Graphs

Definitions

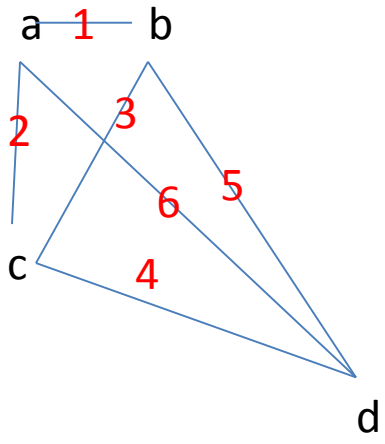
- A graph G is a structure with a set of nodes (*vertices* V) and *edges* E connecting the vertices*

  so.. G={V,E}
- Two vertices are said to be *adjacent* if there is a single edge connecting them
- There may be a passage from vertex a to b and likewise from b to a. This edge is said to be *undirected*. If the passage is one way, the edge is *directed*
- A graph is connected if every pair of vertices has a path (set of edges) connecting them
- The number of edges touching a vertex is the *degree of the vertex*

*A nuance: a graph cannot have an edge that runs from a vertex back to itself (self-edge). This constraint makes an important distinction between a graph and a Markov Chain.

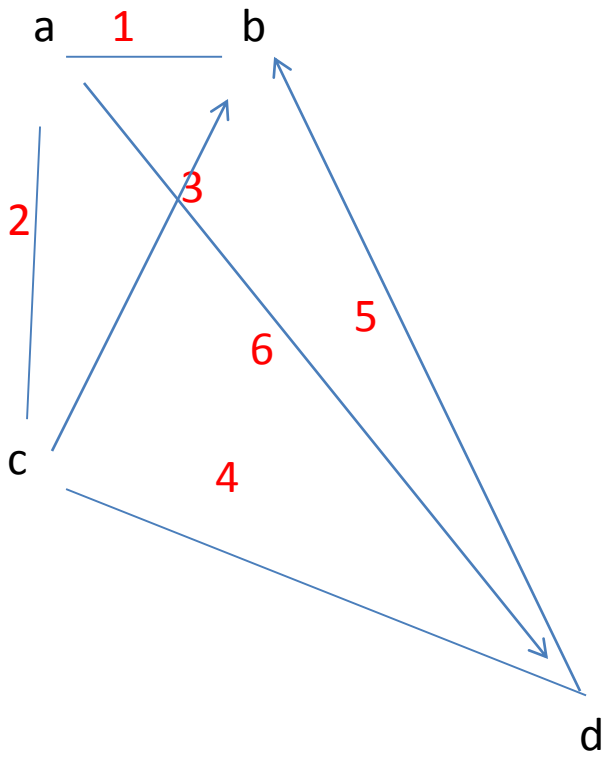
How to Implement a Graph on a Computer: The Adjacency Matrix

- Fundamental to algorithms on graphs
- The matrix index is the vertex; the matrix entry is the edge measure



	a	b	c	d
a	0	1	2	6
b	1	0	3	5
c	2	3	0	4
d	6	5	4	0

Here is an undirected graph and its adjacency matrix



	a	b	c	d
a	0	1	2	6
b	1	0	∞	∞
c	2	3	0	4
d	∞	5	4	0

Here is a directed graph and its adjacency matrix

Graph Traversal

Basic algorithms

- Linked list
  - Expensive
  - Good for sparse graphs
- Adjacency Matrix
  - Always requires n^2 space and time to construct
  - Traverse in linear time

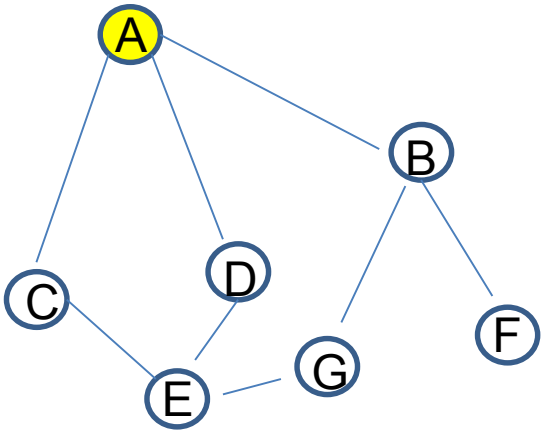
Graph Traversal

- Depth-first searching is one of 2 ways to traverse a graph (*vis à vis* breadth-first search)
  - The idea is to visit as many vertices (edges) as possible.
    1. Travel as far as possible down into the graph
    2. Back up and visit an unvisited vertex
    - Repeat 1 and 2 until exhausted

