https://algs4.cs.princeton.edu

4. Graphs and Digraphs II

- *breadth-first search (in digraphs)*
- *breadth-first search (in graphs)*
- *topological sort*
- *challenges*

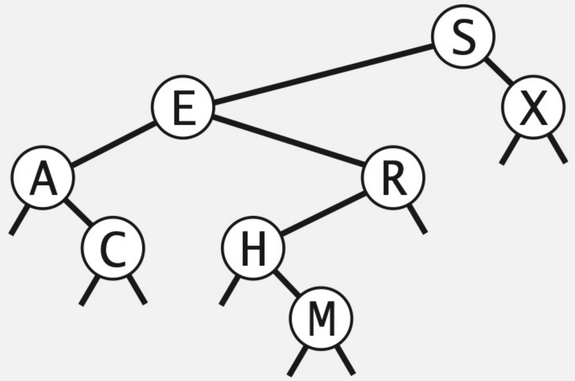
Graph search

Tree traversal. Many ways to explore a binary tree.

- Inorder: A C E H M R S X
- Preorder: S E A C R H M X
- Postorder: C A M H R E X S
- Level-order: S E X A R C H M

stack/recursion

queue



Graph search. Many ways to explore a graph.

- DFS preorder: vertices in order of calls to `dfs(G, v)`.
- DFS postorder: vertices in order of returns from `dfs(G, v)`.
- Breadth-first: vertices in increasing order of distance from `s`.

stack/recursion

queue

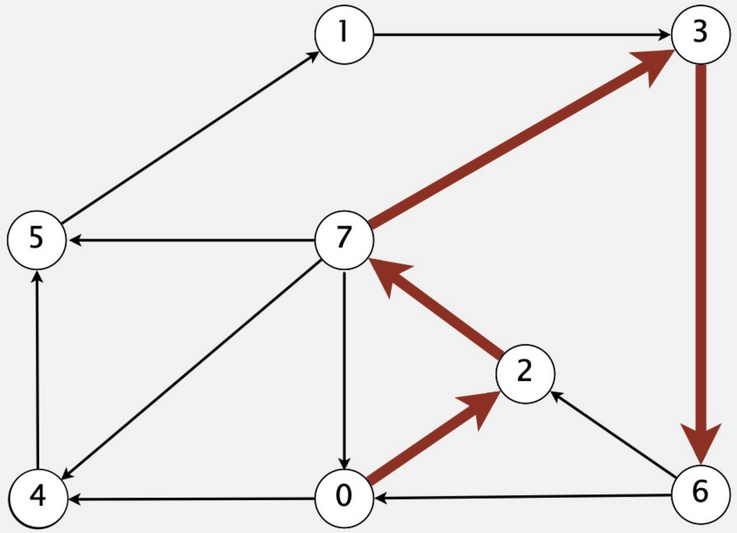
https://algs4.cs.princeton.edu

4. Graphs and Digraphs II

- *breadth-first search (in digraphs)*
- *breadth-first search (in graphs)*
- *topological sort*
- *challenges*

Shortest paths in a digraph

Problem. Find directed path from s to each other vertex that uses the fewest edges.



directed paths from 0 to 6

$0 \to 2 \to 7 \to 4 \to 5 \to 1 \to 3 \to 6$

$0 \to 4 \to 5 \to 1 \to 3 \to 6$

$0 \to 2 \to 7 \to 3 \to 6$

$0 \to 2 \to 7 \to 0 \to 2 \to 7 \to 3 \to 6$

Note: shortest paths must be simple (no repeated vertices)

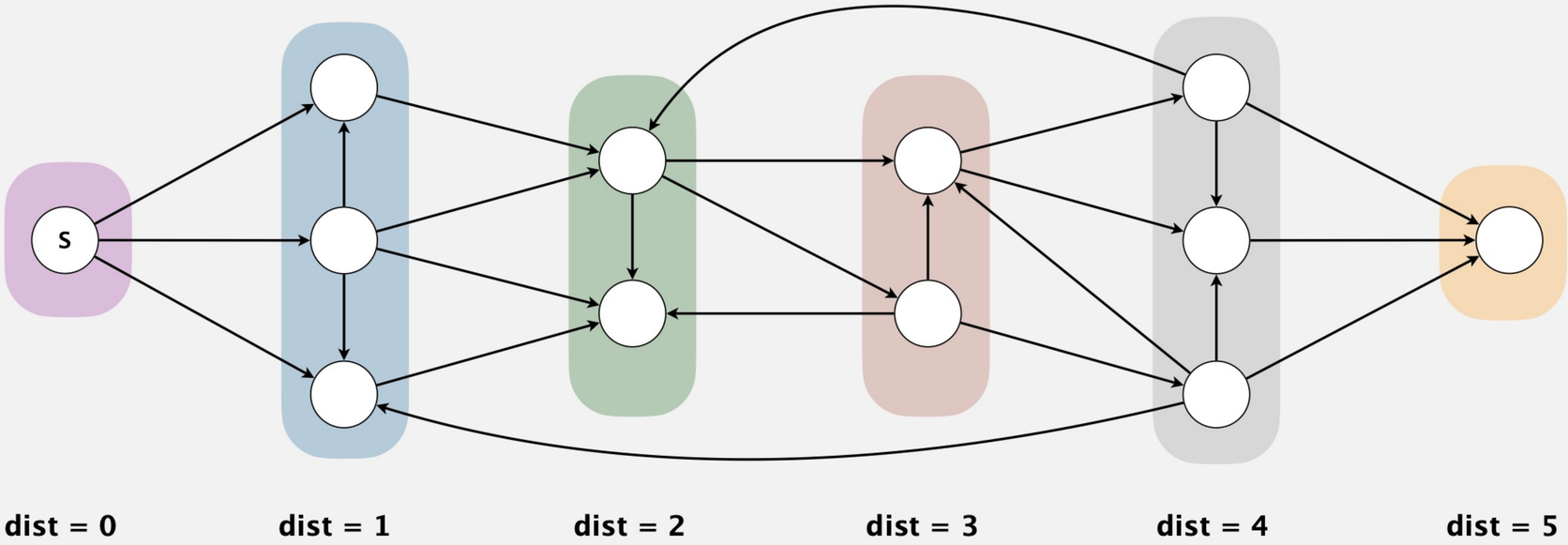
shortest path from 0 to 6 (length = 4)

