Graph App.

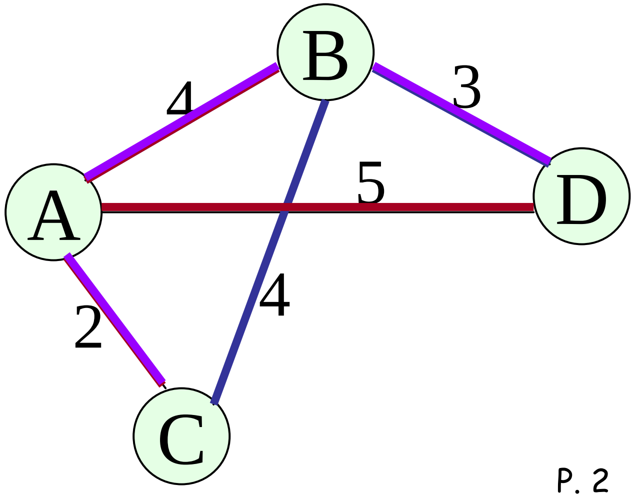
- **Topological Sort**
- **Spanning Tree**
  - Minimum Spanning Tree
- **Shortest Paths**

Minimum Spanning Tree: *Definition*

Data Structures

- **Cost of spanning tree**
  – Sum of the edge weights on a spanning tree
- **A *minimum spanning tree* of a connected undirected graph has a minimal edge-weight sum**
  – A particular graph could have *several* minimum spanning trees

DFS: 4+4+3=11
BFS: 4+2+5=11
MST: 4+2+3=9



Minimum Spanning Trees: *Algorithms*

- **Find a *minimum spanning tree* that begins at any given vertex** **[Robert Prim, 1957]**
  1. Find the least-cost edge (*v*, *u*) from a visited vertex *v* to some unvisited vertex *u*
  2. Mark *u* as visited
  3. Add the vertex *u* and the edge (*v*, *u*) to the *minimum spanning tree*
  4. Repeat the above steps until all vertices are visited

Prim's Algorithm

Minimum Spanning Tree (MST): AC AB BD

BD BC AC

PrimAlgorithm(Vertex v)

Mark v as visited; count=0;

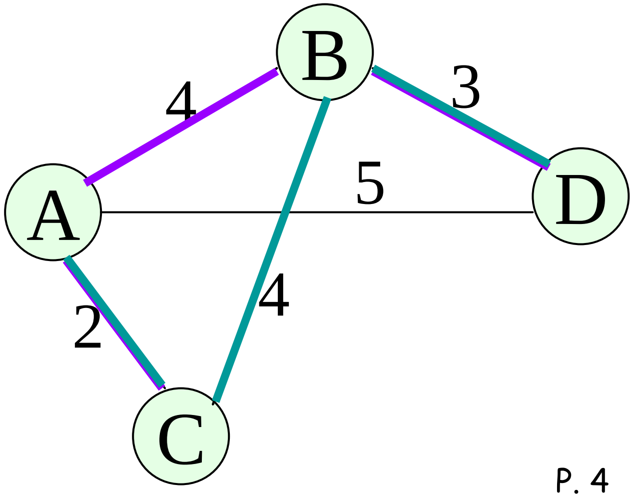
while (count < |V|-1)

(v,u) = the least-cost edge from visited to unvisited

Mark u as visited;

Add ***(v,u)*** into MST;

count++;



