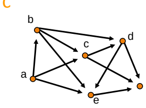
CS/ENGRD 2110
Object-Oriented Programming
and Data Structures

Spring 2011
Thorsten Joachims

Lecture 19:
Shortest Paths

Graph Definitions

- A directed graph (or digraph) is a pair (V, E) where
  - V is a set
  - E is a set of ordered pairs (u,v) where u,v in V
    - Usually require u ≠ v (i.e., no self-loops)
- An element of V is called a vertex (pl. vertices) or node
- An element of E is called an edge or arc
- |V| = size of V, often denoted n
- |E| = size of E, often denoted m

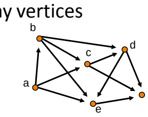


Some Graph Terminology

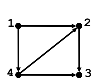
- Vertices u and v are called the source and sink of the directed edge (u,v), respectively
- Vertices u and v are called the endpoints of (u,v)
- Two vertices are adjacent if they are connected by an edge
- The outdegree of a vertex u in a directed graph is the number of edges for which u is the source
- The indegree of a vertex v in a directed graph is the number of edges for which v is the sink
- The degree of a vertex u in an undirected graph is the number of edges of which u is an endpoint

More Graph Terminology

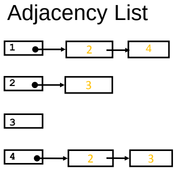
- A path is a sequence $v_0,v_1,v_2,\dots,v_p$ of vertices such that (v_i,v_{i+1}) in E, $0 \le i \le p - 1$
- The length of a path is its number of edges
  - In this example, the length is 5
- A path is simple if it does not repeat any vertices
- A cycle is a path $v_0,v_1,v_2,\dots,v_p$ such that $v_0 = v_p$
- A cycle is simple if it does not repeat any vertices except the first and last
- A graph is acyclic if it has no cycles
- A directed acyclic graph is called a dag



Graphs



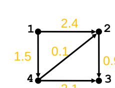
Adjacency List



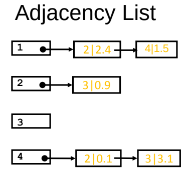
Adjacency Matrix

	1	2	3	4
1	0	1	0	1
2	0	0	1	0
3	0	0	0	0
4	0	1	1	0

Weighted Graphs



Adjacency List



Adjacency Matrix

	1	2	3	4
1	0	2.4	0	1.5
2	0	0	0.9	0
3	0	0	0	0
4	0	0.1	3.1	0

Shortest Paths in Graphs

- Finding the shortest (min-cost) path in a graph is a problem that occurs often
  - Best flight from Ithaca, NY to Duesseldorf, Germany?
  - How closely are two people connected on Facebook?
  - Driving directions from Ithaca, NY to Queens, NY?
  - Result depends on our notion of cost
    - Number of hops
    - Least mileage
    - Least time
    - Cheapest
    - Least boring
  - All of these "costs" can be represented as edge weights
- How do we find a shortest path?

Breadth-First Search for Shortest Paths Unweighted Graphs

- Input: start node s, destination node t
- Put start s node into queue and mark s as visited.
- While queue not empty
  - Poll n off queue.
  - FOR all (unmarked) successors n' of n
    - IF n' equals t THEN return path
    - Put n' into queue
    - Mark n' as visited.
- Time complexity:
  - O(m) time

8

Why does BFS find Shortest Path?

- Any node in distance 1 is visited before any node at 2 hops, before any node at distance 3 hops, …
- Whenever a node is at the top of the queue for the first time, we must have gotten there with the minimum number of hops.
- How do we keep track of the path that got BFS there?
  - Store predecessor node on path for each node in graph.

Breadth-First Search for Shortest Paths Weighted Graphs

- Input: start node s, destination node t
- Put start (s,0,null) into min-priority queue.
- While queue not empty
  - Poll minimum element (n,c,prev) off queue and mark n as visited.
  - IF n equals t THEN return path
  - FOR all (unmarked) successors n' of n
    - Put (n',c+weight(n,n'),n) into priority queue
- Time complexity:
  - O(m log m) time using heap and adjacency lists
  - Can be improved → Dijkstra's Algorithm

10