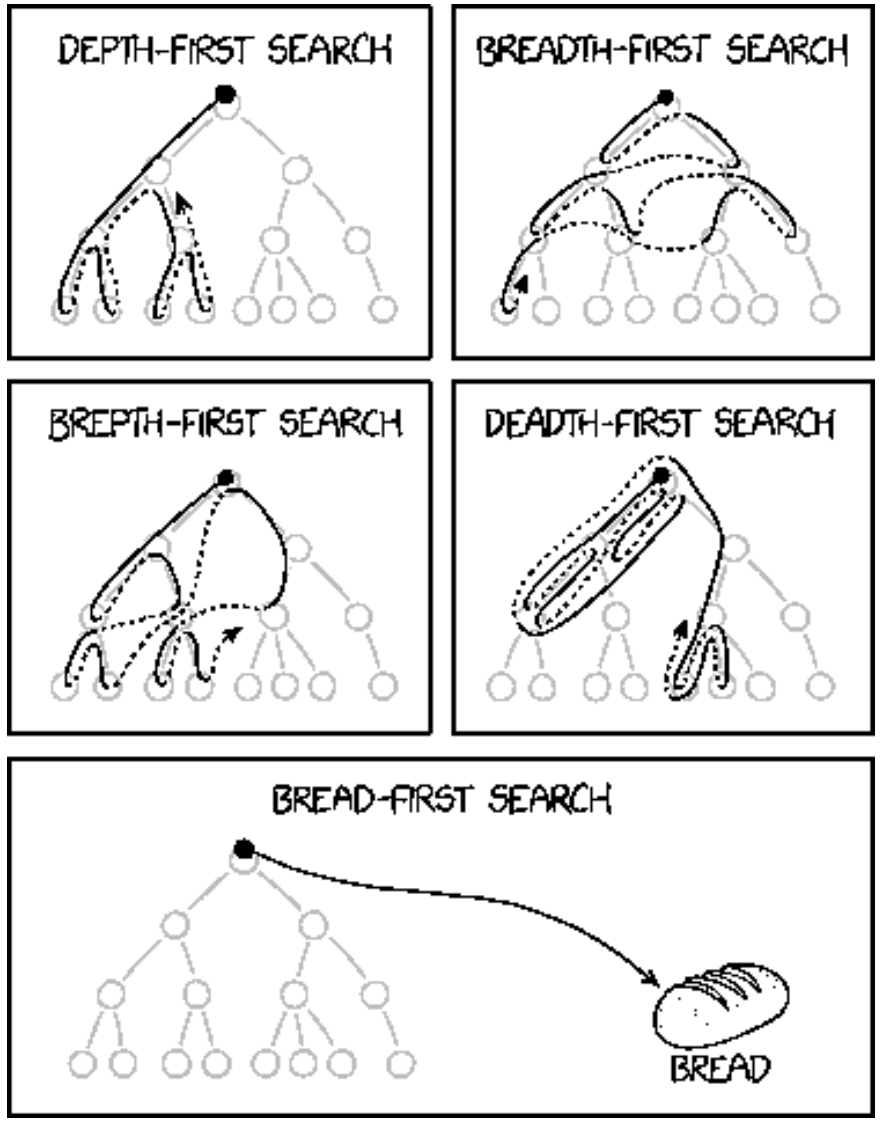
CSCI 3104: Algorithms

Lecture 14: Minimum Spanning Tree, Kruskal's Algorithm, Prim's Algorithm

Rachel Cox

Department of Computer Science



What will we learn today?

- ❑ Minimum Spanning Trees
- ❑ Kruskal's Algorithm
- ❑ Prim's Algorithm

Intro to Algorithms, CLRS:
Sections 22.2, 23.1, 24.3

Graphs – A few Definitions

- Here are a few definitions involving Graphs (taken from our Week 8 reading by Prof. Aaron Clauset

simple graph: A graph with no self-loops and no multi-edges

multigraph: A graph in which a pair of nodes i, j can have multiple, distinct connections

weighted graph: One in which an edge (i, j) is annotated with a scalar weight w_{ij} or by a weight function $w(e)$ for $e \in (i, j) \in E$