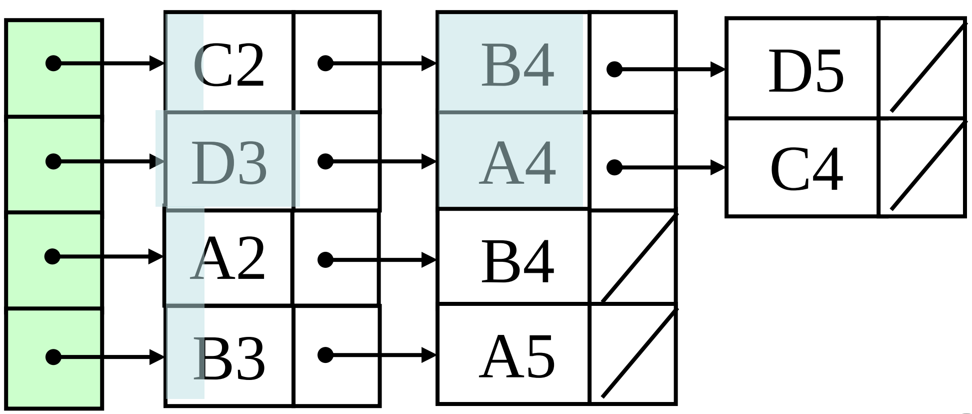
Prim's Algorithm

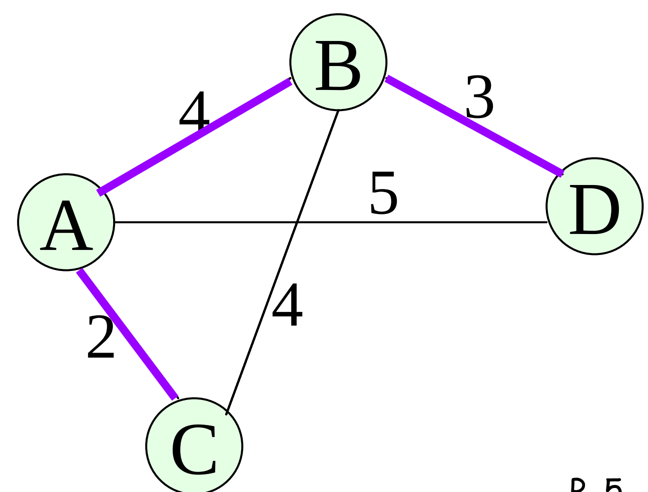
Minimum Spanning Tree (MST): AC AB BD

Visited

T	A
T	B
T	C
T	D

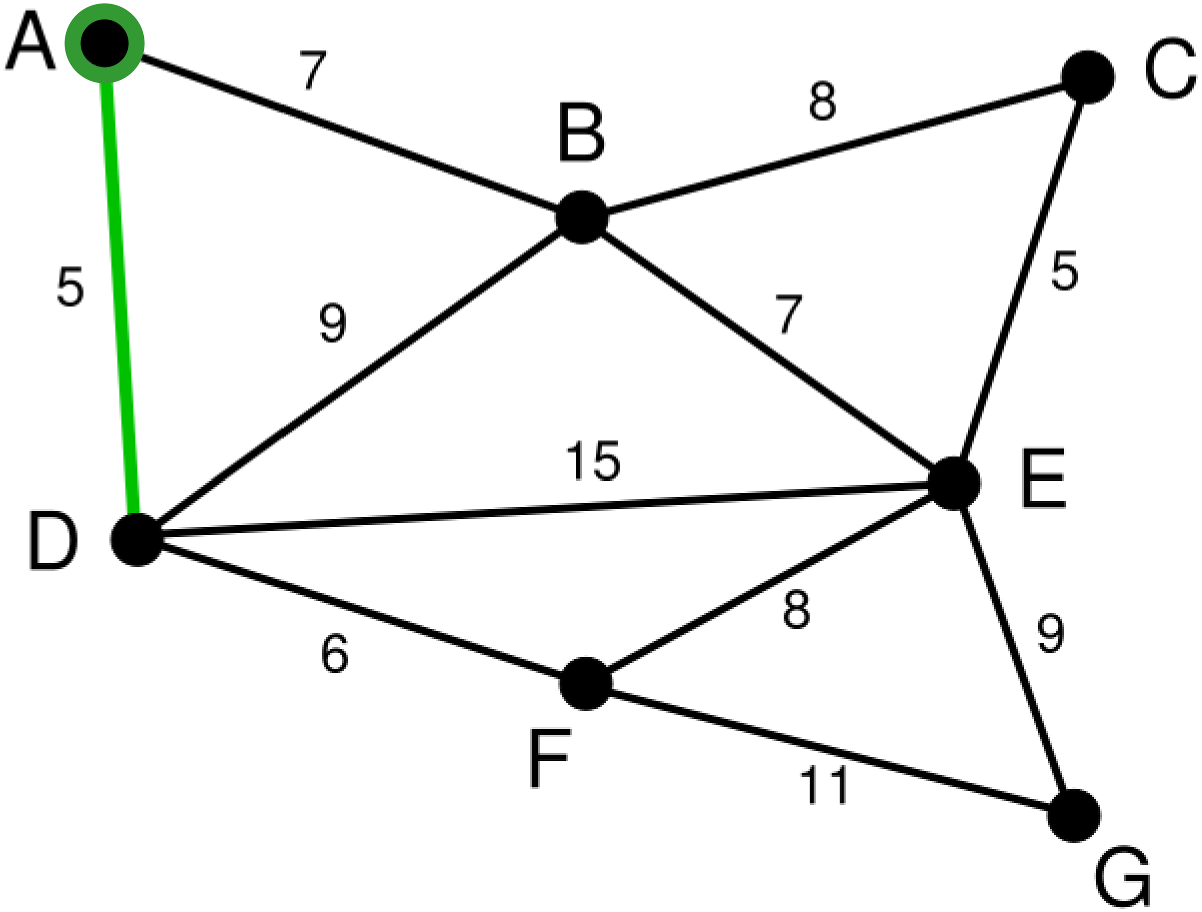
adjacency lists





Q&A: Which data structure would be helpful?

Practice 6: *Prim's Algorithm*



wikipedia ©

Minimum Spanning Trees: *Algorithms*

- **Find a *minimum spanning tree* that begins at any given vertex** **[Joseph Kruskal, 1956]**

1. Create a forest, where each vertex is a tree
2. Find the least-cost edge (*v*, *u*) where vertex *v* and vertex *u* are from two different trees
