

Hamiltonian Graph

Hamiltonian Graph

Definitions

A **Hamiltonian cycle** in a graph G is a cycle that includes all the vertices of the graph.

Hamiltonian Cycle: Cycle that covers every vertex once

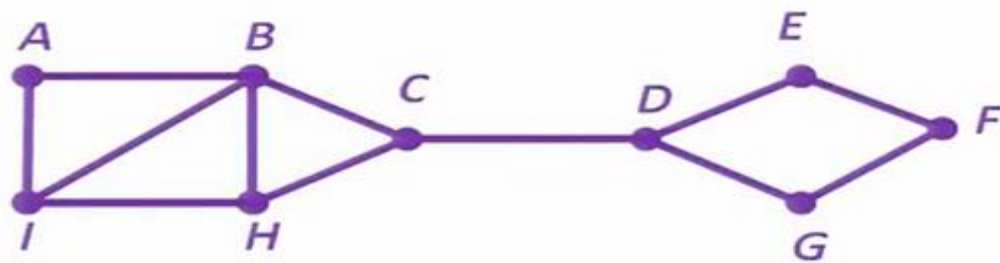
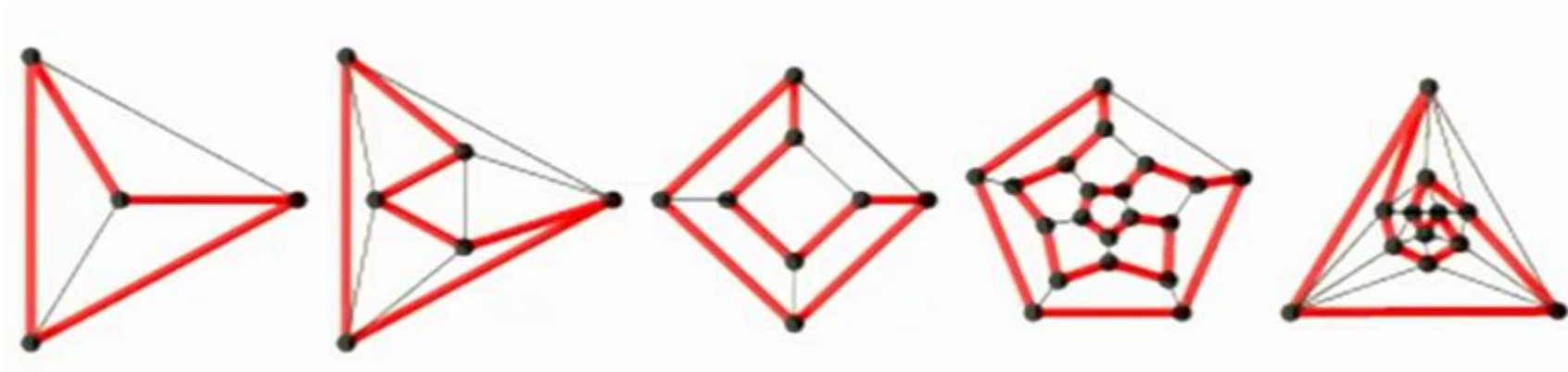
Hamiltonian Path: Path that covers every vertex once

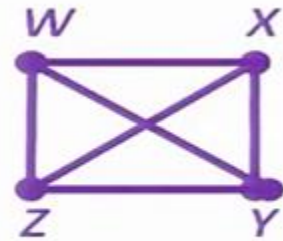
Hamiltonian Graph: A graph containing a Hamiltonian Cycle is called a Hamiltonian Graph

Nonhamiltonian Graph: A nonhamiltonian graph is a graph that is not Hamiltonian

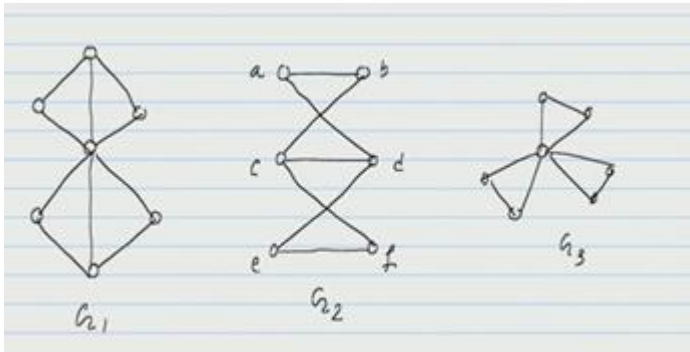
Hamiltonian cycle can be converted to a Hamiltonian path by removing one edge.

Examples

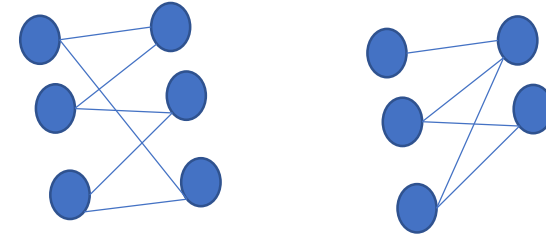




Complete Graphs



Necessary Conditions



Unlike Eulerian graphs, there is no known condition that can be used to characterize if a graph is Hamiltonian

- **Every Hamiltonian graph is 2-connected**
 - Because deleting a vertex leaves a subgraph with a spanning path.
- Bipartite graphs suggest a way to strengthen this necessary condition.
- Bipartite graphs. **A spanning cycle** in a bipartite graph visits the **two partite sets alternately**,
 - So there can be no such cycle **unless the partite sets have the same size**.
 - Alternatively, we can argue that the cycle returns to different vertices of one partite set after each visit to the other partite set.

Hence $K_{m,n}$ is Hamiltonian only if $m = n$

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)

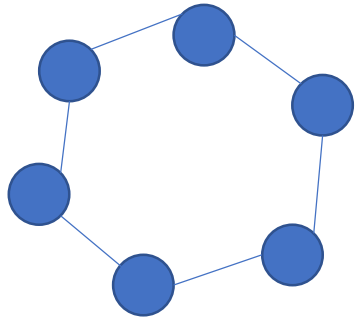
- To show: **If G has at most one nontrivial component and every vertex has even degree, then (G) is Eulerian.**
- Assume that the given conditions hold.
- We prove that (G) has an Eulerian circuit by **induction on the number of edges, m.**
- **Basis step: $m=0$.** A closed trail consisting of a single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



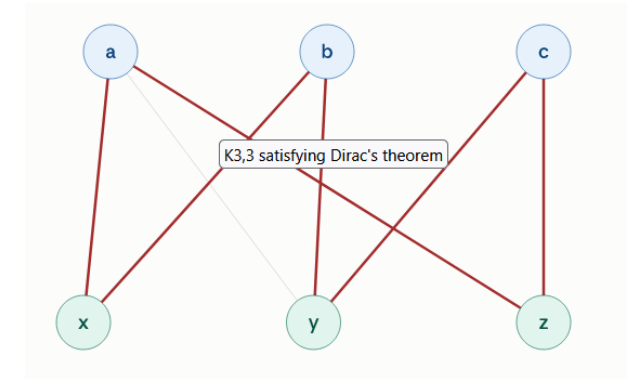
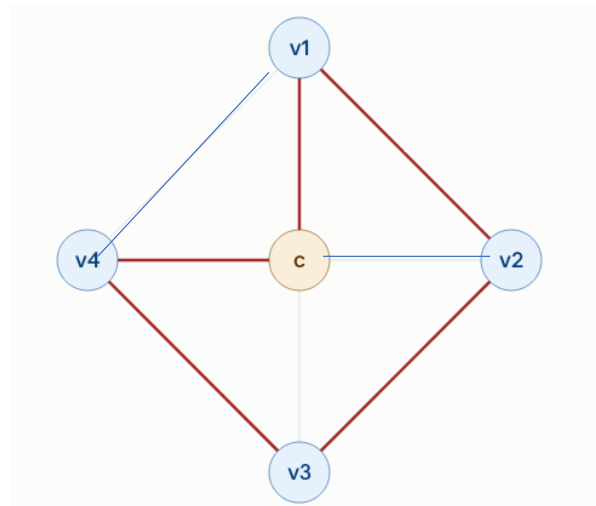
Sufficient conditions: Dirac's Theorem

Theorem: A sufficient condition for a simple graph G to have a Hamiltonian cycle is that the degree of every vertex of G be at least $n/2$, where n = no. of vertices in G (≥ 3)

$d(u) \geq n/2 \Rightarrow$ Hamiltonian cycle



$N=6$



Dirac's Theorem

Theorem: A sufficient condition for a simple graph G to have a Hamiltonian cycle is that the degree of every vertex of G be at least $n/2$, where n = no. of vertices in G (≥ 3)

$d(u) \geq n/2 \rightarrow$ Hamiltonian cycle

G is connected

Dirac's Theorem

Theorem: A sufficient condition for a simple graph G to have a Hamiltonian cycle is that the degree of every vertex of G be at least $n/2$, where $n = \text{no. of vertices in } G (\geq 3)$