directed graph: A graph in which connections can be acyclic

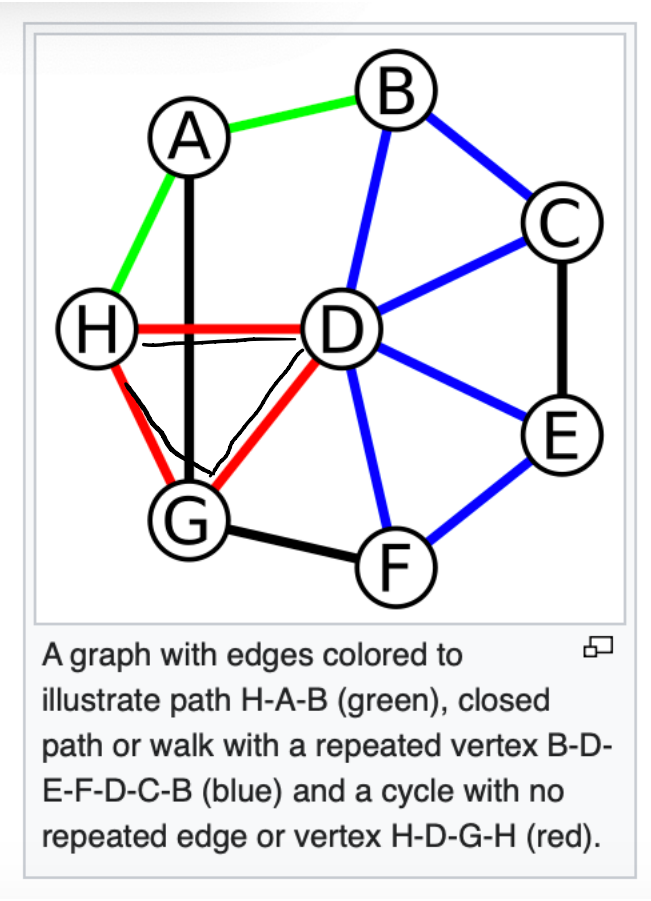
acyclic graph: A special kind of directed graph that contains no cycles.

Graphs – A few Definitions

What is a cycle?

A **cycle** in a graph is a non-empty trail in which the only repeated vertices are the first and last vertices. (Wikipedia)

Note: A trail is a walk in which all edges are distinct.



A graph with edges colored to illustrate path H-A-B (green), closed path or walk with a repeated vertex B-D-E-F-D-C-B (blue) and a cycle with no repeated edge or vertex H-D-G-H (red).

Dijkstra's Algorithm

Dijkstras(G, S)

Initialize $distance, previous$
$distance(S) = 0$
$distance(V) = \infty$

$Q = \min priority\ queue$
$Q.add(V)$

$while\ Q\ !\ empty$

$u = Q.pop()$

$for\ each\ v\ in\ u.adjacent$
$d = distance(u) + e_{u,v}$

$if\ d < distance(v)$
$distance(v) = d$
$previous(v) = u$

m iterations

At most n-1 iterations

$return\ G$

Runtime:

n = number of vertices
m = number of edges

Note: Each vertex can be connected to at most n-1 other vertices

$\Rightarrow$ worst case $\Theta(nm)$

But with: priority queue: $\Theta(m \lg n)$

Minimum Spanning Trees

Given connected, undirected graph $G = (V, E)$, a spanning tree is a subgraph that is an undirected tree and contains all vertices in G.

In a weighted graph, the weight of the subgraph is the sum of edges in the subgraph.

A **minimum spanning tree** (MST) is a spanning tree with minimum weight.