$0 \to 2 \to 7 \to 3 \to 6$

Shortest paths in a digraph

Problem. Find directed path from s to each other vertex that uses the fewest edges.

Key idea. Visit vertices in increasing order of distance from s.

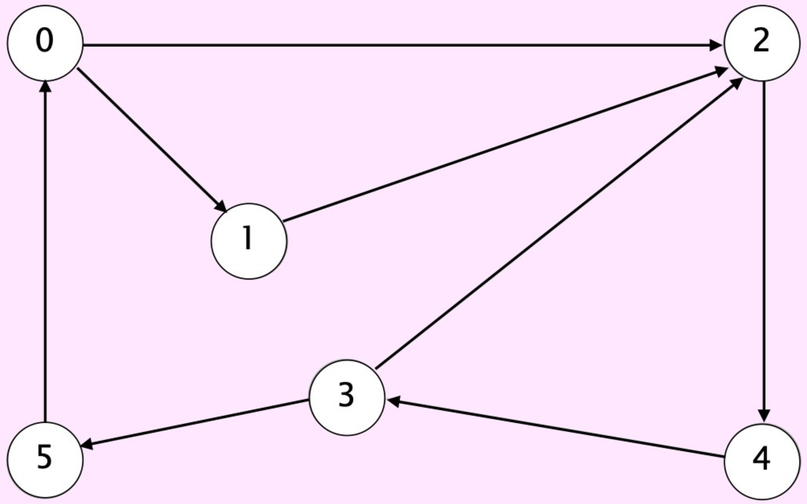


Key data structure. Queue of vertices to visit.

Breadth-first search demo

Repeat until queue is empty:

- Remove vertex v from queue.
- Add to queue all unmarked vertices adjacent from v and mark them.



