CSC 172– Data Structures and Algorithms

Lecture #22

Spring 2018

Please put away all electronic devices

Announcements

- Project 3 due tomorrow (04/13)
- Project 4 will be out next week.
  - You may work with a partner

Announcements

- Optional Project:
  - Any game of your choice!
  - Maximum team size: 4
  - Criteria:
    - A Good GUI
    - Have both components:
      - User Interaction
      - Computer's Suggestion
    - Preferably related to Graphs (our next data structure)
    - You need to meet me with a one-page written proposal for approval/suggestions before you proceed
    - Each member may get between 2 pts and 6 pts for completing this project.
    - The adjustment would be applied without harming other's grades.
    - You need to demo your project during reading period.

Should you try this optional project

- You should if:
  - You love coding and hate exams
  - You think the course is too easy
  - You want to build your own project
  - You have done poorly in Project 1 and 2 but you think you have the potential (and I know all of you have!)

How should you form the team

- Use Piazza
- Ideally, half of the team should work on GUI and other half on Algorithm before you put them together.
- Designing and Distribution of work are the key components.
-

Before We Start!

- Data Structures we have covered:
  - Arrays
  - Lists
  - Trees
    - Heap
    - Binary Search Tree
    - Balanced Binary Search Tree
  - Stack
  - Queues
    - Priority Queues
  - Maps
    - Tree Maps
    - Hash Maps (We are yet to cover the implementation details)

REAL LIFE EXAMPLES

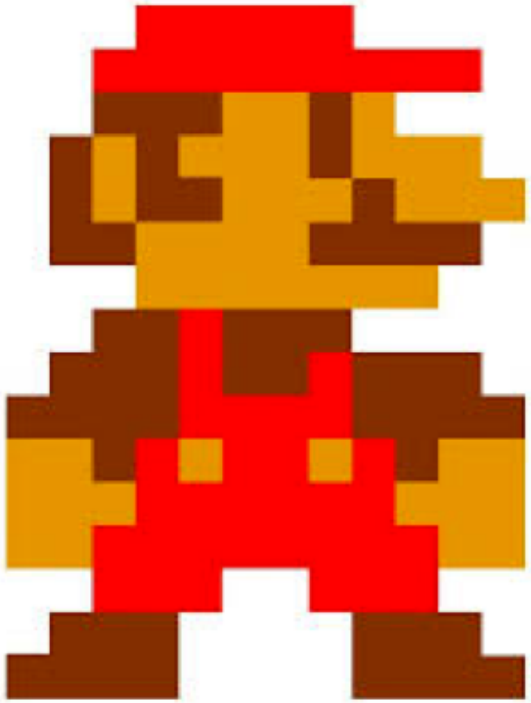
Undo/Redo operations in a word processor.

Evaluate an expression

$$\frac{3x + 27y + 15z}{x + 9y + 5z} = ?$$

Storing an image

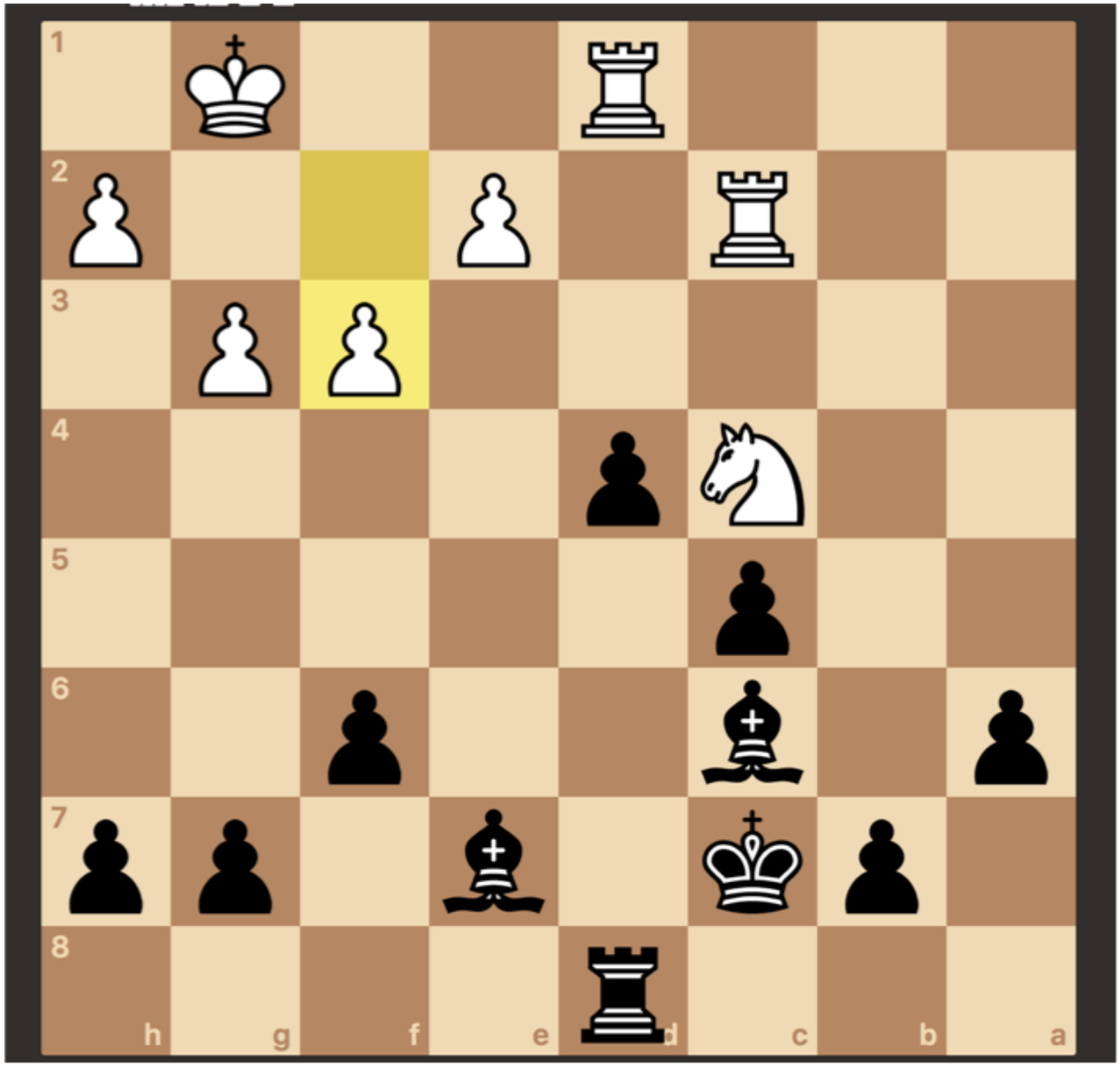
- (1000 by 1000 pixels) as a bitmap.