- number of edges = number of vertices − 1

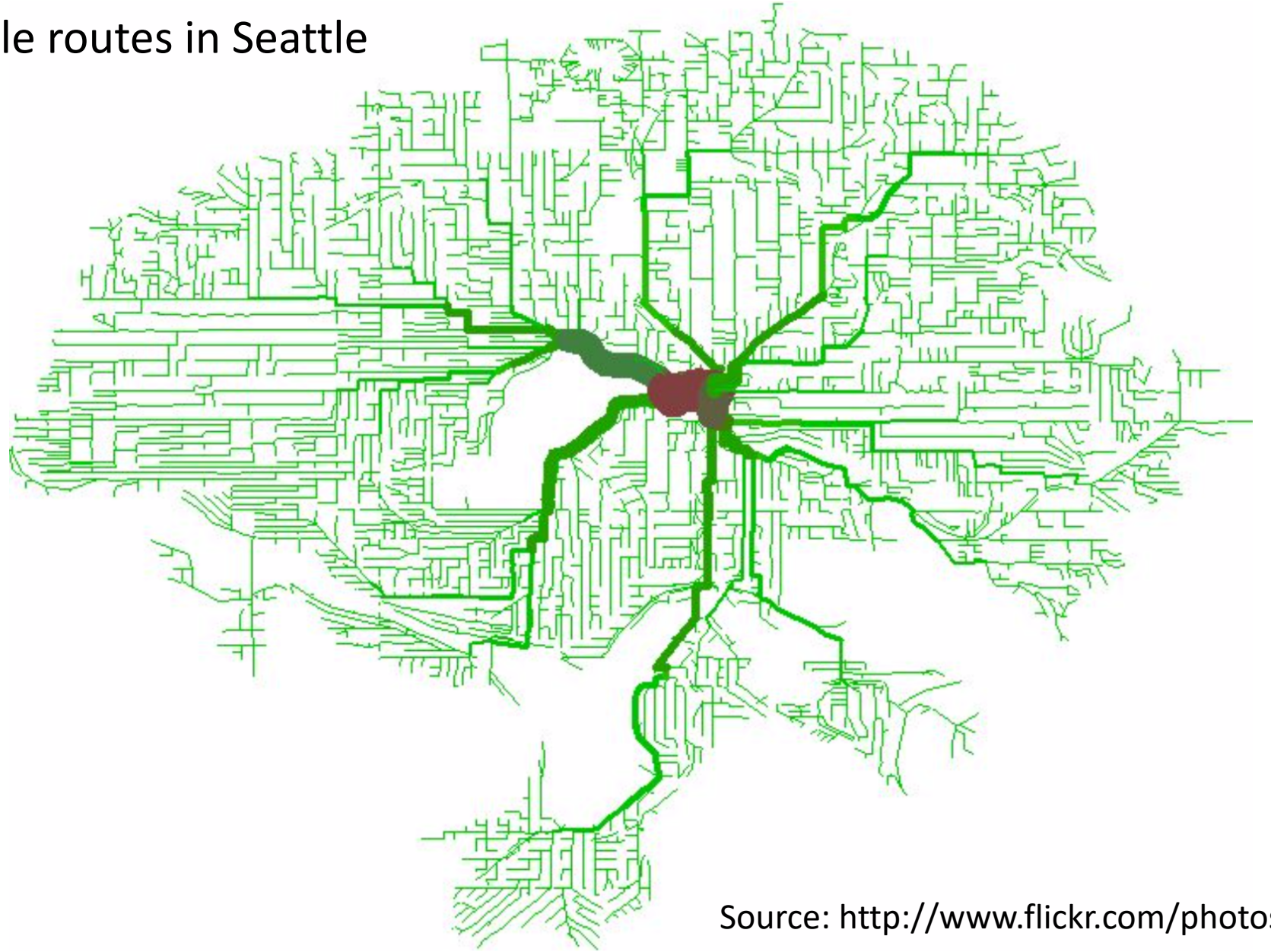
spanning tree

- Let A be a set of edges such that $A \subseteq T$ where T is a MST.
  An edge (u,v) is a "safe edge" for A, if $A \cup \{(u,v)\}$ is also a subset of some MST.

- Trying to "grow" a MST.

