CS4102 Algorithms

Spring 2019

Warm up:

Show that the sum of degrees of all nodes in any undirected graph is even

Show that for any graph $G = (V, E)$, $\sum_{v \in V} \deg(v)$ is even

$\sum_{v \in V} \deg(v)$ is always even

- $\deg(v)$ counts the number of edges incident v
- Consider any edge $\mathrm{e} \in E$
- This edge is incident 2 vertices (on each end)
- This means $2 \cdot |E| = \sum_{v \in V} \deg(v)$
- Therefore $\sum_{v \in V} \deg(v)$ is even

Today’s Keywords

- Greedy Algorithms
- Choice Function
- Graphs
- Minimum Spanning Tree
- Kruskal’s Algorithm
- Prim’s Algorithm
- Cut Theorem

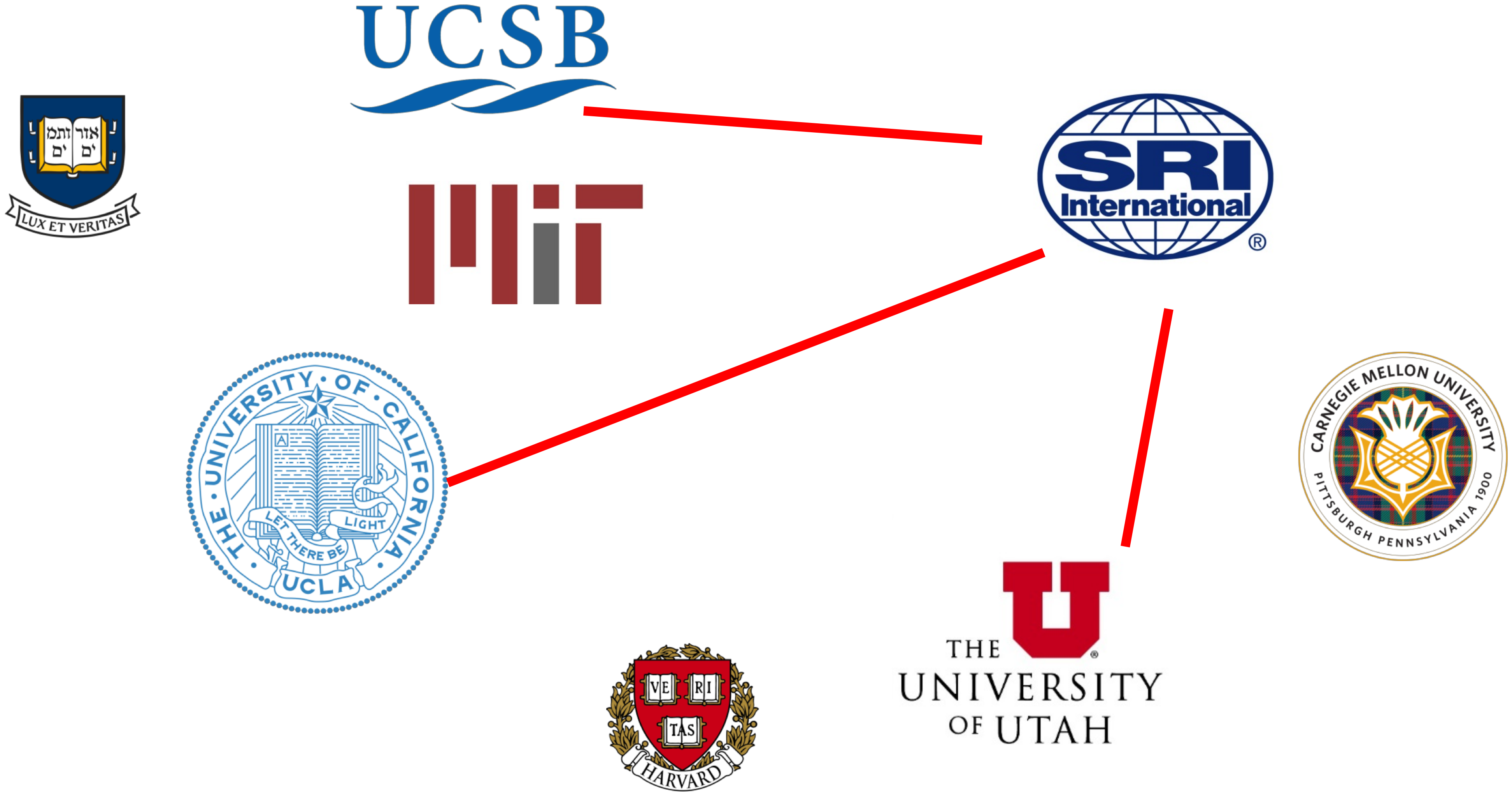
CLRS Readings

- Chapter 22
- Chapter 23

Homeworks

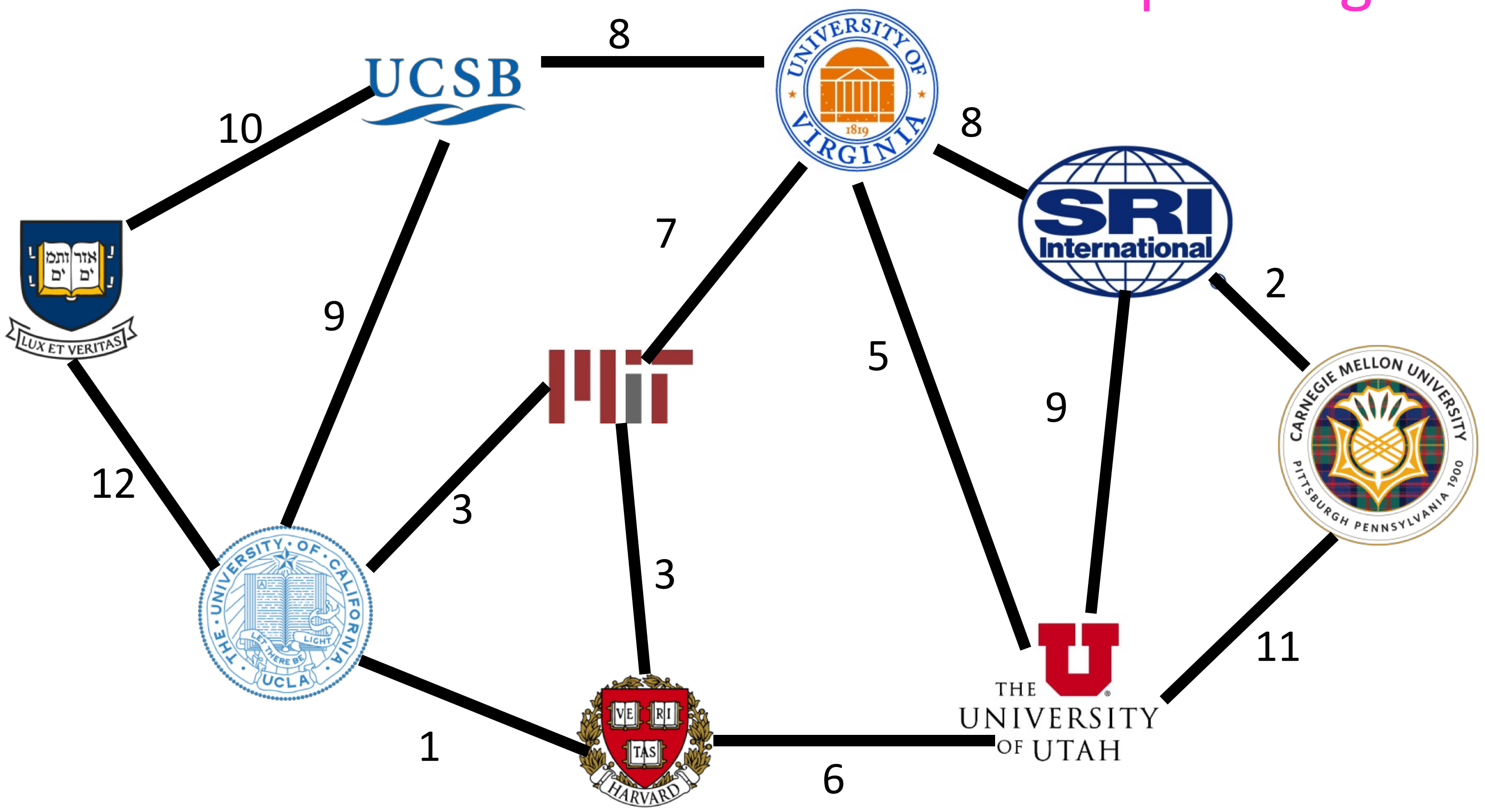
- HW7 Due **Tuesday** April 16 @11pm
  - Written (use latex)
  - Graphs
  - Released tonight

ARPANET


UCSB
LUX ET VERITAS
SRI International
THE UNIVERSITY OF CALIFORNIA
LET THERE BE LIGHT
UCLA
CARNEGIE MELLON UNIVERSITY
PITTSBURGH PENNSYLVANIA 1900
VE RI TAS
HARVARD
THE UNIVERSITY OF UTAH

Problem

Find a
Minimum
Spanning Tree



We need to connect together all these places into a network
We have feasible wires to run, plus the cost of each wire
Find the cheapest set of wires to run to connect all places

