CSC345 Discussion 6

Graphs

Announcements

Test on 2/22, Thursday next week in class usual time.

Review Session on 2/19, Tuesday next week 6pm GS906

Project Due, 2/28, a little less than 2 weeks

Directed & Undirected Graphs

Directed & Undirected Graphs

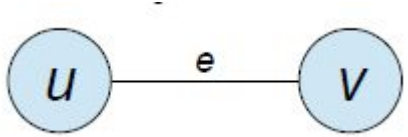
In an undirected graph, each edge is a set of two vertices; order does not matter.

In a directed graph (digraph), each edge is an ordered pair: it has a direction.

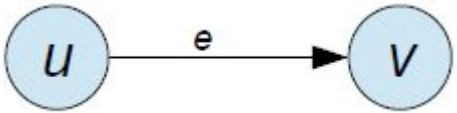
Terminology

Incident and Adjacent

– In an undirected graph with edge e = {u, v}, we say that e is incident to vertex u and vertex v, and that vertices u and v are adjacent.



In a directed graph with edge e = (u, v), we say that edge e is an out-edge of vertex u, and an in-edge of vertex v, and that v is adjacent to u.



Terminology

Reachable:

Connected:

Terminology

Reachable:

– Vertex w is reachable from vertex u if a path exists from u to w.

Connected:

– An undirected graph is connected if a path exists between every pair of vertices.

– A directed graph is strongly connected if a path exists between every pair of vertices.

Matrix vs List

Adjacency List:

Adjacency Matrix:

Matrix vs List

Adjacency List:

– More space efficient for sparse graphs, that is, graphs with $|E| << |V|^2$.

– O(|E|) time to test adjacency of two vertices.

Adjacency Matrix:

– O(1) access time to test adjacency of two vertices.

– Wastes a lot of space for a very sparse graph.

– Space efficient if graph is dense (many edges) -- one bit per edge in an unweighted graph.

Depth-First Search & Breadth First Search

Depth-First Search & Breadth First Search

DFS: Depth First Search

– Like preorder traversal in a rooted tree.

– Start at a vertex, find any neighbor.

– Visit it, and recurse from there.

– After recursion is done, recurse into next neighbor.

BFS: Breadth First Search

– Start at some vertex, find all (adjacent) neighbors.

– Visit each neighbor, – . . . and only then visit their neighbors, and so on.

Should You Use BFS or DFS

Depends on what kind of spanning tree you want

DFS tree is tall, thin.

– Recursive code is also simple.

BFS tree is wider, shorter.

– Minimum number of edges to root

Handy Visualizations for DFS and BFS

DFS

https://www.cs.usfca.edu/~galles/visualization/DFS.html

BFS

https://www.cs.usfca.edu/~galles/visualization/BFS.html

Dijkstra’s

Dijkstra's algorithm computes the distance

> the cost of a shortest path from a source to every other vertex in a directed graph.
>
> Also, it's relatively easy to get the paths themselves.
>
> Basically, it's BFS + edge weights, and greedy choices

Dijkstra's

Priority Queue implementation:	*Unsorted array*	*Binary Min-heap*
Time cost of operations below:		
Filling the queue	$\Theta(\|V\|)$	$\Theta(\|V\|)$
All *extract_min*() operations	$O(\|V\|) \cdot \|V\|$	$O(\log \|V\|) \cdot \|V\|$
All *update_priority*() operations	$O(1) \cdot O(\|E\|)$	$O(\log \|V\|) \cdot O(\|E\|)$
Total time cost	$\boldsymbol{O(\|V\|^2)}$	$\boldsymbol{O((\|V\|+\|E\|) \log \|V\|)}$

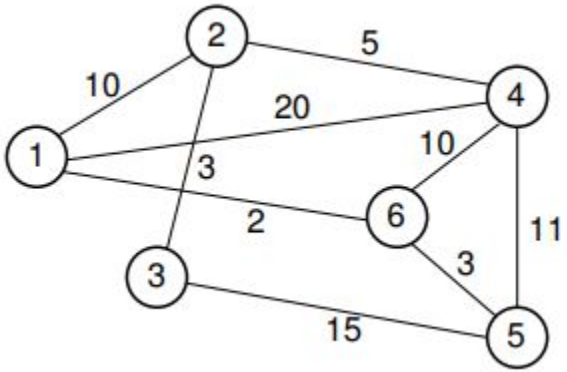
Practice Question

Prove by induction that a graph with n vertices has at most $n(n-1)/2$ edges.