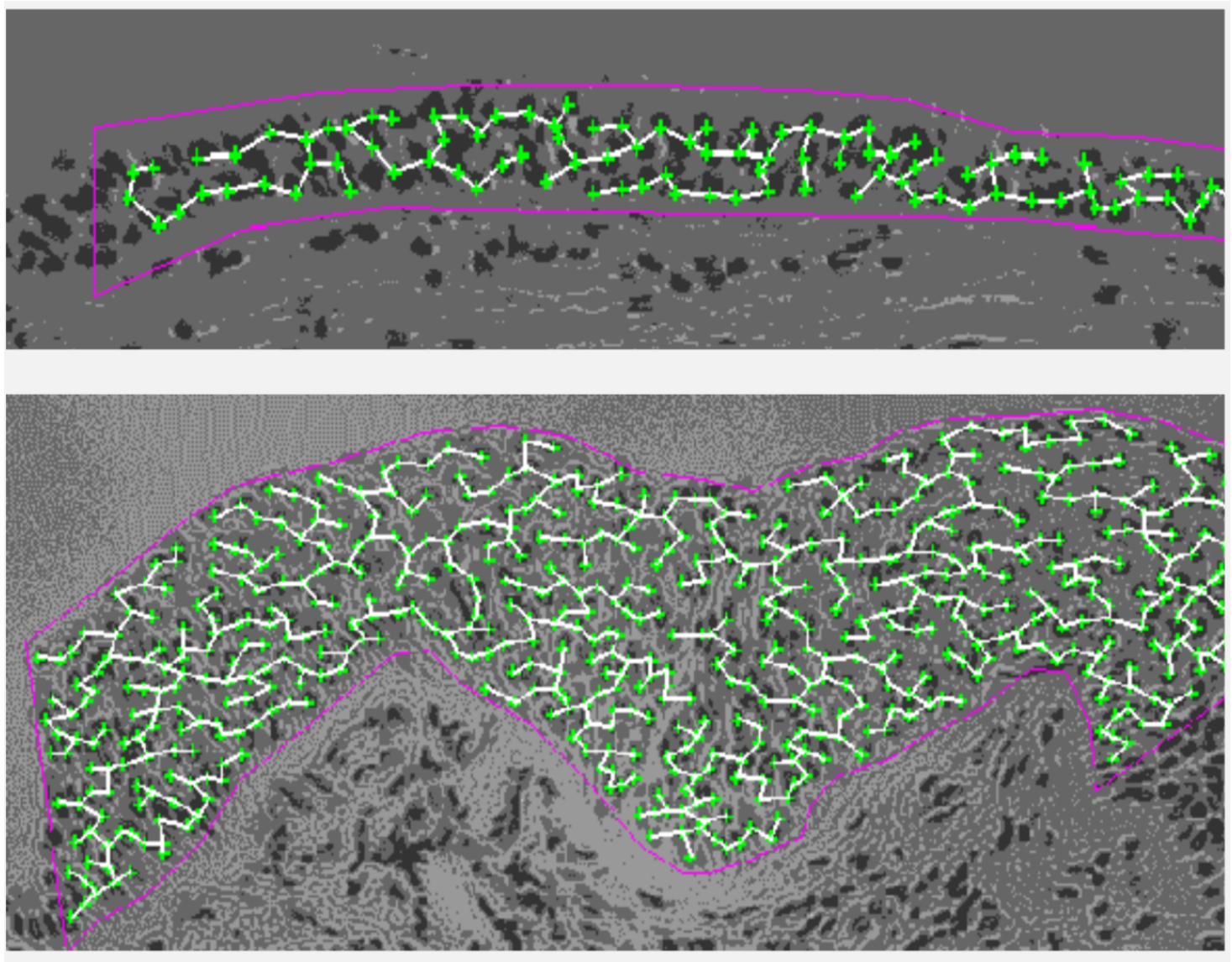
Minimum Spanning Trees

MST of bicycle routes in Seattle



Source: http://www.flickr.com/photos/ewedistrict/21980840

Minimum Spanning Trees



MST describes arrangement of nuclei in the epithelium for cancer research

Source: http://www.bccrc.ca/ci/ta01_archlevel.html

Minimum Spanning Trees

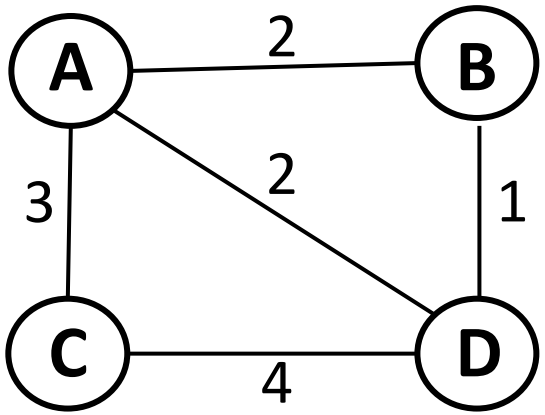
Many Applications:

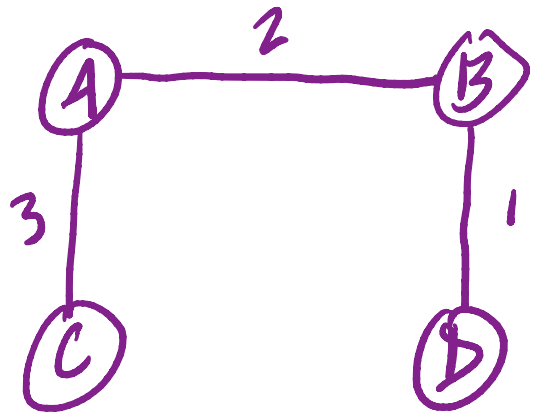
- Cluster Analysis
- Maximum bottleneck paths
- Real-time face verification
- LDPC codes for error correction
- Find road networks in satellite and aerial imagery.
- Reducing data storage in sequencing amino acids in a protein.
- Autoconfig protocol for Ethernet bridging to avoid cycles in a network
- Approximation algorithms for NP-hard problems
- Network design