Graphs

Vertices/Nodes

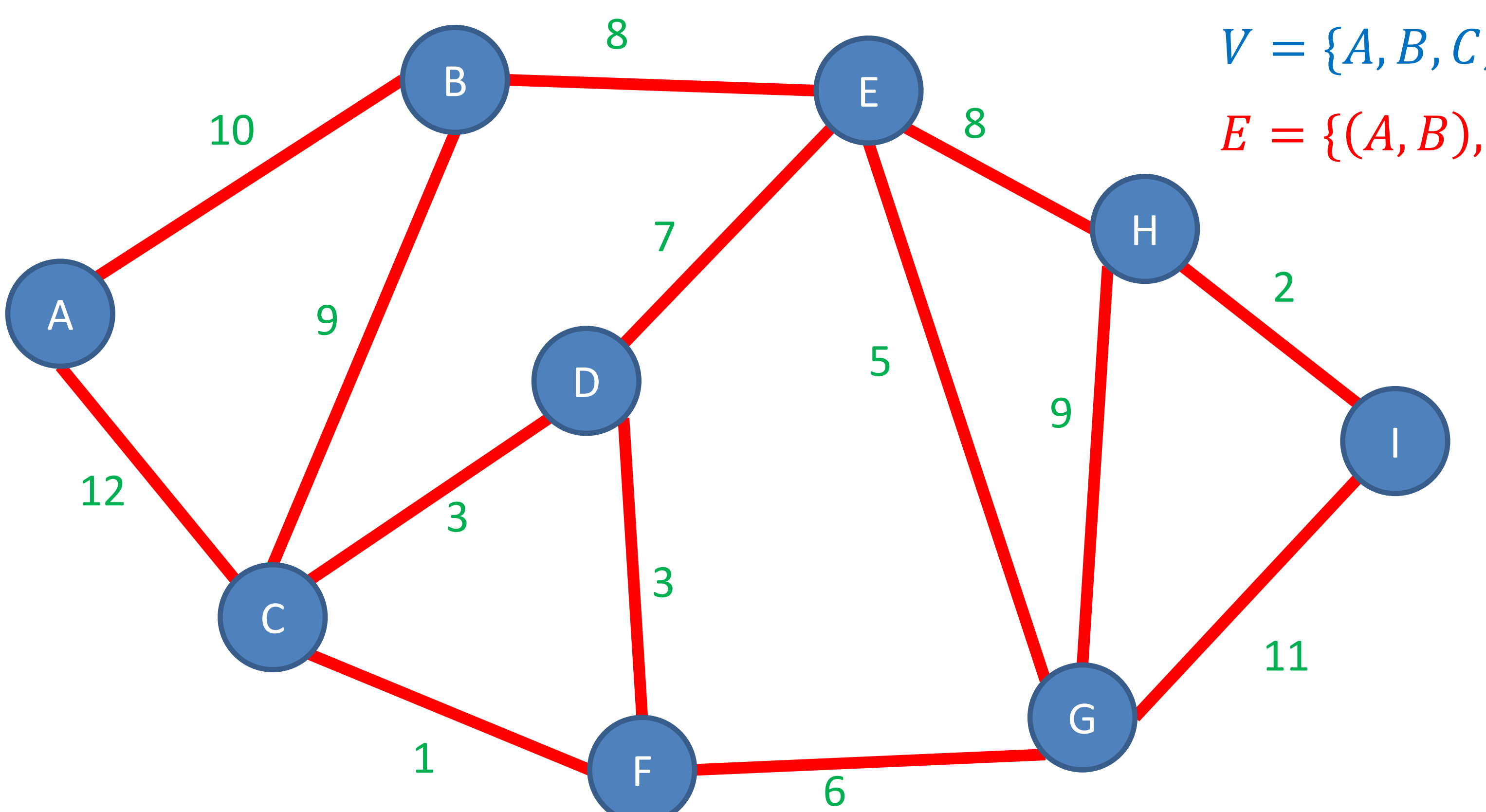
Definition: $G = (V, E)$

Edges

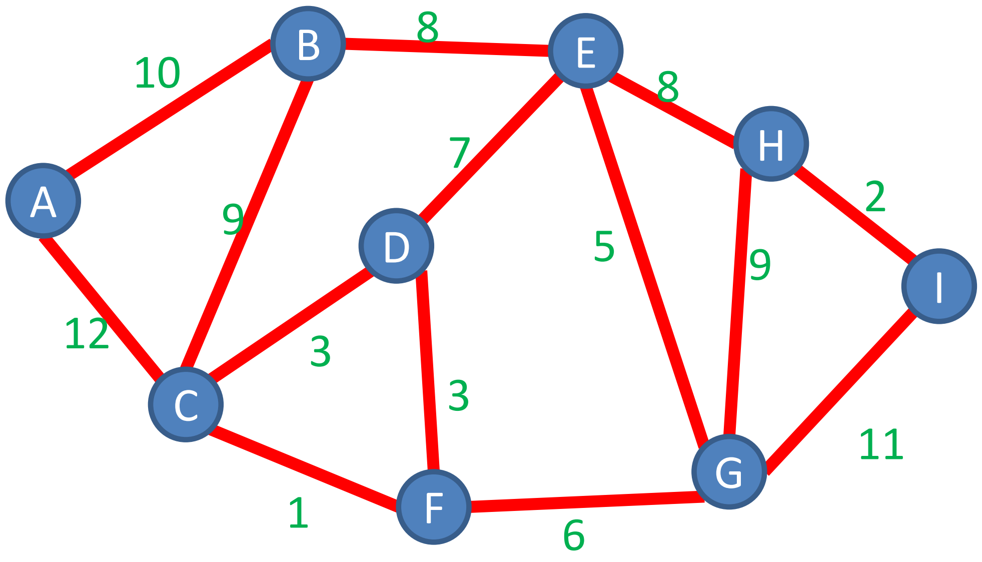
$w(e)$ = weight of edge e

$V = \{A, B, C, D, E, F, G, H, I\}$

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



Adjacency List Representation



A	B	C		
B	A	C	E	
C	A	B	D	F
D	C	E	F	
E	B	D	G	H
F	C	D	G	
G	E	F	H	I
H	E	G	I	
I	G	H		

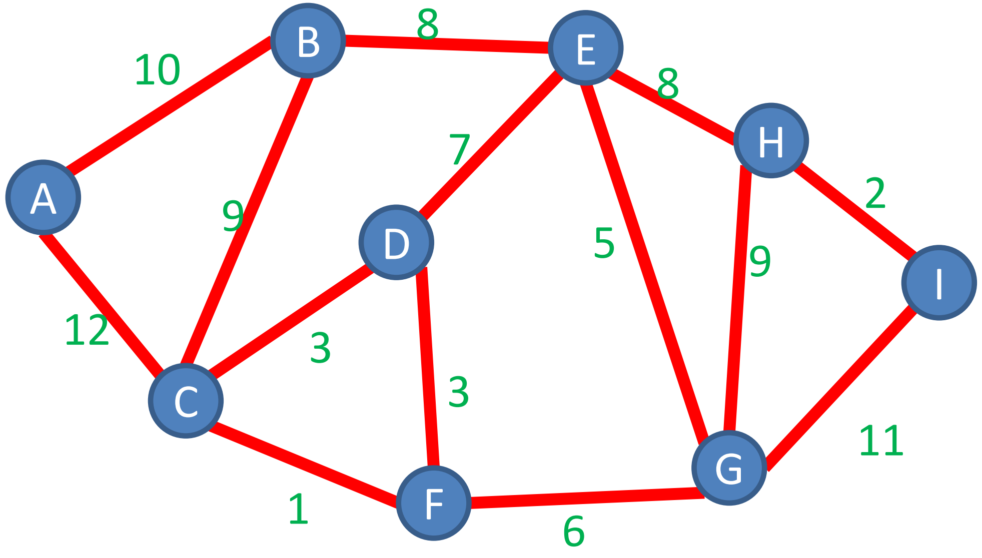
<u>Tradeoffs</u>

Space: $V + E$

Time to list neighbors: $Degree(A)$

Time to check edge (A, B):$Degree(A)$

Adjacency Matrix Representation



	A	B	C	D	E	F	G	H	I
A		1	1						
B	1		1		1				
C	1	1		1		1			
D			1		1	1			
E		1		1			1	1	
F			1	1			1		
G					1	1		1	1
H					1		1		1
I							1	1	