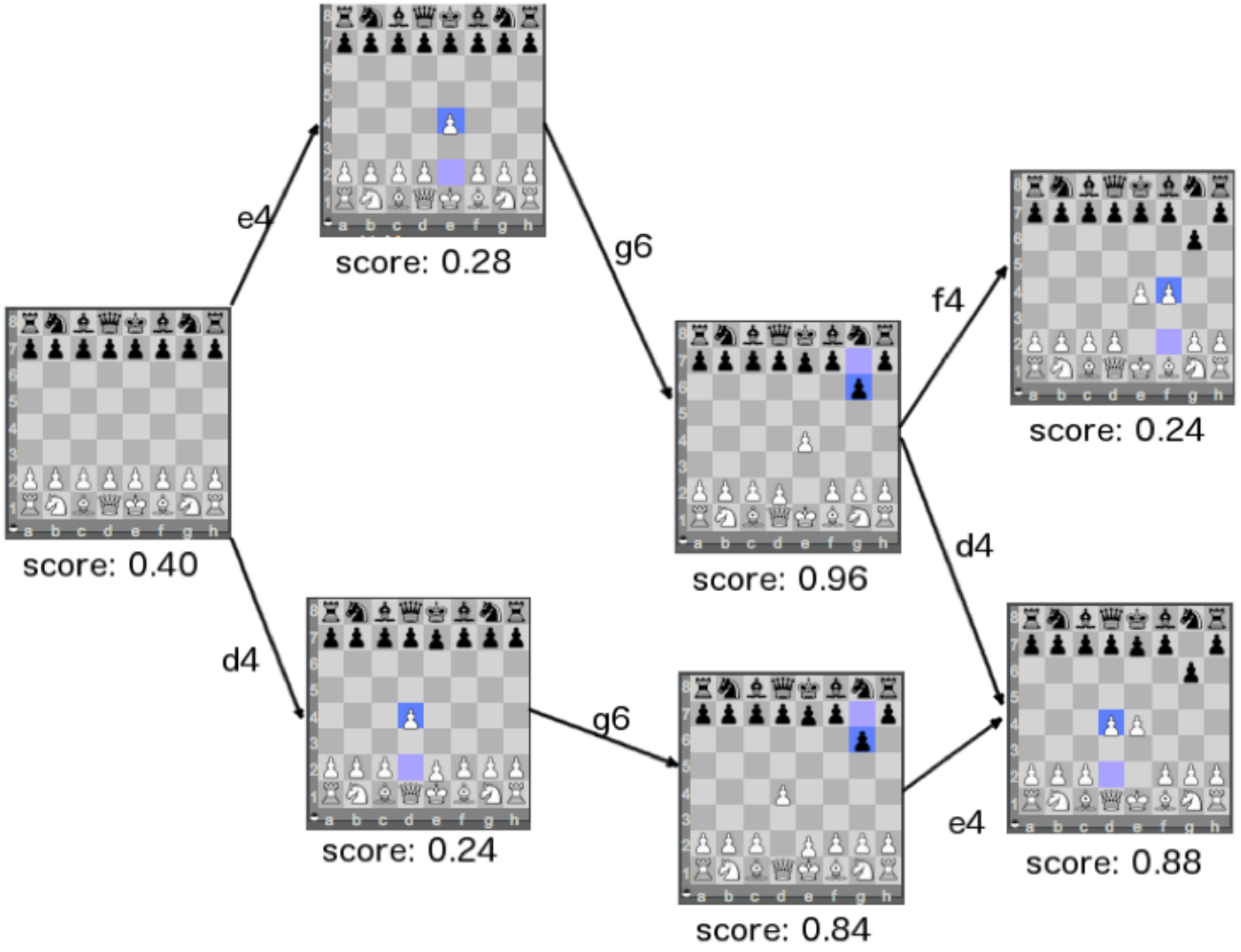
Implementing Printer Scheduler

- so that jobs can be printed in the order of their arrival.

Thinking Ahead in a Chess game



Thinking Ahead in a Chess game



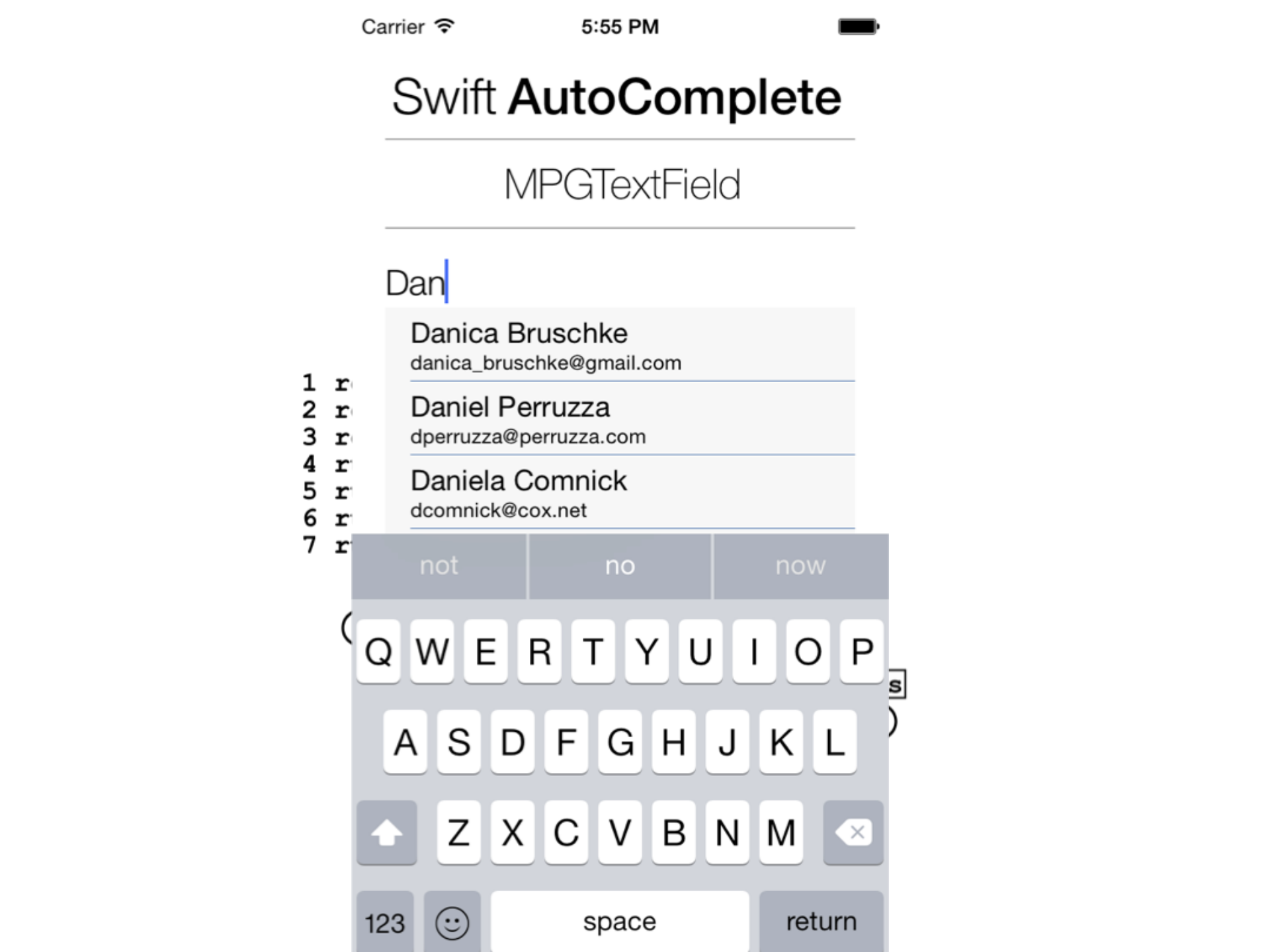
Operating System: Page Replacement Algorithm