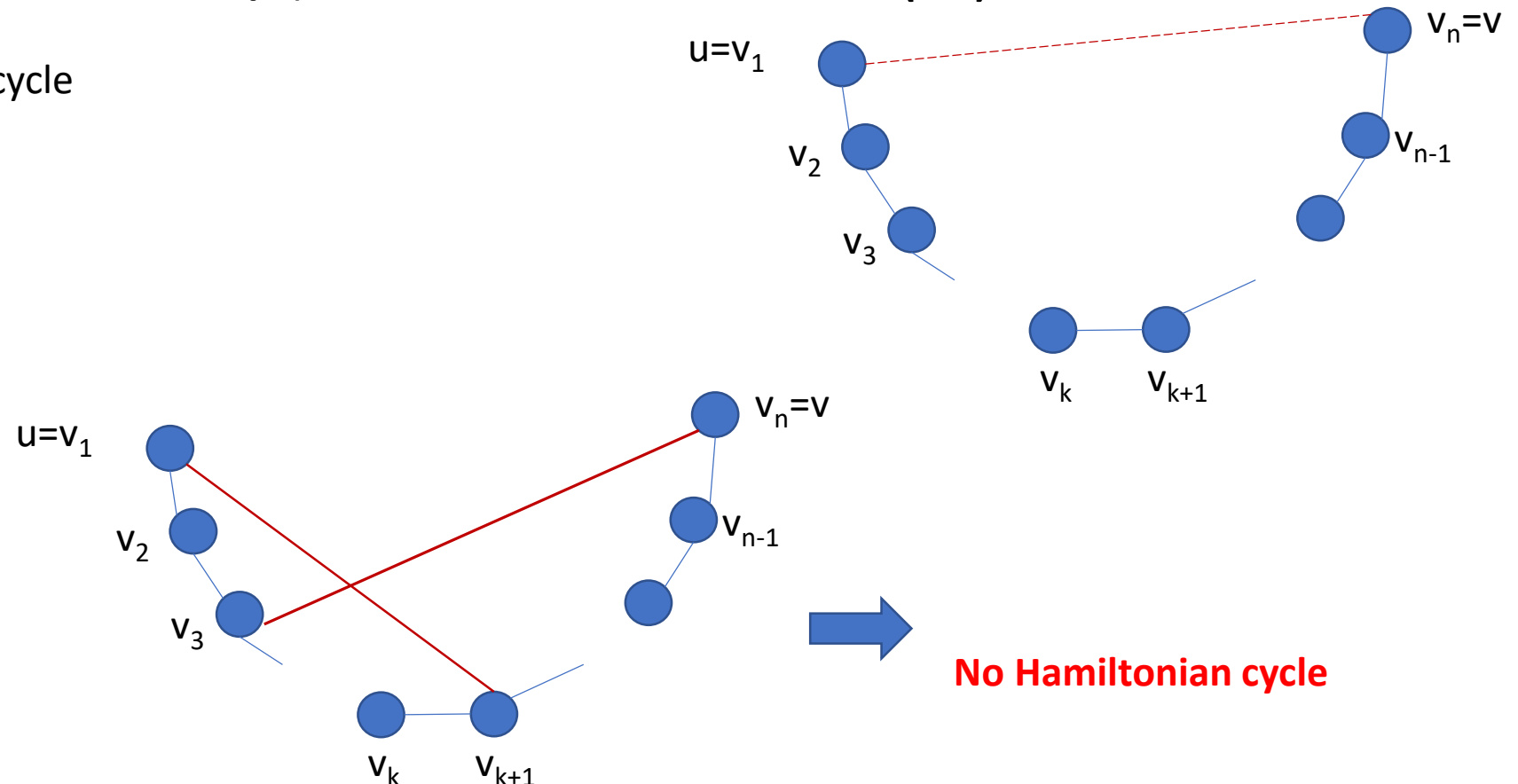
$d(u) \geq n/2 \Rightarrow$ Hamiltonian cycle

Consider a maximal path

$P = \{v_1, v_2, \dots, v_k, v_{k+1}, \dots, v_n\}$

v_k, v_{k+1} are adjacent vertices

Every neighbor of u, v lie on P



Dirac's Theorem

Theorem: A sufficient condition for a simple graph G to have a Hamiltonian cycle is that the degree of every vertex of G be at least $n/2$, where $n = \text{no. of vertices in } G (\geq 3)$

Consider a maximal path

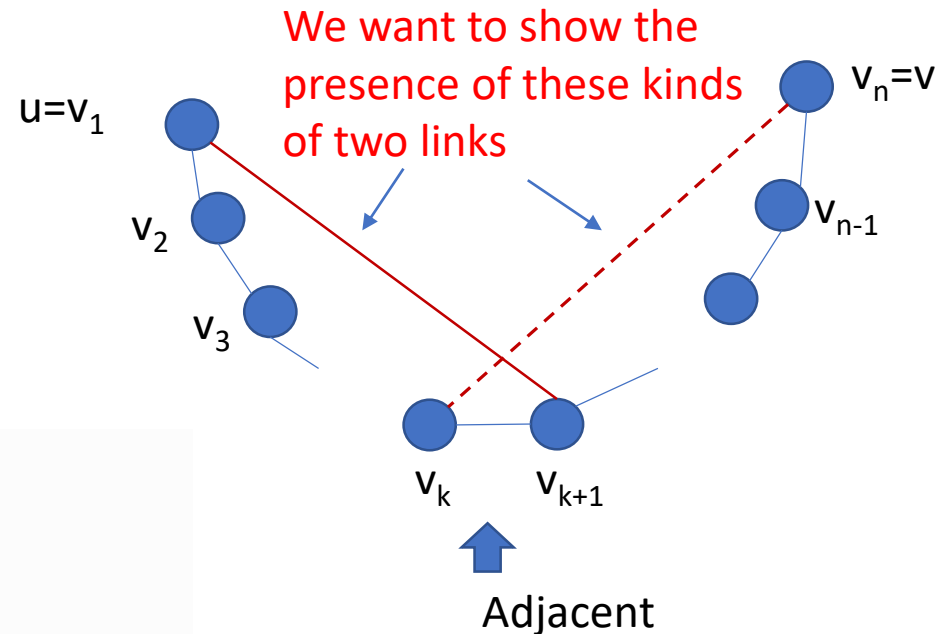
$P = \{v_1, v_2, \dots, v_k, v_{k+1}, \dots, v_n\}$

v_k, v_{k+1} are adjacent vertices

The common index k gives us:

$$u \sim v_{k+1} \quad \text{and} \quad v_k \sim v$$

Those two edges allow us to close the Hamiltonian path into a Hamiltonian cycle.



Dirac's Theorem

Focus on u

We look at **all neighbors of u** .

Suppose u is connected to v_2, v_4, v_7 , for example.

Instead of recording the neighbors themselves, we record the **position immediately before each neighbor**:

$$A = \{1, 3, 6, \dots\}.$$

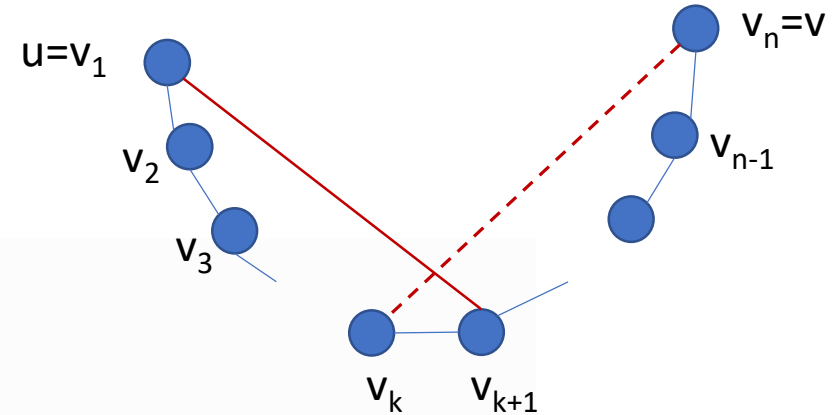
Formally,

$$A = \{k : u \sim v_{k+1}\}.$$



So:

$$k \in A \iff u \sim v_{k+1}.$$



Dirac's Theorem

Focus on v

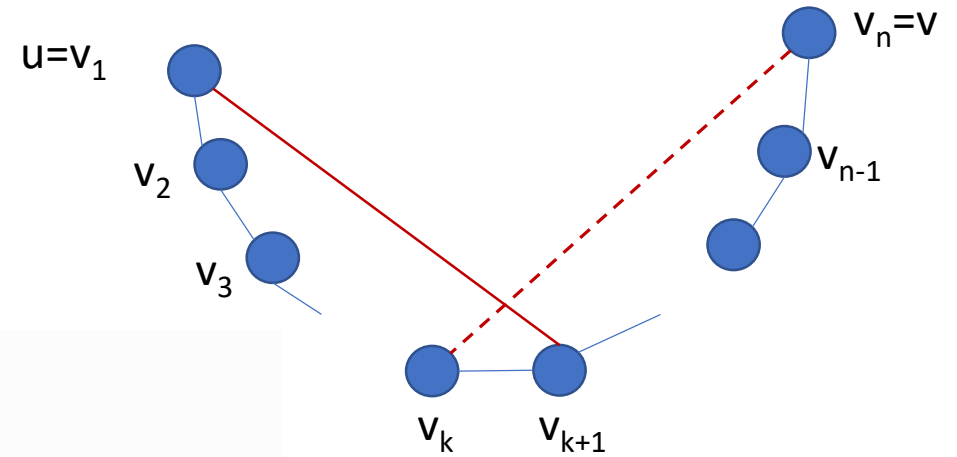
Now we do something similar with v .

We look at all neighbors of v , and record their positions directly:

$$B = \{ k : v_k \sim v \}.$$

So:

$$k \in B \iff v_k \sim v.$$



$$|A| = d(u) \geq \frac{n}{2}$$

and

$$|B| = d(v) \geq \frac{n}{2}.$$

But A and B are both subsets of

$$\{1, 2, \dots, n-1\},$$

which has only $n-1$ elements.

