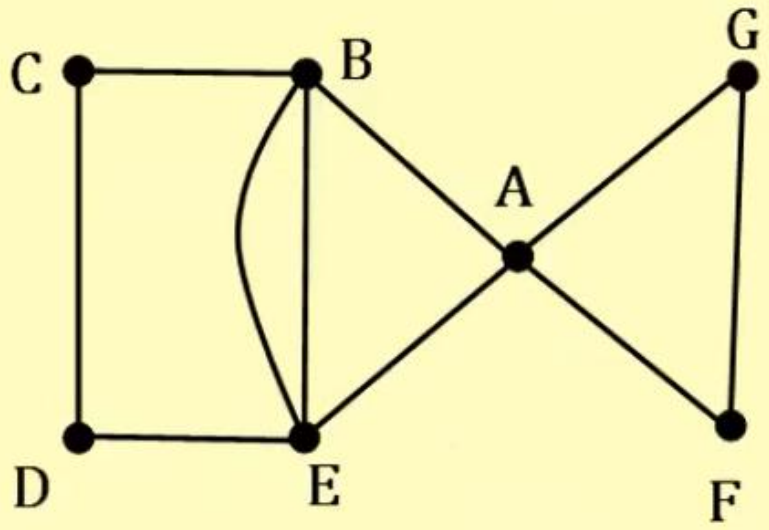




Fleury's Algorithm



Example

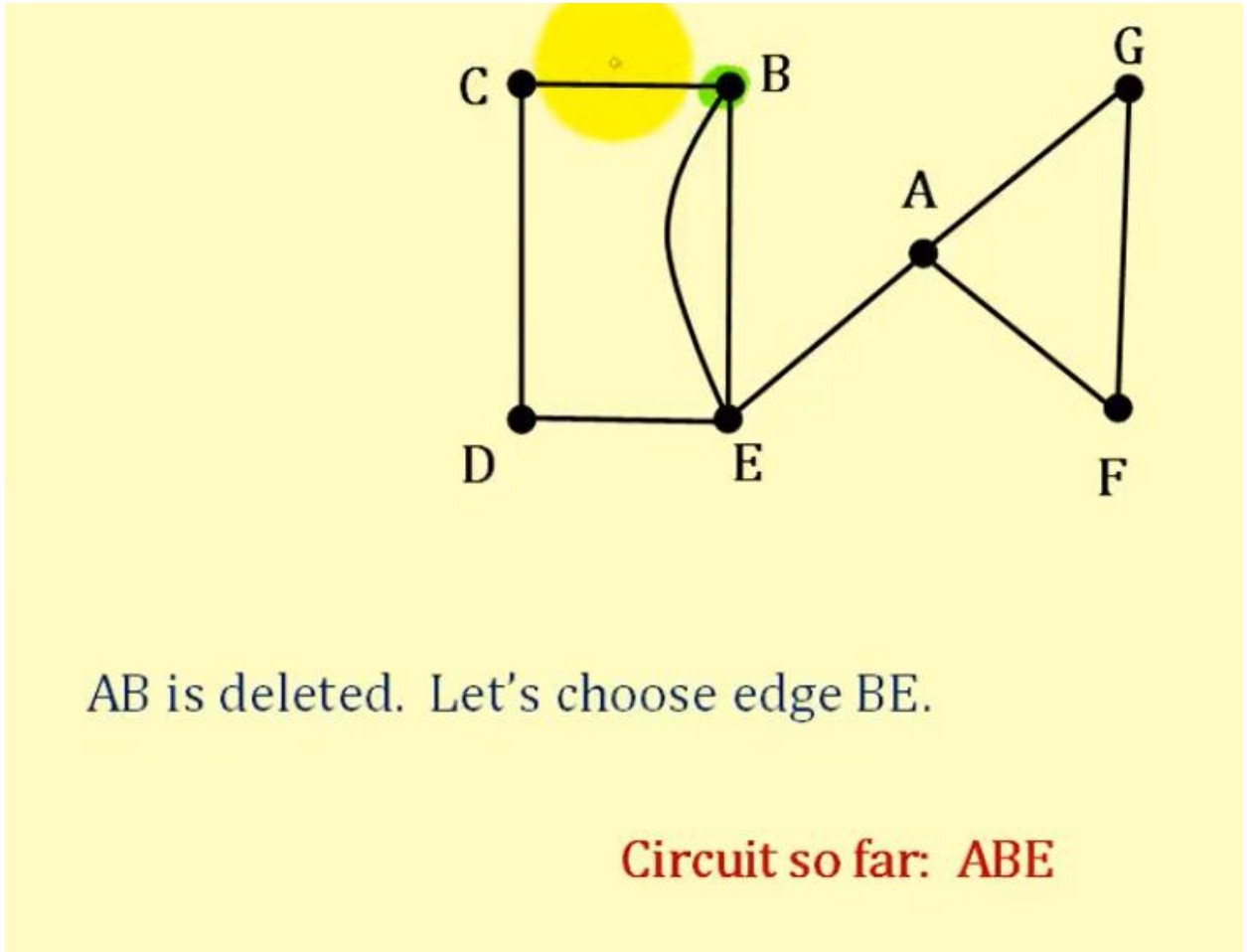


A graph will contain an Euler circuit if all vertices have even degree.

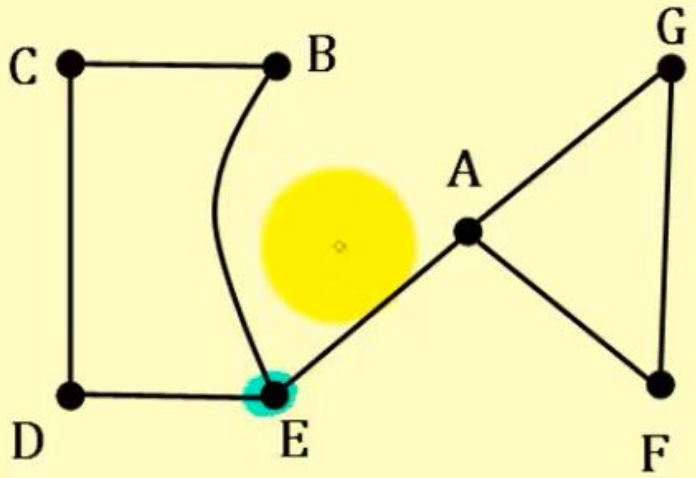
Let's begin by choosing edge AB.