Source: http://www.ics.uci.edu/~eppstein/gina/mst.htm

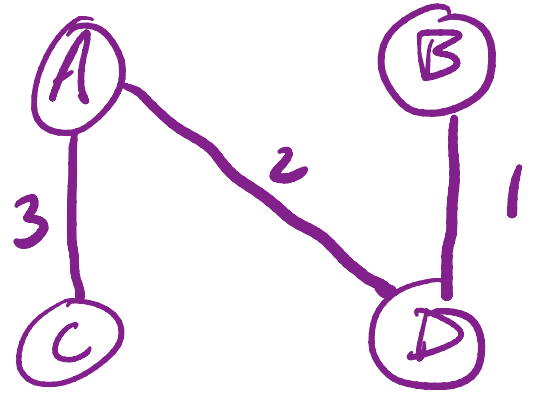
Minimum Spanning Trees

Example: Consider the following graph. Find an example of a Minimum Spanning Tree.

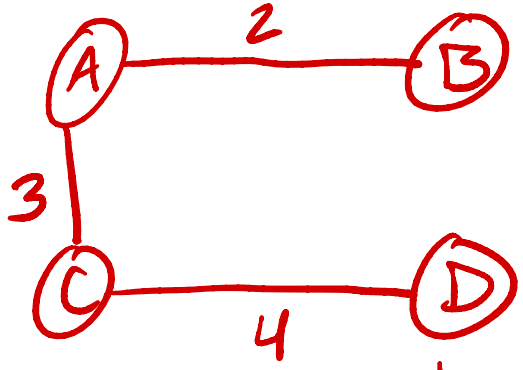




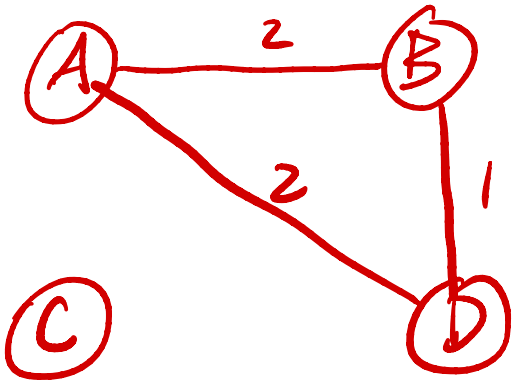
MST: 6



MST = 6



• Spanning tree but MST



not a Spanning tree

Kruskal's Algorithm

Two Algorithms in this Lecture

Prim's & Kruskal's – Greedy Algo for growing a MST

Kruskal's Algorithm

1. Sort all edges in ascending order with respect to their weight
2. Select the smallest edge. Check if this smallest edge forms a cycle. If not, include the edge, otherwise remove it.
3. Repeat until there are V-1 edges in the spanning tree. (assume the graph has V vertices to begin with)

Kruskal's Algorithm

Kruskal's Algorithm

MST

$T = \emptyset$

for $i = 1$ to m ← $O(m)$ iterations

  if $T \cup \{i\}$ has no cycles } $O(n)$ time to check for cycles

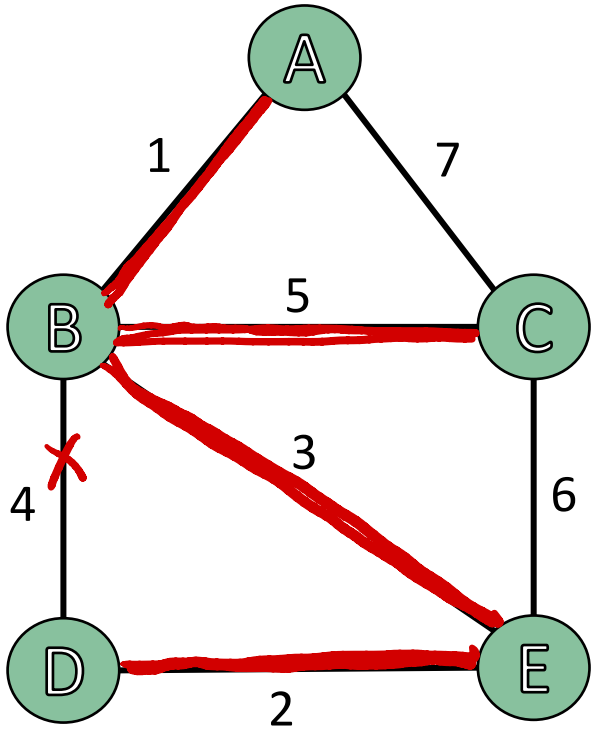
    Add i to T

return T

worst case : $O(mn)$

Kruskal's Algorithm

Example: Use Kruskal's algorithm to find the minimum spanning tree (MST) of the graph below.



exclude because it will form a cycle

Note: $n = 5$
. we run Kruskals until T has $n-1$ edges.