- Finding a ‘Page’ which is Most Recently Used (MRU)

In a operating systems that use paging for memory management, page replacement algorithm are needed to decide which page needed to be replaced when new page comes in. Whenever a new page is referred and not present in memory, page fault occurs and Operating System replaces one of the existing pages with newly needed page. Different page replacement algorithms suggest different ways to decide which page to replace. The target for all algorithms is to reduce number of page faults.

Source: https://www.geeksforgeeks.org/operating-system-page-replacement-algorithm/

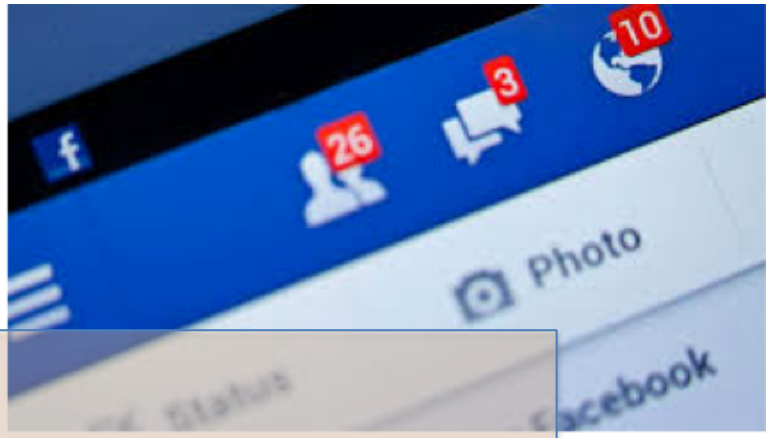
Last 4 digits of your SSN


Carrier
5:55 PM
Swift AutoComplete
MPGTextField
Dan
Danica Bruschke
danica_bruschke@gmail.com
Daniel Perruzza
dperruzza@perruzza.com
Daniela Comnick
dcomnick@cox.net
not
no
now
Q W E R T Y U I O P
A S D F G H J K L
Z X C V B N M
123
space
return

If you can't figure out what data structure you should use, probably you should use a Graph!!!

