EEL 4837
Programming for Electrical Engineers II

Ivan Ruchkin
Assistant Professor
Department of Electrical and Computer Engineering
University of Florida at Gainesville
iruchkin@ece.ufl.edu
http://ivan.ece.ufl.edu

Spanning Tree Problem

Readings:

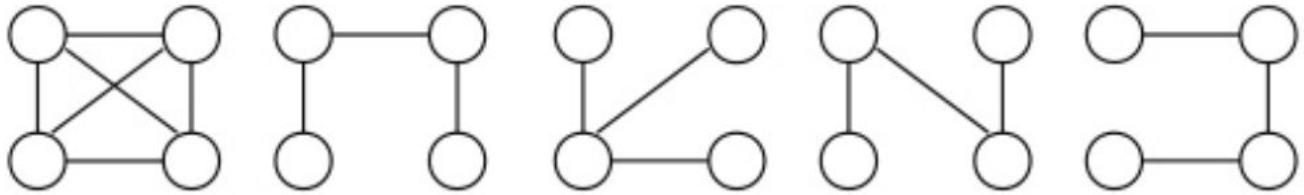
- Weiss 9.5
- Horowitz 4.5
- Cormen 23

Spanning Tree

Suppose you have a connected undirected graph

- Connected: every node is reachable from every other node
- Undirected: edges do not have an associated direction

...then a spanning tree of the graph is a connected subgraph in which there are no cycles



A connected, undirected graph

Four of the spanning trees of the graph

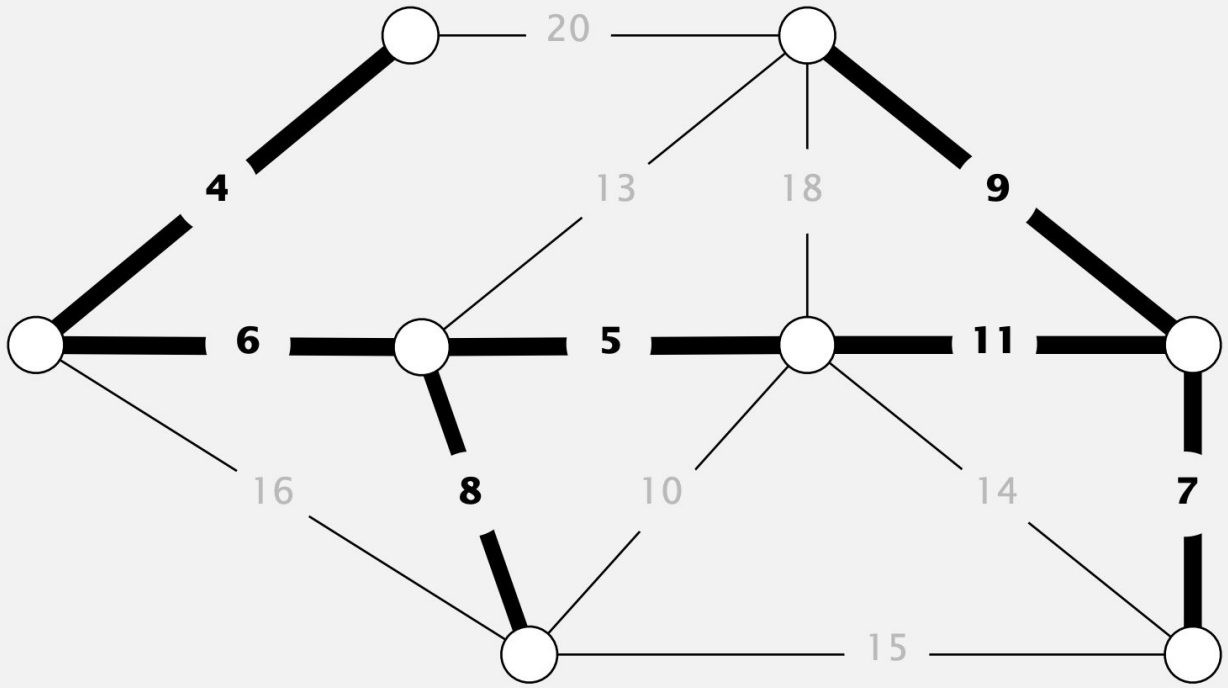
A (non-rooted) tree:
a connected acyclical graph

Minimum Spanning Tree Problem

Input. Connected, undirected graph G with positive edge weights.