[n-2 nodes, + either u or v]

Therefore,

$$|A| + |B| \geq n.$$

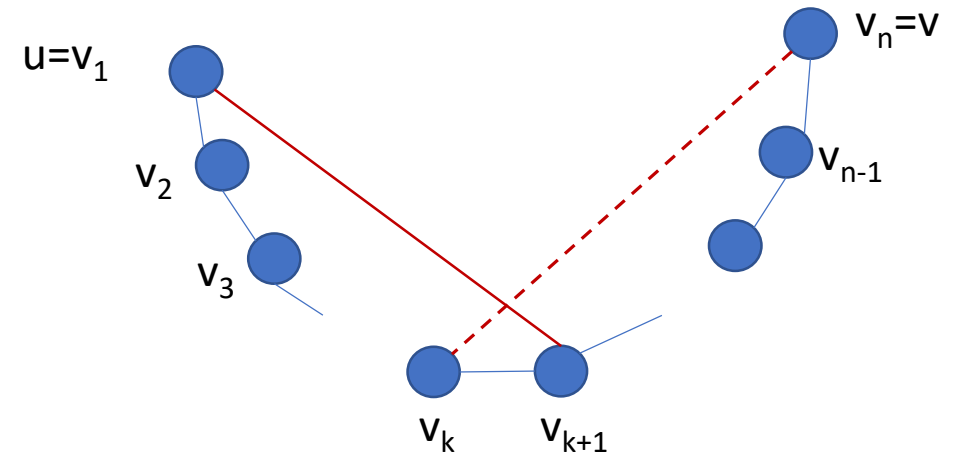
But there are only $n-1$ possible indices.

[n-2 nodes, + either u or v]

So A and B cannot be disjoint.

Hence:

$$A \cap B \neq \emptyset.$$



One common index k in A and B

One v_{k+1} neighbor of u

One v_k neighbor of v

Consider a maximal path

$$P = \{v_1, v_2, \dots, v_k, v_{k+1}, \dots, v_m\}$$

But perhaps

$$m < n.$$

Could there be some vertices outside this cycle?

Suppose there is a vertex x outside the cycle.

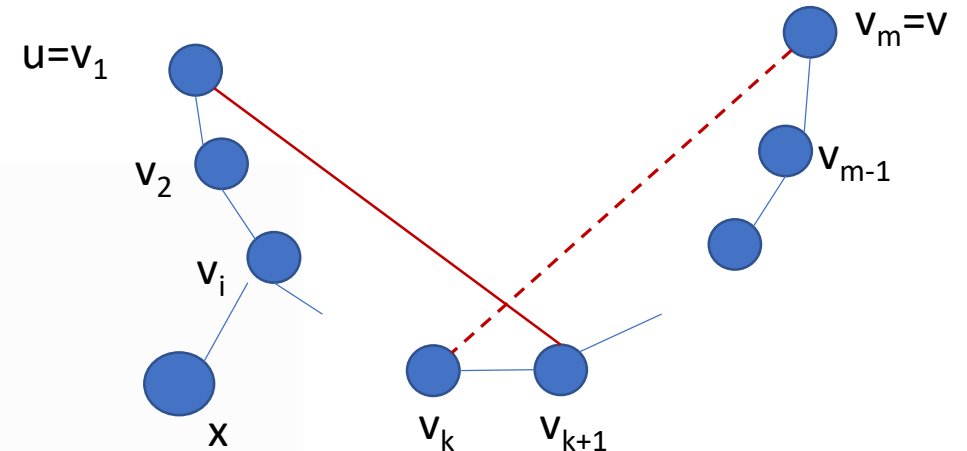
Because

$$d(x) \geq \frac{n}{2} > 0,$$

x has a neighbor. More generally, we need G to be connected; Dirac's condition guarantees that for $n \geq 3$.

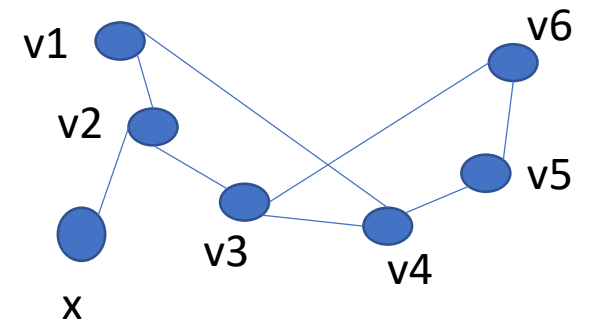
So there is an edge from some outside part toward the cycle. Take a vertex x outside the cycle adjacent to a cycle vertex v_i .

Then we can **break the cycle at v_i** and use x to create a path with more than m vertices.



$$P = \{v_1, v_2, v_3, v_4, v_5, v_6\}$$

$$P = \{v_2, v_3, v_6, v_5, v_4, v_1\}$$



$$P' = \{x, v_2, v_3, v_6, v_5, v_4, v_1\}$$

$$P' > P!!$$

Dirac's Theorem

This gives a path with

$$m + 1$$

vertices.

But P was supposed to be a **longest path**.

Contradiction.

Therefore there cannot be any vertex outside the cycle.

Hence

$$m = n.$$

So the cycle we found contains all n vertices.

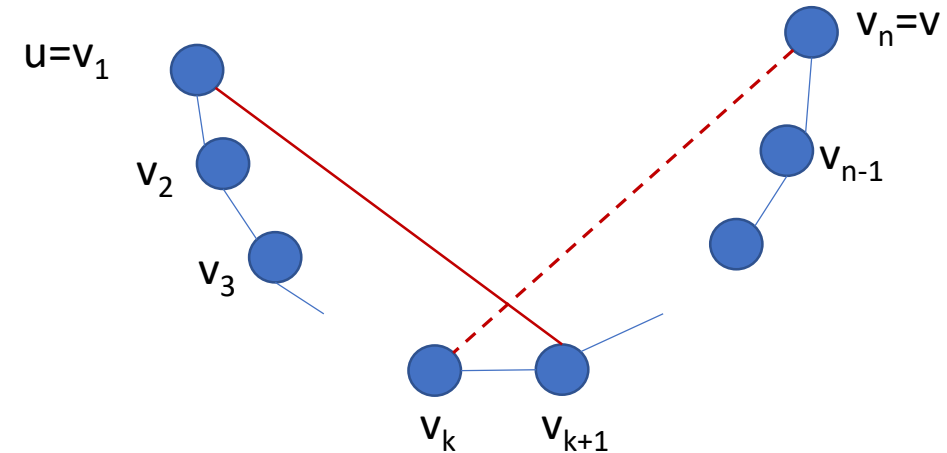
Ore Lemma

Lemma: If $d(u) + d(v) \geq n$ for every pair of non-adjacent vertices u and v of a simple graph G , then G is Hamiltonian.

Assume: Does not contain Hamiltonian cycle

Same way we construct the set A and set B

- Consider set B (neighbors of v)
 $d(v) \leq n-1$
- If v_k is adjacent to v (in B), then **u should not be** neighbor of v_{k+1}
- These **B nodes should not be adjacent to neighbors of u** (i.e. A) ---- should not be adjacent to $d(u)$ nodes
 - Since, each element of $d(u)$ creates **one** neighbor of u
- Note: we are not telling “These B nodes should not be adjacent to u ”



$$d(v) \leq n-1-d(u)$$

Theorem

Theorem: Let (G) be a simple graph on n ($n \geq 3$) vertices. Suppose $(u, v \in V)$ and such that $u-v \notin E(G)$ and $d(u)+d(v) \geq n$.

In this case, G is Hamiltonian if and only if $G+(u,v)$ is Hamiltonian

Proof: Sufficient condition

If G is Hamiltonian $\rightarrow G+(u,v)$ is Hamiltonian.

Necessary condition:

Given that $G+(u,v)$ is Hamiltonian, we need to prove that G is Hamiltonian.

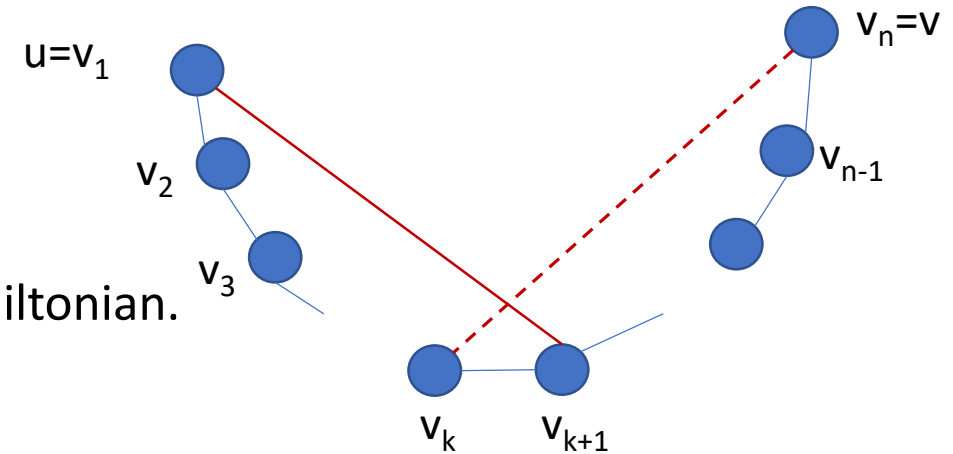
Suppose G is not Hamiltonian.

Then there is a Hamiltonian path/spanning path in G joining u and v .

If u is adjacent to v_{k+1} , then v cannot be adjacent to v_k .

Not possible as per Ore's Lemma ($d(u)+d(v) \geq n$)

So G is Hamiltonian.

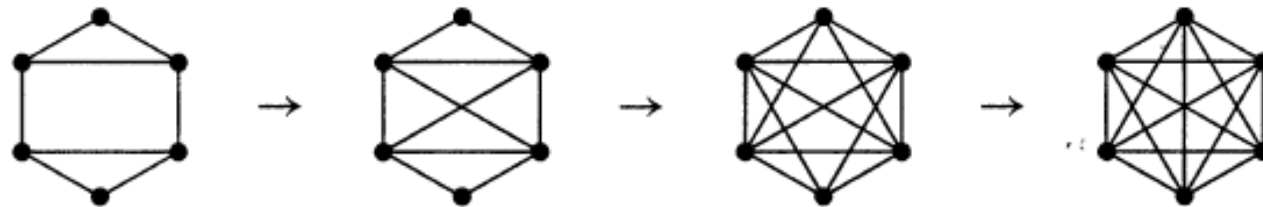


(Ham) closure $C(G)$ of a graph

Definition:

The closure $C(G)$ of a graph G is obtained from G by repeatedly joining pairs of non-adjacent vertices (u,v) for which $d(u)+d(v) \geq n$ until no such pair remains.