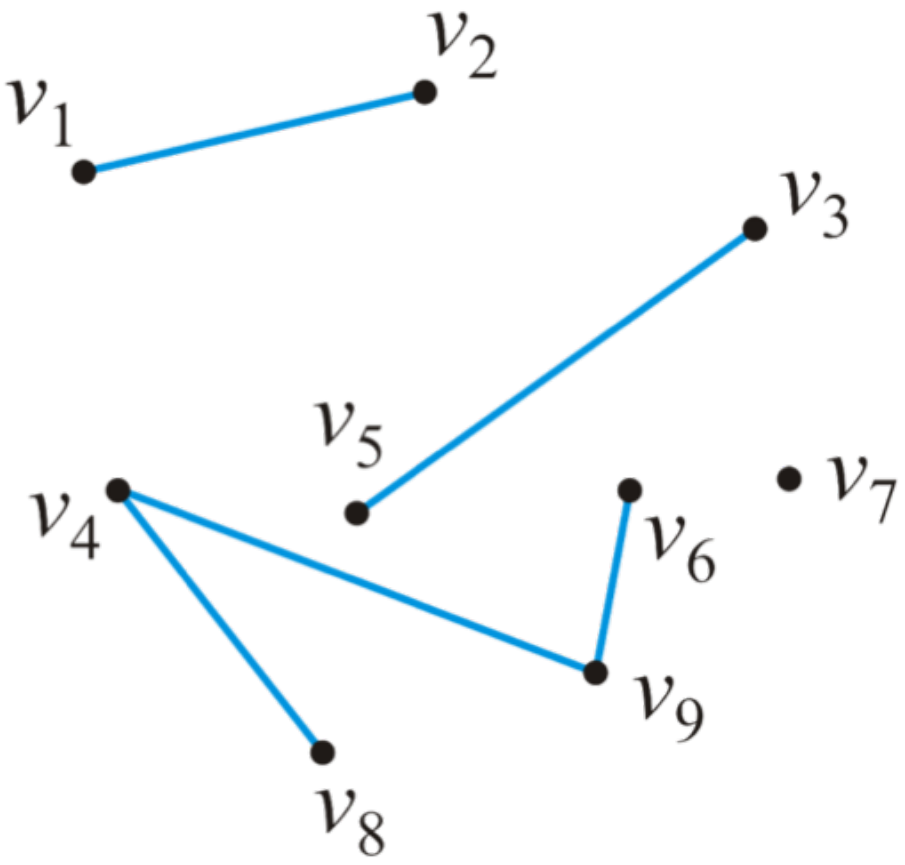
GRAPHS

Example

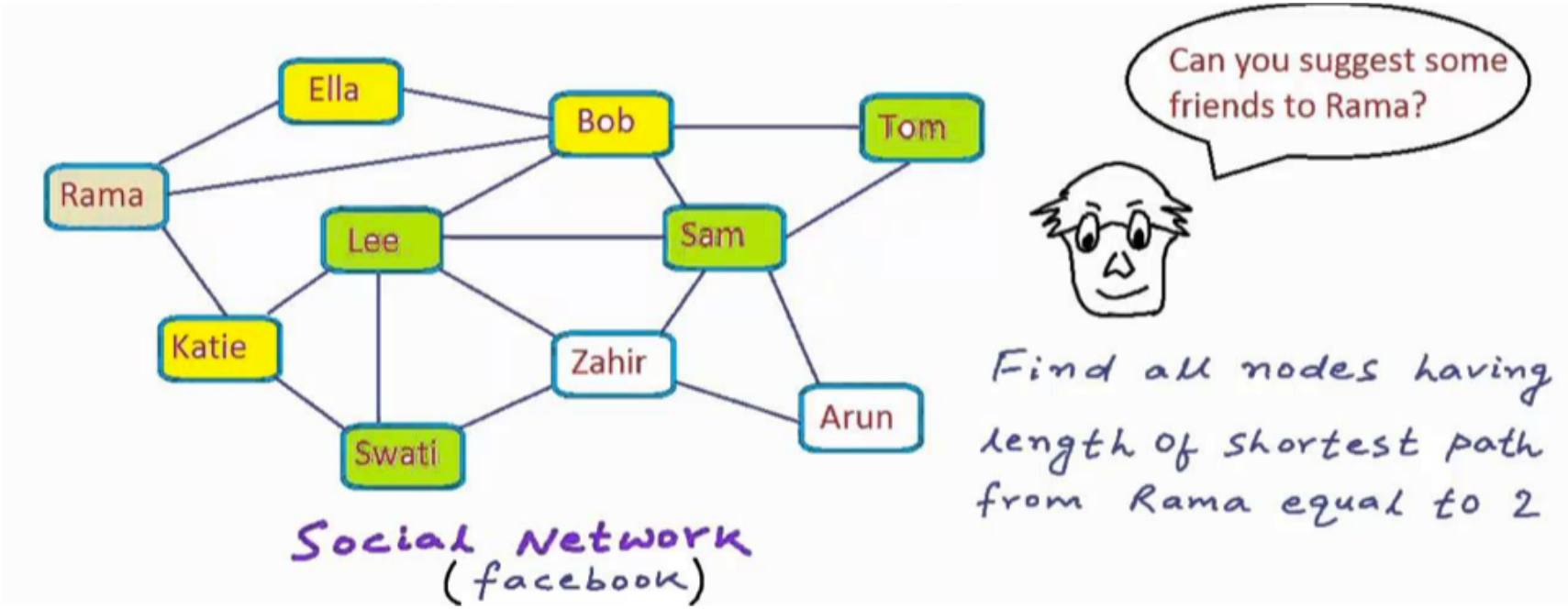


Friend Suggestions

Example of a Graph



Example



https://goo.gl/images/G7q4Y4

Outline

A graph is an abstract data type for storing adjacency relations

- We start with a few definitions:
  - Vertices, edges
- We will describe paths in graphs
  - Simple paths and cycles
- Definition of connectedness
- Weighted graphs

Outline

We will define an Undirected Graph ADT as a collection of *vertices*

$$V = \{v_1, v_2, \ldots, v_n\}$$

- The number of vertices is denoted by

$$|V| = n$$

- Associated with this is a collection E of unordered pairs $\{v_i, v_j\}$ termed *edges* which connect the vertices

There are a number of data structures that can be used to implement abstract undirected graphs

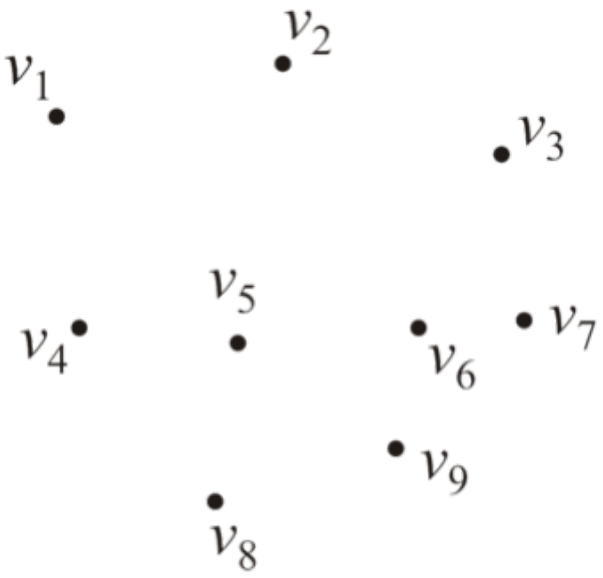
- Adjacency matrices
- Adjacency lists

Graphs

Consider this collection of vertices

$$V = \{v_1, v_2, \ldots, v_9\}$$

where $|V| = n = 9$

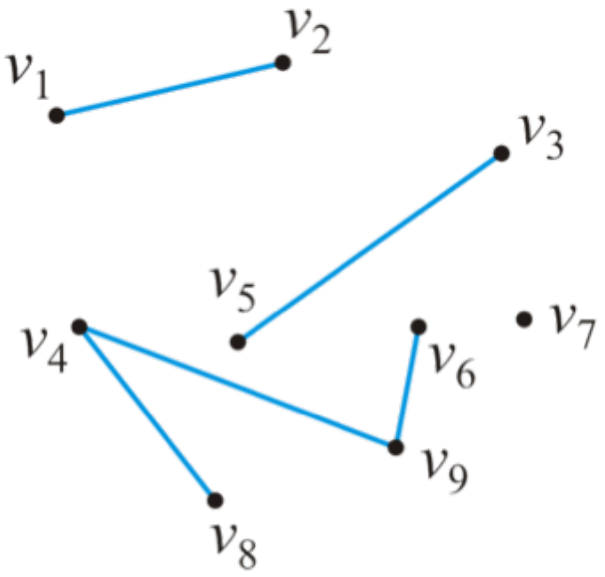


Undirected graphs

Associated with these vertices are $|E| = 5$ edges

$$E = \{\{v_1, v_2\}, \{v_3, v_5\}, \{v_4, v_8\}, \{v_4, v_9\}, \{v_6, v_9\}\}$$

– The pair $\{v_j , v_k\}$ indicates that both vertex v_j is adjacent to vertex v_k and vertex v_k is adjacent to vertex v_j