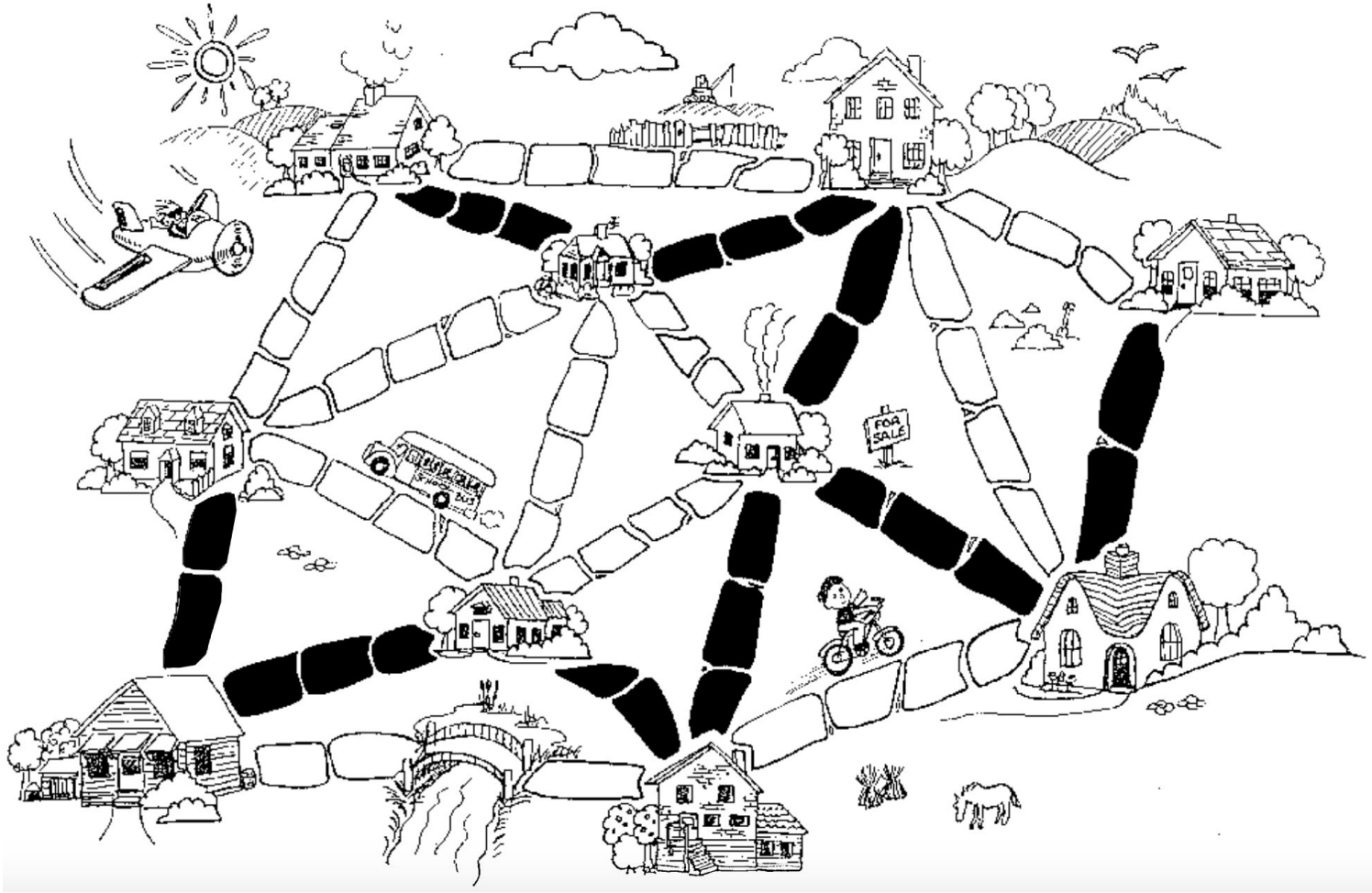
Output. A spanning tree of minimum weight.

Brute force: Try all spanning trees and find the minimum



minimum spanning tree T
(weight = 50 = 4 + 6 + 8 + 5 + 11 + 9 + 7)

MST Applications: Network Design



Problem: determine which network links to build based on their length/difficulty/cost

Properties of Spanning Trees

Let T be a spanning tree of a connected graph G with V nodes.
Which of the following is true?

A. T contains exactly V – 1 edges

B. Removing any edge from T makes T disconnected

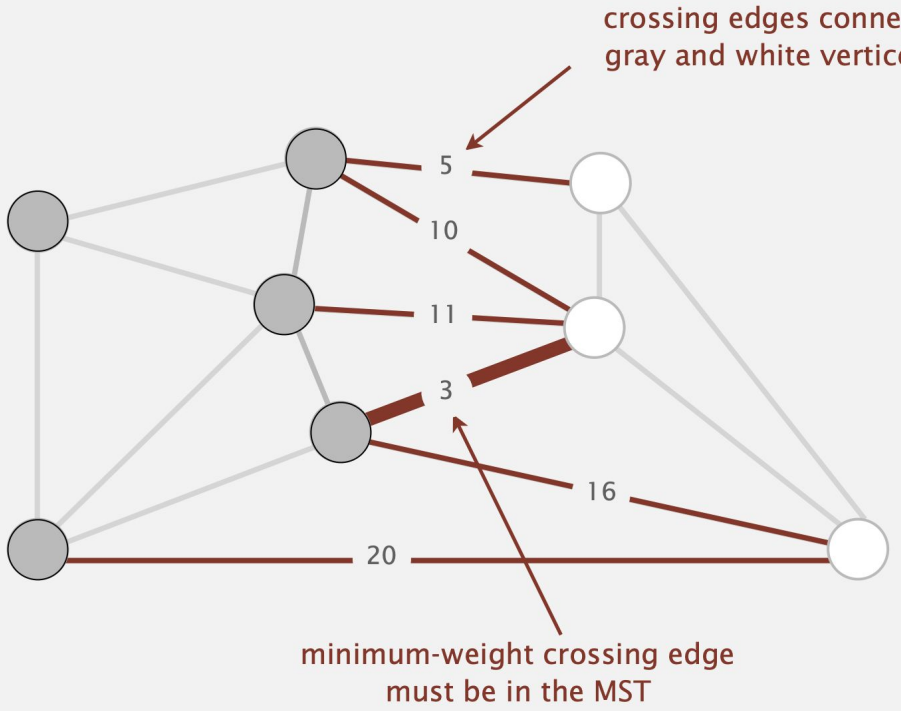
C. Adding any edge to T creates a cycle

Cut Property

Def. A cut in a graph is a partition of its vertices into two (nonempty) sets.

Def. A crossing edge connects a vertex in one set with a vertex in the other.

Cut property. Given any cut, the crossing edge of min weight is in the MST.



Cut Property

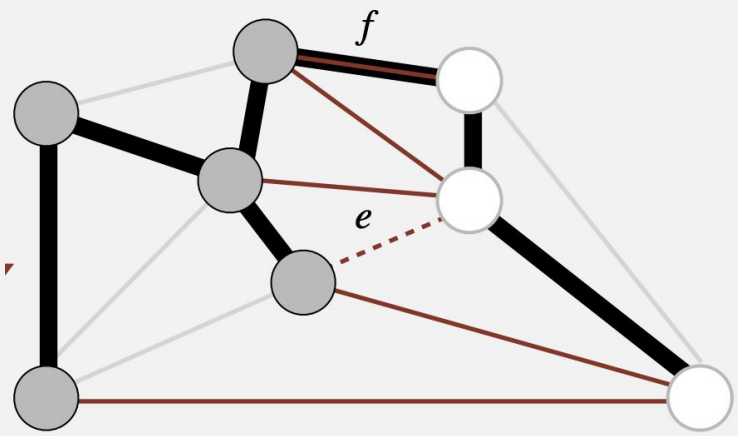
Def. A cut in a graph is a partition of its vertices into two (nonempty) sets.

Def. A crossing edge connects a vertex in one set with a vertex in the other.

Cut property. Given any cut, the crossing edge of min weight is in the MST.

Pf. [by contradiction] Suppose e is not in the MST.