① $T = \emptyset$

② $T = \emptyset \cup (A,B)$ ← edge

↳ e_{AB}

③ $T = (A,B) \cup (D,E)$

④ $T = (A,B) \cup (D,E) \cup (B,E)$

⑤ $T = (A,B) \cup (D,E) \cup (B,E) \cup (B,C)$

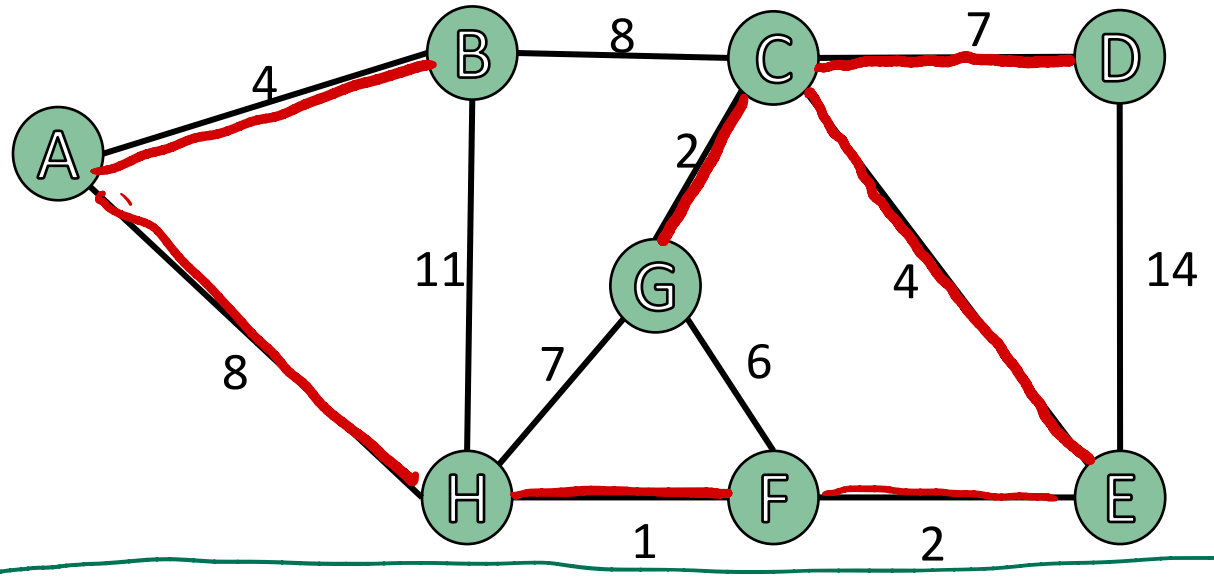
$n-1$ edges, thus we stop.

$T = \{(A,B), (D,E), (B,E), (B,C)\}$

Kruskal's Algorithm

• Add edge
x exclude edge

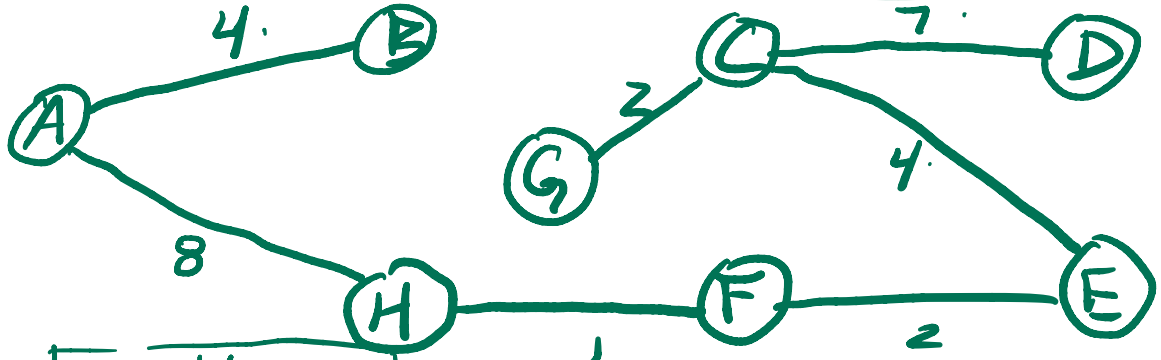
Example: Use Kruskal's algorithm to find the MST of the graph below.