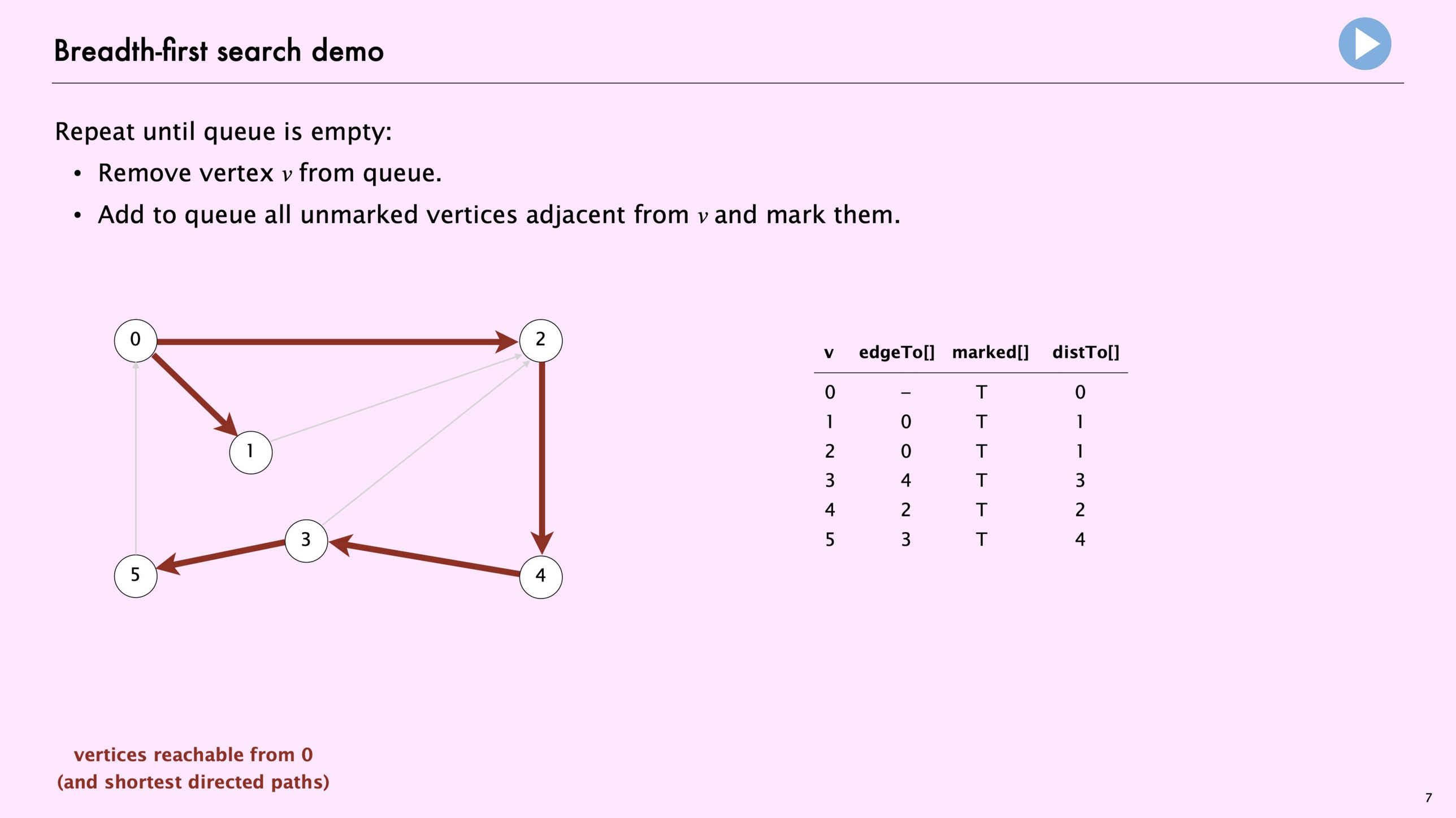
graph G

tinyDG2.txt

V → 6 E → 8

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

Breadth-first search demo



Repeat until queue is empty:

- Remove vertex v from queue.
- Add to queue all unmarked vertices adjacent from v and mark them.

v	edgeTo[]	marked[]	distTo[]
0	–	T	0
1	0	T	1
2	0	T	1
3	4	T	3
4	2	T	2
5	3	T	4

vertices reachable from 0
(and shortest directed paths)

Breadth-first search

Repeat until queue is empty:

- Remove vertex v from queue.
- Add to queue all unmarked vertices adjacent from v and mark them.

← visit vertex v

BFS (from source vertex s)

Add s to FIFO queue and mark s.

Repeat until the queue is empty:

- **remove the least recently added vertex v**
- **for each unmarked vertex w adjacent from v:**
  add w to queue and mark w.

Breadth-first search: Java implementation

```
public class BreadthFirstDirectedPaths
{
   private boolean[] marked;
   private int[] edgeTo;
   private int[] distTo;
   …

   private void bfs(Digraph G, int s) {
      Queue<Integer> queue = new Queue<>();
      queue.enqueue(s);
      marked[s] = true;
      distTo[s] = 0;

      while (!queue.isEmpty()) {
         int v = queue.dequeue();
         for (int w : G.adj(v)) {
            if (!marked[w]) {
               queue.enqueue(w);
               marked[w] = true;
               edgeTo[w] = v;
               distTo[w] = distTo[v] + 1;
            }
         }
      }
   }

}
```

initialize FIFO queue of vertices to explore

found new vertex w via edge $v \rightarrow w$

https://algs4.cs.princeton.edu/42digraph/BreadthFirstDirectedPaths.java.html

Breadth-first search properties

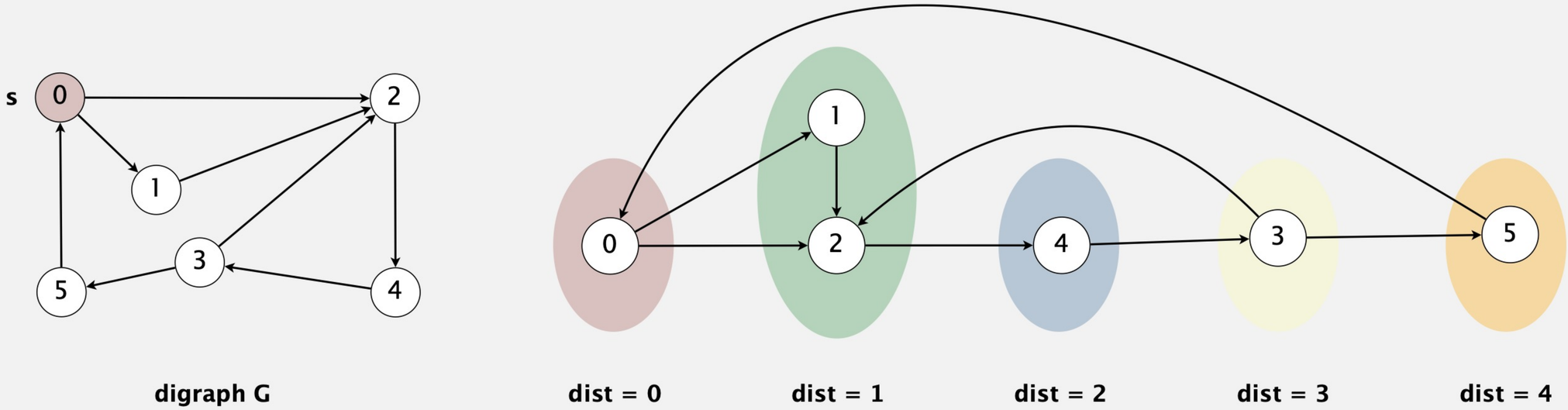
Proposition. In the worst case, BFS takes $\Theta(E + V)$ time.