- Some other edge f in the MST must be a crossing edge.
- Removing f and adding e is also a spanning tree.
- Since weight of e is less than the weight of f, that spanning tree has lower weight.
- Contradiction. ▪



Minimum Spanning Tree Algorithms

- There are two basic algorithms for finding minimum-cost spanning trees, and both are greedy algorithms

- **Kruskal's algorithm:**

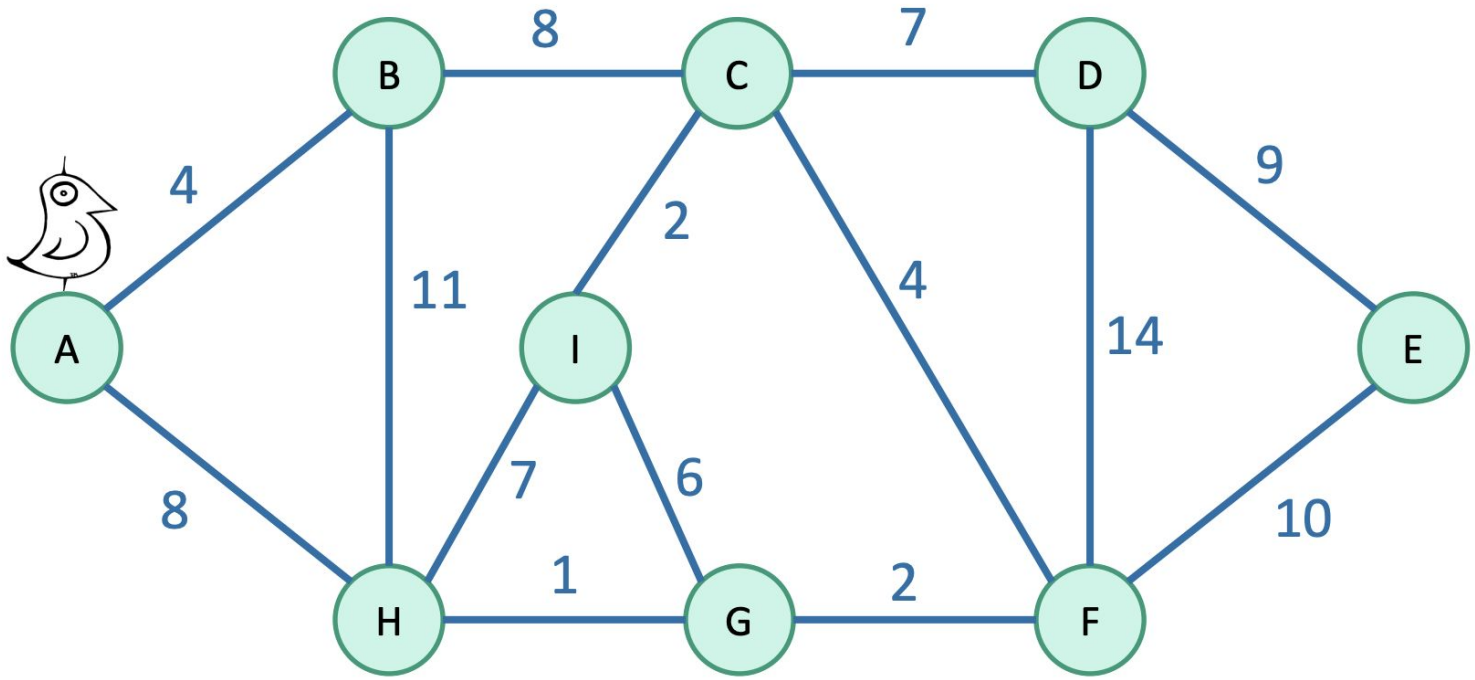
  Created in 1957 by Joseph Kruskal

- **Prim's algorithm**

  Created by Robert C. Prim

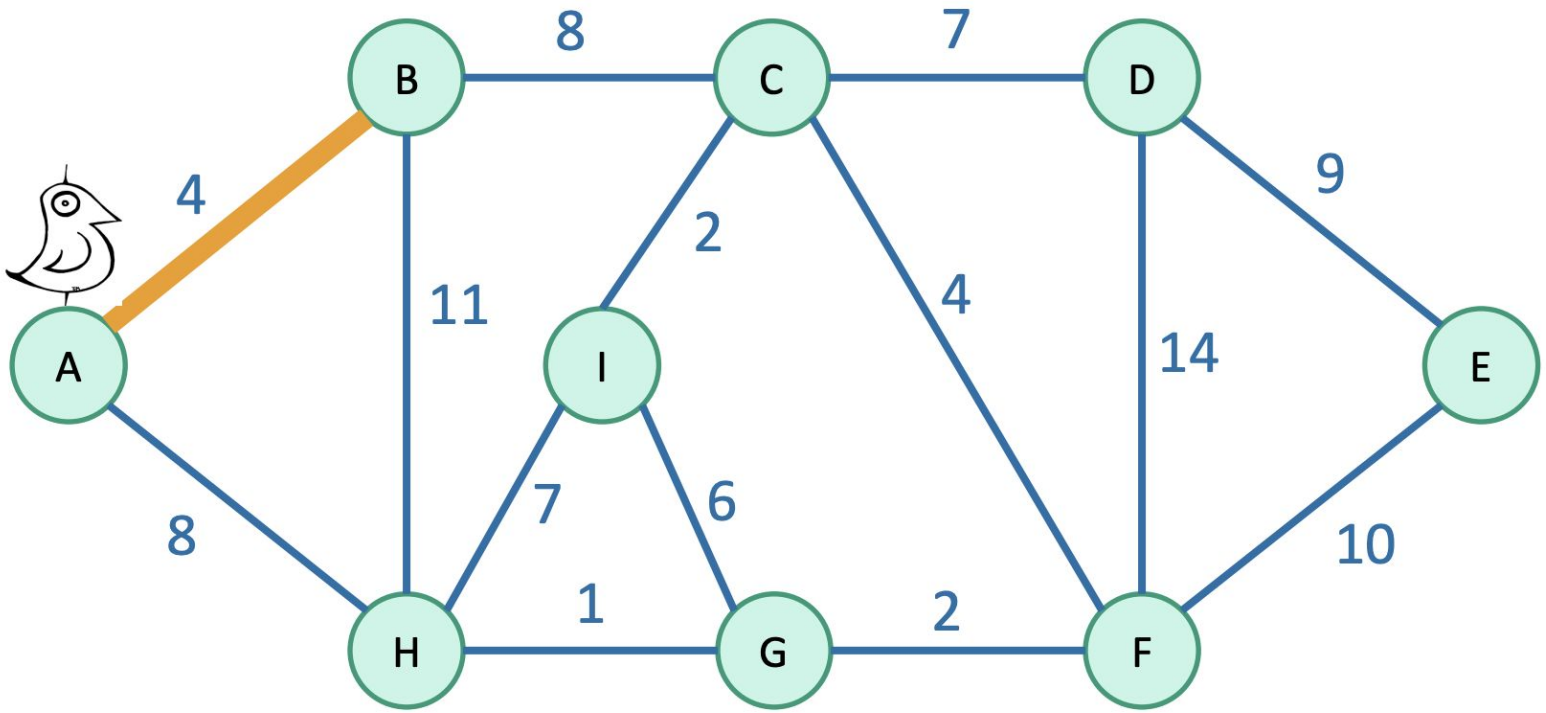
Prim's Algorithm Walkthrough (1)