Circuit so far: AB

Fleury's Algorithm



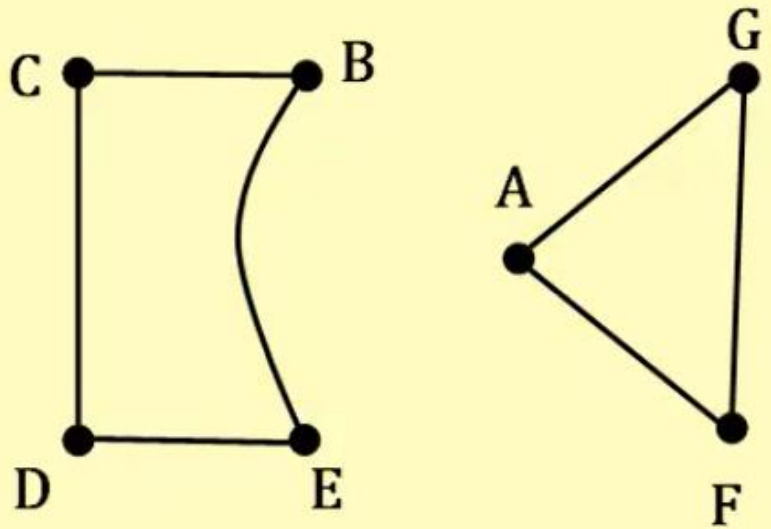
Fleury's Algorithm



AB and BE are deleted. Notice we can not choose edge EA because this would separate the graph into two disconnected sets of edges. Choose edge ED.