Pf. Each vertex reachable from s is visited once.

Proposition. BFS computes shortest paths from s.

Pf idea. BFS examines vertices in increasing distance (number of edges) from s.

invariant: queue contains vertices of distance k from s,
followed by ≥ 0 vertices of distance k+1 (and no other vertices)



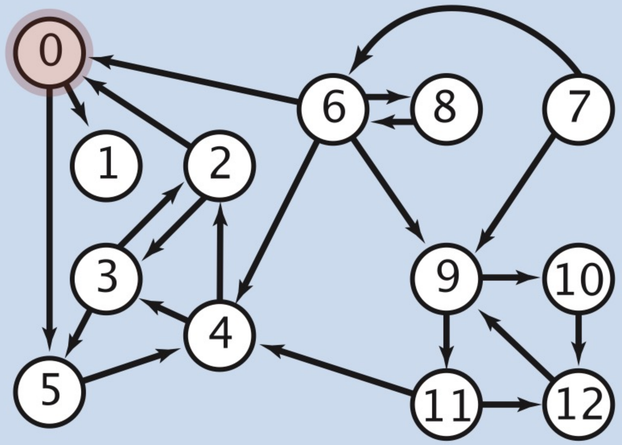
digraph G

SINGLE-SINK SHORTEST PATHS

Given a digraph and a target vertex t, find shortest path from every vertex to t.

Ex. $t = 0$

- Shortest path from 7 is $7 \to 6 \to 0$.
- Shortest path from 5 is $5 \to 4 \to 2 \to 0$.
- Shortest path from 12 is $12 \to 9 \to 11 \to 4 \to 2 \to 0$.



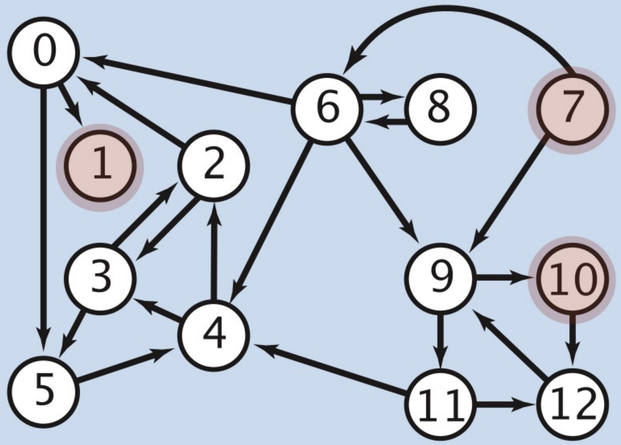
Q. How to implement single-target shortest paths algorithm?

MULTIPLE-SOURCE SHORTEST PATHS

Given a digraph and a set of source vertices, find shortest path from any vertex in the set to every other vertex.

Ex. $S = \{ 1, 7, 10 \}$.

- Shortest path to 4 is $7 \rightarrow 6 \rightarrow 4$.
- Shortest path to 5 is $7 \rightarrow 6 \rightarrow 0 \rightarrow 5$.
- Shortest path to 12 is $10 \rightarrow 12$.



needed for WordNet assignment

Q. How to implement multi-source shortest paths algorithm?

Graphs and digraphs: quiz 2

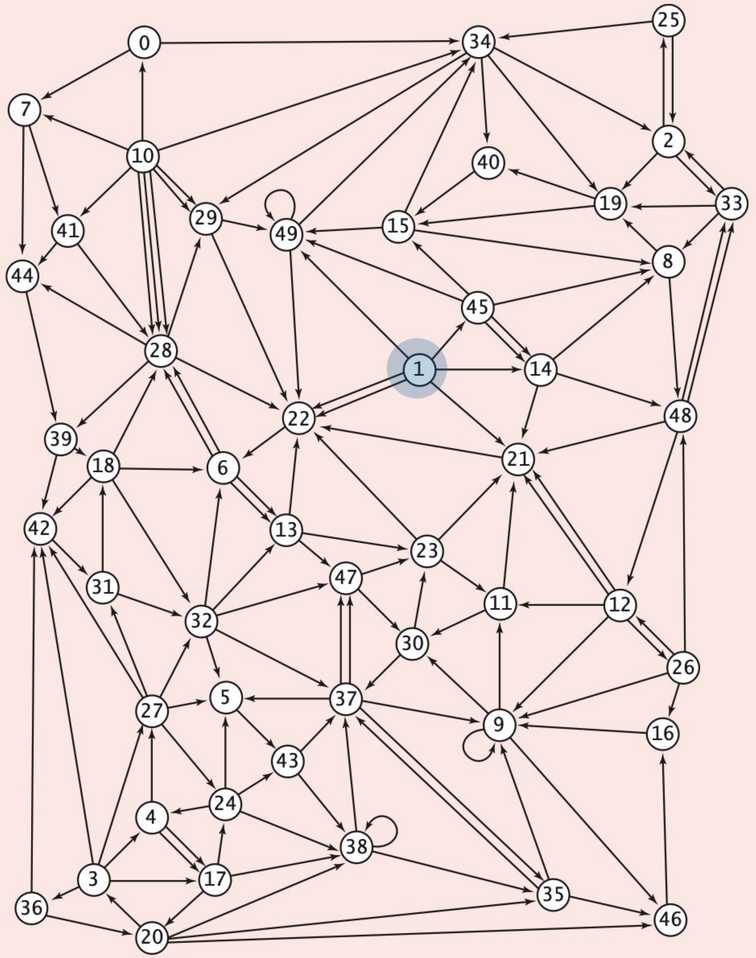
Suppose that you want to design a web crawler. Which algorithm should you use?