Start growing a tree, greedily add the shortest edge we can to grow the tree.



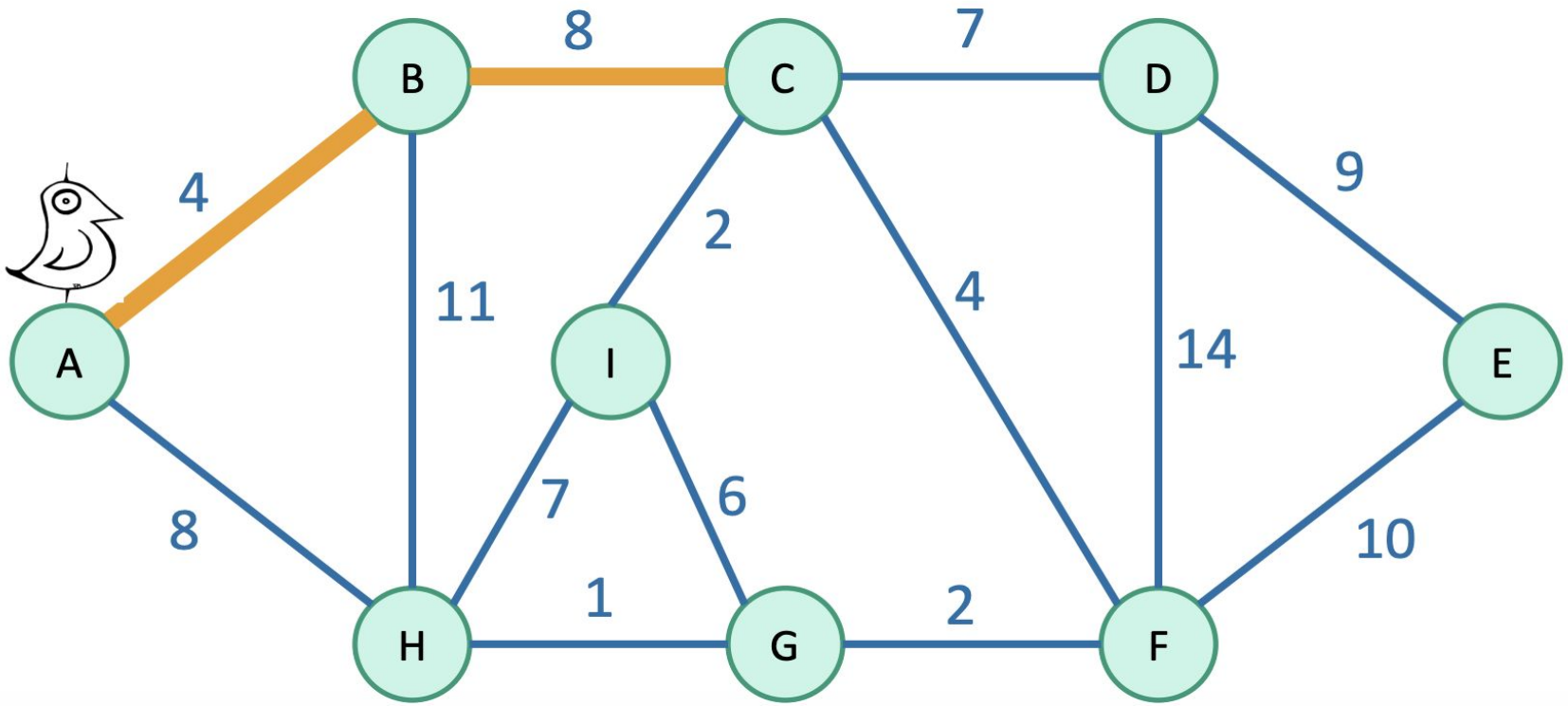
Prim's Algorithm Walkthrough (2)

Start growing a tree, greedily add the shortest edge we can to grow the tree.