<u>Tradeoffs</u>

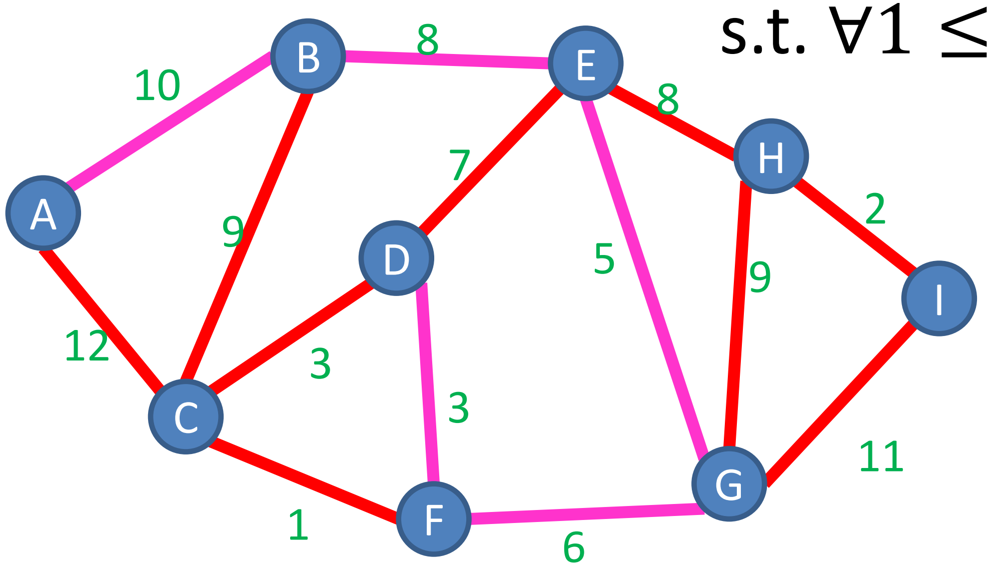
Space: V^2

Time to list neighbors: V

Time to check edge (A, B):$O(1)$

Definition: Path

A sequence of nodes $(v_1, v_2, \dots, v_k)$
s.t. $\forall 1 \le i \le k-1, (v_i, v_{i+1}) \in E$

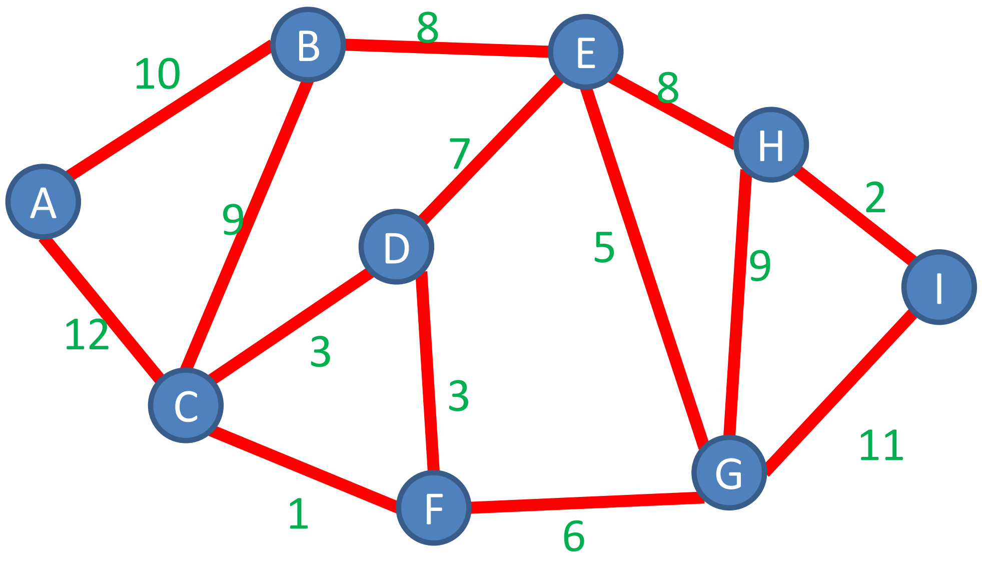


<u>Simple Path:</u>
A path in which each node appears at most once

<u>Cycle:</u>
A path of > 2 nodes in which $v_1 = v_k$

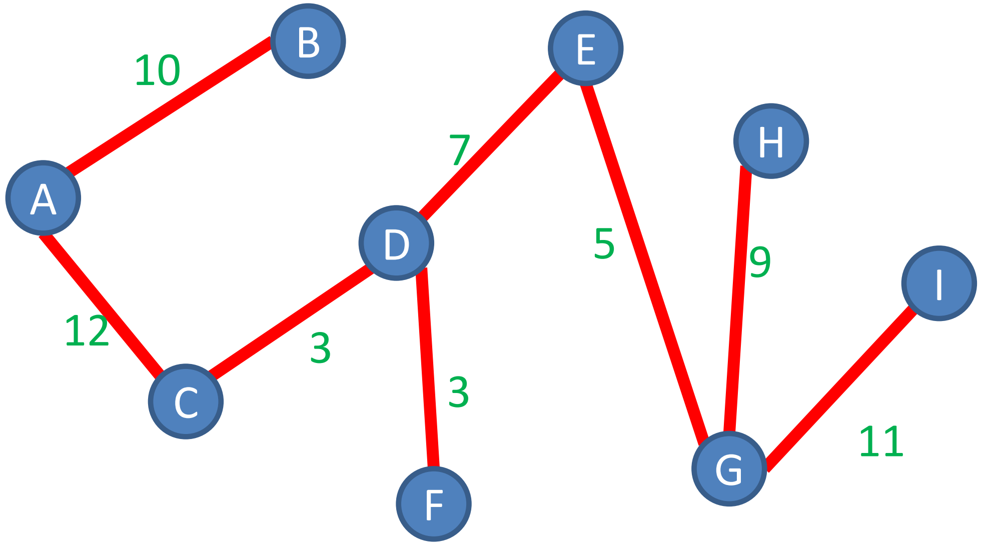
Definition: Connected Graph

A Graph $G = (V, E)$ s.t. for any pair of nodes $v_1, v_2 \in V$ there is a path from v_1 to v_2



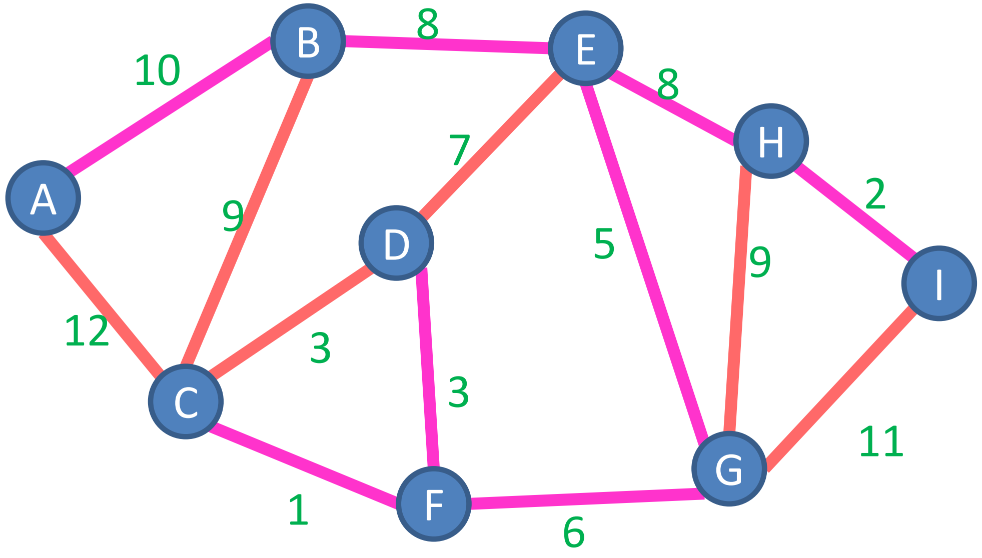
Definition: Tree

A connected graph with no cycles



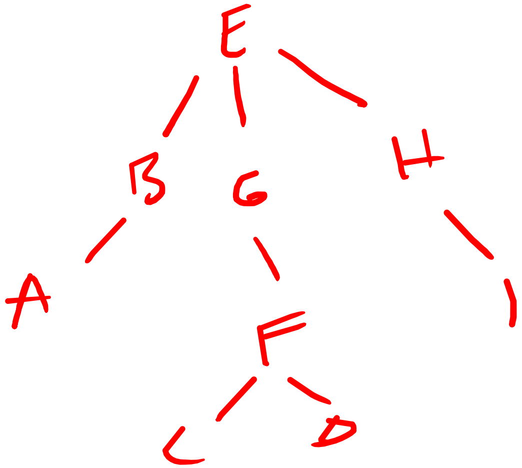
Definition: Spanning Tree

A Tree $T = (V_T, E_T)$ which connects ("spans") all the nodes in a graph $G = (V, E)$



How many edges does T have?

