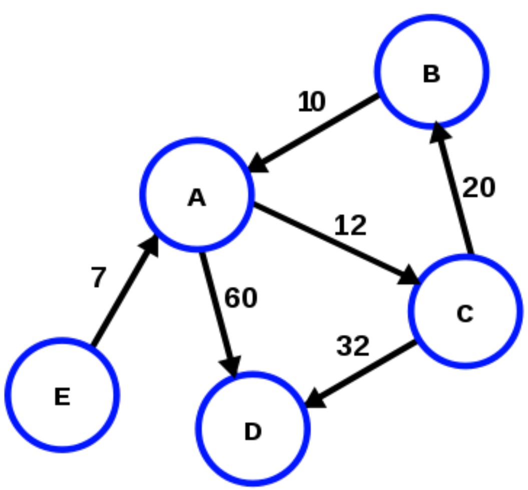
Graphing

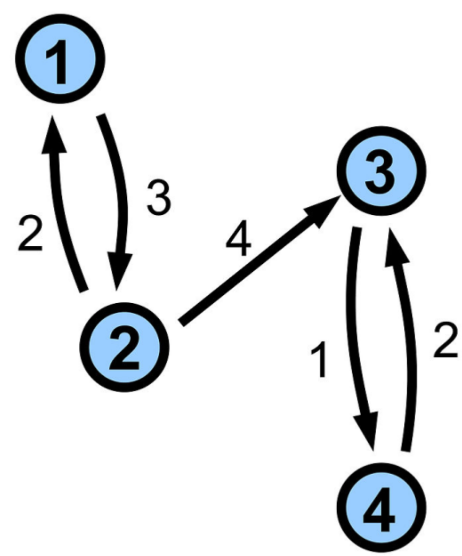
Problem Solving Club

Oct 26, 2016

What is a graph?

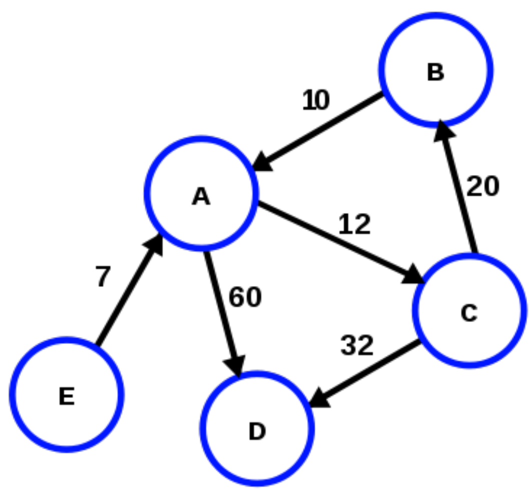
- A graph is a set of vertices and edges
- Edges can be:
  - One-directional (directed) or bidirectional
  - Weighted or unweighted





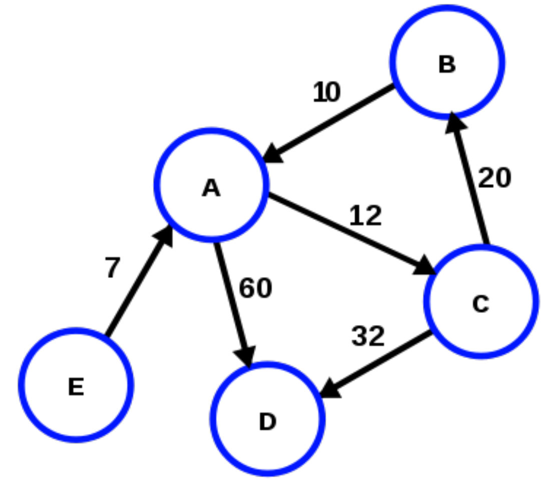
Graph representation #1
Adjacency matrix

	A	B	C	D	E
A			12	60	
B	10				
C		20	32		
D					
E	7				



- What is the main problem?
- Large waste of space. What is the space usage in terms of V and E?
- Requires $O(V^2)$ space