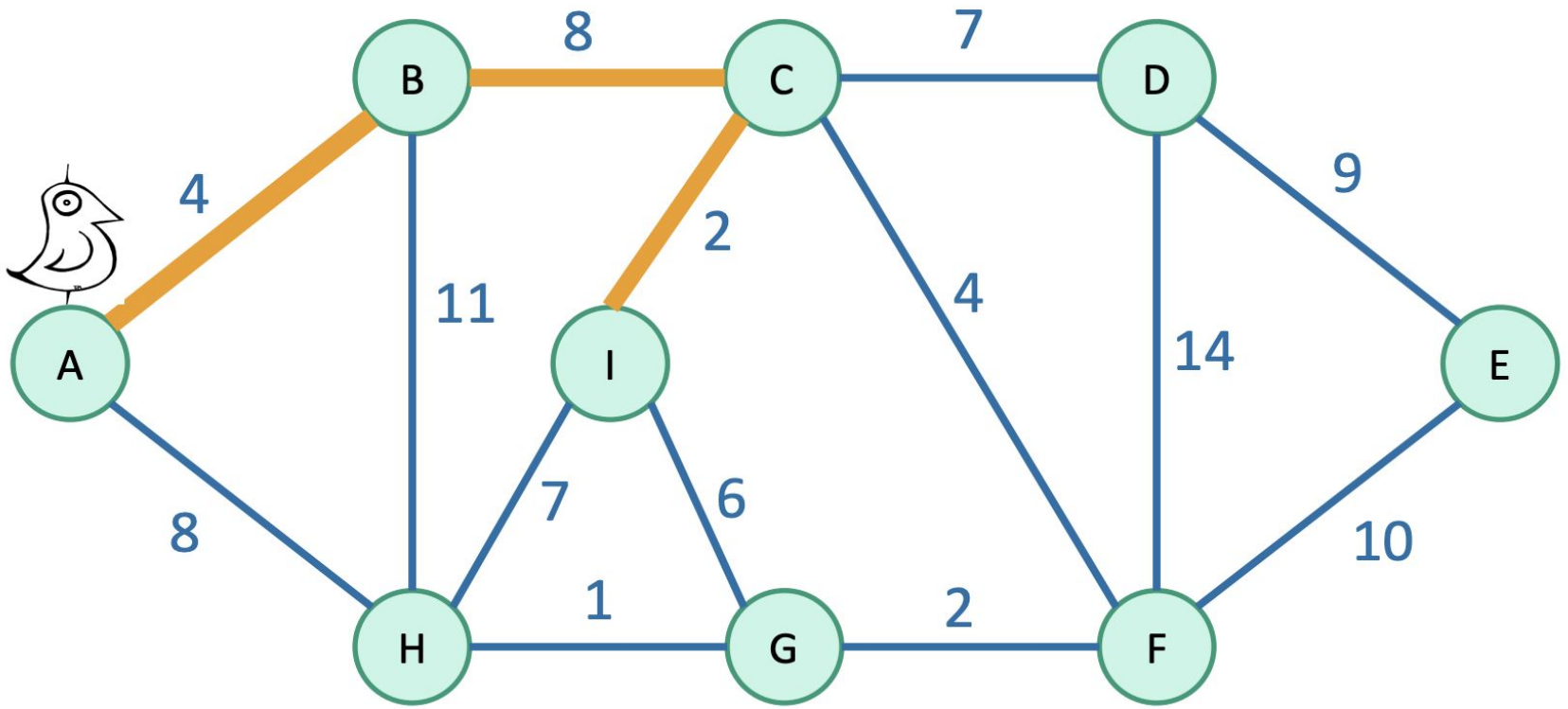
Prim's Algorithm Walkthrough (3)

Start growing a tree, greedily add the shortest edge we can to grow the tree.



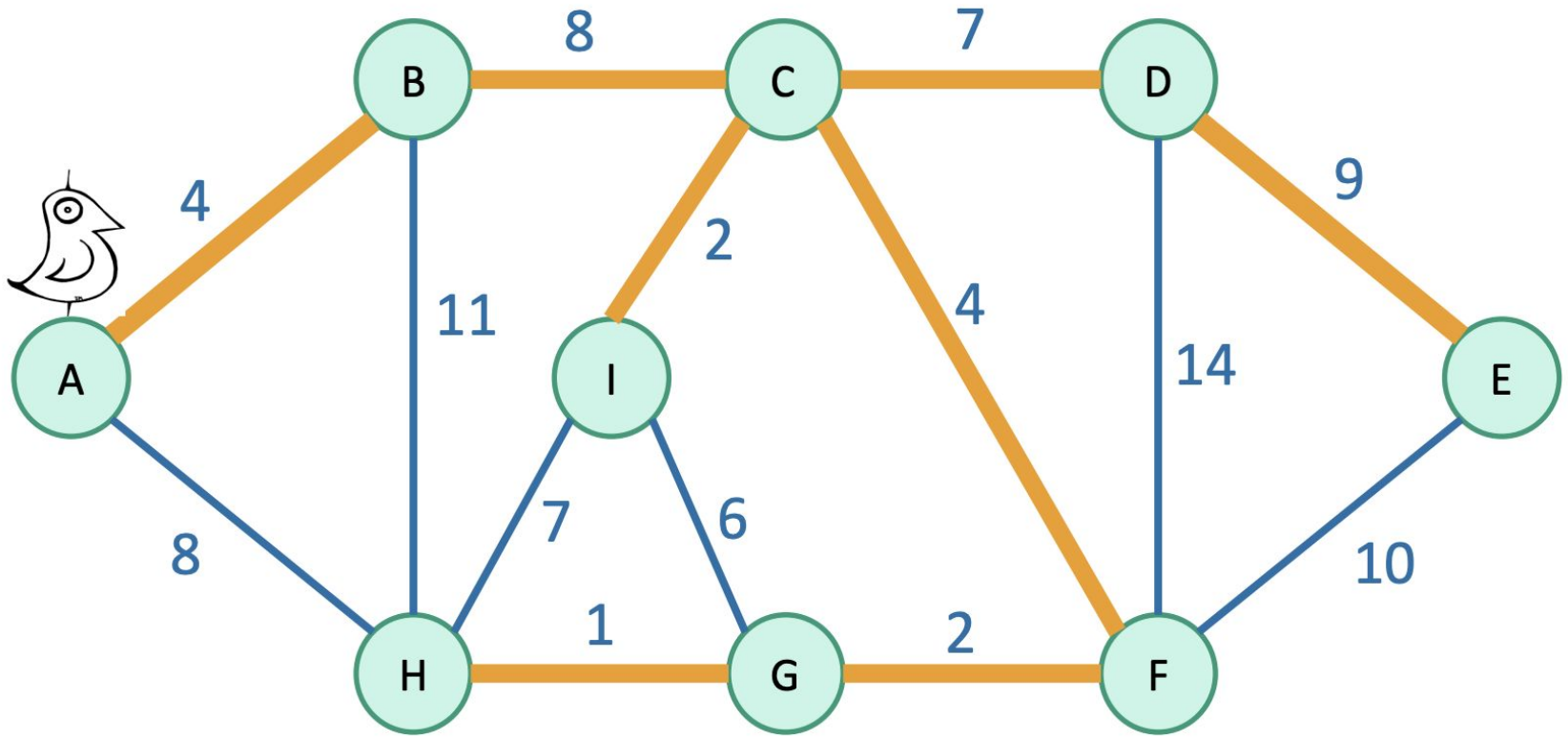
Prim's Algorithm Walkthrough (4)

Start growing a tree, greedily add the shortest edge we can to grow the tree.