A. Depth-first search.

B. Breadth-first search.

C. Either A or B.

D. Neither A nor B.



Web crawler output

BFS crawl

```
http://www.princeton.edu
http://www.w3.org
http://ogp.me
http://giving.princeton.edu
http://www.princetonartmuseum.org
http://www.goprincetontigers.com
http://library.princeton.edu
http://helpdesk.princeton.edu
http://tigernet.princeton.edu
http://alumni.princeton.edu
http://gradschool.princeton.edu
http://vimeo.com
http://princetonusg.com
http://artmuseum.princeton.edu
http://jobs.princeton.edu
http://odoc.princeton.edu
http://blogs.princeton.edu
http://www.facebook.com
http://twitter.com
http://www.youtube.com
http://deimos.apple.com
http://qeprize.org
http://en.wikipedia.org
...
```

DFS crawl

```
http://www.princeton.edu
http://deimos.apple.com
http://www.youtube.com
http://www.google.com
http://news.google.com
http://csi.gstatic.com
http://googlenewsblog.blogspot.com
http://labs.google.com
http://groups.google.com
http://img1.blogblog.com
http://feeds.feedburner.com
http:/buttons.googlesyndication.com
http://fusion.google.com
http://insidesearch.blogspot.com
http://agoogleaday.com
http://static.googleusercontent.com
http://searchresearch1.blogspot.com
http://feedburner.google.com
http://www.dot.ca.gov
http://www.TahoeRoads.com
http://www.LakeTahoeTransit.com
http://www.laketahoe.com
http://ethel.tahoeguide.com
...
```

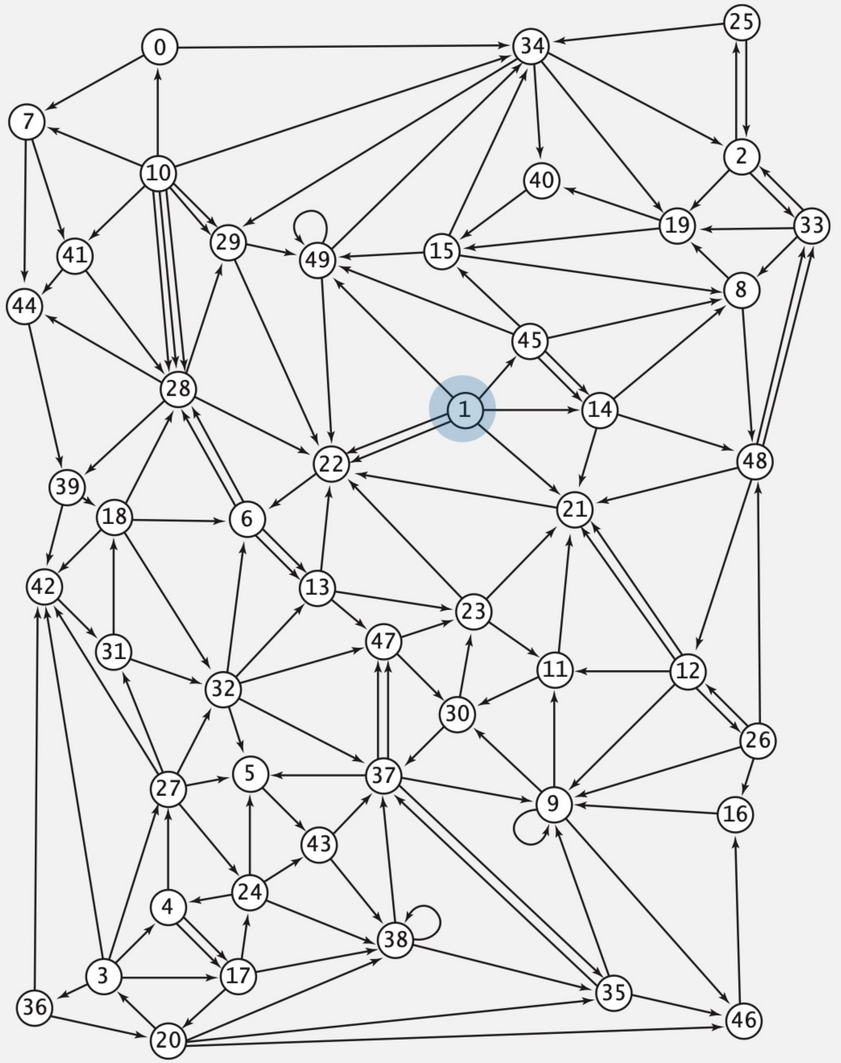
Breadth-first search application: web crawler

Goal. Crawl web, starting from some root web page, say `http://www.princeton.edu`.

Solution. [BFS with implicit digraph]

- Choose root web page as source s.
- Maintain a queue of websites to explore.
- Maintain a set of marked websites.
- Dequeue the next website and enqueue any unmarked websites to which it links.