Example



Starting with the given graph, we obtain its closure $C(G)$ by adding the required edges.

Bondy-Chvatal Theorem

Theorem: A graph G is Hamiltonian iff its closure $C(G)$ is Hamiltonian.

Proof:

Suppose G is Hamiltonian.

- Then clearly $C(G)$ is Hamiltonian, since $C(G)$ contains all the edges of G .

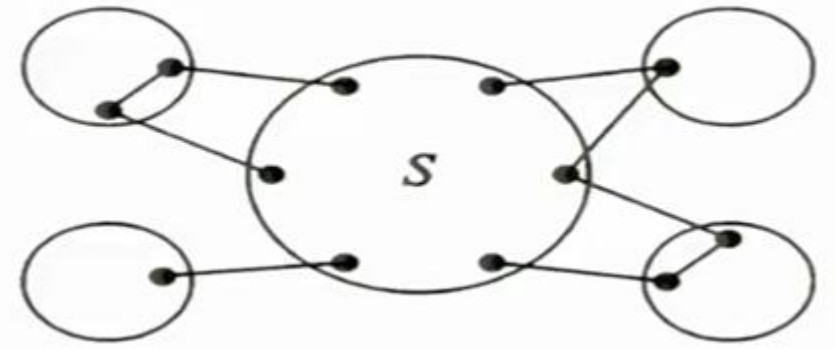
For the converse, **suppose $C(G)$ is Hamiltonian.**

- If $C(G)$ is Hamiltonian, then it **has a Hamiltonian cycle.**
- If the Hamiltonian cycle in **$C(G)$** contains an edge **$u-v$** that was **added during the closure process**,
 - **Drop $u-v$ edge**, obtain G'
 - **u and v were non-adjacent in the graph G' , and $d(u)+d(v) \geq n$.**
- By Ore's theorem, adding an edge $u-v$ does not change whether the graph is Hamiltonian.
- Hence **G' is Hamiltonian** $\rightarrow$ repeat this argument
- Therefore, we can remove the added edges one by one while preserving Hamiltonicity.
- Hence G is Hamiltonian

Corollary

If the closure $C(G)$ of graph G is complete, i.e.,
 $C(G)=K_n$,
then graph G is Hamiltonian,
because K_n is Hamiltonian.

Lemma



- If G has a Hamiltonian cycle, then for each nonempty set $S \subseteq V$, the graph $G - S$ has **at most $|S|$ components**

Let

$$S \subseteq V, \quad S \neq \emptyset.$$

Remove all vertices in S and all edges incident to them. We get

$$G - S.$$

Suppose $G - S$ has r components:

$$C_1, C_2, \dots, C_r.$$

We want to prove

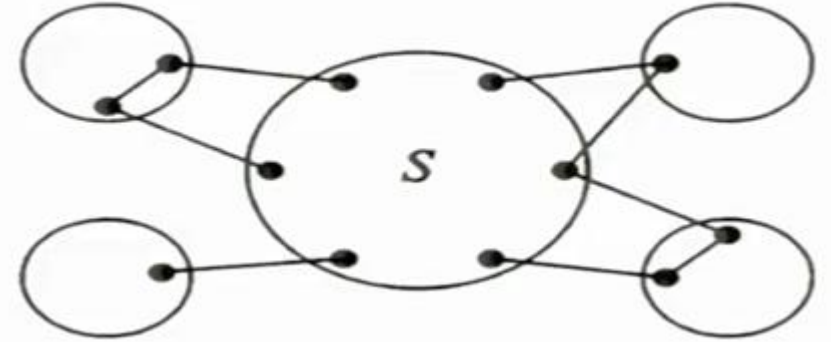
$$\boxed{r \leq |S|}.$$

Lemma

G has a Hamiltonian cycle

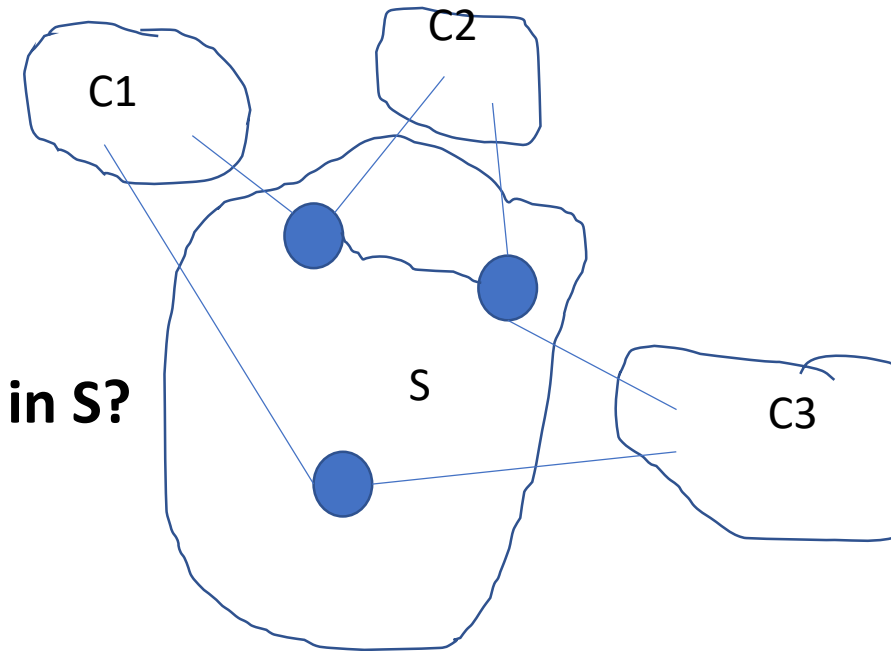
Suppose the Hamiltonian cycle is $\{v_1, v_2, \dots, v_n, v_1\}$

- Because it is Hamiltonian, it contains **every vertex of G exactly once**.
- Now imagine **walking around** this Hamiltonian cycle.
- **Some vertices belong to S , and all the other vertices belong to $(G-S)$**
- Every time we **enter a component $(G-S)$** , we must enter it **from (S)** .
 - Because if we are **at a vertex of $(G-S)$** and the **next vertex on the Hamiltonian cycle is also in $(G-S)$** , we remain **inside component $G-S$** .
 - When we **move from one component of $(G-S)$ to another**, we cannot directly cross between them, because then they would actually be the same component.
- **Leaving a component of $G-S$ requires entering in S .**
- Therefore, the Hamiltonian **cycle must go through S between different components**.



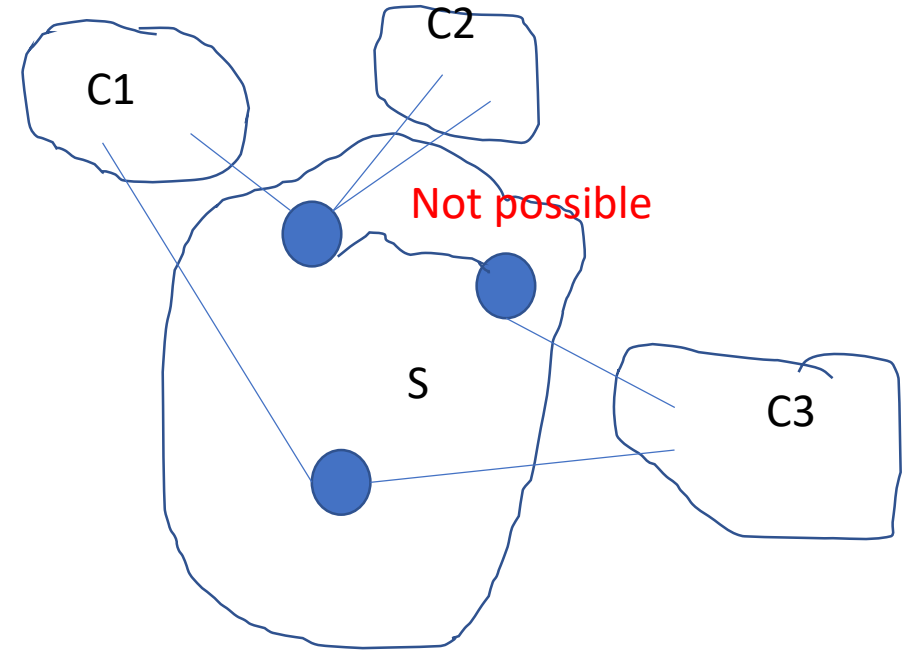
Each arrival in S must be distinct

- Suppose $(G-S)$ has three components:
- C_1, C_2, C_3 .
- Each component needs its own arrival point in S .
- **Why can't C_1 and C_2 both arrive at the same vertex s in S ?**