Undirected graphs

We will assume in this course that a vertex is never adjacent to itself

– For example, $\{v_1, v_1\}$ will not define an edge

The maximum number of edges in an undirected graph is

$$|E| \leq \binom{|V|}{2} = \frac{|V|(|V|-1)}{2} = \mathrm{O}(|V|^2)$$

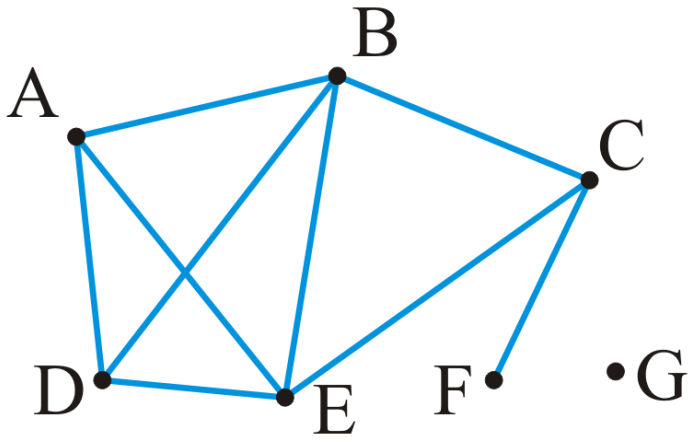
An undirected graph

Example: given the $|V| = 7$ vertices

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

and the $|E| = 9$ edges

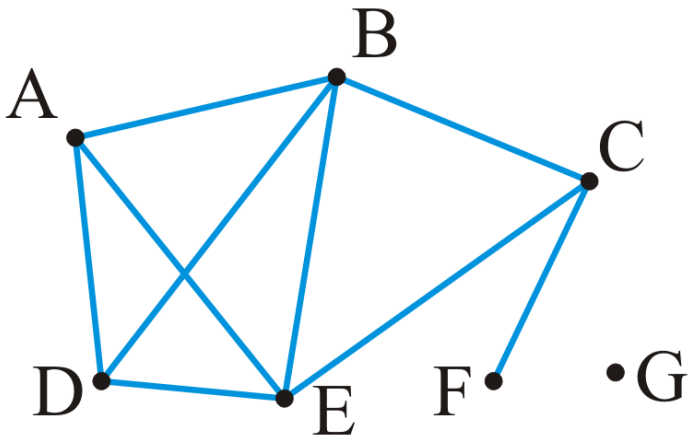
$$E = \{\{\text{A}, \text{B}\}, \{\text{A}, \text{D}\}, \{\text{A}, \text{E}\}, \{\text{B}, \text{C}\}, \{\text{B}, \text{D}\}, \{\text{B}, \text{E}\}, \{\text{C}, \text{E}\}, \{\text{C}, \text{F}\}, \{\text{D}, \text{E}\}\}$$



Degree

The degree of a vertex is defined as the number of adjacent vertices

$$\text{degree}(A) = \text{degree}(D) = \text{degree}(C) = 3$$
$$\text{degree}(B) = \text{degree}(E) = 4$$
$$\text{degree}(F) = 1$$
$$\text{degree}(G) = 0$$



Those vertices adjacent to a given vertex are its *neighbors*

Paths

A path in an undirected graph is an ordered sequence of vertices

$$(v_0, v_1, v_2, ..., v_k)$$

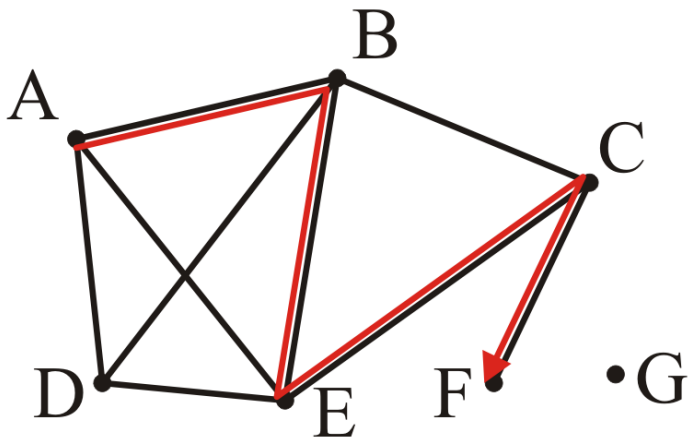
where $\{v_{j-1}, v_j\}$ is an edge for $j = 1, ..., k$

- Termed *a path from* v_0 to v_k
- The length of this path is k

Paths

A path of length 4:

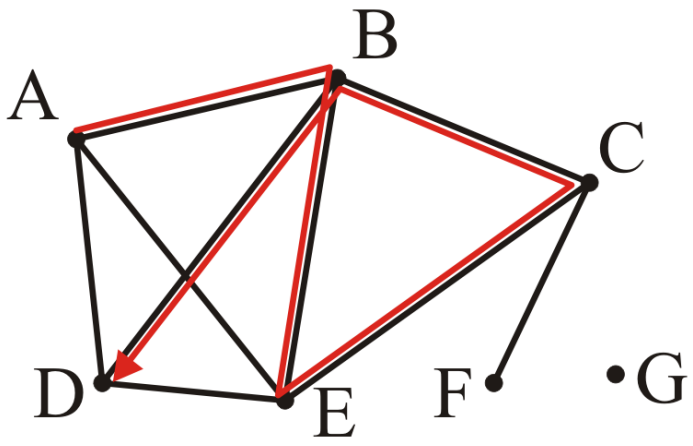
(A, B, E, C, F)



Paths

A path of length 5:

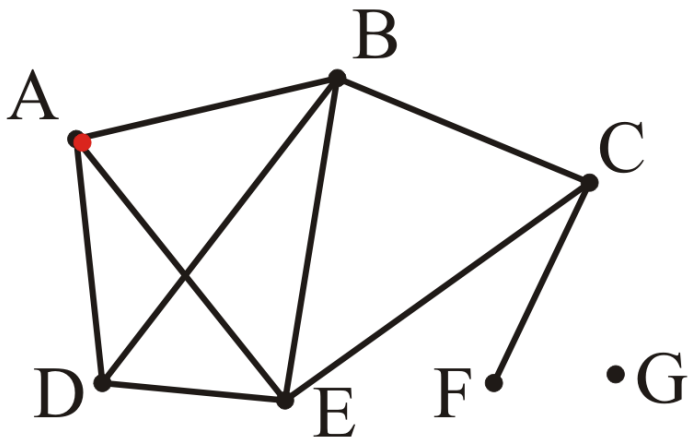
(A, B, E, C, B, D)