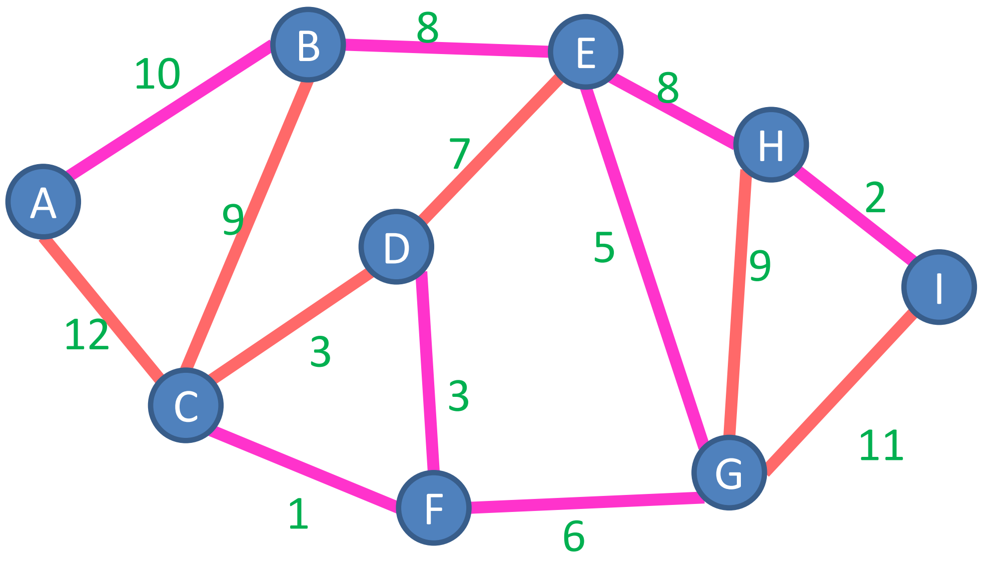
$V - 1$



Definition: Minimum Spanning Tree

A Tree $T = (V_T, E_T)$ which connects ("spans") all the nodes in a graph $G = (V, E)$, that has minimal cost

$$Cost(T) = \sum_{e \in E_T} w(e)$$



How many edges does T have?

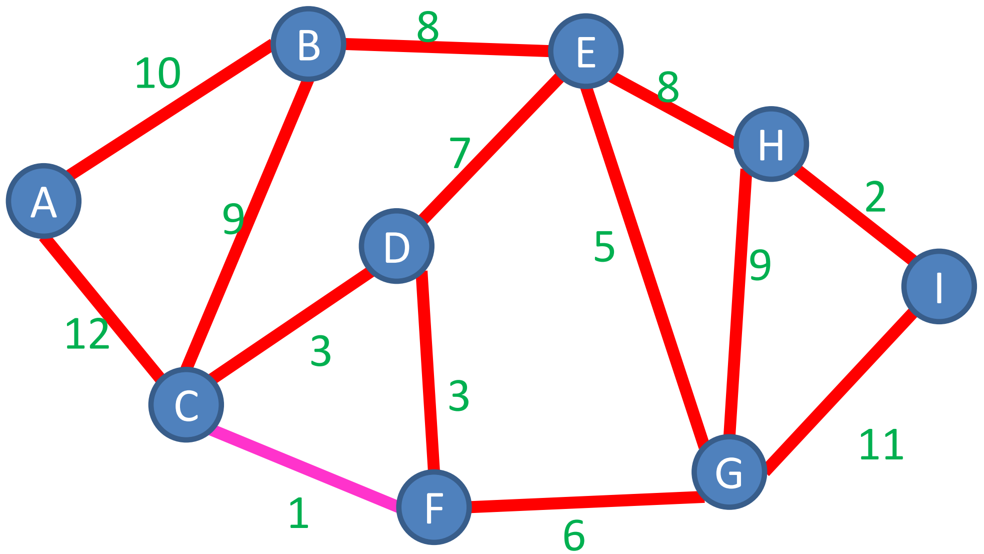
$V - 1$

Greedy Algorithms

- Require Optimal Substructure
  - Solution to larger problem contains the solution to a smaller one
  - Only one subproblem to consider!
- Idea:
  1. Identify a greedy choice property
     - How to make a choice guaranteed to be included in some optimal solution
  2. Repeatedly apply the choice property until no subproblems remain

Kruskal’s Algorithm

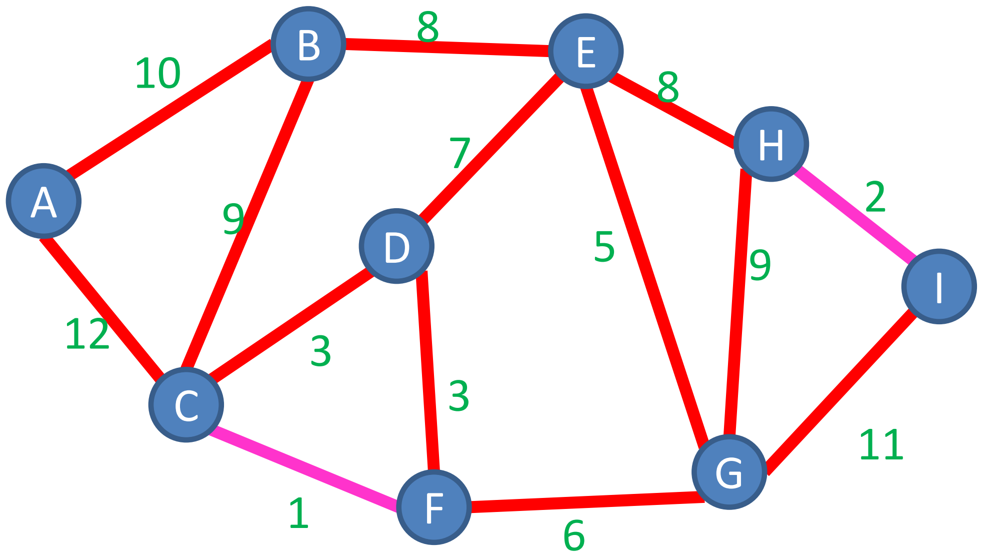
Start with an empty tree A
Add to A the lowest-weight edge that does not create a cycle



Kruskal's Algorithm

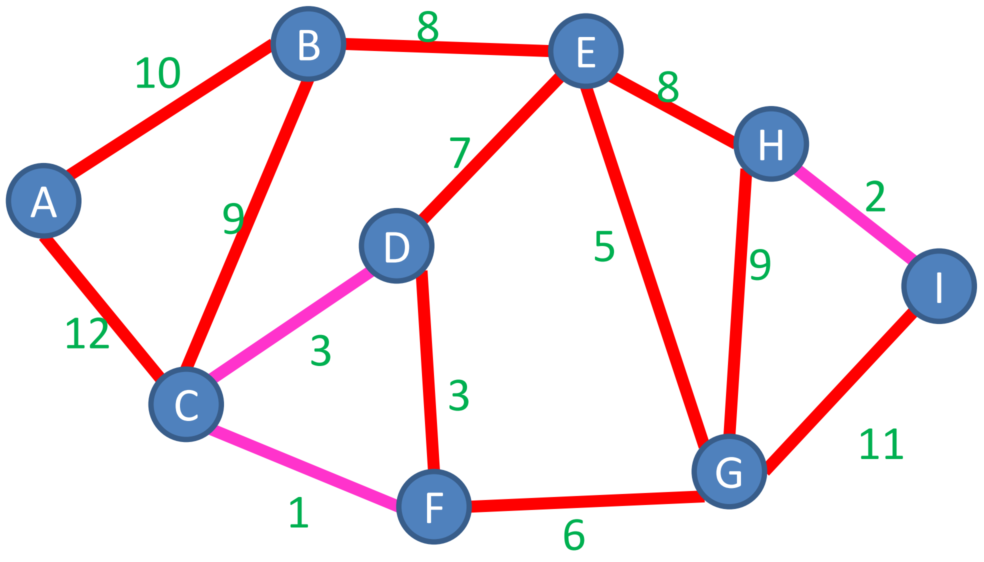
Start with an empty tree A

Add to A the lowest-weight edge that does not create a cycle



Kruskal’s Algorithm

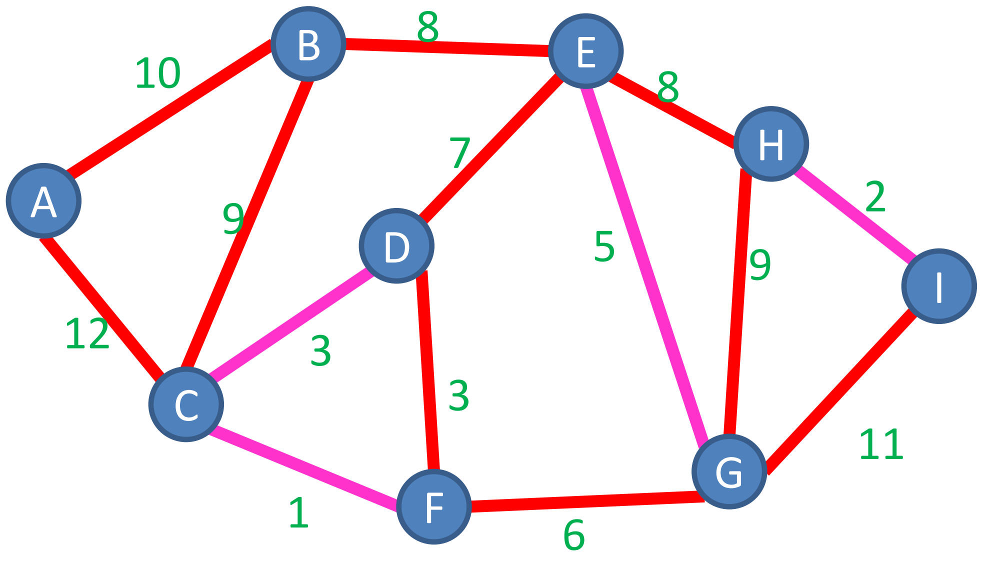
Start with an empty tree A
Add to A the lowest-weight edge that does not create a cycle