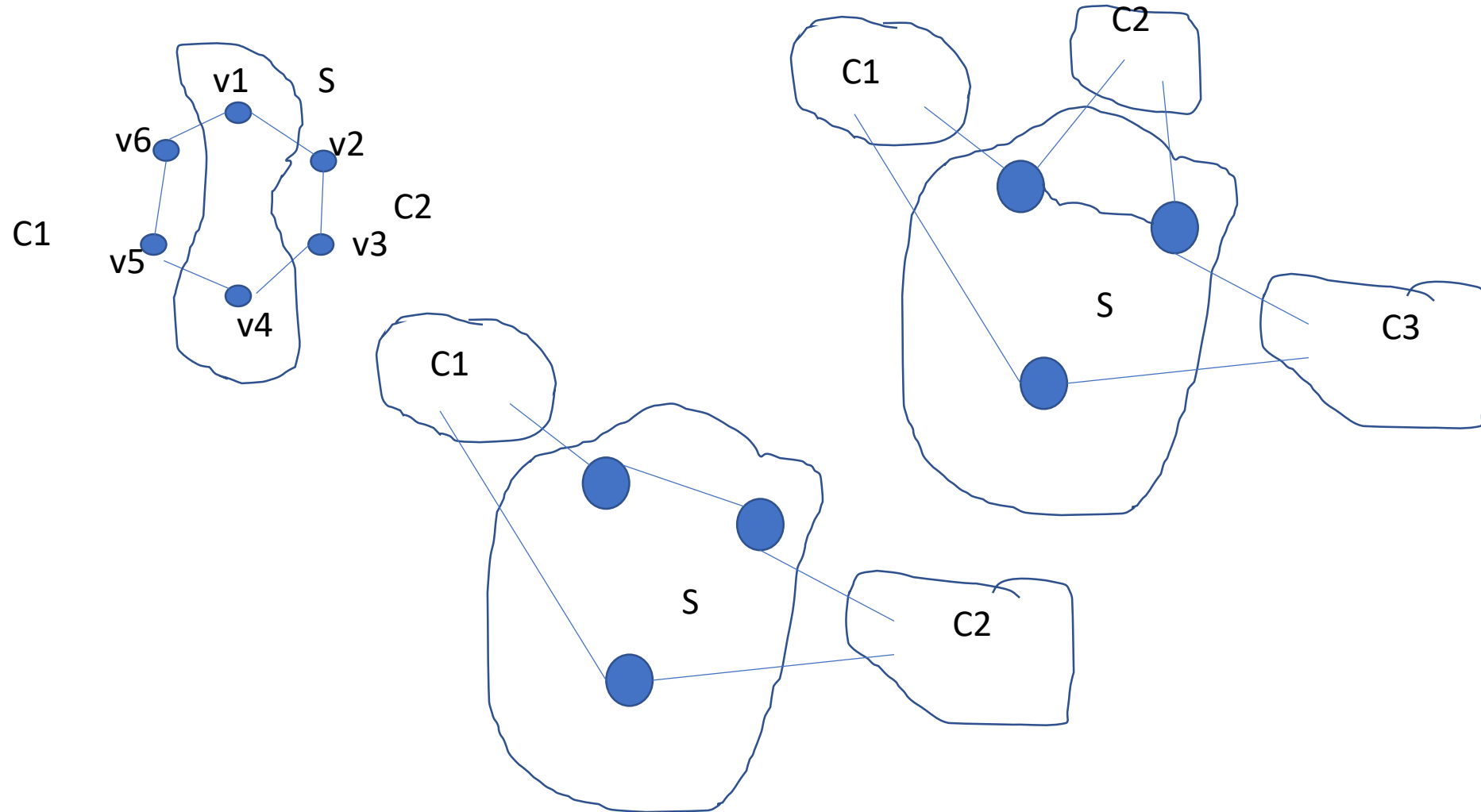
Each arrival in S must be distinct

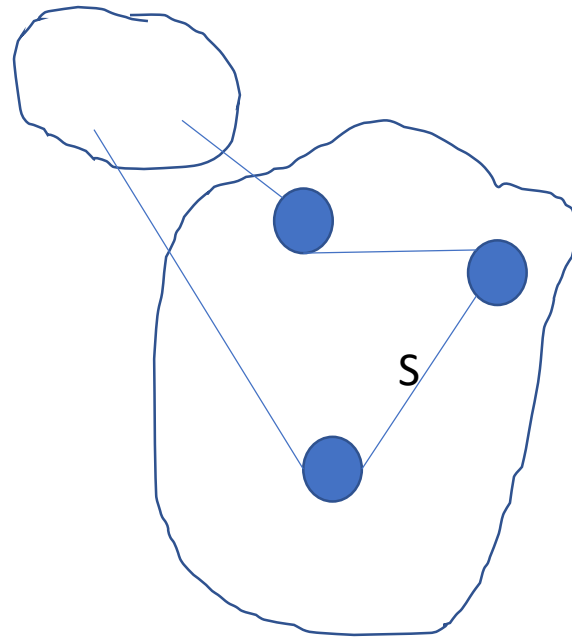
- Suppose $(G-S)$ has three components:
- C_1, C_2, C_3 .
- Each component needs its own arrival point in S .



- **Why can't C_1 and C_2 both arrive at the same vertex s in S ?**
- **Because in the Hamiltonian cycle, every vertex appears only once**
- **So vertex s cannot be the "arrival vertex" for two different components.**
- Therefore, if there are r components, we need **at least r distinct vertices** in S
- $r \leq |S|$.

Therefore, if there are (r) components, we need at least r distinct vertices in (S) :
 $|S| \geq r$.





Therefore, if there are (r) components, we need at least r distinct vertices in (S) :
 $|S| \geq r$.

Traveling Salesman Problem (TSP)

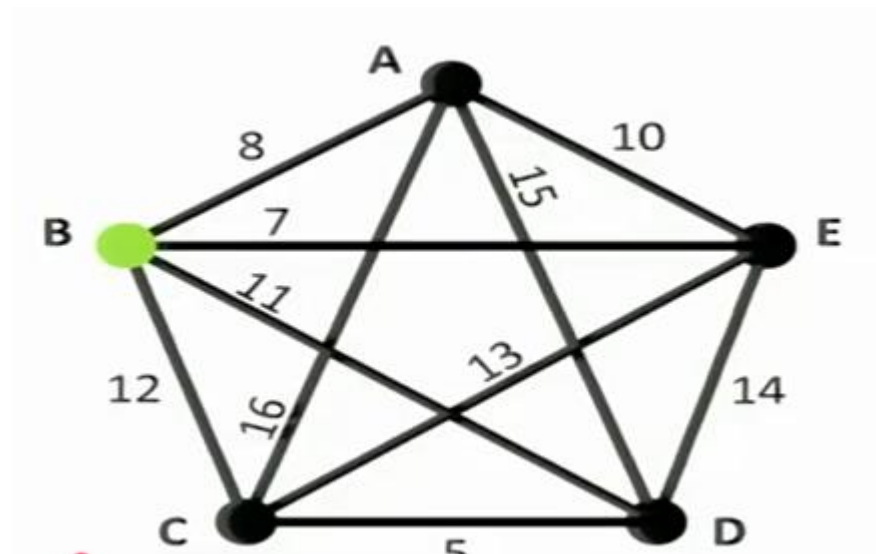
- A salesman plans to visit $n-1$ other cities and return home. The natural objective is to minimize the total travel time.
- If we assign **each edge of K_n** a weight equal to the **travel time** between the corresponding cities,
 - then we seek the **spanning cycle of minimum total weight**.
- This is the famous Traveling Salesman Problem (TSP). Seemingly analogous to the Minimum Spanning Tree problem, the **TSP as yet has no good algorithm.**

Nearest-Neighbor Algorithm (upper bound)

- 1. From the **starting vertex**, choose the **edge** with the **smallest cost** and use that as the first edge in your cycle.
- 2. **Continue in this manner**, choosing among the **edges** that connect from the **current vertex** to vertices you have **not yet visited**.
- 3. When you have **visited every vertex**, return to the **starting vertex**.

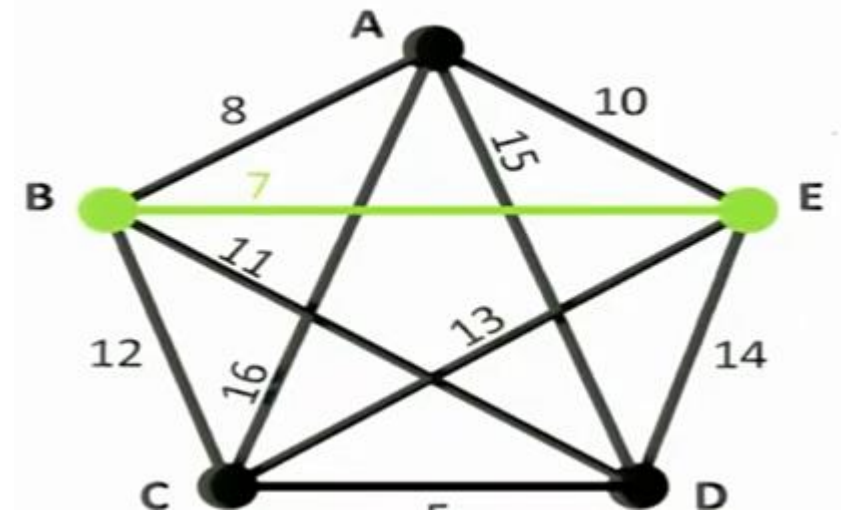
Nearest-Neighbor Algorithm

- Nearest-Neighbor Algorithm for TSP
- We have to have a starting point B
- We will choose our second vertex by finding the "nearest neighbor"



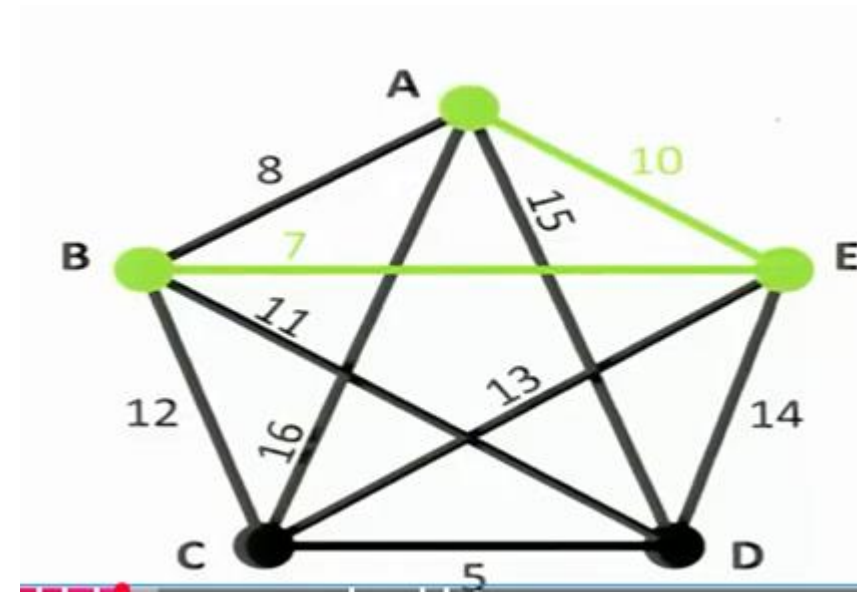
Nearest-Neighbor Algorithm

- Choose the cheapest edge
- In this case, we go from B to E (7)