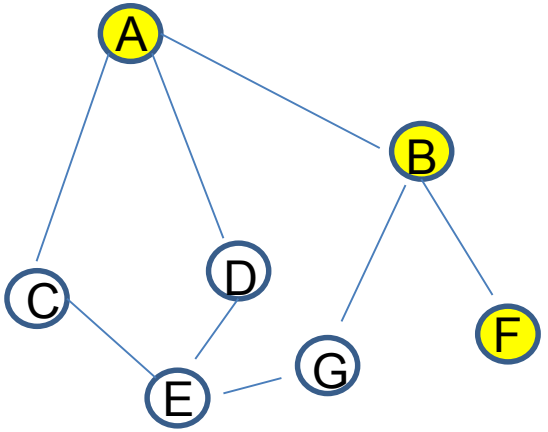
Implementation of DFS*

- Recursion
  - DFS: Starting at some vertex a, visit a, mark a visited, and push a onto the stack
  - For each unvisited vertex v adjacent to a, recurse with DFS until the stack is empty
- Iteration using a stack
  - Visit v, push v and mark v visited
  - While the stack is not empty
    - If no vertex adjacent to the vertex on the top of the stack is unvisited, pop the stack
    - Else select an adjacent unvisited vertex u, visit u, push u and mark visited
  - End while

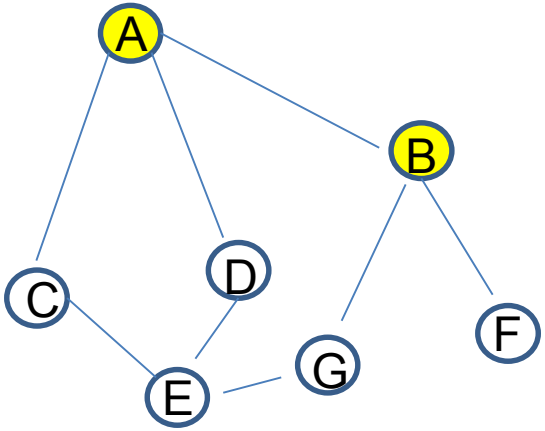
*From Data Abstraction and Problem Solving with C++ Walls and Mirrors , Carrano, Helman and Veroff, Addison Wesley 1998