3. Merge the trees of vertex *v* and vertex *u*, and add the edge (*v*, *u*) to the *minimum spanning tree*
4. Repeat the above steps until |V|-1 edges

Kruskal’s Algorithm

Minimum Spanning Tree (MST): AC BD AB

KruskalAlgorithm()

Assign a unique label to each vertex; count=0;

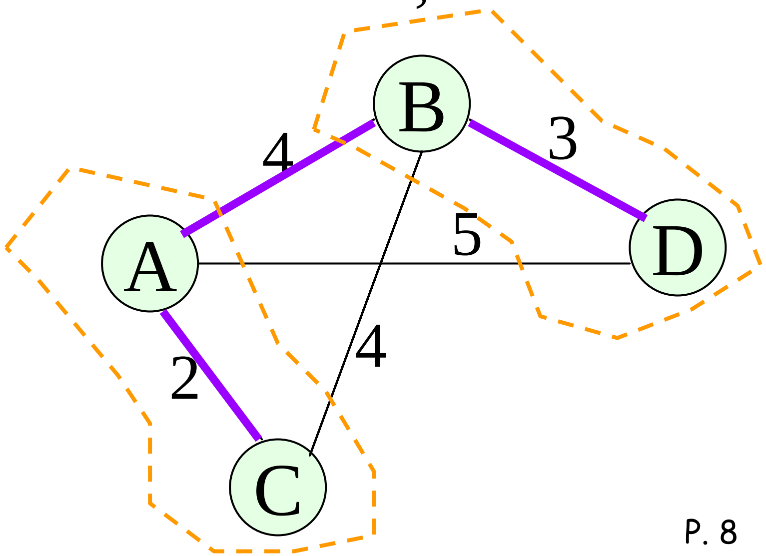
while (count < |V|-1)

(v,u)= the least-cost edge of two vertices with different labels

Assign min(u,v) to all vertices with these two labels;

Add ***(v,u)*** into MST;

count++;



