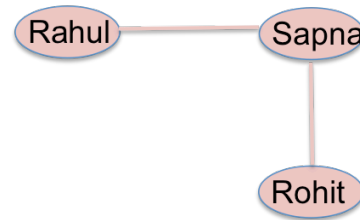
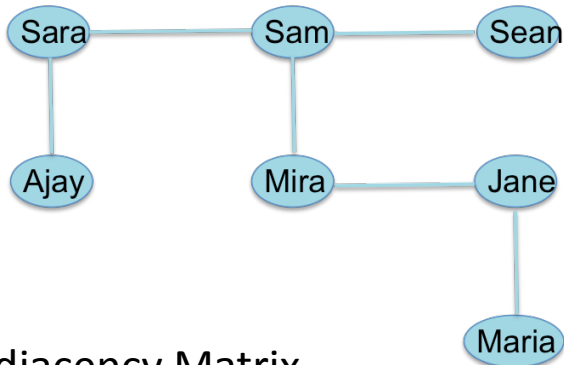


CS 112: Fall 2016

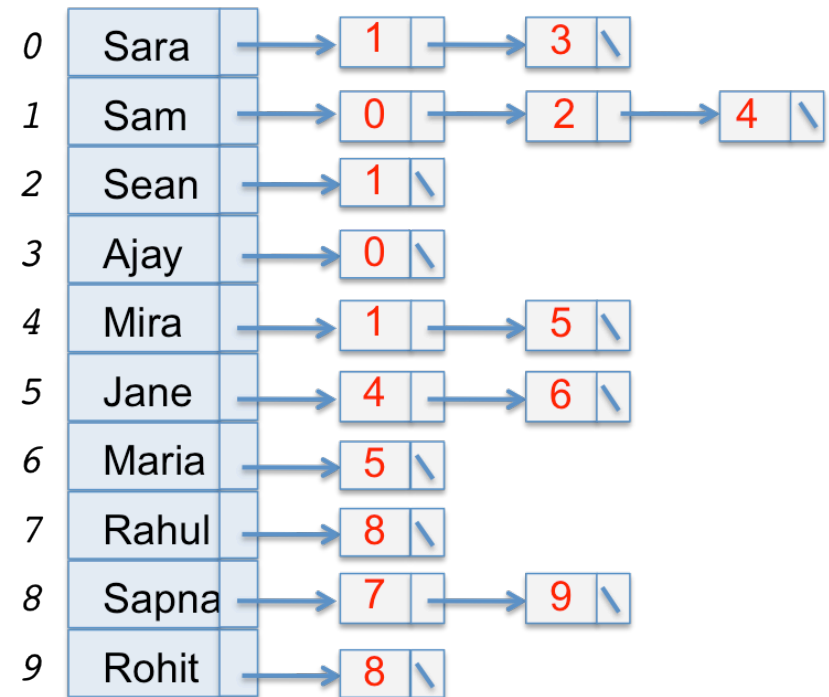
Graphs – Types and Representation Review



Undirected graph,
no edge weights

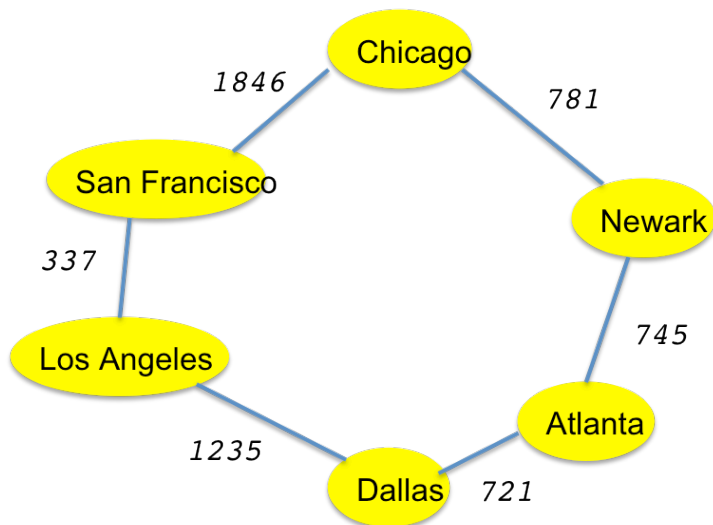
Adjacency Matrix
(vertex names stored separately)

	0	1	2	3	4	5	6	7	8	9
0		T		T						
1	T		T		T					
2		T								
3	T									
4		T				T				
5					T		T			
6						T				
7									T	
8								T		T
9									T	