Practice Question

Draw the adjacency matrix representation for the graph

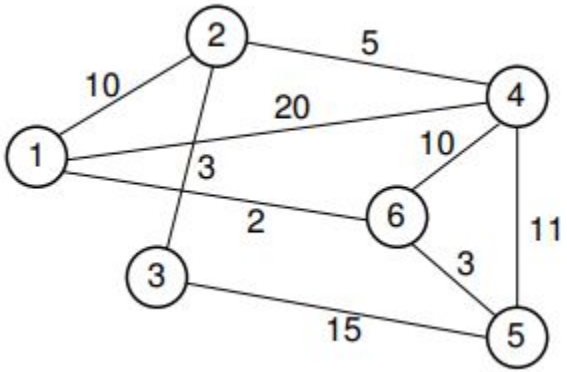
Draw the adjacency list representation for the same graph.



Practice Question

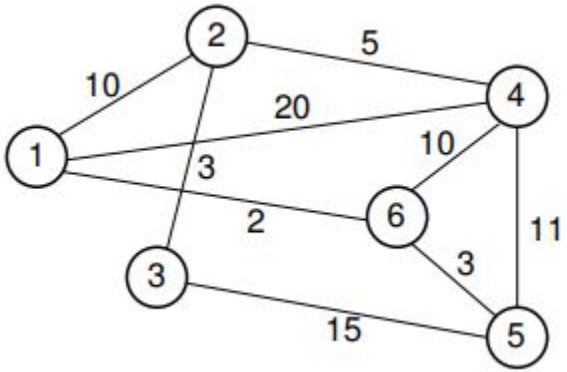
Show the DFS tree for the graph starting at vertex 1.

Show the BFS tree for the graph starting at vertex 1.



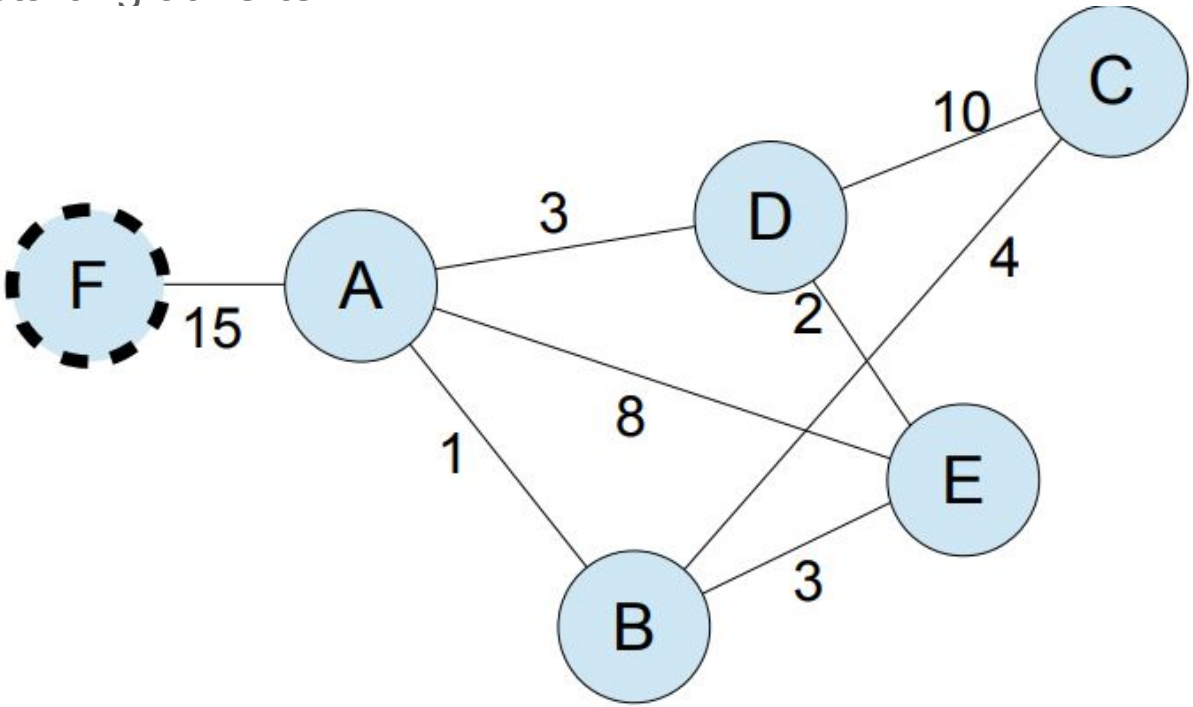
Practice Question

Run Dijkstra starting at vertex 1.



Practice Question

Run Dijkstra starting at vertex F.



Practice Problem

Explain why, in the worst case, Dijkstra’s algorithm is (asymptotically) as efficient as any algorithm for finding the shortest path from some vertex I to another vertex J.