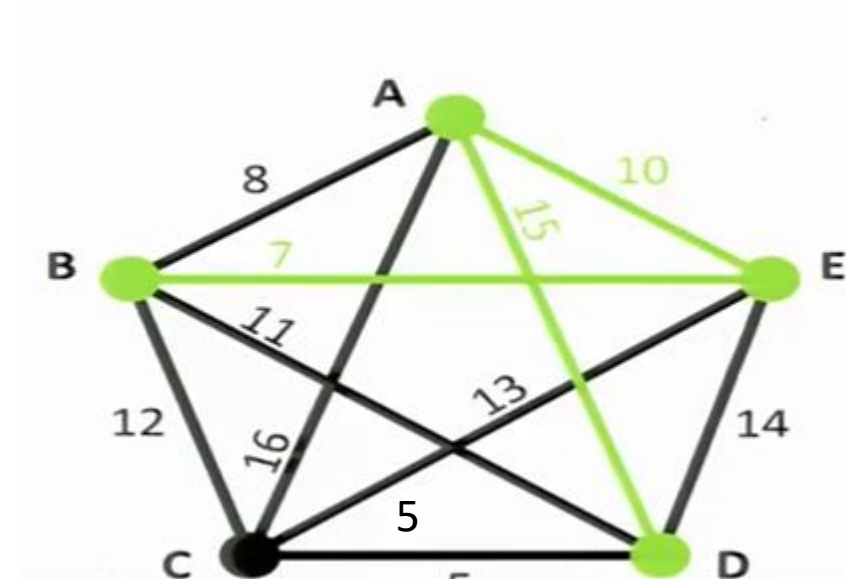
Nearest-Neighbor Algorithm

- Example: Nearest-Neighbor Algorithm for TSP
- Now where do we go?
- We can't go back to B
- Again choose the cheapest edge



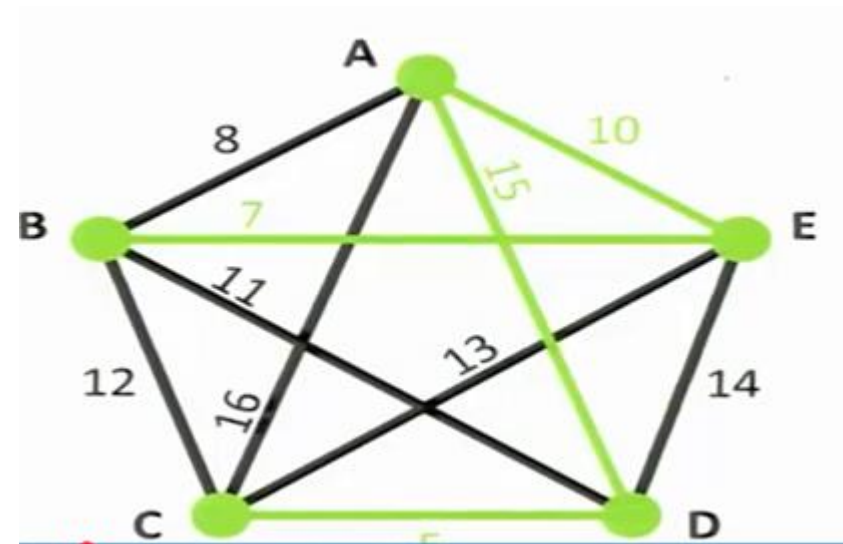
Nearest-Neighbor Algorithm

- Example: Nearest-Neighbor Algorithm for TSP
- The rule is "nearest neighbor": **always choose the lowest cost edge, unless that would take you back to a vertex you have already visited**



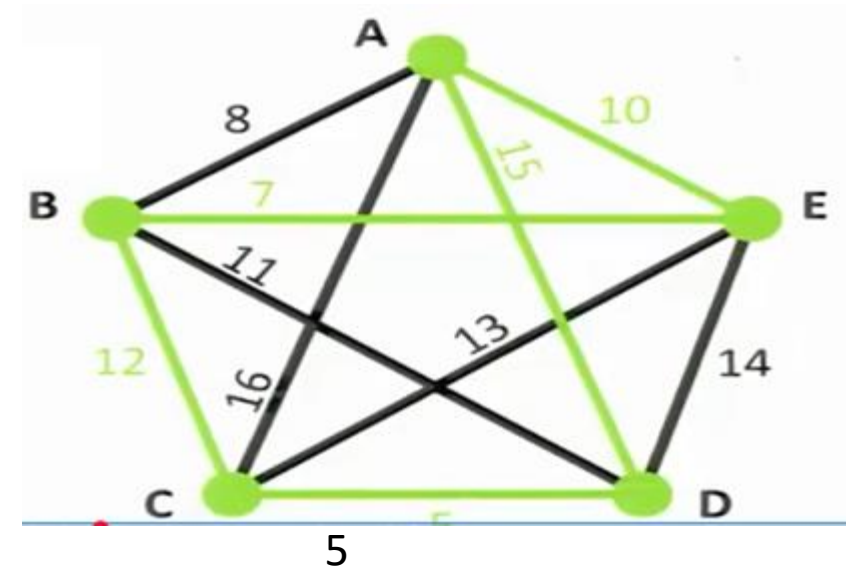
Nearest-Neighbor Algorithm

- Now we only have one choice
- We can't go back to A or E, and we can't return to B because that would leave out C
- So we must go to C



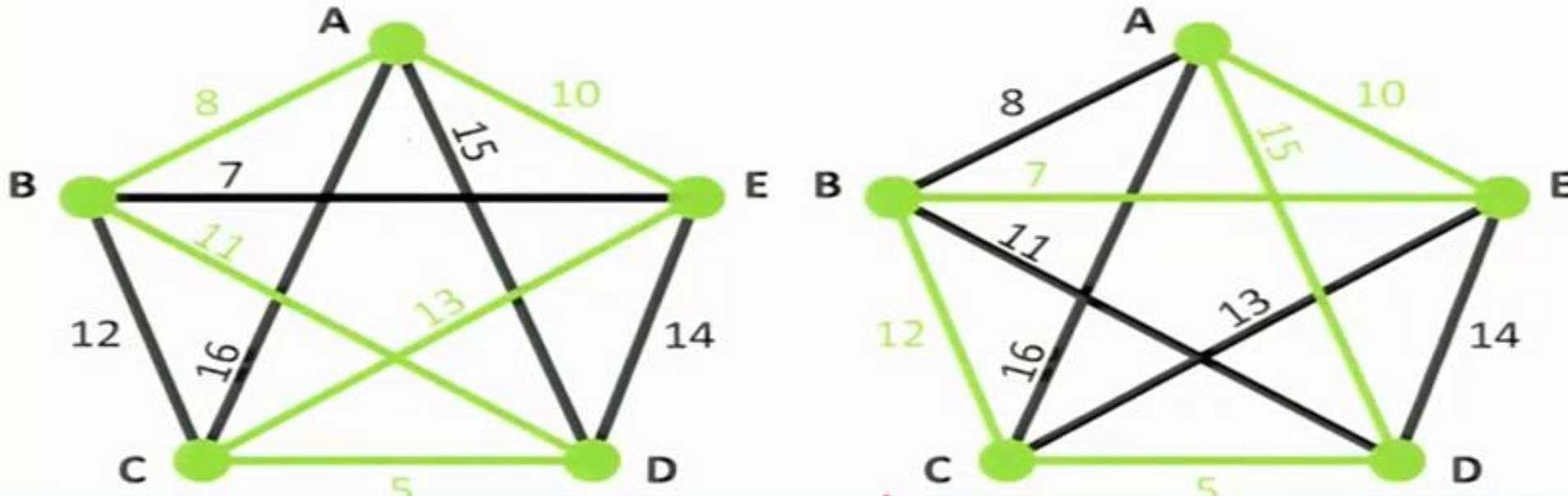
Nearest-Neighbor Algorithm

- Example: Nearest-Neighbor Algorithm for TSP
- We have now visited all of the vertices, so we finally return to B
- This cycle has a total cost of 49
- Is it the best cycle?