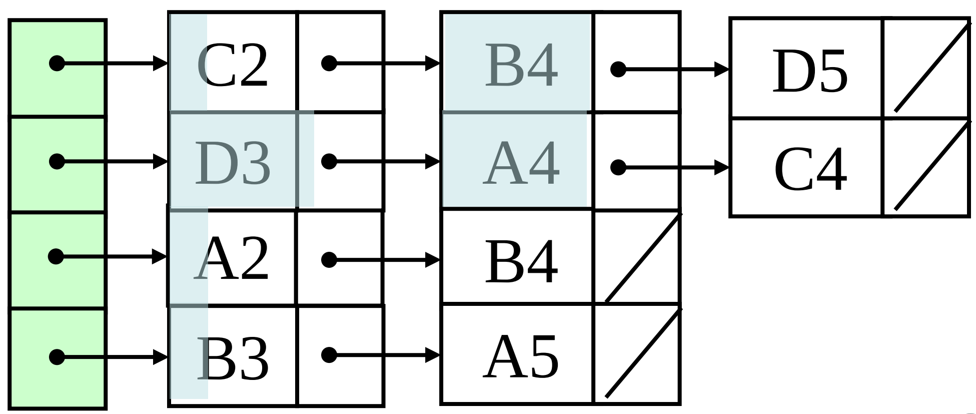
Kruskal's Algorithm

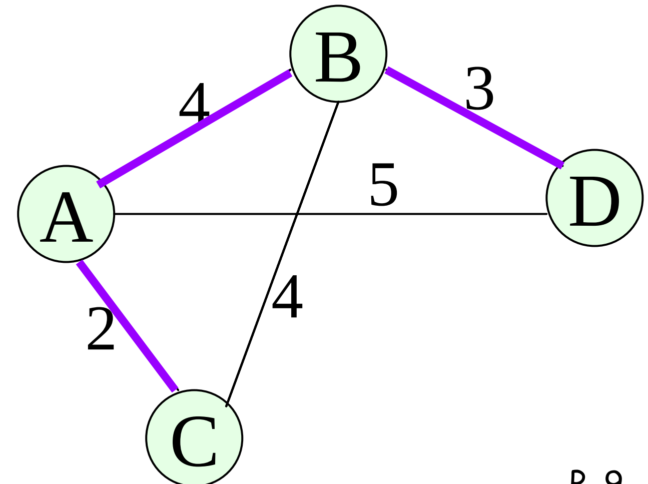
Minimum Spanning Tree (MST): AC BD AB

assigned labels

adjacency lists

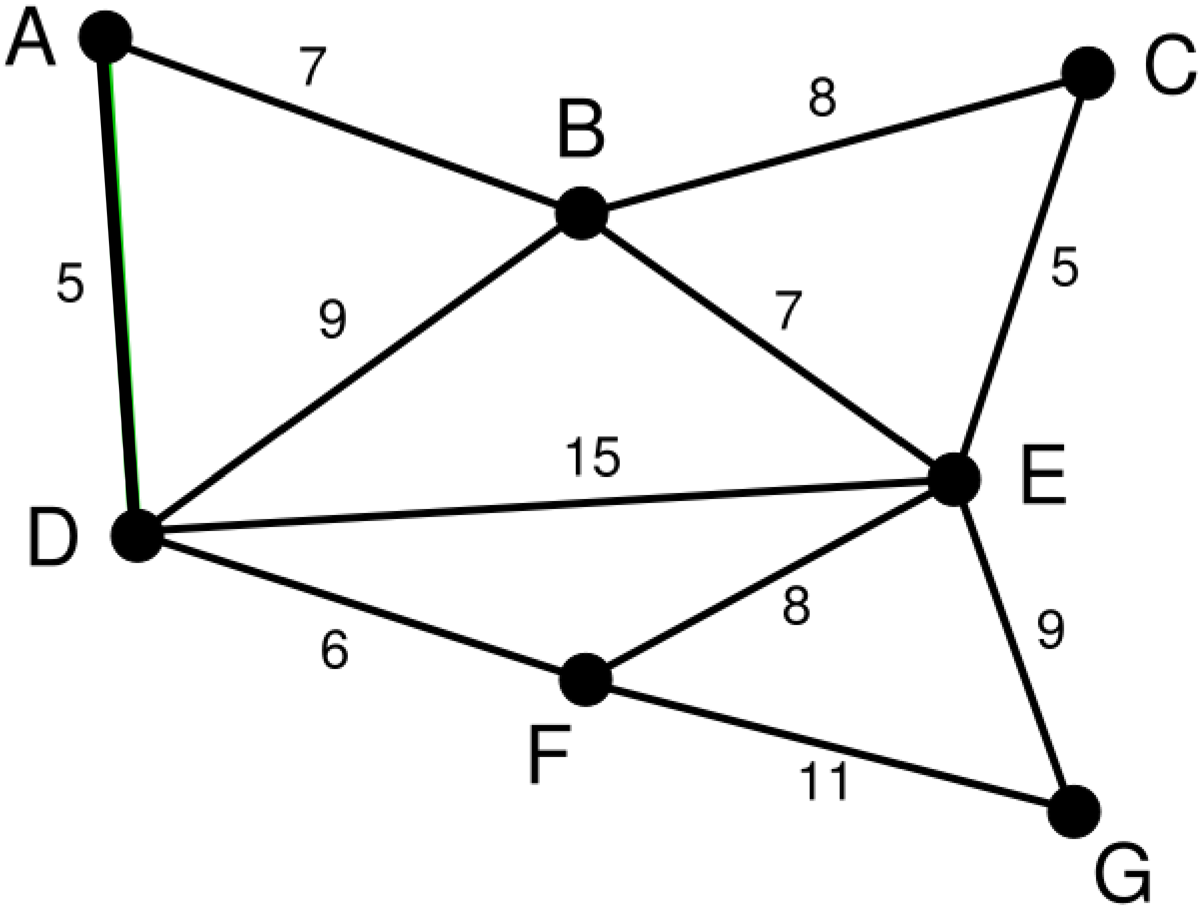
1 A
1 B
1 C
1 D





Q&A: Which data structure would be helpful?

Practice 7: *Kruskal's Algorithm*



Minimum Spanning Trees: *Algorithms*

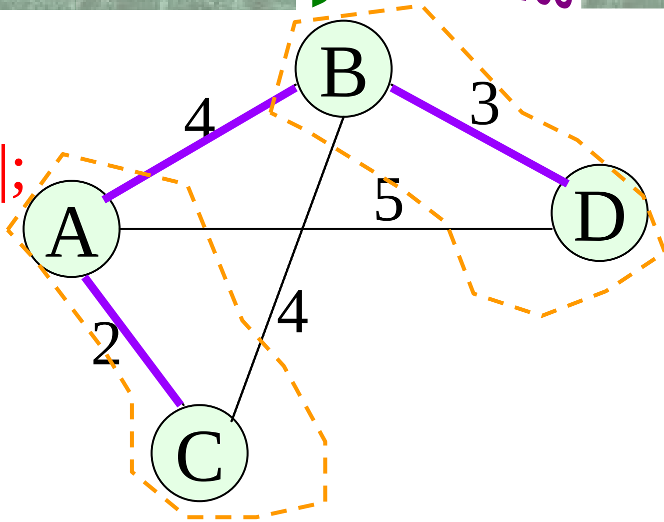
- **Find a *minimum spanning tree* that begins at any given vertex** **[Otakar Borůvka, 1926] [Sollin, 1965]**
  1. Create a forest, where each vertex is a tree
  2. For each tree T, do the following steps:

     2.1 Find the least-cost edge (*v*, *u*) where vertex *v* is in T and vertex *u* is outside T

     2.2 Merge the trees of vertex *v* and vertex *u*, and add the edge (*v*, *u*) to the *minimum spanning tree*
  3. Repeat step 2 until only one tree is left

Sollin's Algorithm

MST: AC BD AB



SollinAlgorithm()
Assign a unique label to each vertex; size = |V|;
while (size > 1)
Initialize Edges[1..size] *as empty sets;*
for each vertex v
L = v.label;
(v,u) = *the least-cost edge from v to u for any vertex with a different label;*
if (Edges[L].weight > **(v,u).weight**)
Edges[L] = **(v,u)**;
for each edge (v,u) in Edges but not in MST
Assign min(v.label, u.label) to vertices in the sets of v and u;
Add (v,u) to MST; size--;