Remark. Industrial-strength web crawlers use more sophisticated algorithms.



Bare-bones web crawler: Java implementation

```
Queue<String> queue = new Queue<>();
SET<String> marked = new SET<>();
```

queue of websites to crawl
set of marked websites

```
String root = "http://www.princeton.edu";
queue.enqueue(root);
marked.add(root);
```

start crawling from root website

```
while (!queue.isEmpty())
{
   String v = queue.dequeue();
   StdOut.println(v);
   In in = new In(v);
   String input = in.readAll();

   String regexp = "http://(\\w+\\.)+(\\w+)";
   Pattern pattern = Pattern.compile(regexp);
   Matcher matcher = pattern.matcher(input);

   while (matcher.find())
   {
      String w = matcher.group();

      if (!marked.contains(w))
      {
         marked.add(w);
         queue.enqueue(w);
      }

   }
}
```

read in raw HTML from next website in queue

use regular expression to find all URLs in website of form `http://xxx.yyy.zzz` [crude pattern misses relative URLs]

if unmarked, mark and enqueue

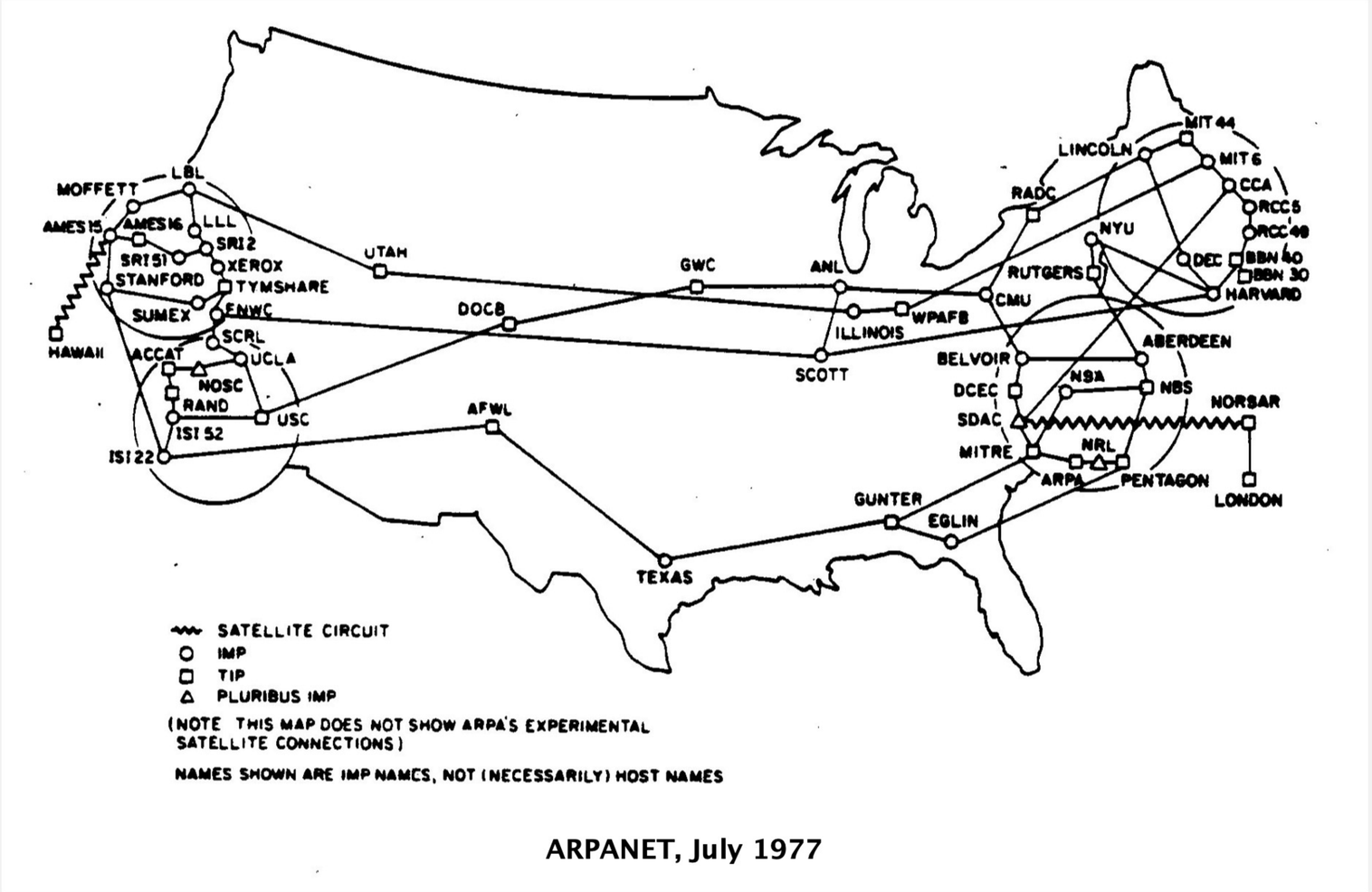
https://algs4.cs.princeton.edu

4. Graphs and Digraphs II

- *breadth-first search (in digraphs)*
- *breadth-first search (in undirected graphs)*
- *topological sort*
- *challenges*

Breadth-first search application: routing

Fewest number of hops in a communication network.