Kruskal’s Algorithm

Start with an empty tree A

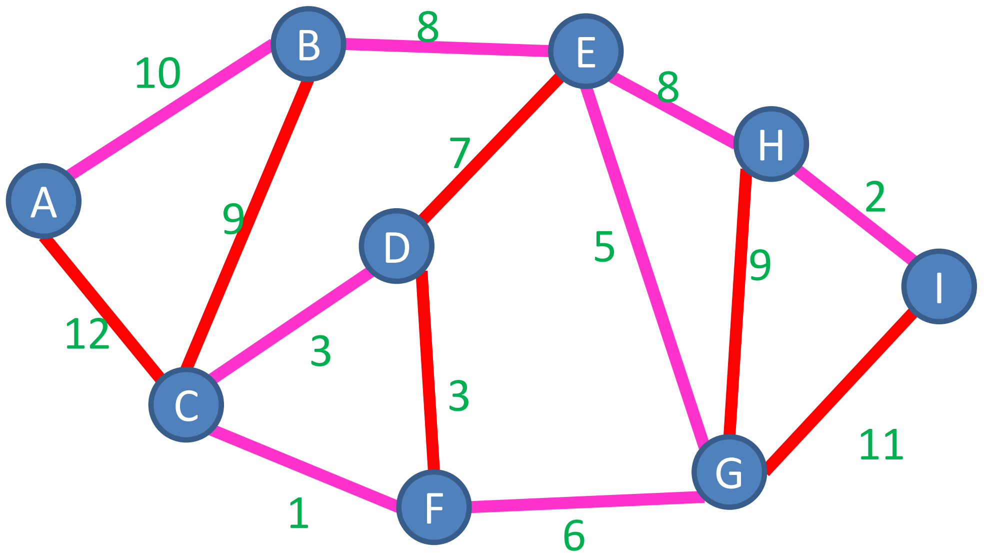
Add to A the lowest-weight edge that does not create a cycle



Kruskal's Algorithm

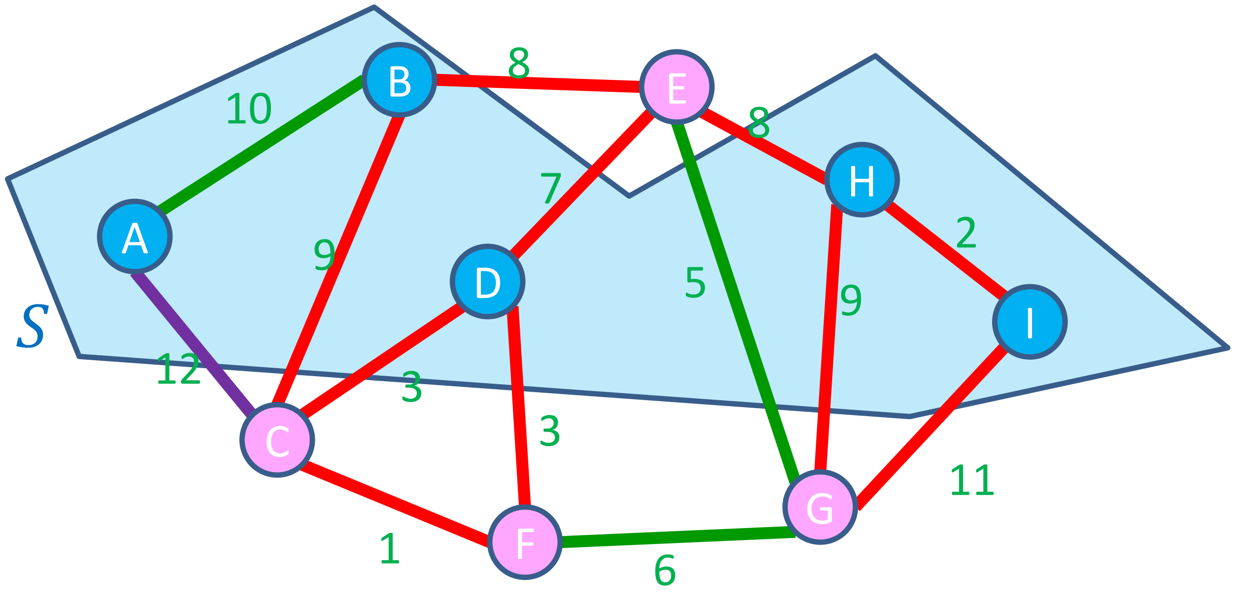
Start with an empty tree A

Add to A the lowest-weight edge that does not create a cycle



Definition: Cut

A Cut of graph $G = (V, E)$ is a partition of the nodes into two sets, S and $V - S$



S

Edge $(v_1, v_2) \in E$ crosses a cut if $v_1 \in S$ and $v_2 \in V - S$ (or opposite), e.g. (A, C)

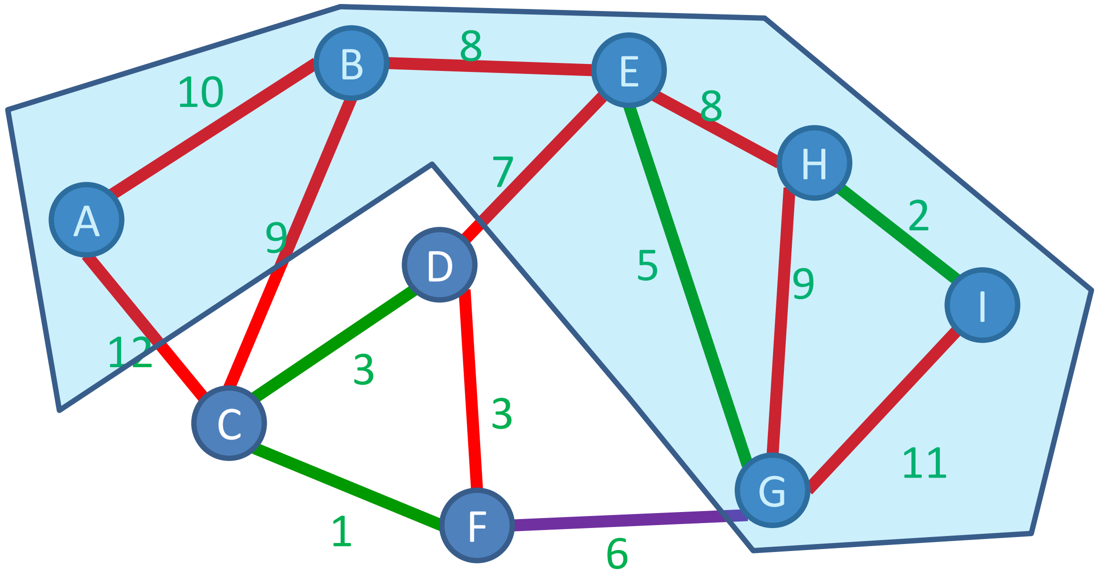
A set of edges R Respects a cut if no edges cross the cut e.g. $R = \{(A, B), (E, G), (F, G)\}$

Exchange argument

- Shows correctness of a greedy algorithm
- Idea:
  - Show exchanging an item from an arbitrary optimal solution with your greedy choice makes the new solution no worse
  - How to show my sandwich is at least as good as yours:
    - Show: “I can remove any item from your sandwich, and it would be no worse by replacing it with the same item from my sandwich”