Paths

A *trivial* path of length 0:

(A)



Simple paths

A *simple path* has no repetitions other than perhaps the first and last vertices

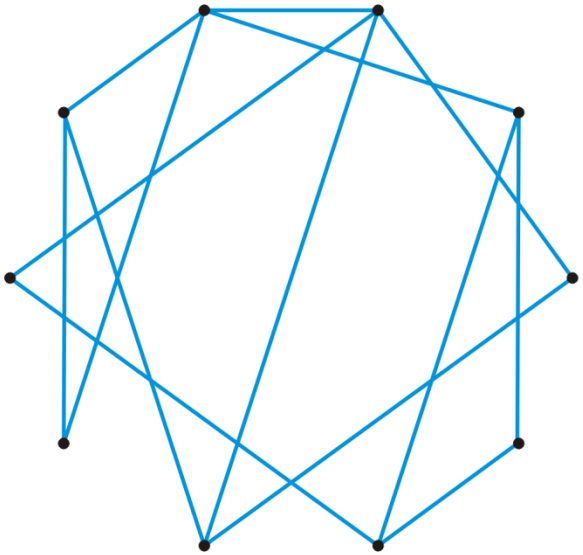
A *simple cycle* is a simple path of at least two vertices with the first and last vertices equal

- Note: these definitions are not universal

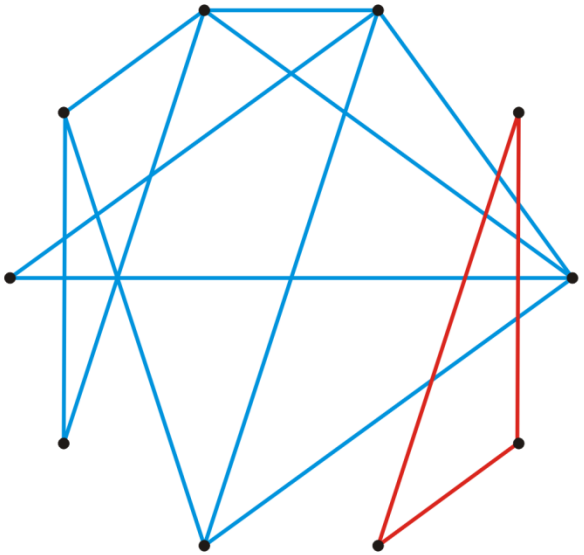
Connectedness

Two vertices v_i, v_j are said to be *connected* if there exists a path from v_i to v_j

A graph is connected if there exists a path between any two vertices



A connected graph



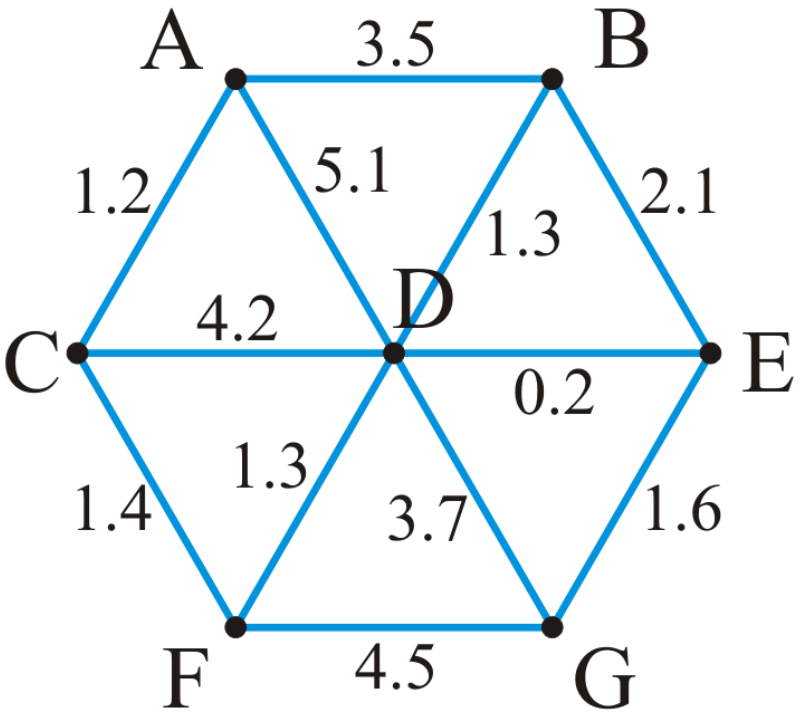
An unconnected graph

Weighted graphs

A weight may be associated with each edge in a graph

- This could represent distance, energy consumption, cost, etc.
- Such a graph is called a *weighted graph*

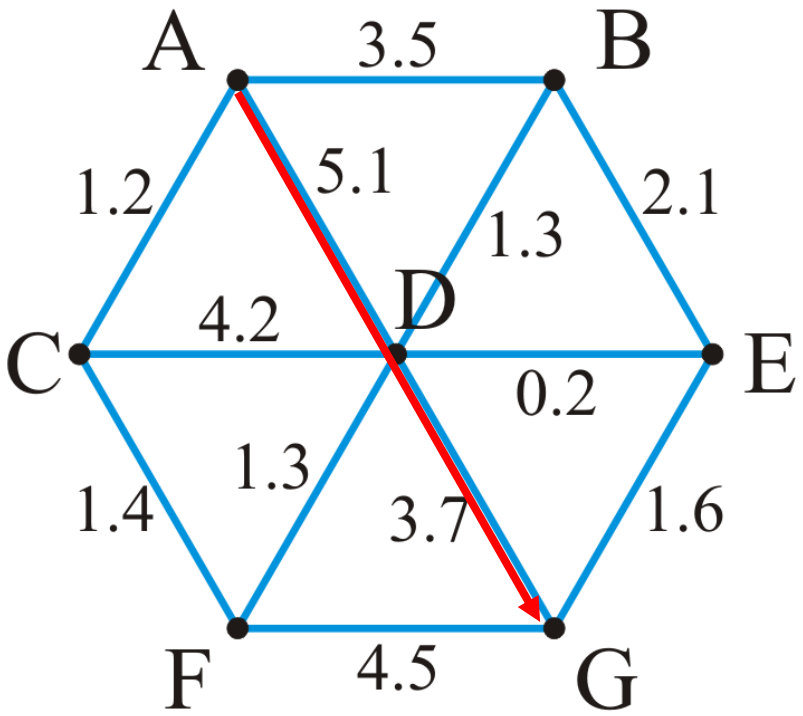
Pictorially, we will represent weights by numbers next to the edges



