Methodic and Practical Foundations of Computer Science 1

11-Introduction to Graphs

Sayed Ahmad Sahim

Kandahar University
Computer Science Faculty

algprog17@gmail.com

November 21, 2017

Objectives

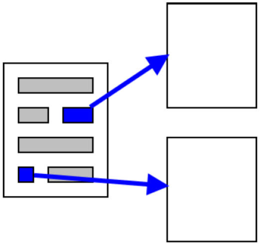
1. Introduction to Graphs
   - Formal Definition
   - Representing Graphs
   - Graph Exploration
2. Question

Motivation

- Represents connection between objects.
- Describe many important phenomena.

Internet

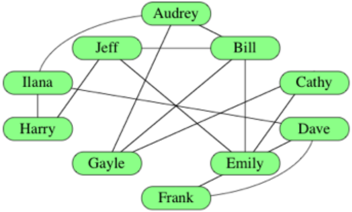
Web pages are connected by links.



This is important for Google's page rank.

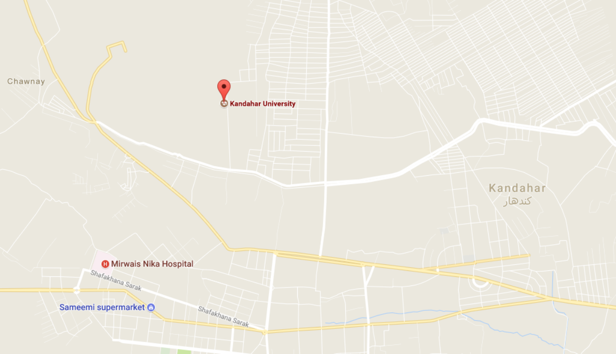
Social

People connected by friendships.



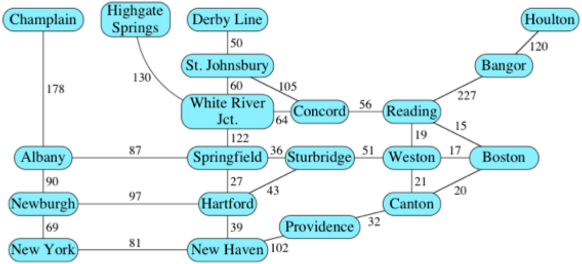
Map

Intersections connected by roads.



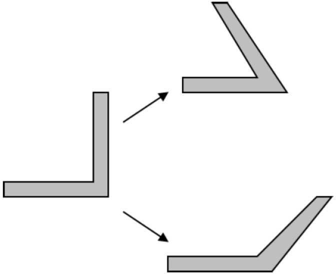
Map

Vertices's, edge and weight



Configuration Spaces

Possible configurations connected by motions



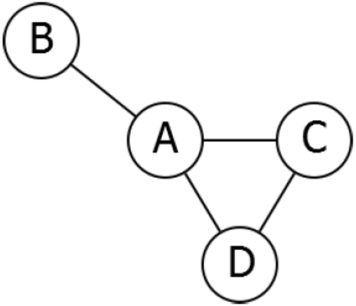
Formal Definition

Definition

An (Undirected) Graph is a collection V of vertices's, and a collection of E edges each of which connects a pair of vertices's.

Drawing Graphs

Vertices: Points. Edges: Lines.

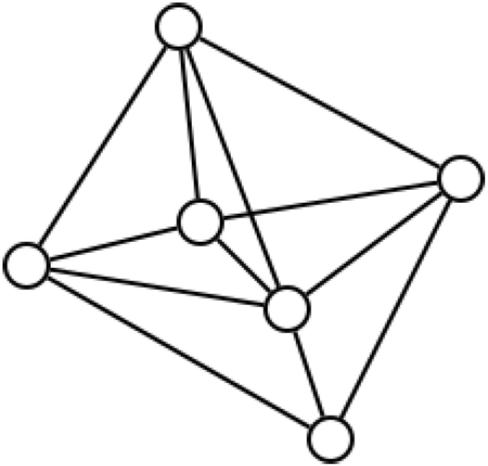


Vertices's: A,B,C,D

Edges: (A, B), (A, C), (A,D), (C,D)