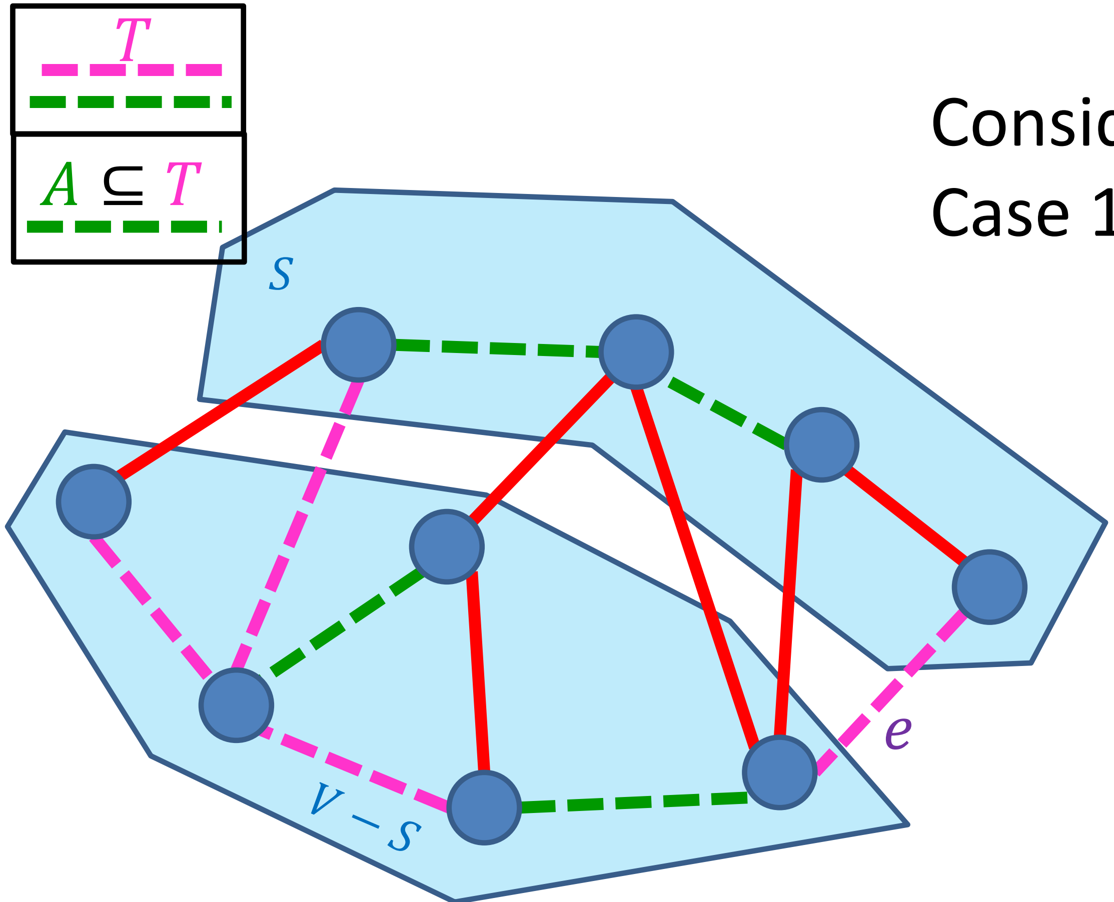
Cut Theorem

If a set of edges A is a subset of a minimum spanning tree T, let $(S, V - S)$ be any cut which A respects. Let e be the least-weight edge which crosses $(S, V - S)$. $A \cup \{e\}$ is also a subset of a minimum spanning tree.



Proof of Cut Theorem

Claim: If A is a subset of a MST T, and e is the least-weight edge which crosses cut $(S, V - S)$ (which A respects) then $A \cup \{e\}$ is also a subset of a MST.



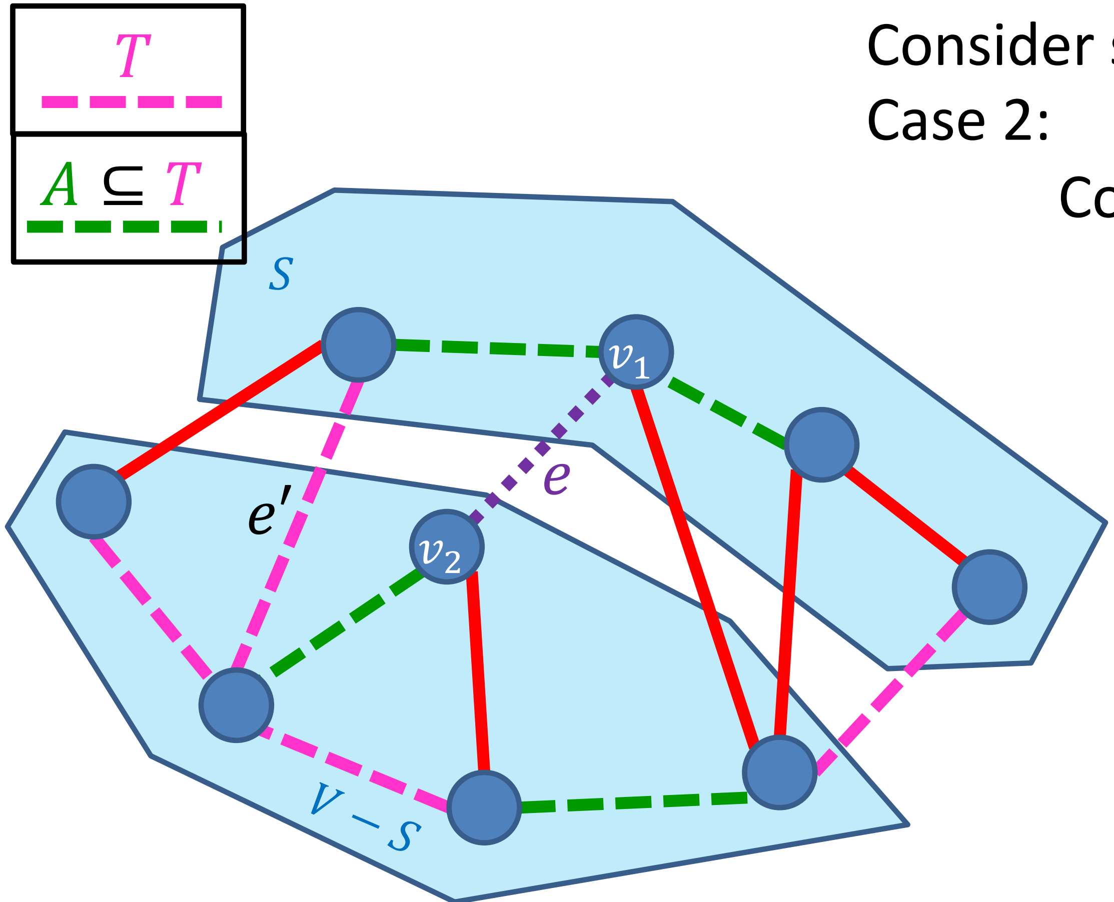
Consider some MST T,

Case 1: (the easy case)

If $e \in T$ Then claim holds

Proof of Cut Theorem

Claim: If A is a subset of a MST T, and e is the least-weight edge which crosses cut $(S, V - S)$ (which A respects) then $A \cup \{e\}$ is also a subset of a MST.



Consider some MST T,

Case 2:

Consider if $e = (v_1, v_2) \notin T$

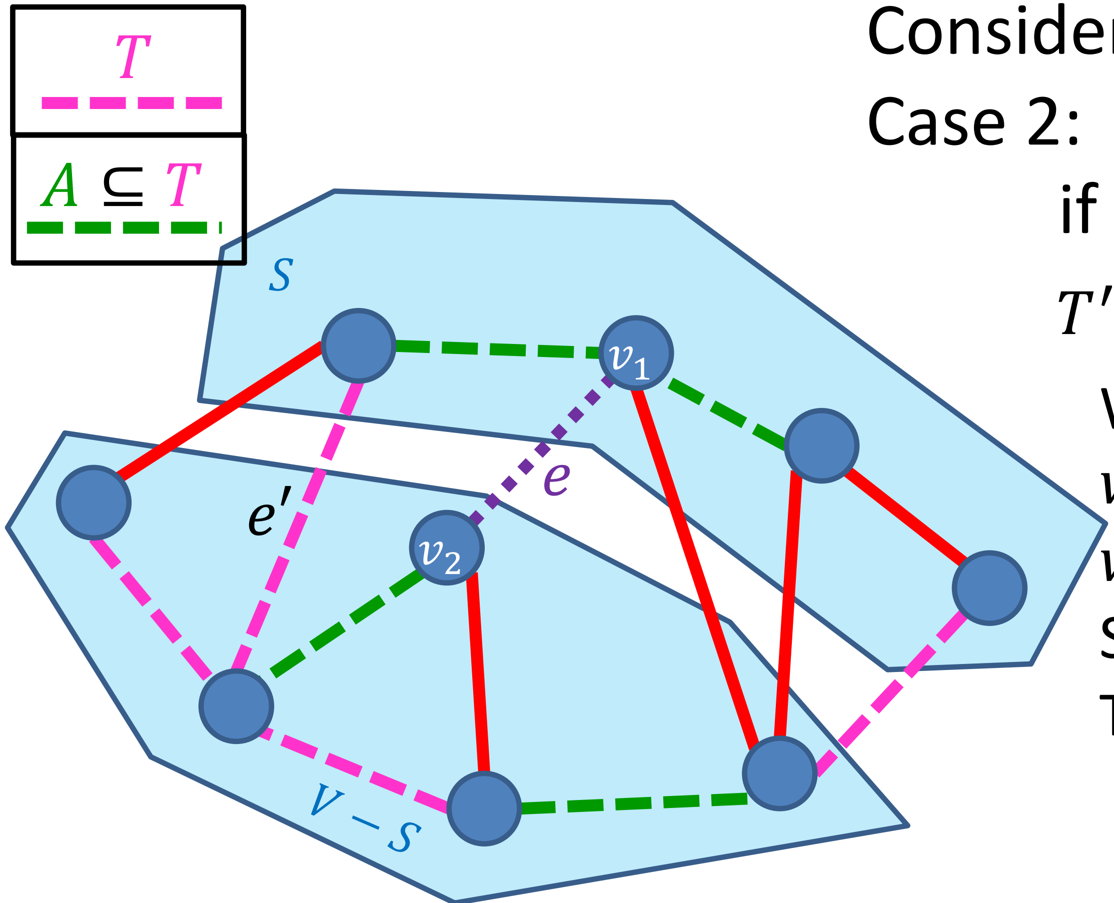
Since T is a MST, there is some path from v_1 to v_2.

Let e' be the first edge on this path which crosses the cut

Build tree T' by exchanging e' for e

Proof of Cut Theorem

Claim: If A is a subset of a MST T, and e is the least-weight edge which crosses cut $(S, V - S)$ (which A respects) then $A \cup \{e\}$ is also a subset of a MST.



Consider some MST T,

Case 2:

if $e = (v_1, v_2) \notin T$

$T' = T$ with edge e instead of e'

We assumed $w(e) \leq w(e')$

$w(T') = w(T) - w(e') + w(e)$

$w(T') \leq w(T)$

So T' is also a MST!