(H,F) = 1 •
(F,E) = 2 •
(C,G) = 2 •
(C,E) = 4 •
(A,B) = 4 •
(G,F) = 6 x makes a cycle
(G,H) = 7 x makes a cycle
(C,D) = 7 •
(A,H) = 8 •
(B,C) = 8 x
(B,H) = 11 x
(D,E) = 14 x

= At this point we have n-1 edges. Stop

$T = \{(H,F), (F,E), (C,G), (C,E), (A,B), (C,D), (A,H)\}$



weight: 28

Prim's Algorithm

- ❖ Both Prim's and Kruskal's form a single tree and add edges to that tree
- ❖ Both are Greedy algorithms

What is the Difference?

- Kruskal's: selects minimum from all possible edges
- Prim's: selects minimum that connects to the existing tree.
  - starting vertex

Prim's Algorithm

$Prim(Graph)$

Initialize $V.cost = \infty,\ V.parent = NULL$
$Queue = Graph.V$
$Graph.root.cost = 0$

while $Queue$! $empty$
  $u = Queue.pop()$
  for each $v \in u.adjacent$
    if $v \in Queue$ and $e_{u,v} < v.cost$
      $v.parent = u$
      $v.cost = e_{u,v}$

} updating costs of edges in the queue

- Remove the minimum cost vertex from Q, Add edge to T.

Prim's Algorithm

initially: {A: ∞, B: ∞, C: ∞, D: ∞}
costs

Example: Use Prim's algorithm to find the MST of the graph given below.

Root = A

A.cost = 0 *

Q = {A: 0, B: ∞, C: ∞, D: ∞}

u = A adjacent nodes: B, C, D

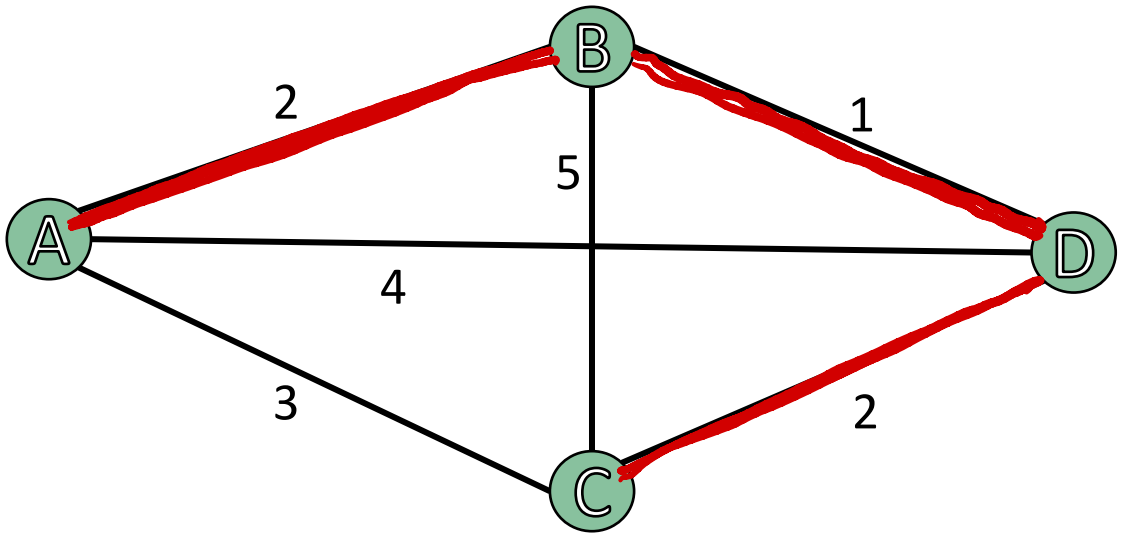
→ B.cost = 2 B.parent = A

C.cost = 3 C.parent = A

D.cost = 4 D.parent = A

u = B adjacent nodes: A, C, D
(A: not in Q)