ARPANET, July 1977

Breadth-first search in undirected graphs

Problem. Find path between s and each other vertex that uses fewest edges.

Solution. Treat as a digraph, replacing each undirected edge with two antiparallel edges.

BFS (from source vertex s)

Add s to FIFO queue and mark s.

Repeat until the queue is empty:

- remove the least recently added vertex v
- for each unmarked vertex w adjacent to v:
  add w to queue and mark w.

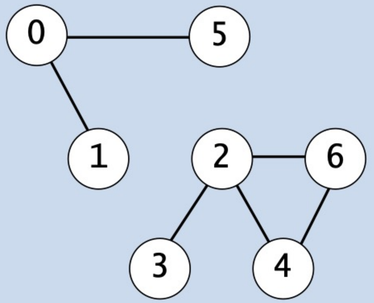
https://algs4.cs.princeton.edu

4. Graphs and Digraphs II

- *breadth-first search (in digraphs)*
- *breadth-first search (in undirected graphs)*
- *topological sort*
- *challenges*

Graph-processing challenge 1

Problem. Identify connected components.



0-1
0-5
2-6
2-3
2-4
4-6

How difficult?

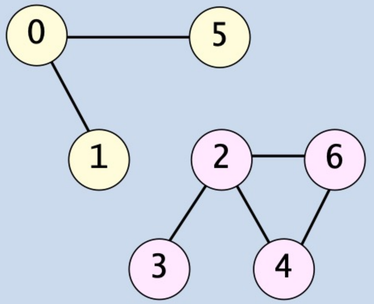
A. Any programmer could do it.

B. Diligent algorithms student could do it.

C. Hire an expert.

D. Intractable.

E. No one knows.



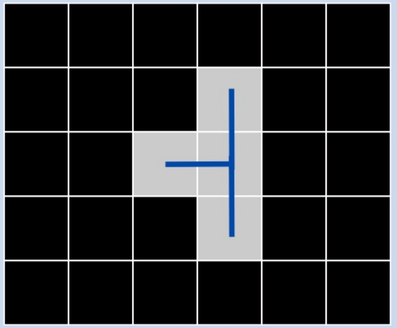
v	id[]
0	0
1	0
2	1
3	1
4	1
5	0
6	1

Graph-processing challenge 1

Problem. Identify connected components.

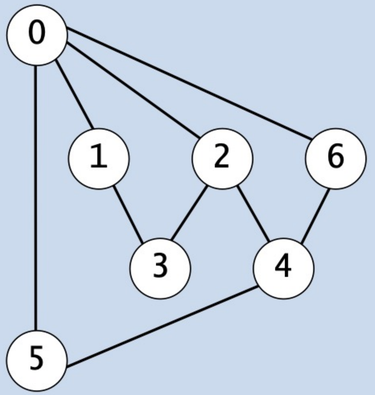
Particle detection. Given grayscale image of particles, identify “blobs.”

- Vertex: pixel.
- Edge: between two adjacent pixels with grayscale value ≥ 70.
- Blob: connected component of 20–30 pixels.



Graph-processing challenge 2

Problem. Is a graph bipartite?



0–1
0–2
0–5
0–6
1–3
2–3
2–4
4–5
4–6

How difficult?

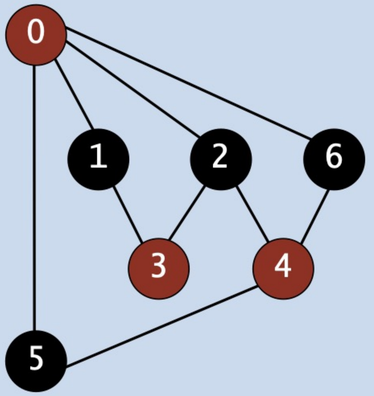
A. Any programmer could do it.

B. Diligent algorithms student could do it.

C. Hire an expert.

D. Intractable.

E. No one knows.

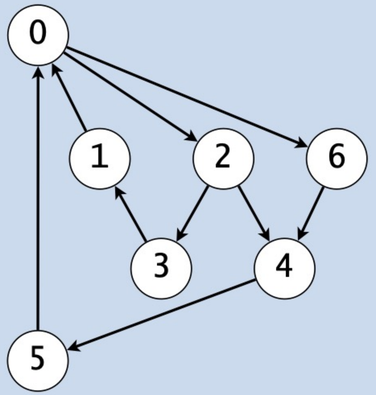


0–1
0–2
0–5
0–6
1–3
2–3
2–4
4–5
4–6

{ 0, 3, 4 }

Graph-processing challenge 3

Problem. Find the girth of a digraph (length of a shortest directed cycle).



0→2→3→1→0

0→2
0→6
1→0
2→3
3→1
2→4
4→5
5→0
6→4

How difficult?

A. Any programmer could do it.

B. Diligent algorithms student could do it.

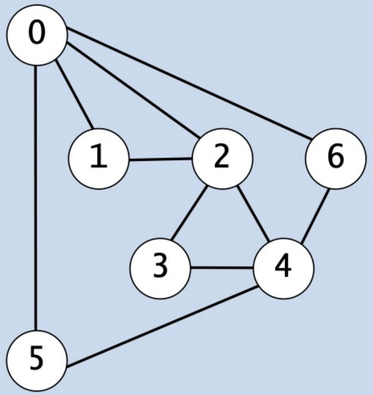
C. Hire an expert.