Prim's Algorithm Walkthrough (5)

Start growing a tree, greedily add the shortest edge we can to grow the tree.



Prim's Algorithm: Pseudocode

- slowPrim(G = (V,E), starting vertex s):
  - Let (s,u) be the lightest edge coming out of s.
  - MST = { (s,u) }
  - verticesVisited = { s, u }
  - **while** |verticesVisited| < |V|:
    - find the lightest edge (x,v) in E so that:
      - x is in verticesVisited
      - v is not in verticesVisited
    - add (x,v) to MST
    - add v to verticesVisited
  - **return** MST

n iterations of this while loop.

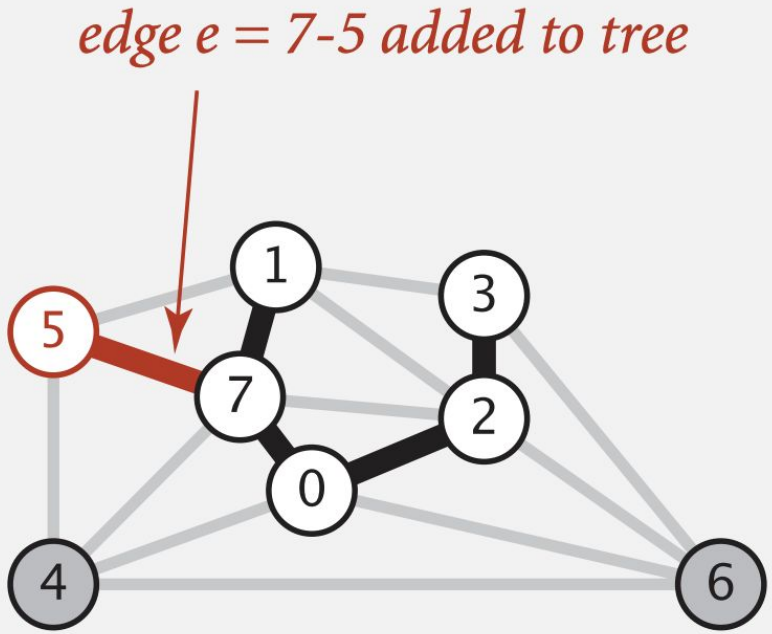
Maybe take time m to go through all the edges and find the lightest.

Question: What is the time complexity? (assume adjacency matrix for the graph)

Prim's Algorithm: Correctness

Proposition.

Prim's algorithm computes the MST.



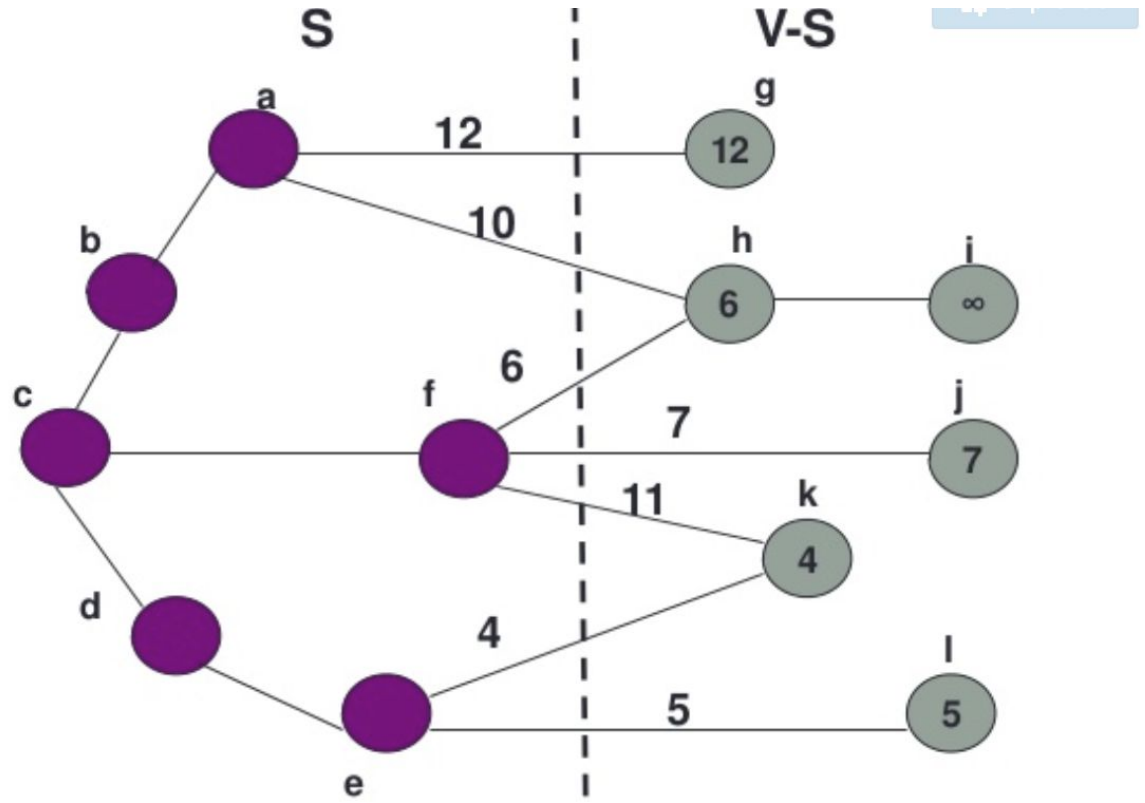
Pf.

- Cut = set of vertices in T.
- The edges crossing this cut are precisely those considered by Prim's algorithm (edges with exactly one endpoint in T).
- Cut property $\Rightarrow$ edge added by Prim's algorithm must be in the MST.