update: D.cost = 1
D.parent = B



TREE = {(A,B), (B,D), (C,D)}

Q: {C: 3, D: 1} not in Q

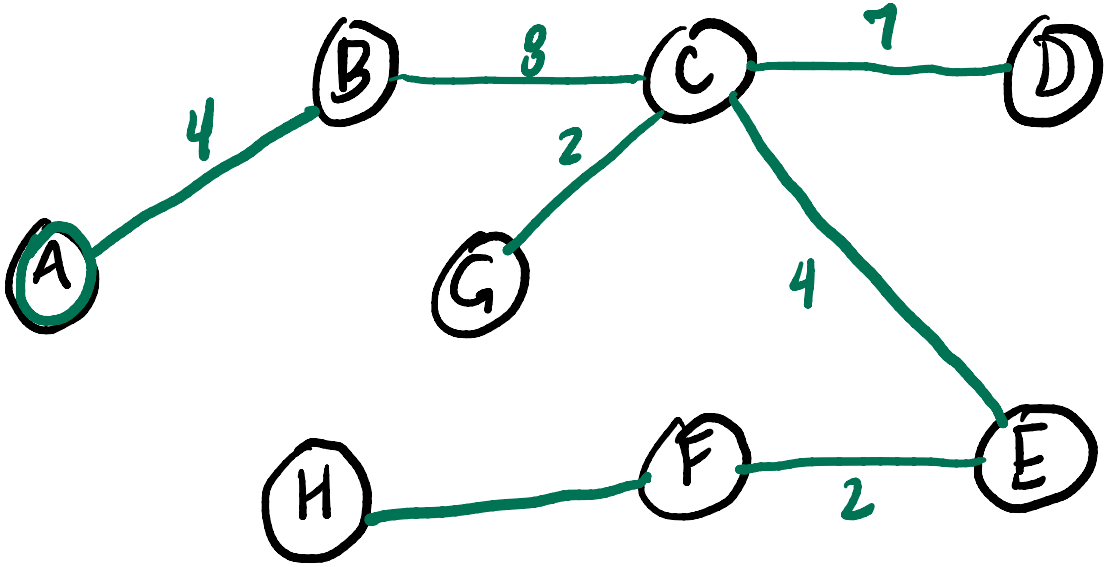
u = D adjacent nodes: A, C

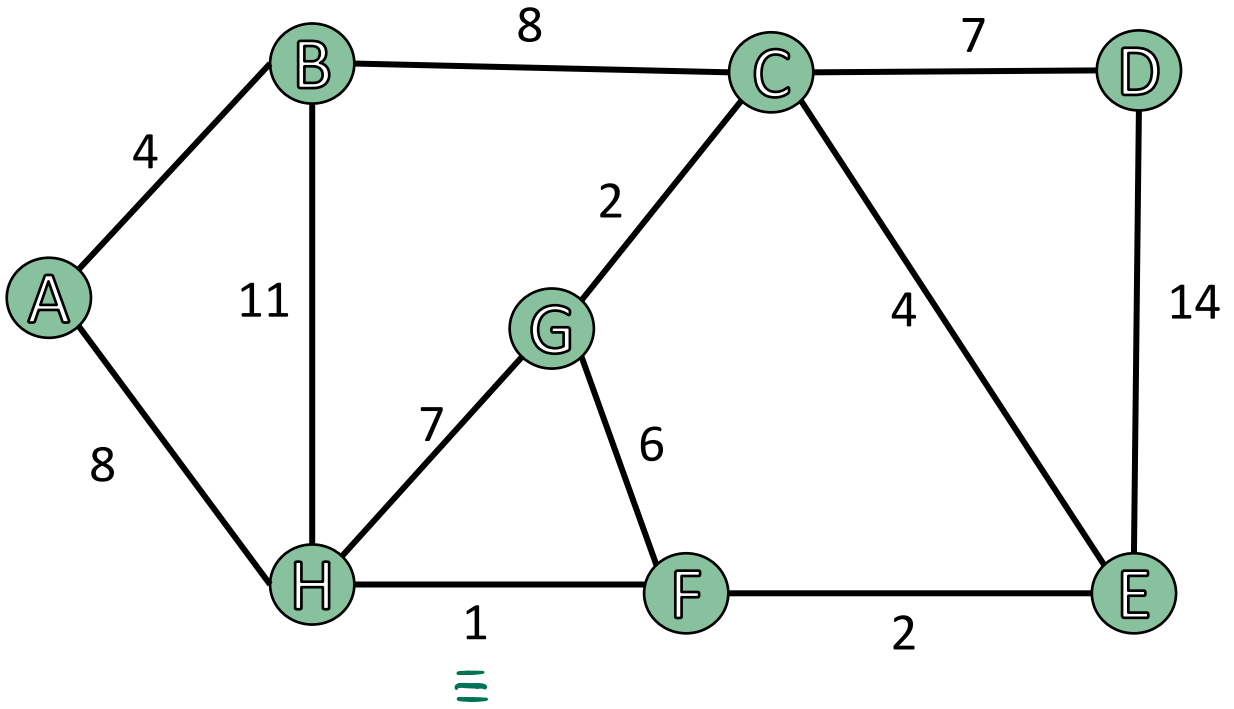
C.cost = 2

C.parent = D

Prim's Algorithm

Example: Use Prim's algorithm to find the MST of the graph given below.





$T = \{(A,B), (B,C), (C,G), (C,E), (E,F), (F,H), (C,D)\}$