Adjacency Linked Lists

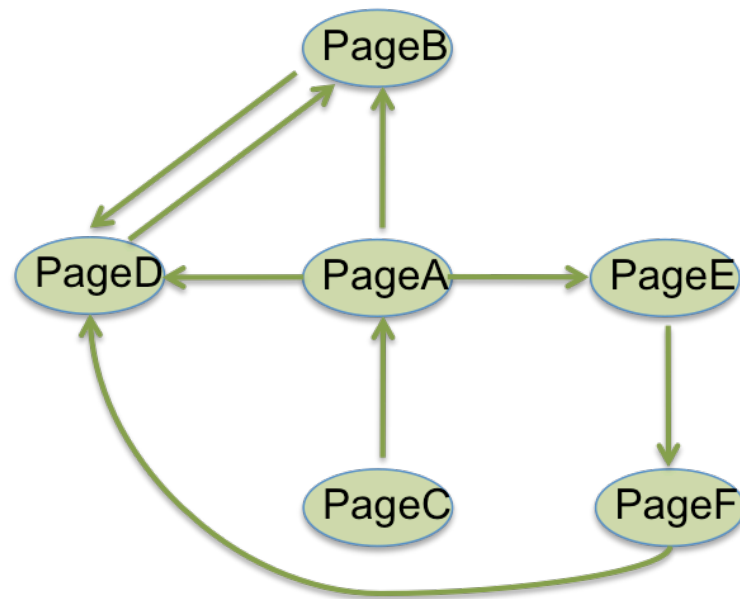
Undirected graph, with edge weights



Adjacency Matrix
(vertex names stored separately)

	0	1	2	3	4	5
0	-1	781	-1	-1	-1	745
1	781	-1	1846	-1	-1	-1
2	-1	1846	-1	337	-1	-1
3	-1	-1	337	-1	1235	-1
4	-1	-1	-1	1235	-1	721
5	745	-1	-1	-1	721	-1

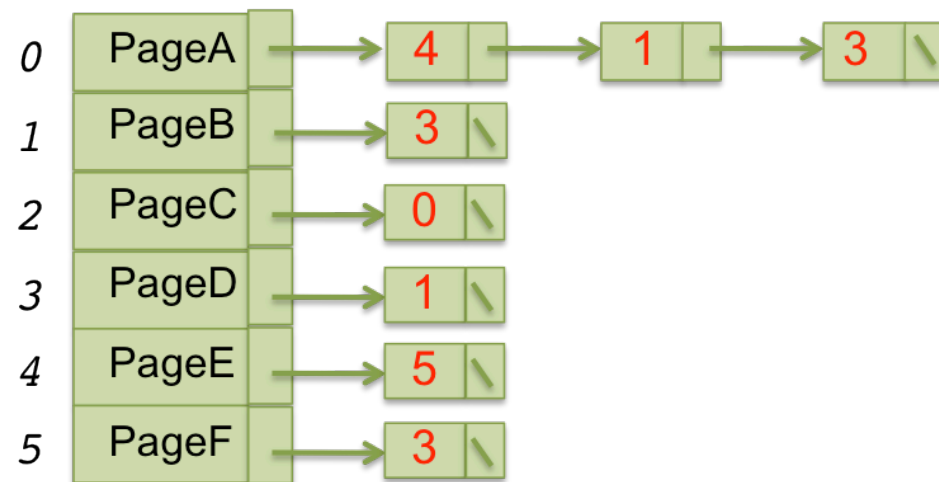
Directed graph,
no edge weights



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

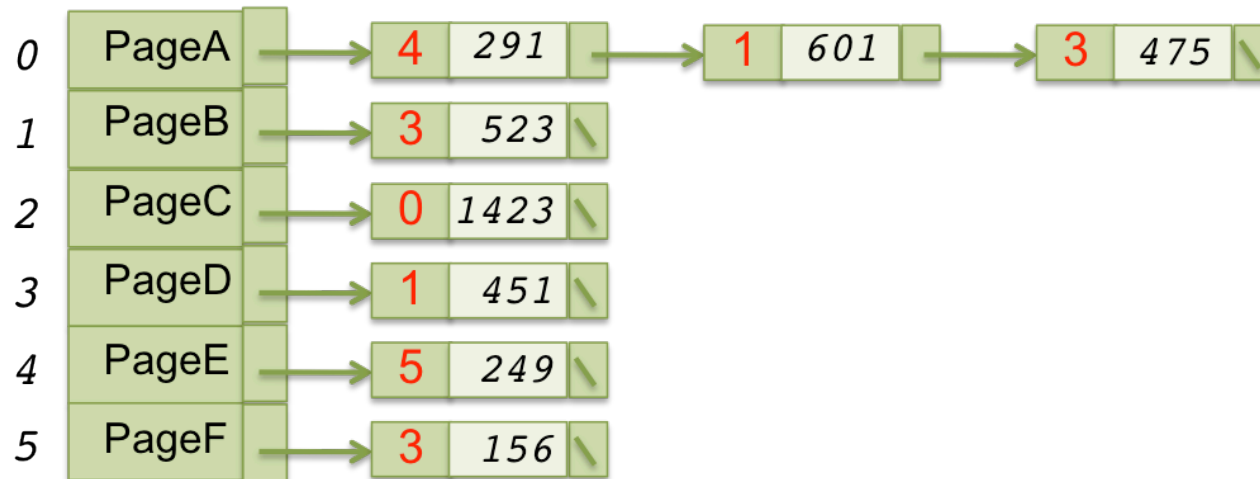
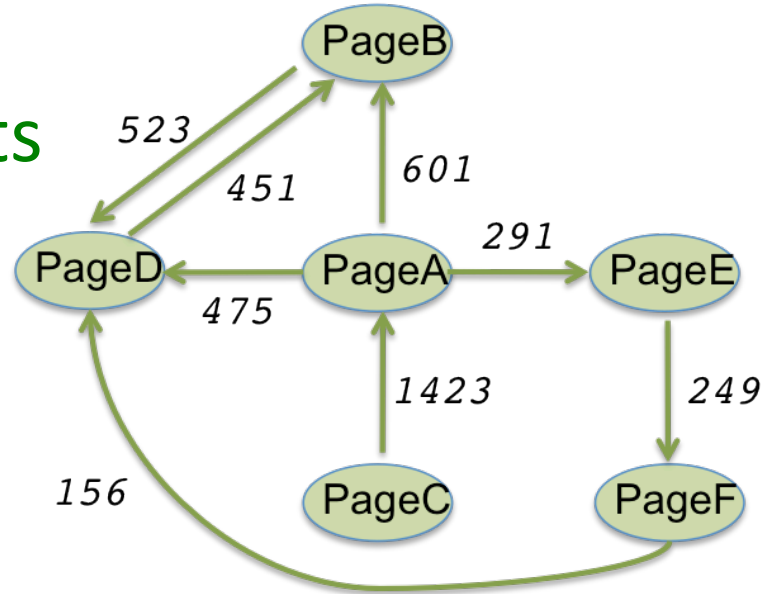
Adjacency Matrix