D. Intractable.

E. No one knows.

Graph-processing challenge 4

Problem. Is there a (non-simple) cycle that uses every edge exactly once?



0-1
0-2
0-5
0-6
1-2
2-3
2-4
3-4
4-5
4-6

0-1-2-3-4-2-0-6-4-5-0

How difficult?

A. Any programmer could do it.

B. Diligent algorithms student could do it.

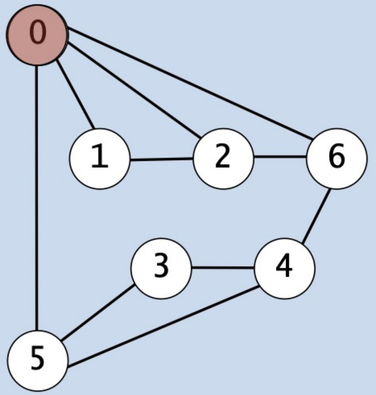
C. Hire an expert.

D. Intractable.

E. No one knows.

Graph-processing challenge 5

Problem. Is there a cycle that uses every vertex exactly once?



0-1
0-2
0-5
0-6
1-2
2-6
3-4
3-5
4-5
4-6

0-5-3-4-6-2-1-0

How difficult?

A. Any programmer could do it.

B. Diligent algorithms student could do it.

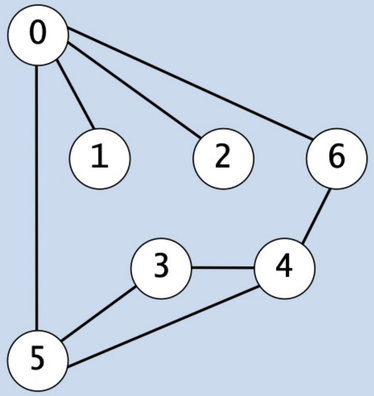
C. Hire an expert.

D. Intractable.

E. No one knows.

Graph-processing challenge 6

Problem. Are two graphs identical except for vertex names?



0–1
0–2
0–5
0–6
3–4
3–5
4–5
4–6

How difficult?

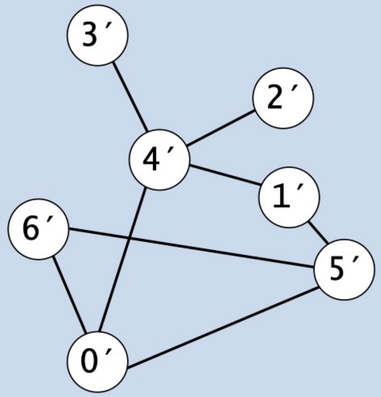
A. Any programmer could do it.

B. Diligent algorithms student could do it.

C. Hire an expert.

D. Intractable.

E. No one knows.



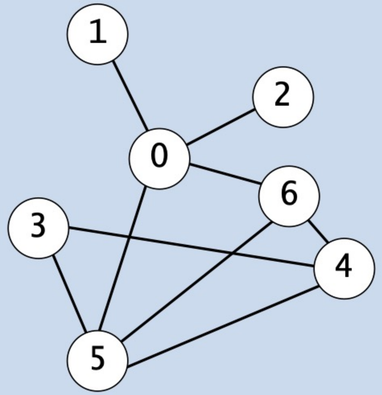
0′–4′
0′–5′
0′–6′
1′–4′
1′–5′
2′–4′
3′–4′
5′–6′

0⇔4′, 1⇔3′, 2⇔2′, 3⇔6′, 4⇔5′, 5⇔0′, 6⇔1′

Graph-processing challenge 7

Problem. Can you draw a graph in the plane with no crossing edges?

try it yourself at https://www.jasondavies.com/planarity/



0–1
0–2
0–5
0–6
3–4
3–5
4–5
4–6
5–6

How difficult?

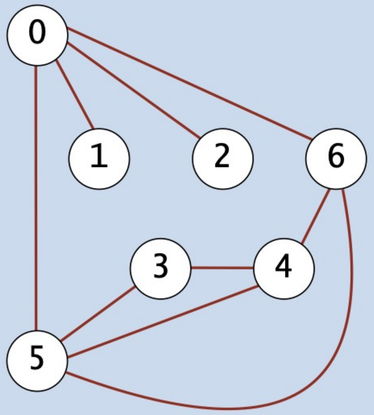
A. Any programmer could do it.

B. Diligent algorithms student could do it.

C. Hire an expert.

D. Intractable.

E. No one knows.



Graph traversal summary

BFS and DFS enables efficient solution of many (but not all) graph and digraph problems.

graph problem	BFS	DFS	time
s–t path	✔	✔	$E + V$
shortest s–t path	✔		$E + V$
shortest directed cycle (girth)	✔		$E\,V$
Euler cycle		✔	$E + V$
Hamilton cycle			$2^{1.657\,V}$
bipartiteness (odd cycle)	✔	✔	$E + V$
connected components	✔	✔	$E + V$
strong components		✔	$E + V$
planarity		✔	$E + V$
graph isomorphism			$2^{c \ln^3 V}$