Weighted graphs

The *length* of a path within a weighted graph is the sum of all of the edges which make up the path

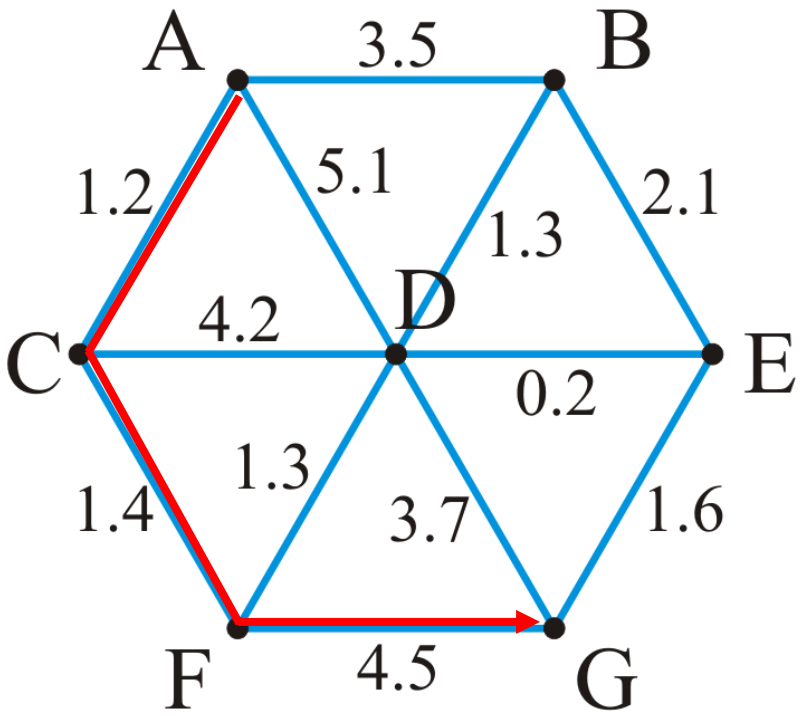
– The length of the path (A, D, G) in the following graph is $5.1 + 3.7 = 8.8$



Weighted graphs

Different paths may have different weights

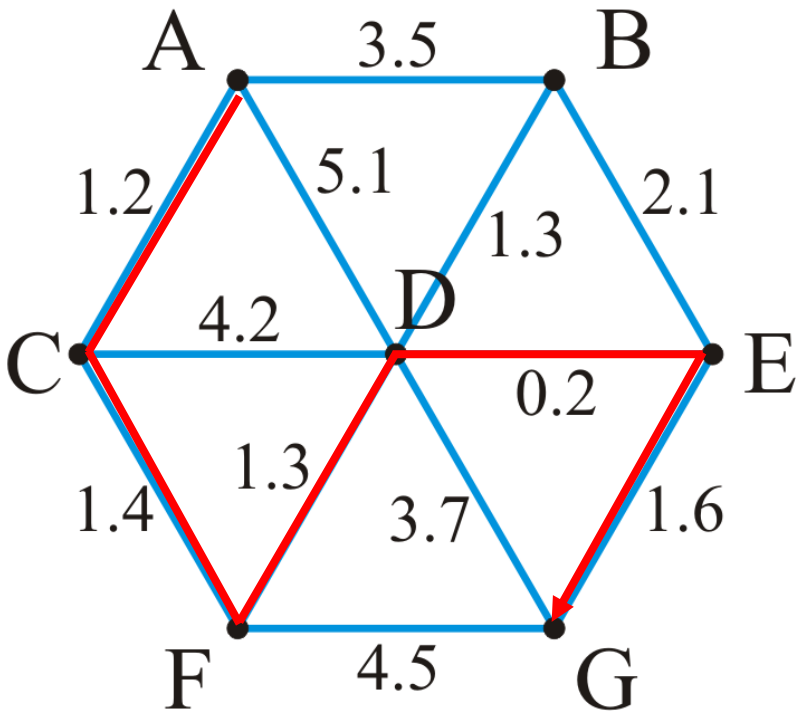
– Another path is (A, C, F, G) with length $1.2 + 1.4 + 4.5 = 7.1$



Weighted graphs

Problem: find the shortest path between two vertices

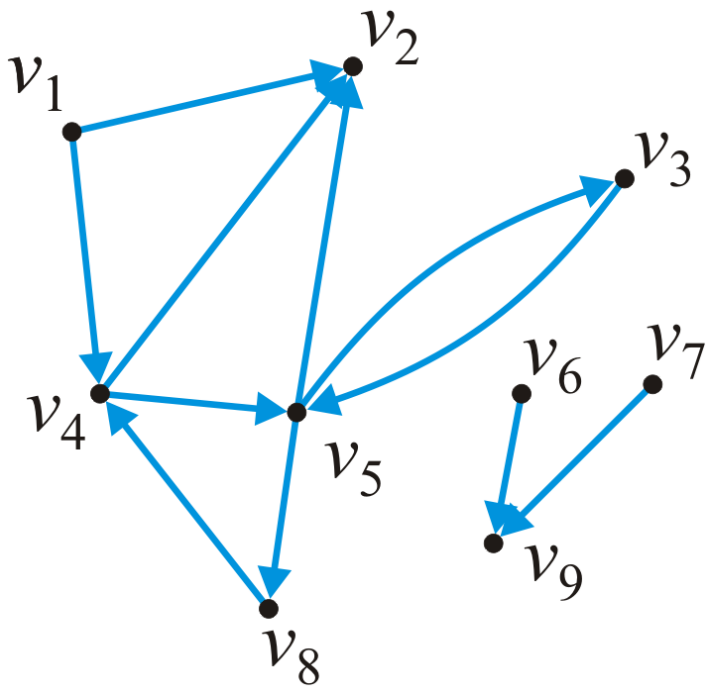
– Here, the shortest path from A to G is (A, C, F, D, E, G) with length 5.7



Representations

How do we store the adjacency relations?