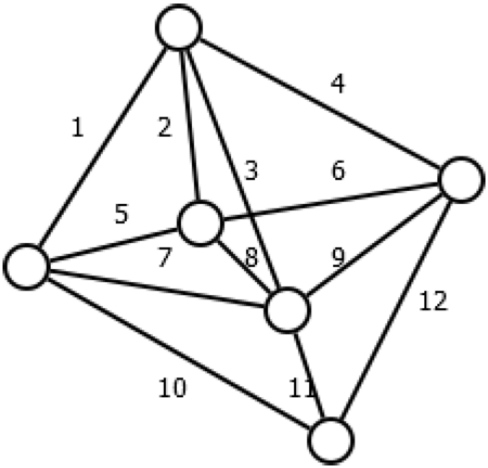
Problem

How many edges are in the graph given below?



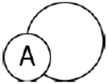
Answer

There are 12 edges

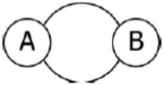


Loops and Multiple Edges

Loops connect a vertex to itself.



Multiple edges between same vertices.



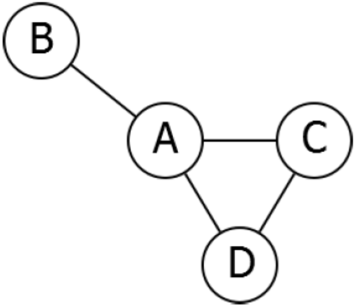
If a graph has neither, it is simple.

Representing Graphs

To compute things about graphs we first need to represent them.
There are many ways to do this.

Edge List

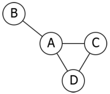
List of all edges:



Edges: (A, B), (A, C), (A,D), (C,D)

Adjacency Matrix

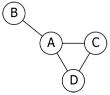
Matrix. Entries 1 if there is an edge, 0 if there is not.



	A	B	C	D
A	0	1	1	1
B	1	0	0	0
C	1	0	0	1
D	1	0	1	0

Adjacency List

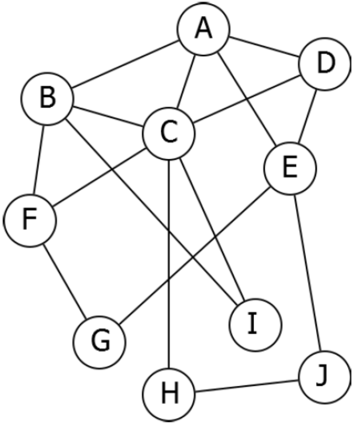
For each vertex, a list of adjacent vertices.



A adjacent to B, C,D
B adjacent to A
C adjacent to A,D
D adjacent to A, C

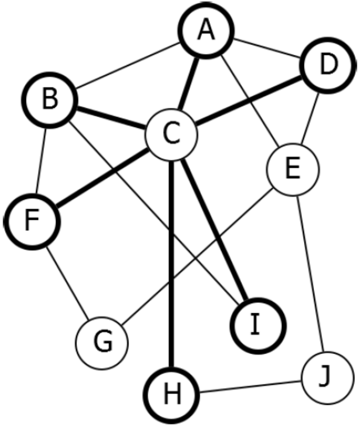
Problem

What are the neighbors of C?



Solution

A,B,D,F ,H,I .



Algorithm Runtime

Graph algorithm runtimes depend on $|V|$ and $|E|$.
For example, $O(|V| + |E|)$ (Linear time),
$O(|V| + |E|), O(|V|^{3/2})$,
$O(|V| log(|E|) + |E|)$.

Density

Which is faster, $O(|V|^{3/2}$ or $O(|E|)$?

Density

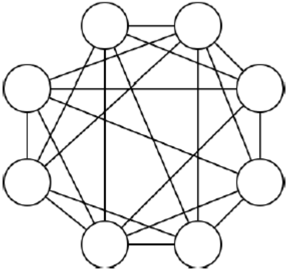
Which is faster, $O(|V|^{3/2}$ or $O(|E|)$?
Depends on graph! Depends on the density, namely how many edges you have in terms of the numbers of vertices's.