Thus the claim holds

Kruskal's Algorithm

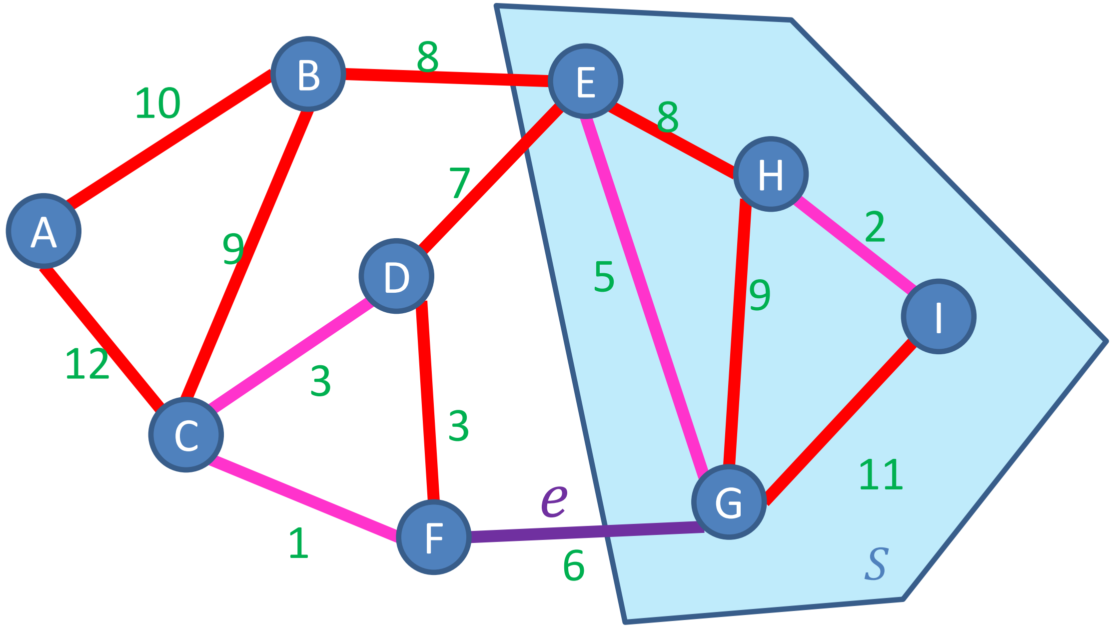
Start with an empty tree A

Repeat $V-1$ times:

Add the min-weight edge that doesn't cause a cycle

Keep edges in a Disjoint-set data structure (very fancy)

$O(E \log V)$



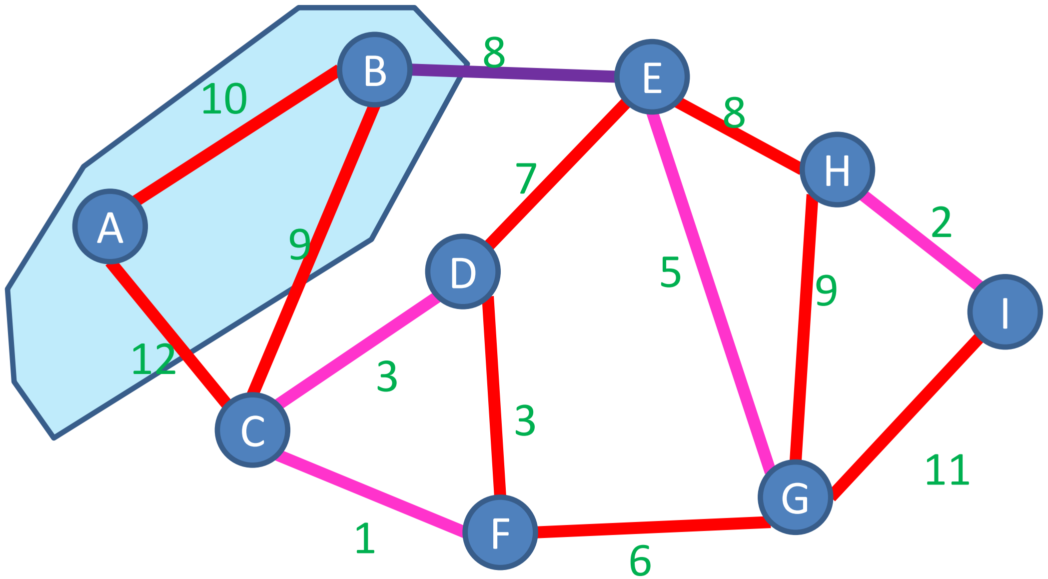
General MST Algorithm

Start with an empty tree A

Repeat $V - 1$ times:

Pick a cut $(S, V - S)$ which A respects

Add the min-weight edge which crosses $(S, V - S)$



Prim's Algorithm

Start with an empty tree A

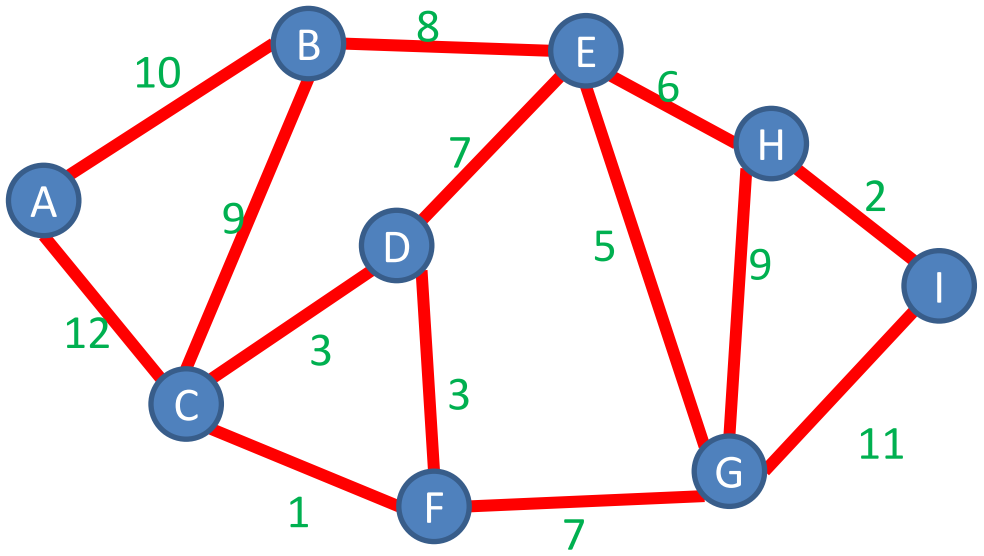
Repeat $V - 1$ times:

Pick a cut $(S, V - S)$ which A respects

Add the min-weight edge which crosses $(S, V - S)$

S is all endpoint of edges in A

e is the min-weight edge that grows the tree



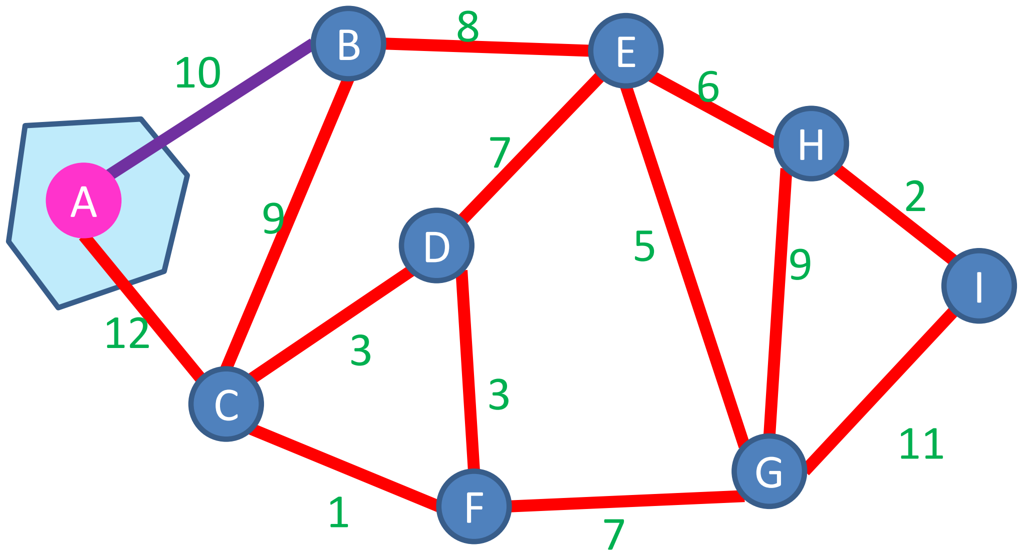
Prim's Algorithm

Start with an empty tree A

Pick a start node

Repeat $V - 1$ times:

Add the min-weight edge which connects to node in A with a node not in A



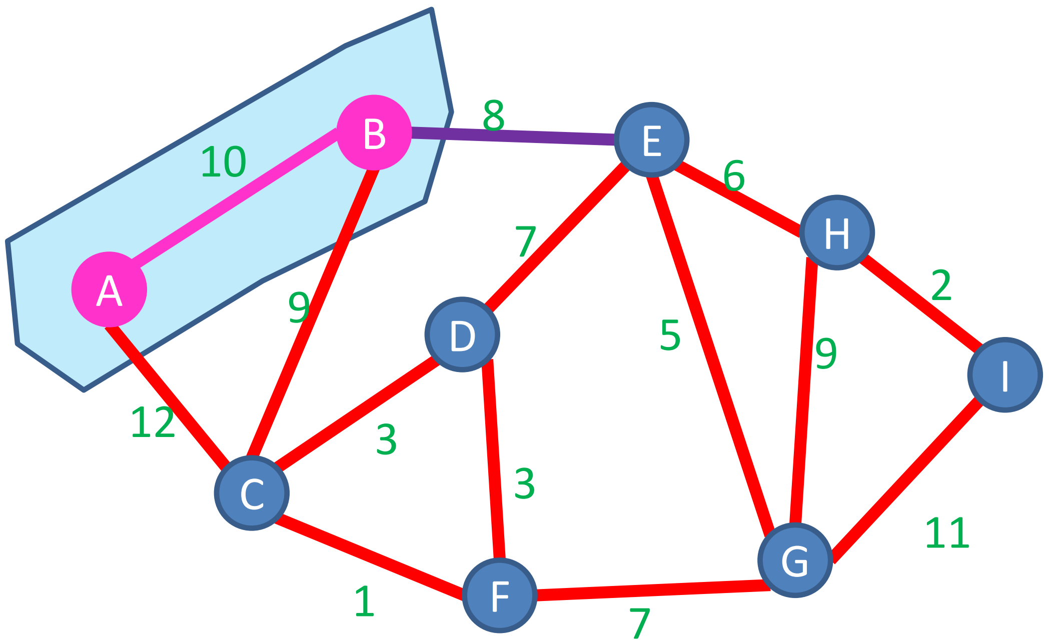
Prim’s Algorithm

Start with an empty tree A

Pick a start node

Repeat $V - 1$ times:

Add the min-weight edge which connects to node in A with a node not in A



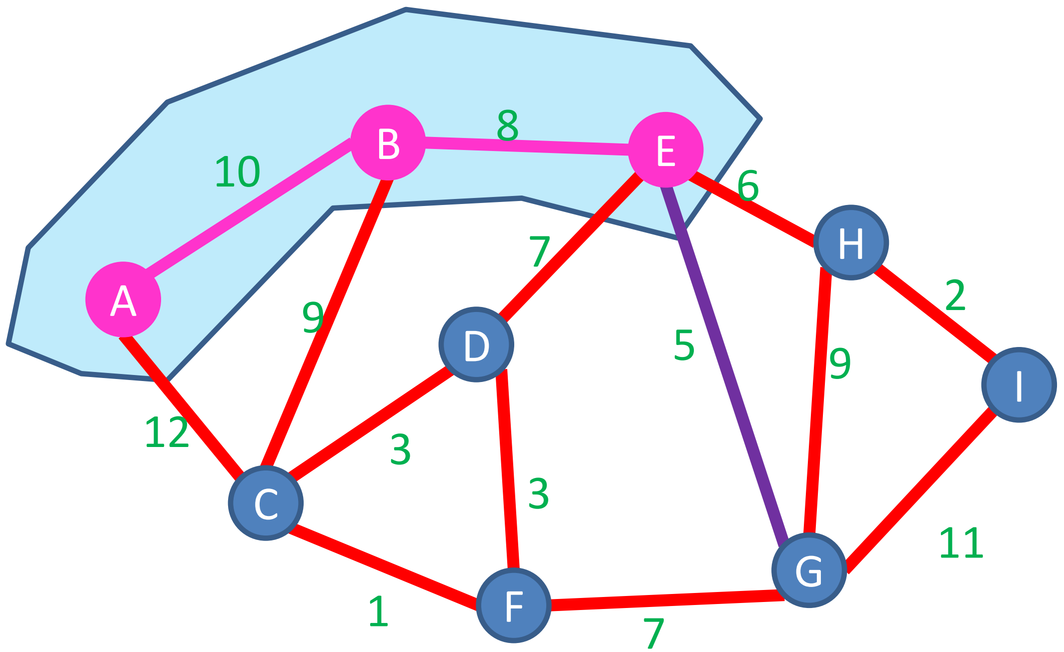
Prim’s Algorithm

Start with an empty tree A

Pick a start node

Repeat $V-1$ times:

Add the min-weight edge which connects to node in A with a node not in A



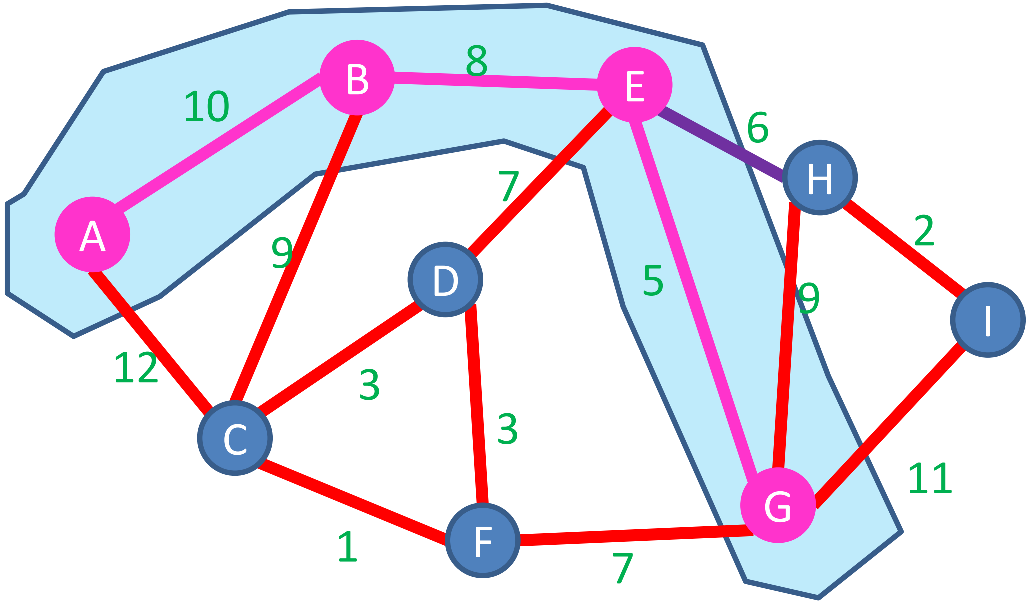
Prim's Algorithm

Start with an empty tree A

Pick a start node

Repeat $V - 1$ times:

Add the min-weight edge which connects to node in A with a node not in A



Prim's Algorithm

Start with an empty tree A

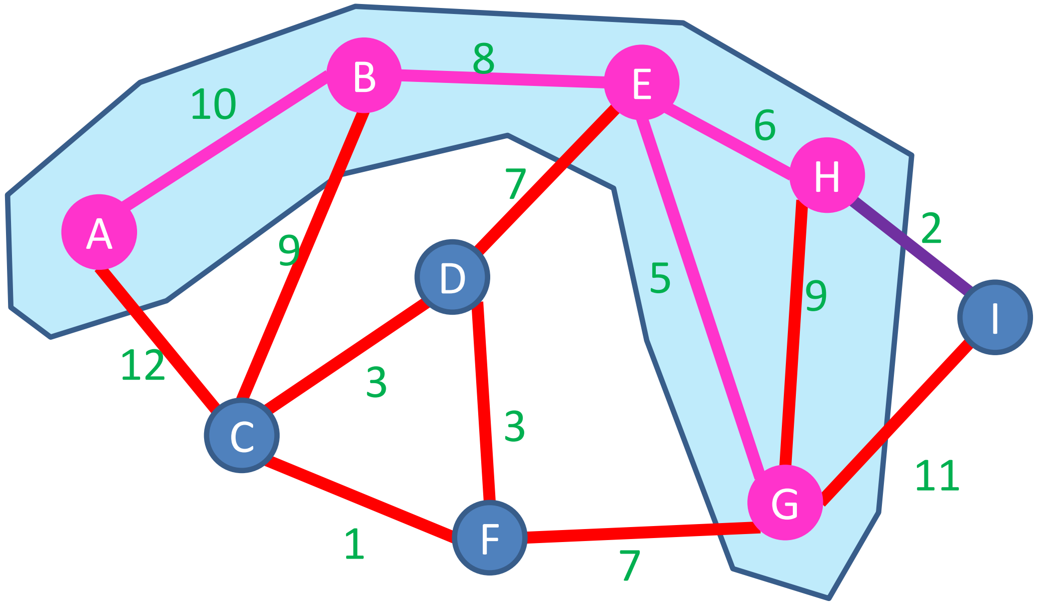
Pick a start node

Repeat $V - 1$ times:

Add the min-weight edge which connects to node in A with a node not in A

Keep edges in a Heap

$O(E \log V)$



Summary of MST results

- Fredman-Tarjan ‘84: $\Theta(E + V \log V)$
- Gabow et al ‘86: $\Theta(E \log \log^* V)$
- Chazelle ‘00: $\Theta(E\alpha(V))$
- Pettie-Ramachandran ’02: $\Theta(?)$ (optimal)
- Karger-Klein-Tarjan ‘95: $\Theta(E)$ (randomized)

- [read and summarize any/all for EC]