Nearest-Neighbor Algorithm (Upper bound)

- It is *not* the best! The solution on the left has a total cost of 47



Nearest-Neighbor Algorithm

Starting Vertex	Path	Total Cost
A	A-B-E-C-D-A	48
B	B-E-A-D-C-B	49
C	C-D-B-E-A-C	49
D	D-C-B-E-A-D	49
E	E-B-A-D-C-E	48

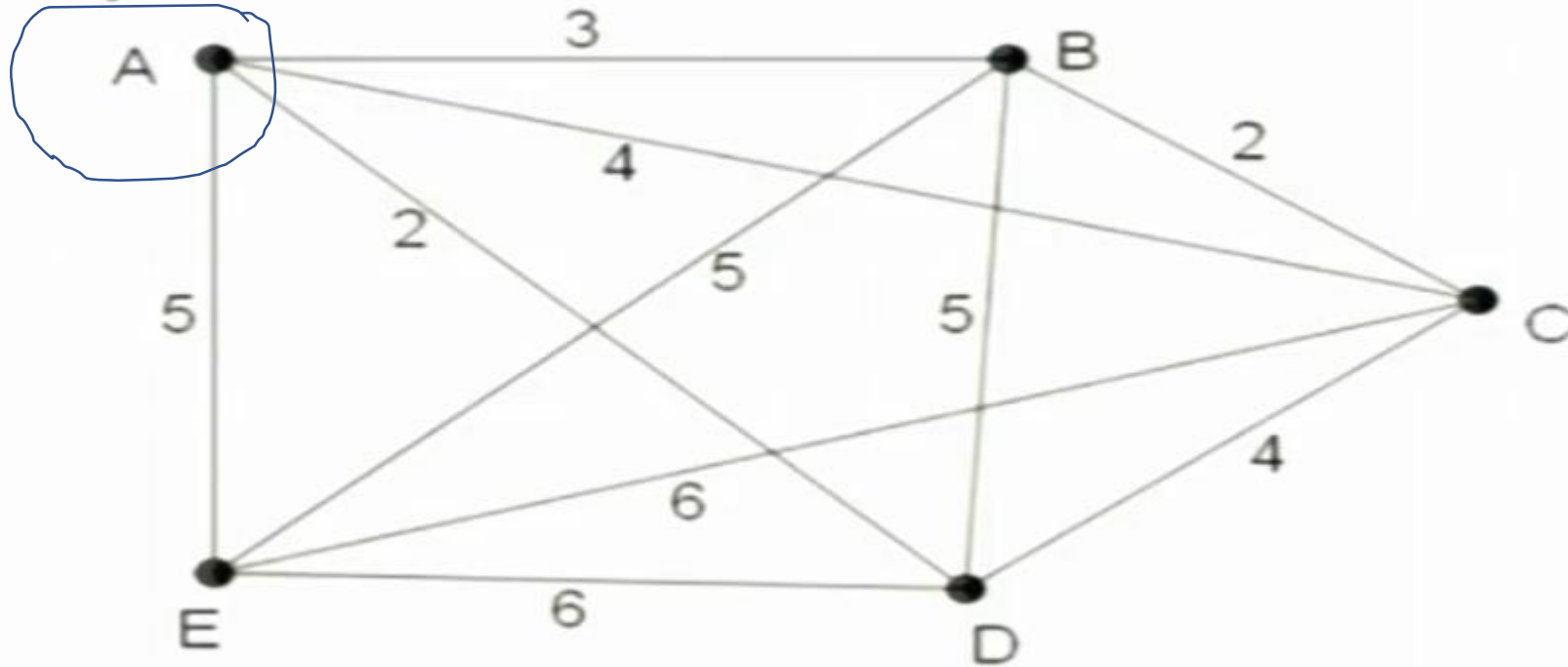
Travelling Salesman (Lower Bound)

Finding a **lower bound**

- To find a lower bound for the weight of the minimum Hamiltonian cycle
- Step 1: Choose an arbitrary node. Delete that node and all its edges.
- Step 2: Find the length of the minimum connector for the remaining arcs (minimum spanning tree) --- Prim's or Kruskal's Algo
- Add the **weights of the two least weight edges** from the **deleted node** to the weight of the minimum spanning tree.

Finding a lower bound

Example



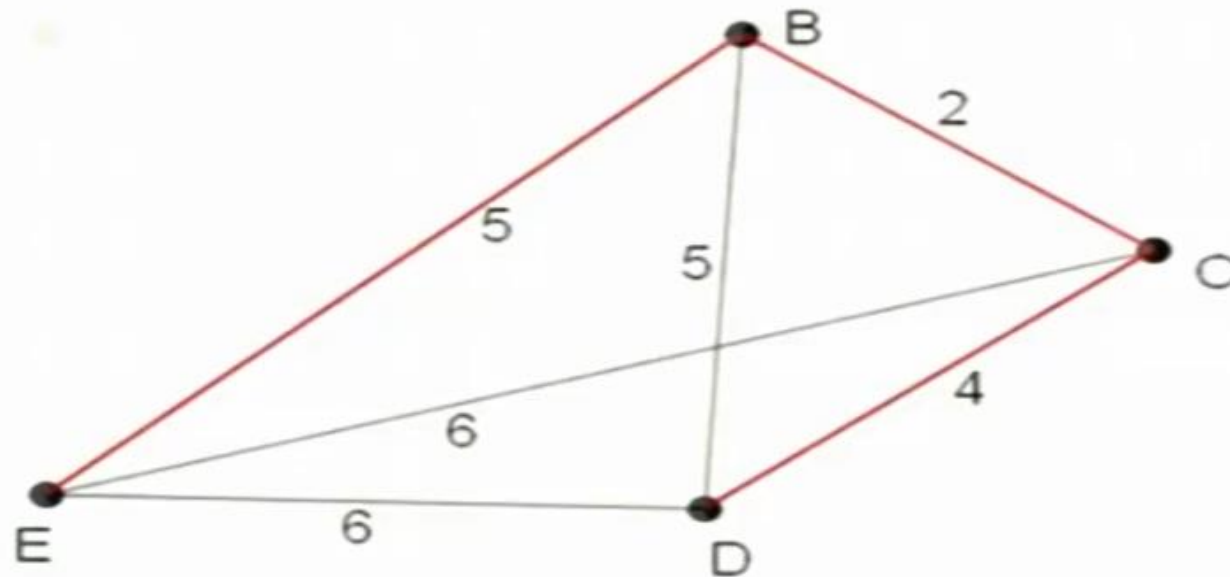
First, delete A and its arcs.

Finding a lower bound

Finding a lower bound

Weight of minimum connector = 11

Example

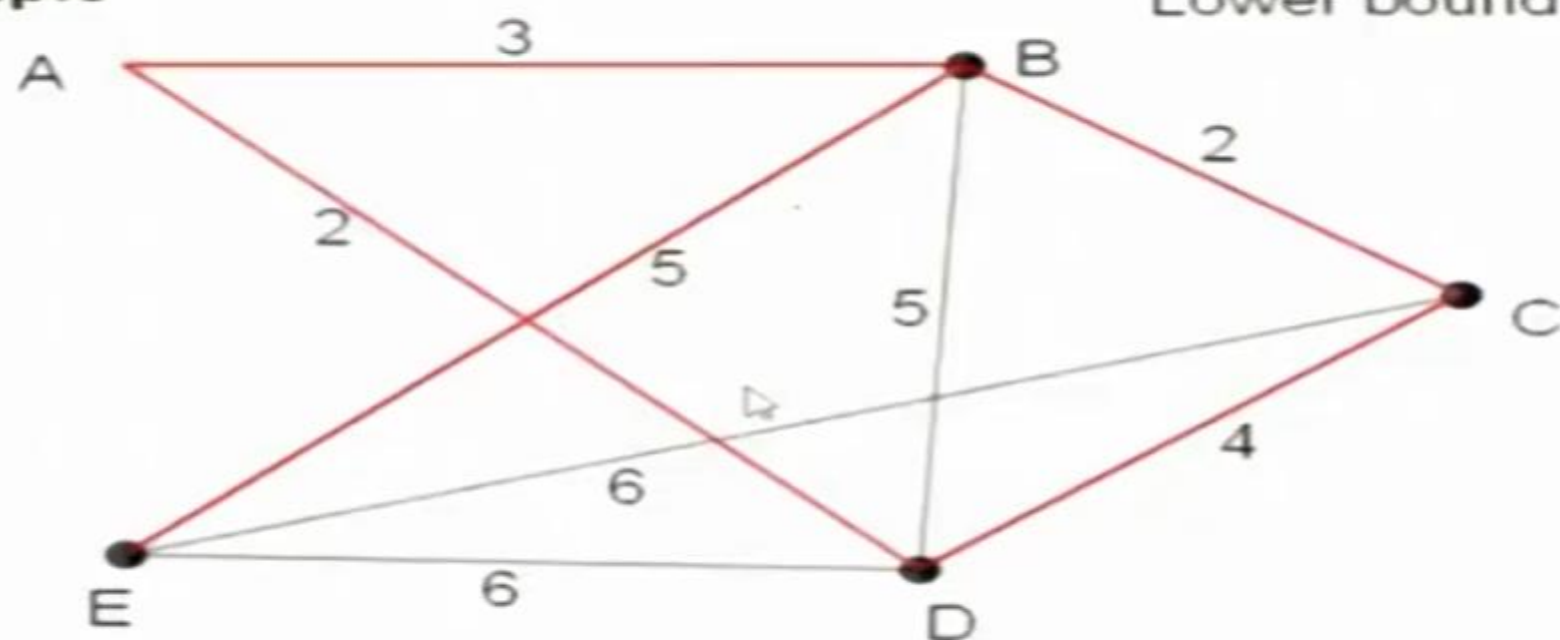


First, delete A and its arcs.

Find the weight of the minimum connector for the remaining network.