Sollin's Algorithm

Minimum Spanning Tree (MST): AC BD AB

size = 2

Sets

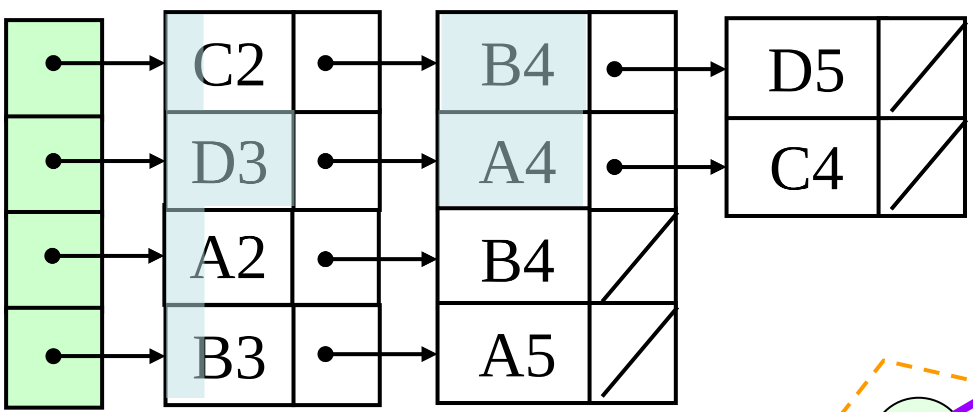
1 A

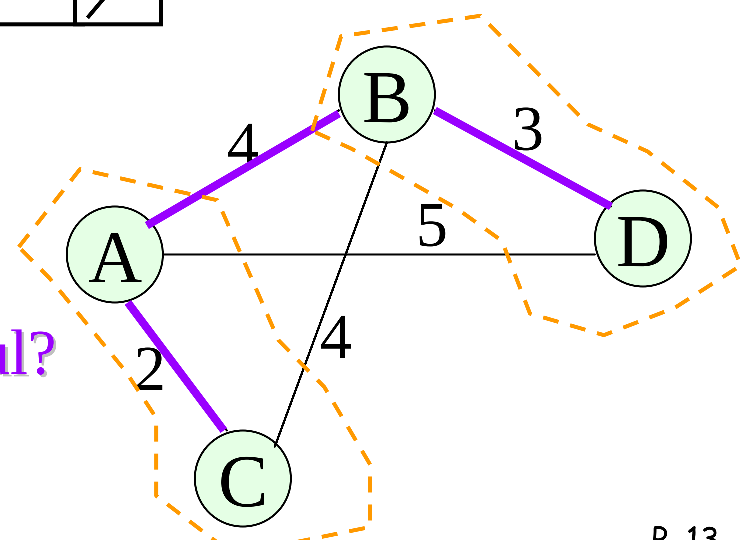
1 B

1 C

1 D

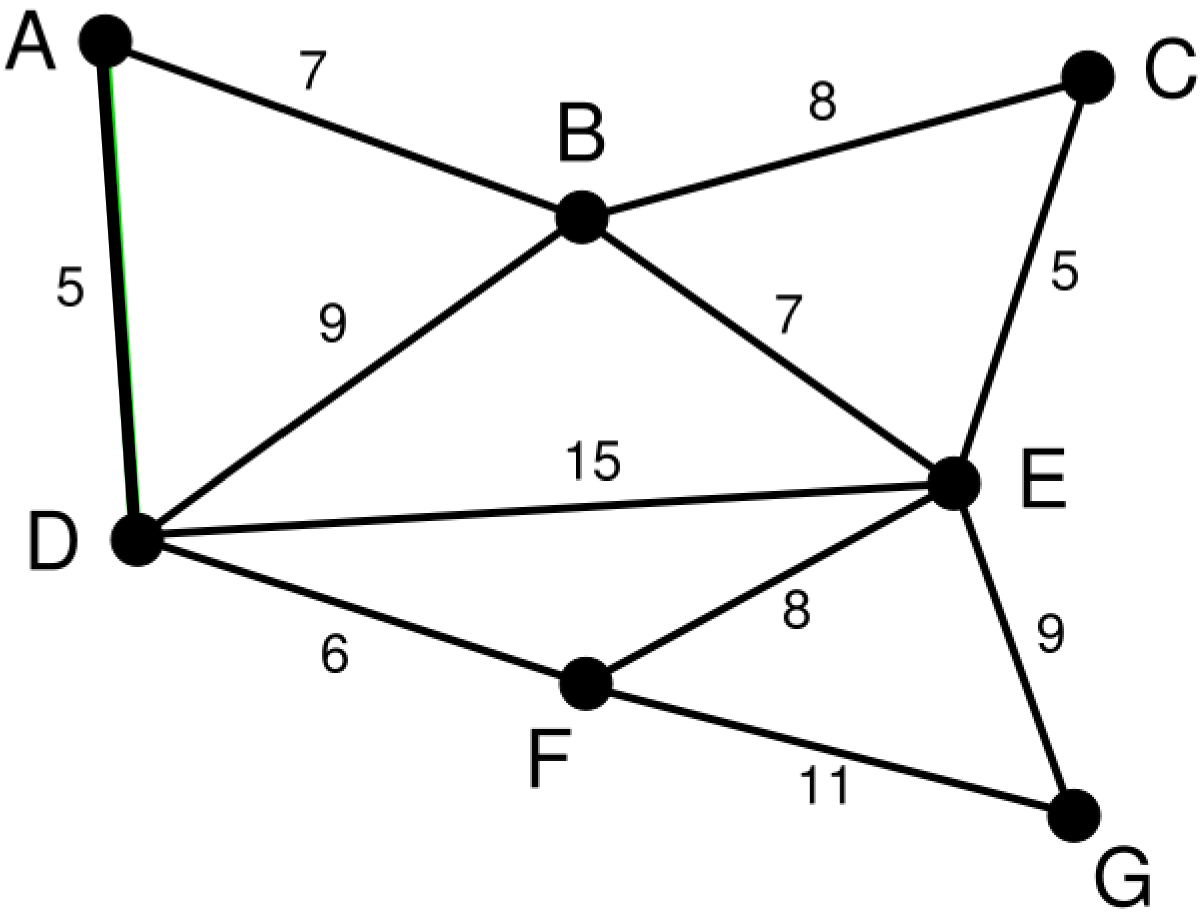
adjacency lists





Q&A: Which data structure would be helpful?

Practice 8: *Sollin's Algorithm*

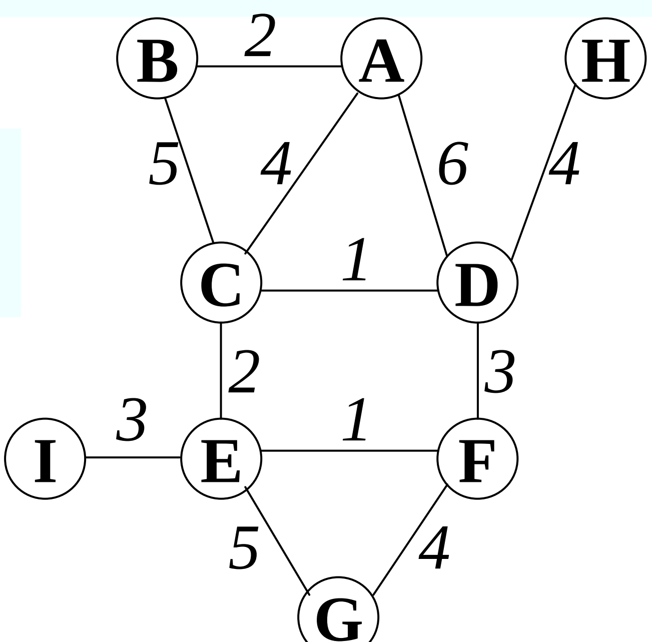


Self-exercise 5

1. Consider the **minimum spanning tree** in the graph to answer the following:

(a) Using ***Prim's algorithm*** by starting at vertex **A**, write the order of visiting vertices.

(b) What is the **Prüfer sequence** of the MST?



If you have multiple choices,

visit the vertex with the smallest label first.