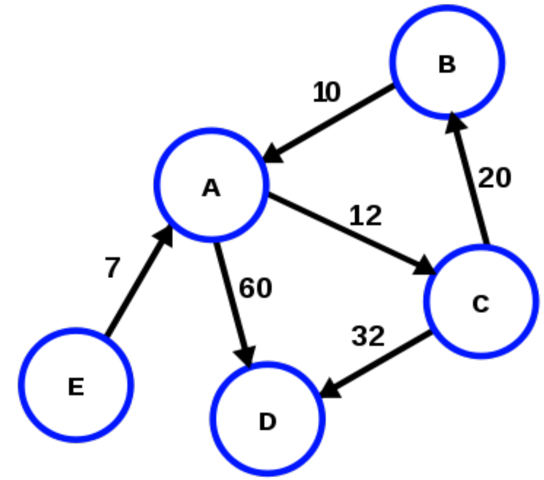
Graph representation #2
Adjacency list



- A: {(C, 12), (D, 60)}
- B: {(A, 10)}
- C: {(B, 20), (D, 32)}
- D: {}
- E: {(A, 7)}
- What are some advantages over adjacency matrix?
- Uses less space. Faster to iterate over graph.
- What are some disadvantages?

Graph representation #3
Edge list

- {(A, C, 12), (A, D, 60), (B, A, 10), (C, B, 20), (C, D, 32), (E, A, 7)}
- Why would you use this?
- Mainly for specialized algorithms, like Kruskal's (minimum spanning tree)
- What are the disadvantages?
- Cannot easily find neighbors, or if edge exists.



Summary: Graph representations

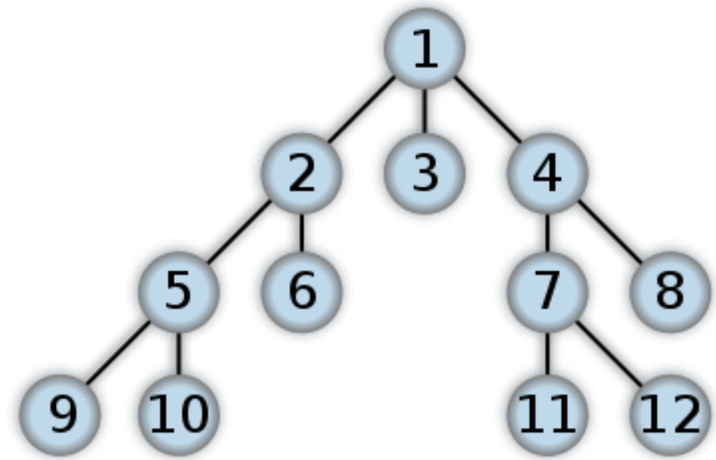
	Description	When to use in programming contest
Adjacency matrix	Store the graph in a matrix. Requires $O(V^2)$ space.	$V \leq 2000$
Adjacency list	For each vertex, store a list of adjacent vertices. Requires $O(V+E)$ space.	$V > 2000$
Edge list	For each edge, store an entry in a list.	When required by a specific algorithm, like Kruskal's (minimum spanning tree)

Breadth-first search (BFS)

- A common problem in graphs is shortest path. What is the shortest path to get between two vertices?
- Single source shortest path (SSSP) problem: From a given vertex, what is the shortest path to every other vertex in the graph?
- BFS is an algorithm that solves this on an unweighted graph
- It also determines which other vertices are reachable from a given vertex

Breadth-first search (BFS)

- The figure shows the order of vertex traversal in BFS



- Two data structures are required for BFS: a **queue** and a **boolean array**.
- The queue determines the order to visit vertices, and the boolean array keeps track of which vertices have already been visited

Breadth-first search (BFS)

- First, push the root vertex into the queue. Mark it as visited.
- While queue is nonempty:
  - Pop the front of the queue. Let this vertex be v.
  - Go through all edges of v. Let (v, w) be an edge.
    - If w is not visited yet, push it onto the back of the queue. Mark w as visited.