Optimizing Prim's with Priority Queue

Idea: use a **priority queue/ min-heap** to store unvisited vertices

- Use the **adjacency list** representation for easier edge traversal
- Add all the vertices into the heap with INF cost, draw any one & add to the MST
- Repeat:
  - For the adjacent vertices of the drawn still in the heap, update the min cost if needed
  - Draw the min-cost vertex from the heap and add to MST



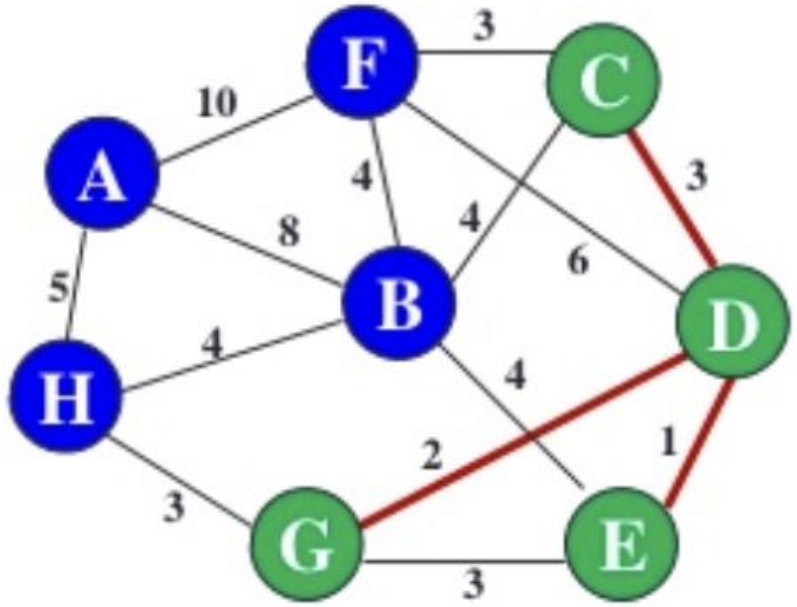
Priority Queue:[(k,4)|(l,5)|(h,6)|(g,12)|(i,∞)]

Question: What is the time complexity?

Kruskal's Algorithm

- Sort edges by increasing edge weight
- Select the first |V| – 1 edges that do not generate a cycle

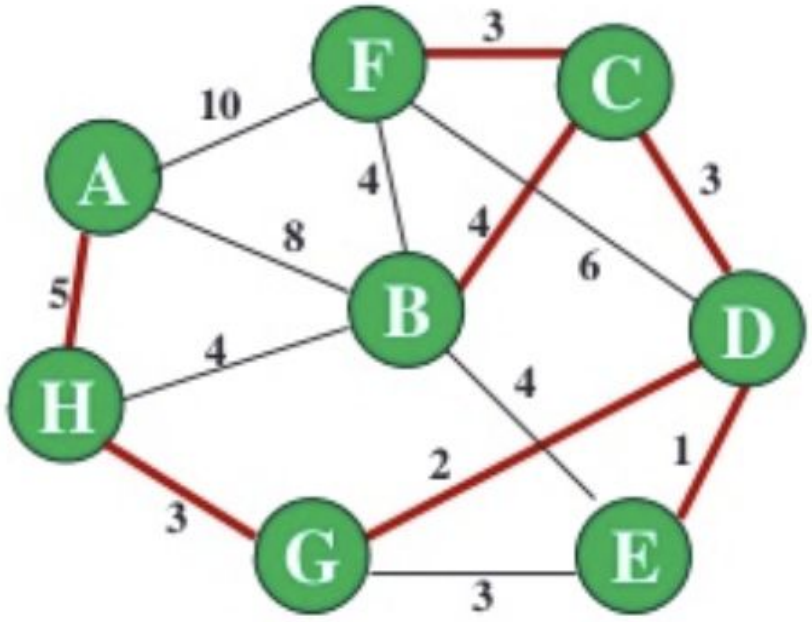
Kruskal's Algorithm Walkthrough (1)



edge	d_v	
(D,E)	1	✓
(D,G)	2	✓
(E,G)	3	χ
(C,D)	3	✓
(G,H)	3	
(C,F)	3	
(B,C)	4	

edge	d_v	
(B,E)	4	
(B,F)	4	
(B,H)	4	
(A,H)	5	
(D,F)	6	
(A,B)	8	
(A,F)	10	

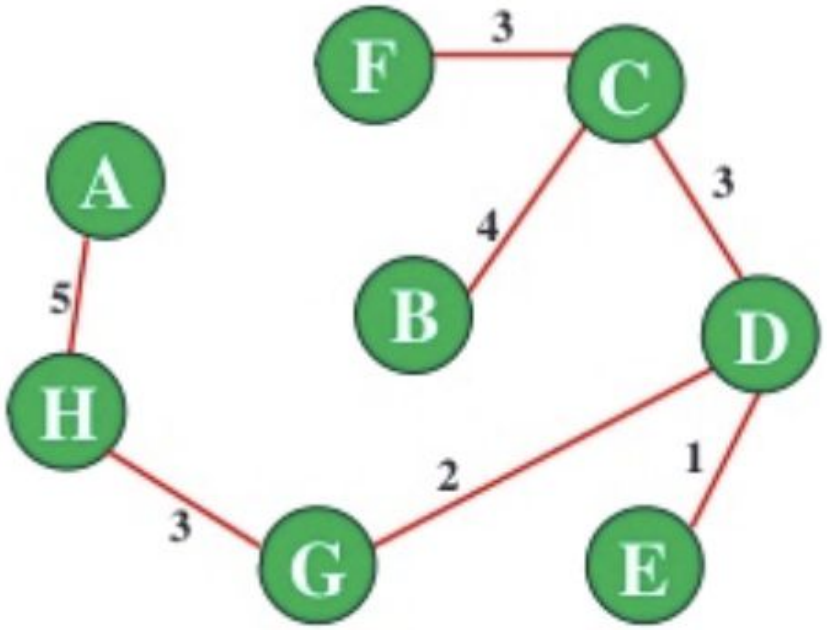
Kruskal's Algorithm Walkthrough (2)



edge	d_v	
(D,E)	1	✓
(D,G)	2	✓
(E,G)	3	χ
(C,D)	3	✓
(G,H)	3	✓
(C,F)	3	✓
(B,C)	4	✓

edge	d_v	
(B,E)	4	χ
(B,F)	4	χ
(B,H)	4	χ
(A,H)	5	✓
(D,F)	6	χ
(A,B)	8	χ
(A,F)	10	χ

Kruskal's Algorithm Walkthrough (3)