Dense Graphs

In dense graphs, $|E| \approx |V|^2$

Dense Graphs

In dense graphs, $|E| \approx |V|^2$



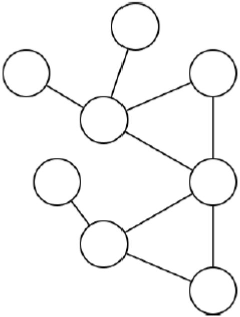
A large fraction of pairs of vertices's are connected by edges.

Sparse Graphs

In sparse graphs, $|E| \approx |V|$

Sparse Graphs

In sparse graphs, $|E| \approx |V|$



Each vertex has only a few edges.

Motivation

You're playing a video game and want to make sure that you've found everything in a level before moving on.
How do you ensure that you accomplish this?

Examples

This notion of exploring a graph has many applications:

- Finding routes
- Ensuring connectivity
- Solving puzzles and mazes

Paths

We want to know what is reachable from a given vertex.

Definition

A path in a graph **G** is a sequence of $vertice's V_0, V_1, ..., V_n$ so that for all i, $(V_i, V_i + 1)$ is an edge of **G**.

Formal Description

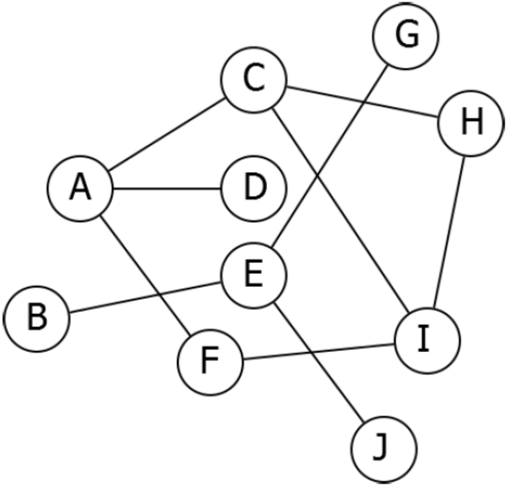
Reachability

Input: Graph **G** and vertex **S**

Output: The collection of vertices's **v** of **G** so that there is a path from s to v.

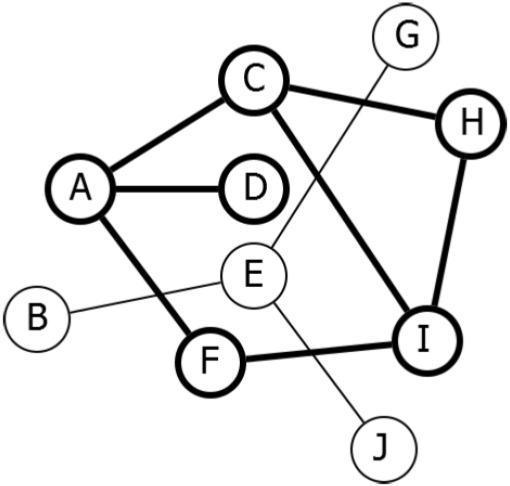
Problem

Which vertices are reachable from A?



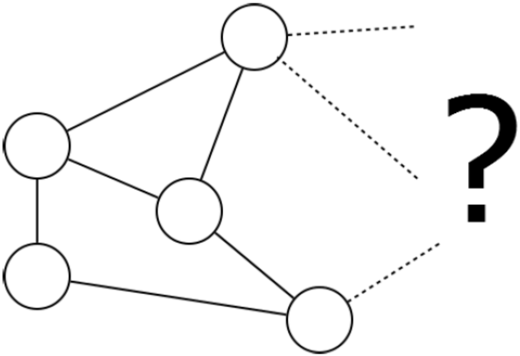
Solution

A,C,D,F,H,I .



Basic Idea

We want to make sure that we have explored every edge leaving every vertex we have found.



Pseudocode

We want to make sure that we have explored every edge leaving every vertex we have found.

Component(s)

DiscoveredNodes $\leftarrow$ s
while there is an edge e leaving
DiscoveredNodes that has not been explored:
add vertex at other end of e to DiscoveredNodes
return DiscoveredNodes

Question