(vertex names stored separately)




Adjacency Linked Lists

Directed graph,
with edge weights



1. The adjacency matrix representation of a graph with n vertices and e edges requires the following amount of space:

A. $O(n)$

B. $O(n^2)$ 

C. $O(n+e)$

D. None of the above

2. Consider a directed graph with 10 vertices in which every vertex can be reached from any other vertex in one or more edge hops. What is the least number of edges this graph should have?

A. 9

B. 10



C. 11

3. In an adjacency matrix for a directed graph what would the occupied cells going down a column represent?

A. Edges going out of the vertex for that column

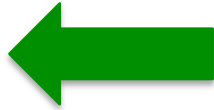
B. Edges coming in to the vertex for that column

C. Neither of the above



4. For an undirected graph with 20 vertices and 20 edges, what is the percentage space utilization if space is allocated ONLY for the lower triangle (including diagonal)?

A. Roughly 10%



B. Roughly 20%

C. Roughly 30%

5. Consider an undirected graph with 8 vertices and 16 edges.
What is the percentage space utilization in the adjacency matrix
used to store this graph?

A. 10%

B. 20%

C. 25%



D. 50%

6. Consider an undirected graph in which there can either be one edge or no edge between any pair of vertices, and there is no edge from a vertex to itself (SIMPLE GRAPH). If such a graph has 4 vertices, it can have a maximum of how many edges?:

A. 4 edges

B. 5 edges

C. 6 edges 

D. None of the above

7. A directed graph with 5 edges is stored in an adjacency linked lists structure. How many nodes will there be in the linked lists?

A. 5 

B. 10


C. 15

D. None of the above

8. A directed graph with n vertices and e edges is stored in *adjacency linked lists*. Its *complement* is computed and stored in another structure of adjacency linked lists. How much storage space (big O) would be taken up by the original graph and its complement? (Complement: If there is an edge (x,y) in the original graph, there won't be one in the complement; if there's no edge (x,y) in the original, there will be one in the complement. Vertices are same in both.)

A. $O(e)$

B. $O(n+e)$

C. $O(n^2)$ 

D. $O((n+e)\log n)$

9. An undirected graph with n vertices, in which every vertex can be reached from all other vertices cannot have fewer than:


A. n edges

B. $n-1$ edges 

C. $n-2$ edges

D. $n+1$ edges

10. If an undirected graph with n vertices and e edges is stored using adjacency linked lists, what is the worst case running time to know if there is an edge (x,y) in the graph, if the number of edges is much greater than the number of vertices?


A. $O(n)$ 

B. $O(e)$

C. $O(n+e)$

D. $O(n^2)$

11. If an undirected graph with n vertices and e edges is stored using adjacency linked lists, what is the worst case running time to know if there is an edge (x,y) in the graph, if there is no prior knowledge as to how many edges there are relative to vertices?

A. $O(\min(n,e))$ 

B. $O(\max(n,e))$

C. $O(n)$

D. $O(e)$

12. In a class of 50 students, each student has 2 friends. In a graph that represents these friendships, what would be the total number of bytes to store it in adjacency linked lists, ignoring the space to store the names of students? Assume 4 bytes to store an integer, and 4 bytes for a pointer.

A. 100

B. 200


C. 800

D. 1000 

13. For every edge (a,b) read from a graph input file, a program first finds the index numbers for the vertices by doing a sequential search for each of them in a vertex names list (unsorted). If there are n vertices and e edges in the graph, what would be the worst case running time to populate the adjacency linked lists?

A. $O(n+e)$

B. $O(n*e)$

C. $O(n^2)$ 

D. $O(e)$