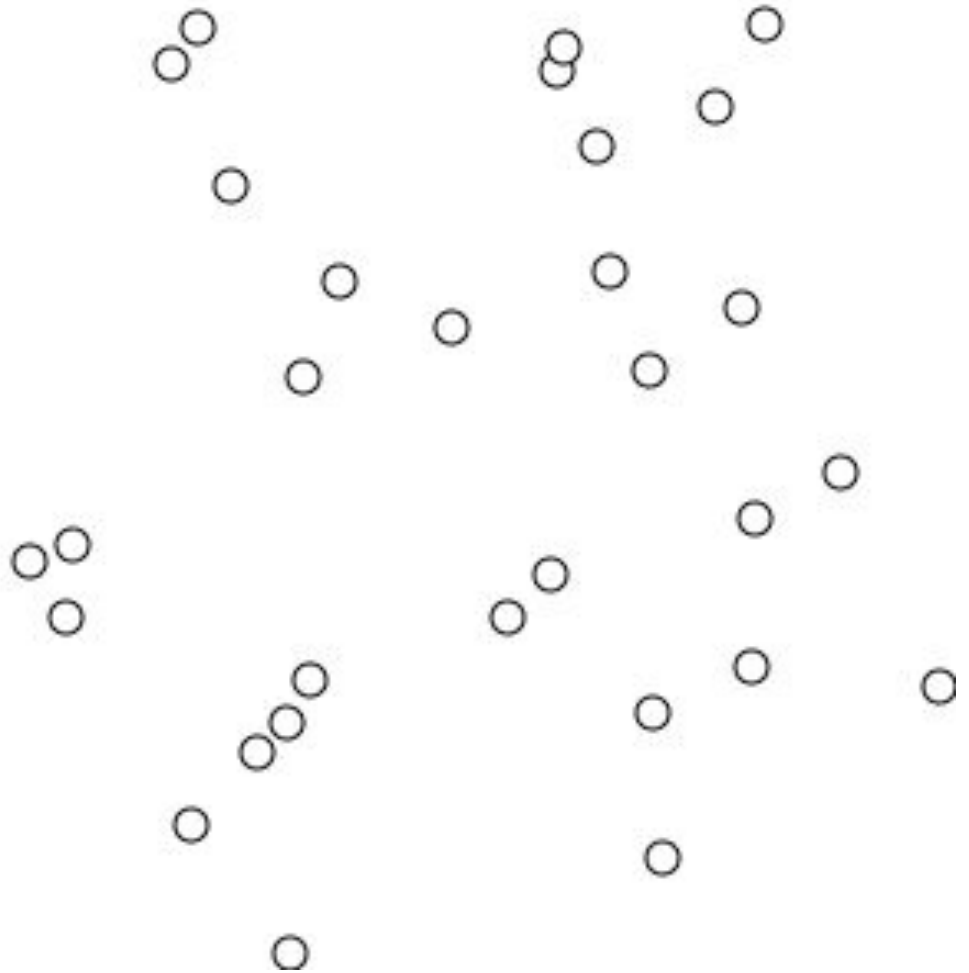
edge	d_v	
(D,E)	1	√
(D,G)	2	√
(E,G)	3	χ
(C,D)	3	√
(G,H)	3	√
(C,F)	3	√
(B,C)	4	√

edge	d_v	
(B,E)	4	χ
(B,F)	4	χ
(B,H)	4	χ
(A,H)	5	√
(D,F)	6	χ
(A,B)	8	χ
(A,F)	10	χ

Done!

Total Cost = $\Sigma\, d_v = 21$

Kruskal's Algorithm Animation



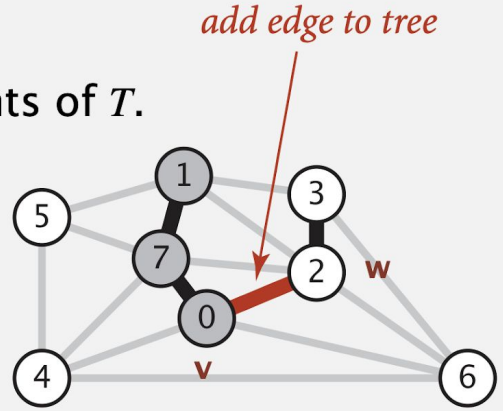
Correctness of Kruskal's Algorithm

Proposition. Kruskal's algorithm computes the MST. Recall: increasing order of edge weights

Pf. Let T be the "tree" at some point during execution, and e the next edge considered.

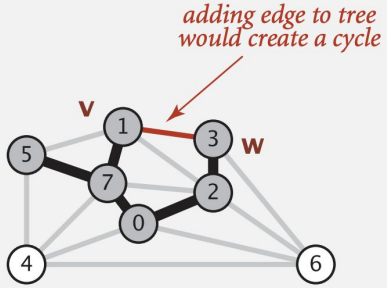
[Case 1] Kruskal's algorithm adds edge $e = v$–w to T.

- Vertices v and w are in different connected components of T.
- Cut = set of vertices connected to v in T.
- By construction of cut, no edge crossing cut is in T.
- No edge crossing cut has lower weight. Why?
- Cut property ⇒ edge e is in the MST.
- ⇒ Kruskal's algorithm correctly adds e to T.



[Case 2] Kruskal's algorithm discards edge $e = v$–w.

- From Case 1, all edges in T are in the MST.
- The MST can't contain a cycle.
- ⇒ Kruskal's algorithm correctly discards e.



Kruskal's Algorithm: Implementation

Will adding an edge e to the current set create a cycle?

- If not, add e.
- Otherwise discard.

Sorting edges: sort a list/array of edges & their costs; or build a heap
Implementing sets: boolean array with edge IDs for indices
Checking for cycles: use DFS to find an edge connecting back to visited vertices
More *advanced implementation* uses the **disjoint-set data structure**

Time complexity: O(E log V), dominated by the sorting/heap

Prim’s vs Kruskal’s: Two Greedy Algorithms

Prim’s Algorithm	Kruskal’s Algorithm
The tree that we are making or growing always remains connected.	The tree that we are making or growing usually remains disconnected.
Prim’s Algorithm grows a solution from a random vertex by adding the next cheapest vertex to the existing tree.	Kruskal’s Algorithm grows a solution from the cheapest edge by adding the next cheapest edge to the existing tree / forest.
Prim’s Algorithm is faster for dense graphs.	Kruskal’s Algorithm is faster for sparse graphs.