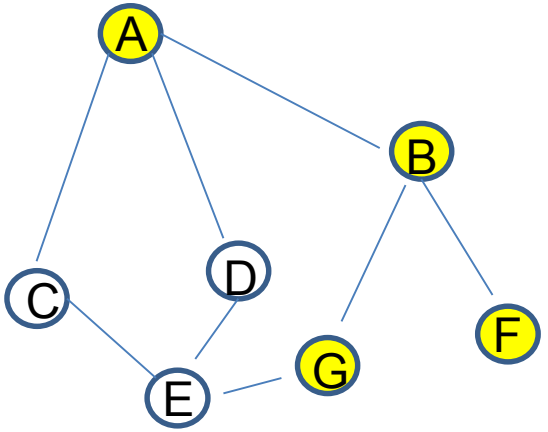
A



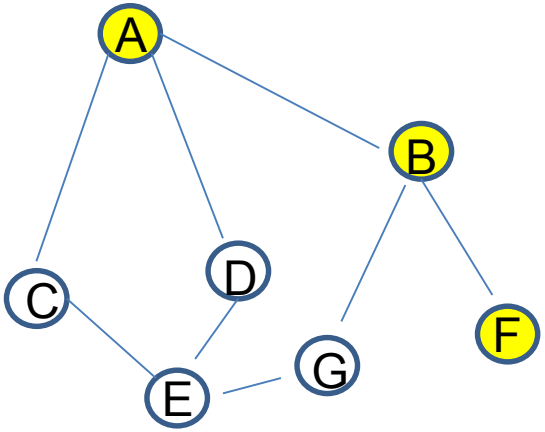
F
B
A



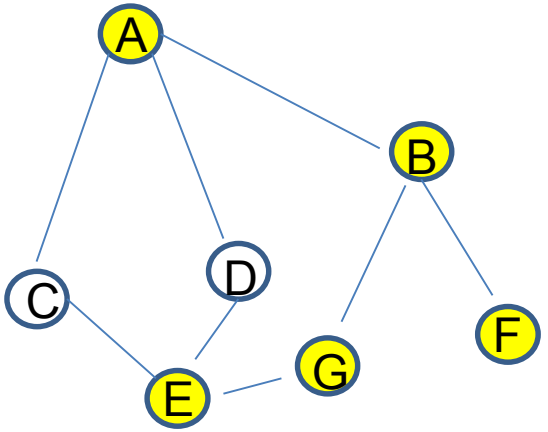
B
A



G
B
A

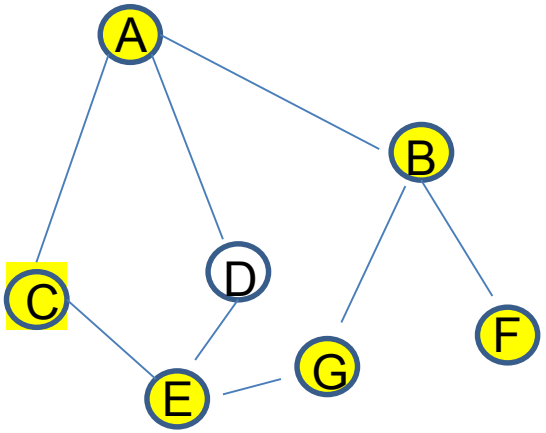


F
B
A



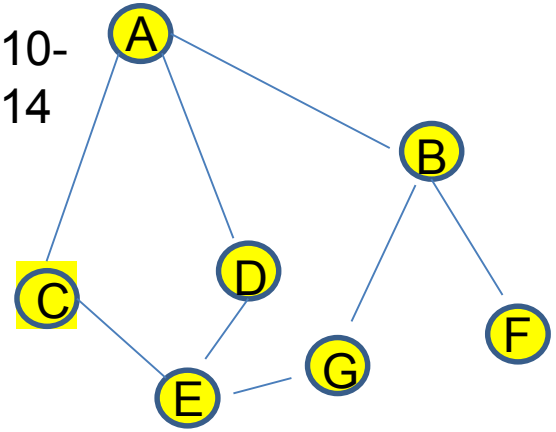
E
G
B
A

7



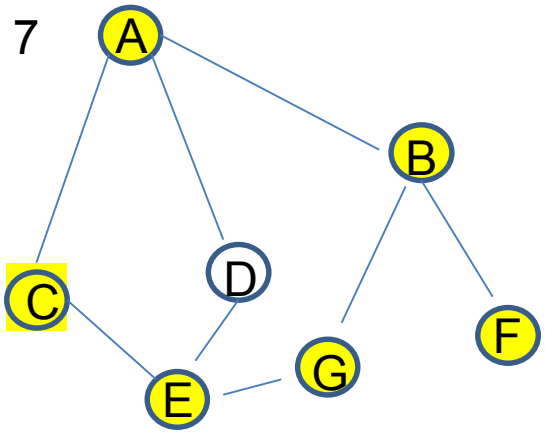
C ↓
E
G
B
A

10-
14



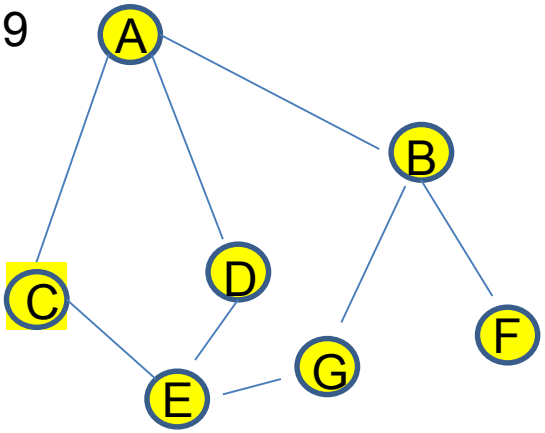
D ↑
E ↑
G ↑
B ↑
A ↑

7



C ↑
E
G
B
A

9



D ↓
E
G
B
A

Some relevance

Some of the computational challenges in this course that can be laid out in graphical representation will involve the concept of a *circuit*