Finding a lower bound

Example



Weight of minimum connector = 11

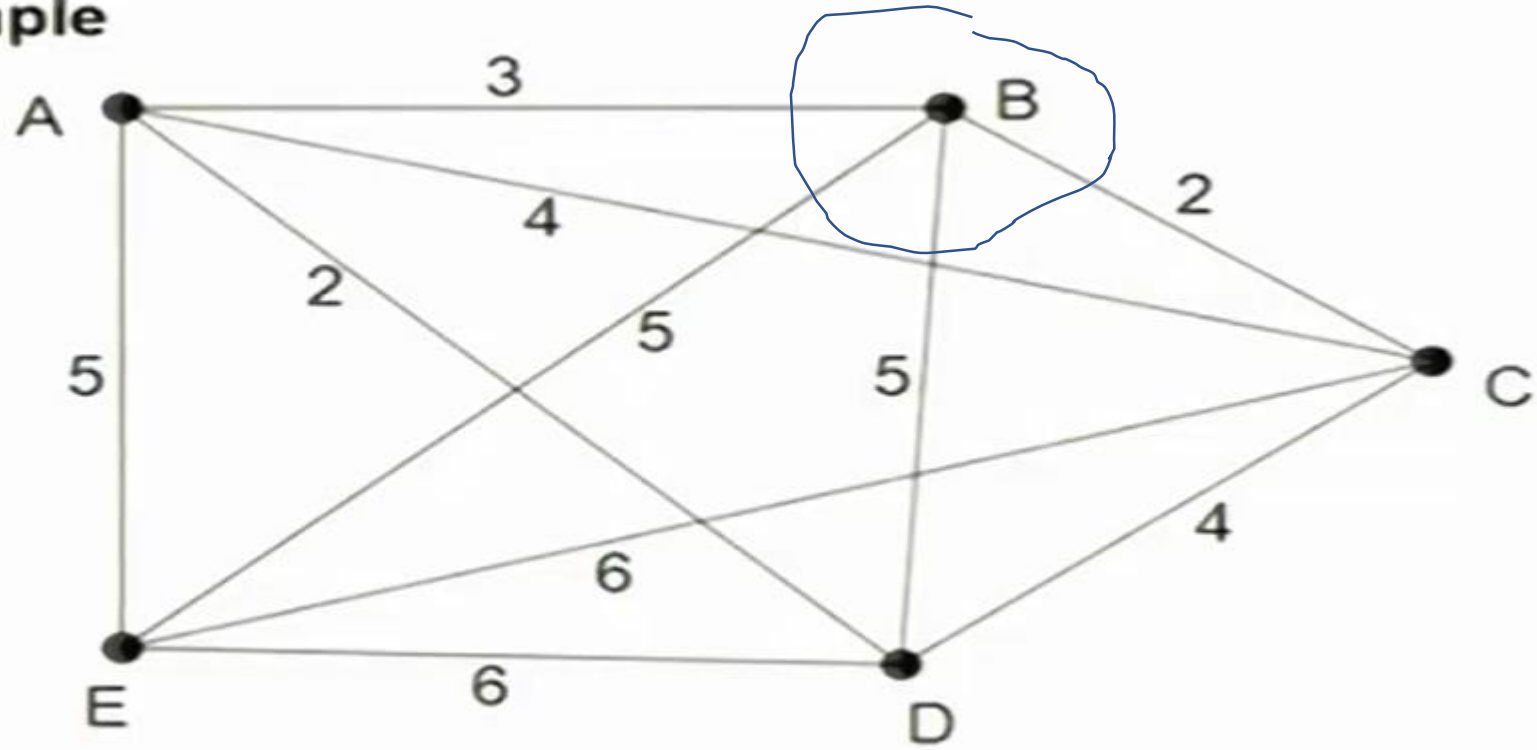
Lower bound = $11 + 2 + 3 = 16$

First, delete A and its arcs.

Find the weight of the minimum connector for the remaining network.

Add the two least weight arcs from A to give a lower bound.

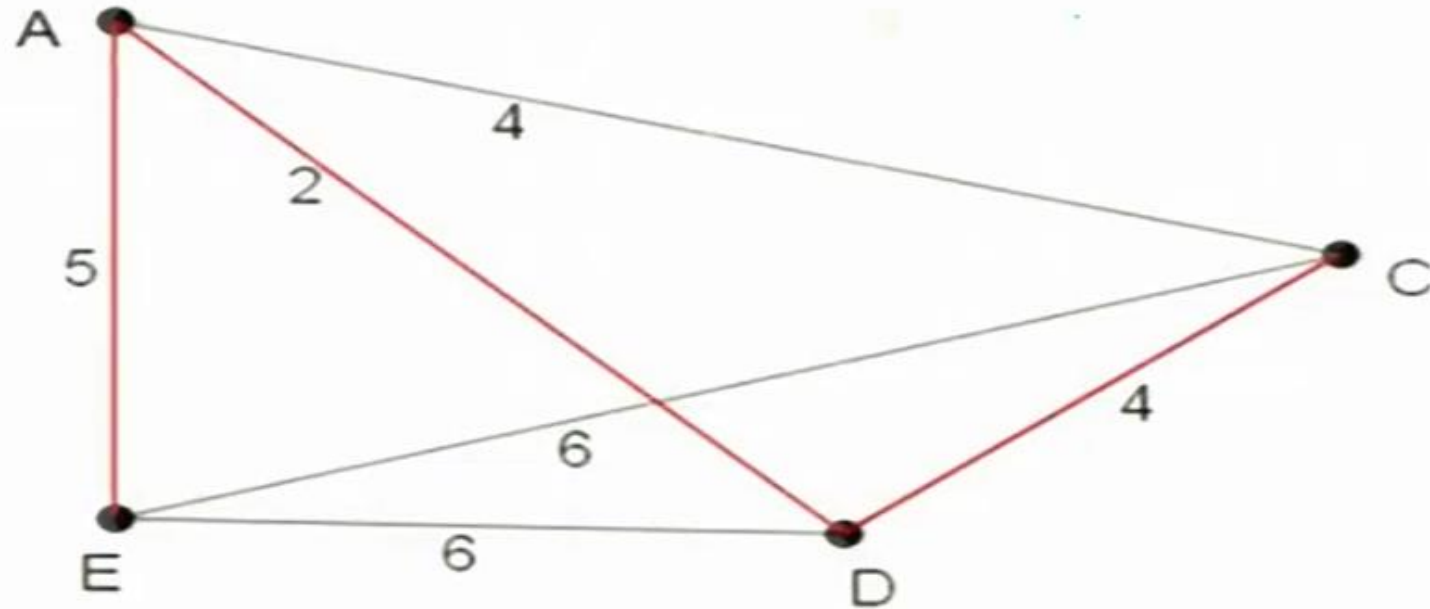
Example



Now, delete B and its arcs.

Weight of minimum connector = 11

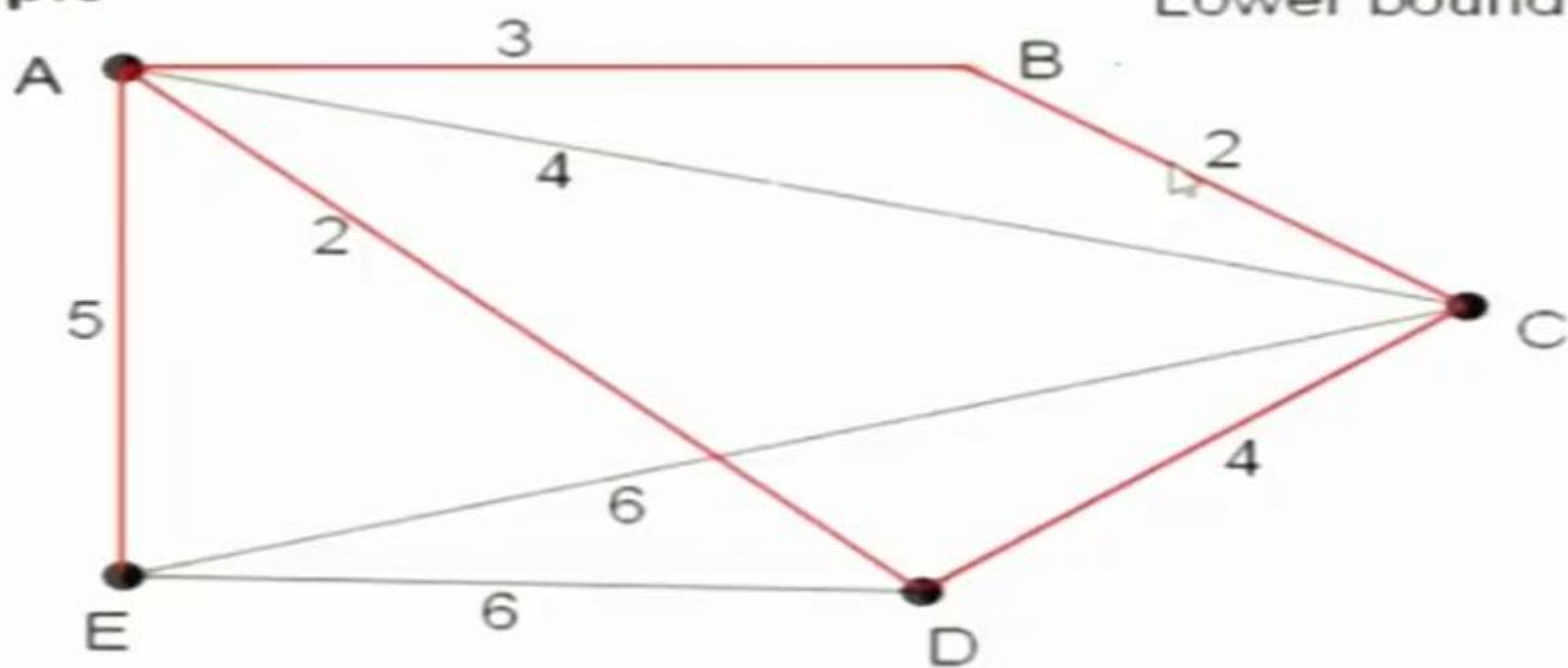
Example



Now, delete B and its arcs.

Find the weight of the minimum connector for the remaining network.

Example



Weight of minimum connector = 11

Lower bound = $11 + 2 + 3 = 16$

Now, delete B and its arcs.

Find the weight of the minimum connector for the remaining network.

Add the two least weight arcs from B to give a lower bound.

Vertex deleted	Lower bound
A	16
B	16
C	16
D	16
E	17

The greatest lower bound is 17, by deleting vertex E.

So the lower bound for the travelling salesperson problem is 17.