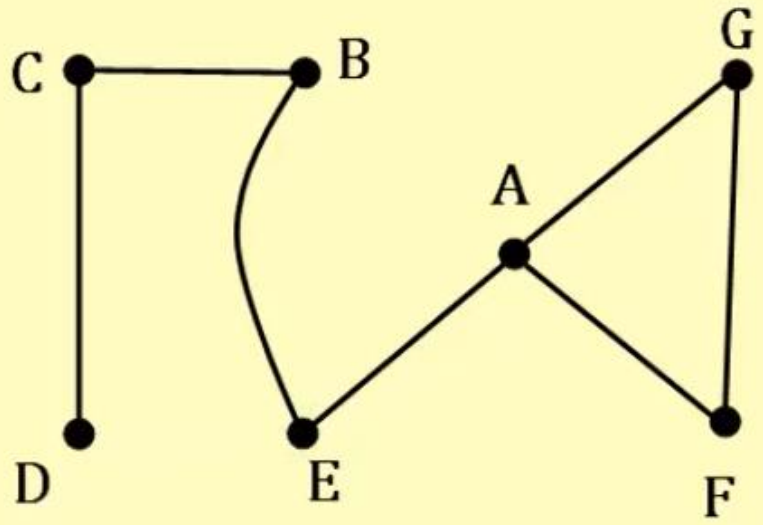
Circuit so far: ABED

Fleury's Algorithm



AB and BE are deleted. Notice we can not choose edge EA. Notice if we did choose EA, the graph is separated. This violates **Fleury's Algorithm.**

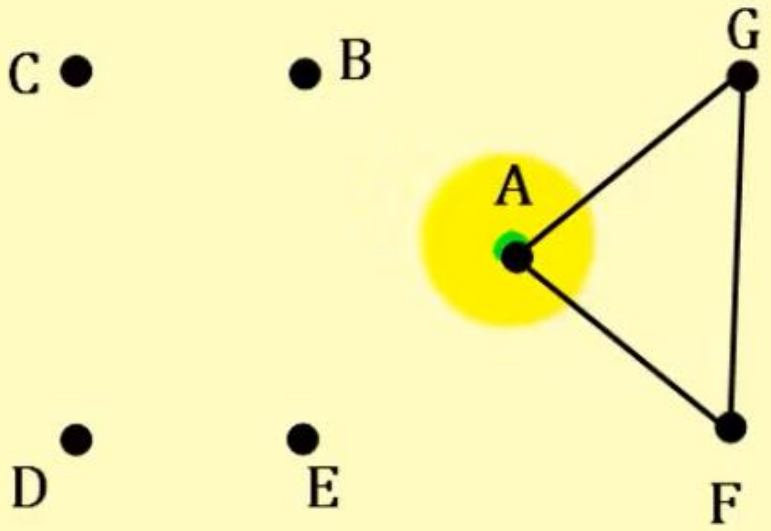
Fleury's Algorithm



AB, BE, and ED are deleted. Notice there is only one option for the next several choices. Choose edge DC, then CB, then BE, and then EA.

Circuit so far: ABEDCBEA

Fleury's Algorithm



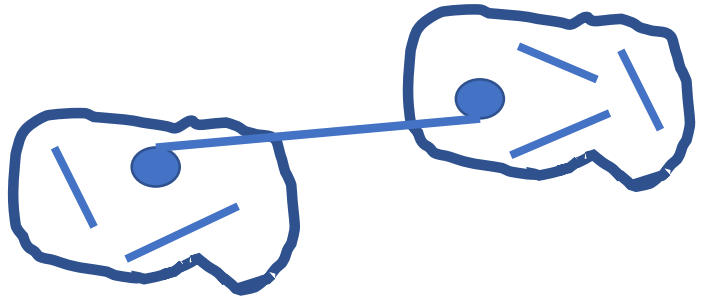
AB, BE, ED, DC, CB, BE, and EA are deleted. We now can choose AF or AG. Let's choose AF.

Circuit so far: ABEDCBEAF

The Key Idea

- **Goal: Traverse every edge exactly once without getting stuck before all edges are visited.**
- At each step, prefer a non-bridge edge whenever one is available.
- As long as there is another available edge, postpone crossing a bridge.
- Instead,
  - First traverse edges that lie inside the current connected part,
  - Use the bridge link until, they become the only available choice.
- Thus, bridges naturally become the last exit from a portion of the graph.

The Key Idea



- **Crossing a bridge too early may disconnect the graph, leaving some edges unreachable**
- A non-bridge can be traversed safely because the remaining graph stays connected.
- Postpone crossing bridges until they become the only available choice.
- Thus, the algorithm always preserves the possibility of completing the Euler circuit.