- A *circuit* is a cycle is a path that visits either every vertex (Hamiltonian) or every edge (Eulerian) precisely once.
- A *cycle* is a circuit that begins and ends at the same vertex (or edge)

Key Facts

- If the graph is undirected and each vertex is of even degree, then
  - An Eulerian circuit exists
  - It can be found in polynomial time
  - An Eulerian cycle contains 2^2n-1-n Hamiltonian circuits
- For an arbitrary graph, a Hamiltonian circuit may or may not exist
  - Making the determination is an NP-hard problem
  - The Traveling Salesman Problem (a classic NP-complete problem) is to find the shortest Hamiltonian circuit when the edges are distances

Recall…

The depth-first search goes as deeply into a graph as it can. It does not stop on a target, unlike algorithms that seek shortest paths

This idea can be exploited to find an Eulerian circuit. If the graph is Eulerian, and the search uses edges instead of vertices, the search will return to the starting vertex, thus defining a cycle.

Piecing together cycles built from untouched edges will yield a circuit.

Finding an Eulerian circuit

- Use a depth-first search, marking edges visited rather than vertices visited to yield a cycle
- Search along the vertices on the cycle until there is one that touches an unvisited edge.
- Use this vertex to start a depth-first search
- Take the cycle so yielded and insert it into the original cycle at the point in the first cycle where the starting vertex of the second cycle is encountered
- Repeat seeking untouched edges until there are no more. We then have visited every edge but once, and have pieced together an Eulerian circuit. Note that this required polynomial time.

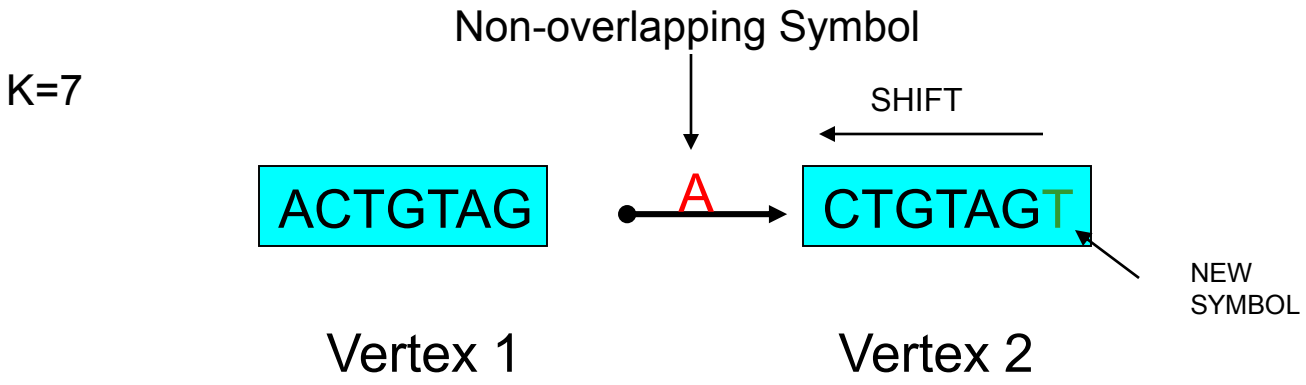
Modern Theory

An advance (*ca.* 1946) in graph theory by the Dutch mathematician Nicholaas Govert DeBruijn has recently been exploited to facilitate DNA sequencing.

De Bruijn graphs are graphs labeled with string data. The graph demonstrates the transformations between all pairs of all strings derived from a prescribed alphabet and string length. Two vertices are related if one can be transformed to the other by a directed edge labeled with the overlap, or shift, usually having the length of one symbol in the alphabet.

