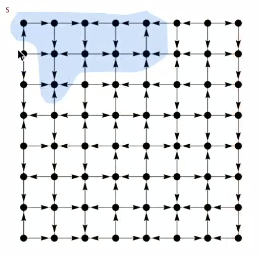
Digraph Search

Problem: find all vertices reachable from ‘s’ along a directed path

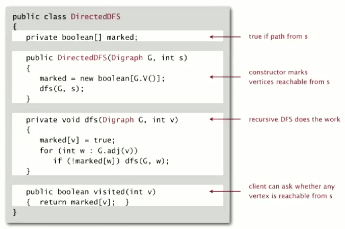


Same method as for undirected graphs:

* Every undirected graph is a digraph (with edges in both directions)
* DFS is a digraph algorithm

DFS(to visit a vertex v)  
Mark v as visited  
Recursively visit all unmarked vertices w pointing from v

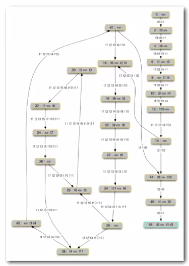
DFS for digraph is the same as - except that the name is changed to DirectedDFS



Applications

Reachability application: program control-flow analysis

* Vertex = basic block of instructions (straight-line program)
* Edge = jump



Dead-code elimination: find and eliminate unreachable code

Infinite-loop detection: determine whether exit is unreachable

Basis for solving difficult digraph problems

* 2-sttisfiability
* Directed Euler path
* Strongly-connected components

BFS in digraphs

Same method as for undirected graphs

* Every undirected graph is a digraph (with edges in both dfirections)
* BFS is a digraph algorithm

---

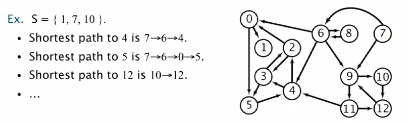
BFS(from source vertex s)  
Put s onto a FIFO queue, and mark s as visited  
Repeat until the queue is empty:

* Remove the least recently added vertex
* For each unmarked vertex pointing from v: add to queue and mark as visited

---

Proposition: BFS computes shortest paths (fewest number of edges) from s to all other vertices n in a digraph in time proportional to E + V (linear)

Multiple source shortest paths: given a digraph and a set of source vertices, find shortest path from any vertex in the set to each other vertex



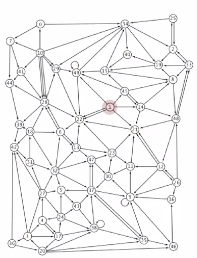
How to implement constructor? Use BFSA, but initialize by enqueuing all source vertices

BFS in digraph applications

Web crawler: crawl web, starting from some root web page

BFS with implicit digraph

* Choose root web page as source s
* Maintain a Queue of websites to explore
* Maintain a SET of discovered websites
* Dequeue the next website and enqueuer websites to which it links (provide you haven’t done so before)



Why not use DFS? Some webpages would trap searches like this by creating recursive links

Implementation