- Binary-relation list
- Adjacency matrix
- Adjacency list



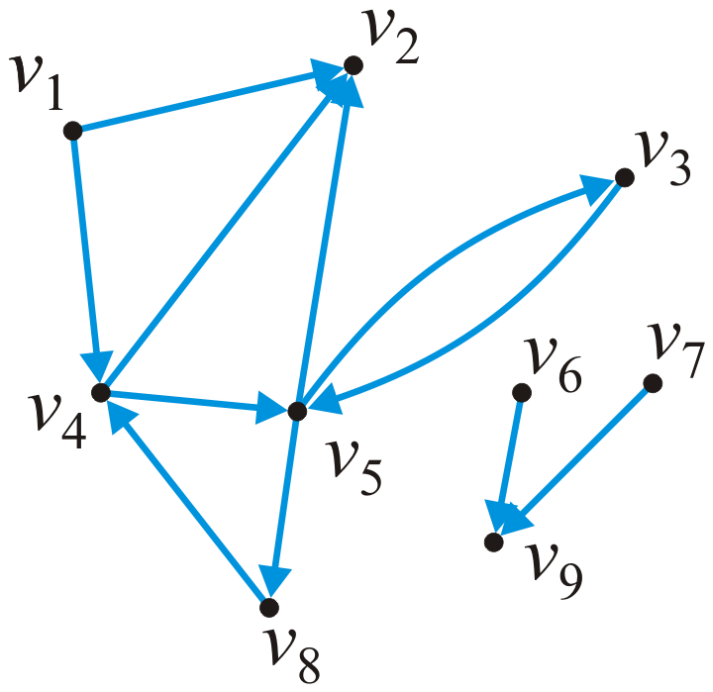
Binary-relation list

The most inefficient is a relation list:

- A container storing the edges

$$\{(1, 2), (1, 4), (3, 5), (4, 2), (4, 5), (5, 2), (5, 3), (5, 8), (6, 9), (7, 9), (8, 4)\}$$

- Requires $\Theta(|E|)$ memory
- Determining if v_j is **adjacent** to v_k is $O(|E|)$
- Finding all **neighbors** of v_j is $\Theta(|E|)$

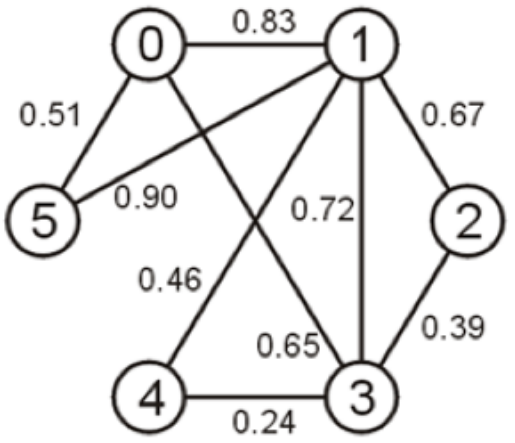


Adjacency Matrix

Define an $n \times n$ matrix $\mathbf{A} = (a_{ij})$ and if the vertices v_i and v_j are connected with weight w, then set $a_{ij} = w$ and $a_{ji} = w$

That is, the matrix is symmetric, *e.g.*,

	0	1	2	3	4	5
0		0.83		0.65		0.51
1	0.83		0.67	0.72	0.46	0.90
2		0.67		0.39		
3	0.65	0.72	0.39		0.24	
4		0.46		0.24		
5	0.51	0.90				



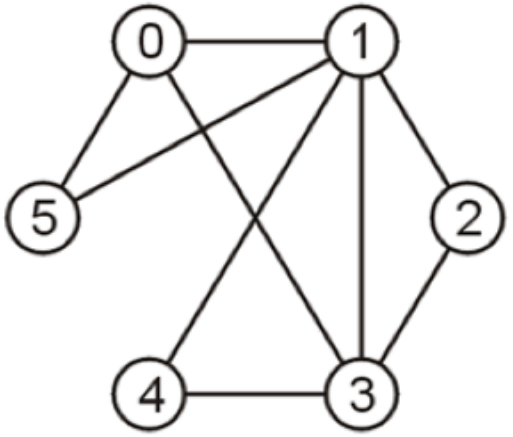
Adjacency Matrix

An unweighted graph may be saved as an array of Boolean values

– vertices v_i and v_j are connected then set

$$a_{ij} = a_{ji} = true$$

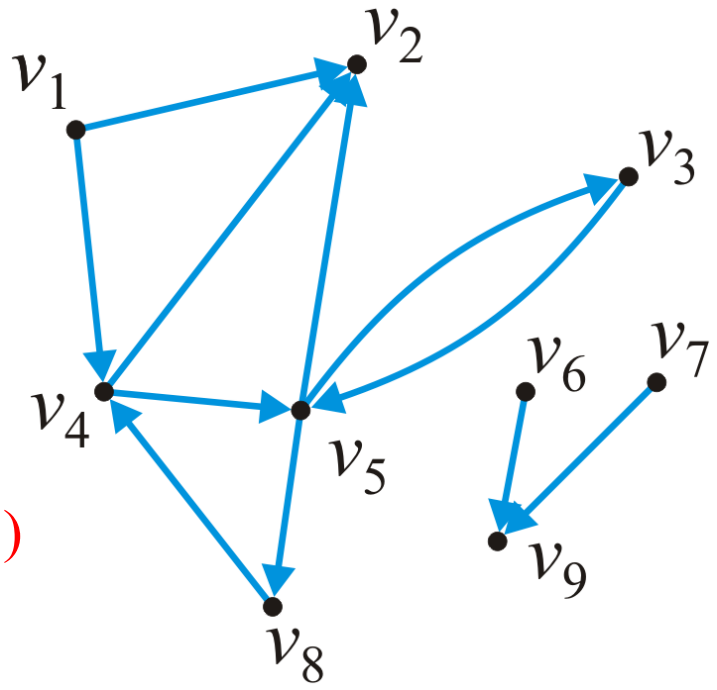
	0	1	2	3	4	5
0		T	F	T	F	T
1	T		T	T	T	T
2	F	T		T	F	F
3	T	T	T		T	F
4	F	T	F	T		F
5	T	T	F	F	F	



Adjacency matrix

The matrix entry (j, k) is set to true if there is an edge (v_j, v_k)

	1	2	3	4	5	6	7	8	9
1		T		T					
2									
3					T				
4		T			T				
5		T	T					T	
6									T
7									T
8				T					
9									



- Requires $\Theta(|V|^2)$ memory
- Determining if v_j is adjacent to v_k is $O(1)$
- Finding all neighbors of v_j is $\Theta(|V|)$

Acknowledgement

- Douglas Wilhelm Harder.
  - Thanks for making an excellent set of slides for ECE 250 *Algorithms and Data Structures* course

- Prof. Hung Q. Ngo:
  - Thanks for those beautiful slides created for CSC 250 (Data Structures) course at UB.

- Many of these slides are taken from these two sources.