The deBruijn Graph

- The vertices contain overlapping substrings of the symbols in k-mers
- A vertex has a directed edge to another vertex if the second vertex is a one-symbol left shift of the symbols of first vertex, with a new symbol added to the end (maintaining k-arity) and a directed edge established



De Bruijn

- Hamiltonian Cycles may not exist
- If they exist, Hamiltonian cycles are expensive to compute (np-Complete)
- Eulerian cycles are cheap and easy

De Bruijn Graph

Exploit Eulerian graph

Given many many k-mers

- Place a prefix of size k-1 on a vertex
- Place the suffix (size k-1) of the same k-mer on another vertex
- Do this for all k-mers
- For each prefix, align the last letter with the first letter of some suffix along a directed edge
- The recovered k-mer is on an edge
- When done, follow the directed edges in order

Prefixes and suffixes from the 3-mers of a sequence of letters

AB

JA

BB

BE

RW

ER

YJ

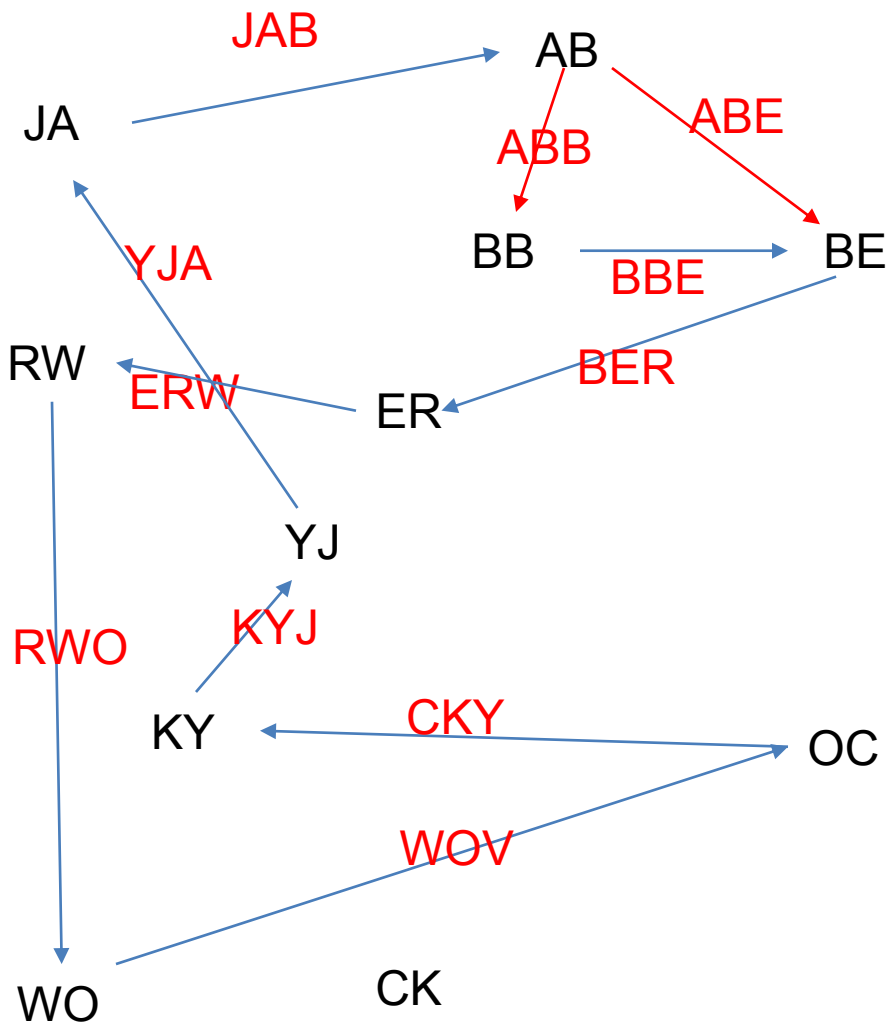
KY

OC

WO

CK

Construction of a deBruijn graph



JAB
ABB
BBE
BER
ERW
RWO
WOC
OCK
CKY
KYJ
YJA

One Eulerian cycle following the directed edges

JAB
ABB
BBE
BER
ERW
RWO
WOC
OCK
CKY
KYJ
YJA

Are there others?

Is the graph for every Eulerian cycle connected?

What about assembling 4-mers for MISSISSIPPI ?

MISS

SSIP

ISSI

SIPP

SSIS

IPPI

SISS

ISSI