Computer Algorithms I, Spring 2020

Graphs review, breadth-first search

March 30, April 1

- review of basic graph concepts
- breadth-first search (BFS)

How to represent a graph?

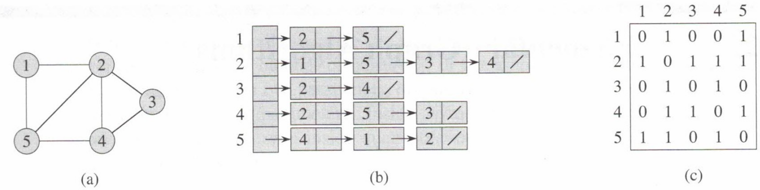
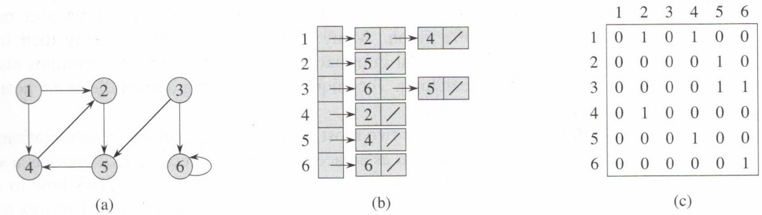


Figure 22.1 Two representations of an undirected graph. **(a)** An undirected graph G with 5 vertices and 7 edges. **(b)** An adjacency-list representation of G. **(c)** The adjacency-matrix representation of G.



Graph basics

graphs: Appendix B.4 (pages 1168-1172), trees: Appendix B.5.1, B.5.2 (pages 1173-1177)

adjacency matrices - for dense graphs (e.g., all edges present - complete graph)

adjacency lists - for sparse graphs (e.g., $O(n)$ edges, note: large networks are sparse)

weights - edges can also have numbers assigned to them, e.g., distance, cost

notation of attributes: e.g., $u.color$ is the color of vertex u

path: $\langle a, b, c, d, e \rangle$: $(a, b), (b, c), (c, d), (d, e)$ are all edges of the graph

length of path: number of edges

shortest path between u and v: path of minimal length from u to v

distance of u and v: length of shortest path between them

Graph searches

for many graph algorithms we need to process the vertices by visiting them in some order

with an adjacency list, we could simply go through the array of vertices - but that is arbitrary

better solutions: visit the vertices in some "natural" order, related to the graph itself

two basic approaches: BFS (breadth-first search) and DFS (depth-first search)

these often serve as skeletons of graph algorithms

Breadth-first search

start with a vertex s, visit its neighbors, then the neighbors' neighbors, etc.

network models of processes on graph (perhaps several starting points, probabilities, ...)

vertices are initially white ("undiscovered", "unprocessed"), then become gray ("discovered but not finished", "under processing") and finally black ("finished", "completely processed")

natural data structure: queue, FIFO - first-in, first-out

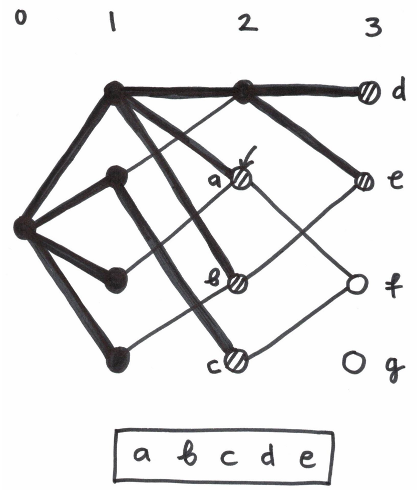
some time after a vertex is discovered, all its undiscovered neighbors become discovered

attributes: $u.color$, $u.d$, $u.\pi$

at the end of the algorithm $u.d$ is the distance of s from u, and $u.\pi$ is the predecessor of u on a shortest path from s to u

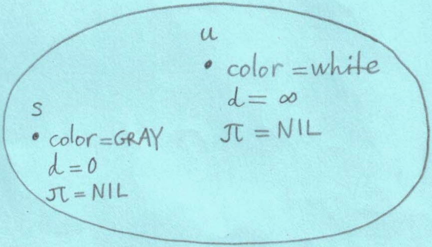
edges $(u.\pi, u)$ form a tree, the breadth-first search tree from s

Breadth-first search example I

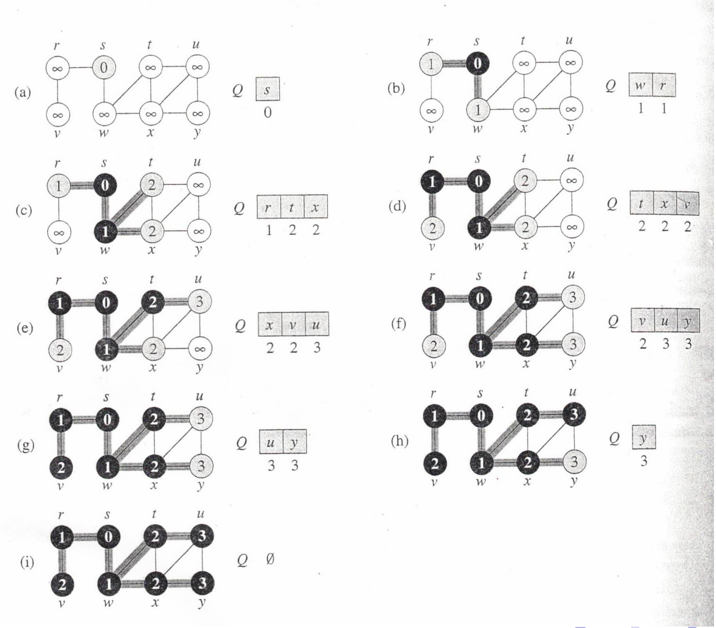


Breadth-first search algorithm

```
BFS(G, s)
for each vertex u ∈ G.V − {s}
    u.color = WHITE
    u.d = ∞
    u.π = NIL
s.color = GRAY
s.d = 0
s.π = NIL
Q = ∅
ENQUEUE(Q, s)
while Q ≠ ∅
    u = DEQUEUE(Q)
    for each v ∈ G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```



Breadth-first search example II



Adjacency lists memory requirement

$G = (V, E)$ undirected graph (similar for directed graphs)

$deg(v)$: degree of vertex, number of its neighbors

size of array + combined size of linked lists

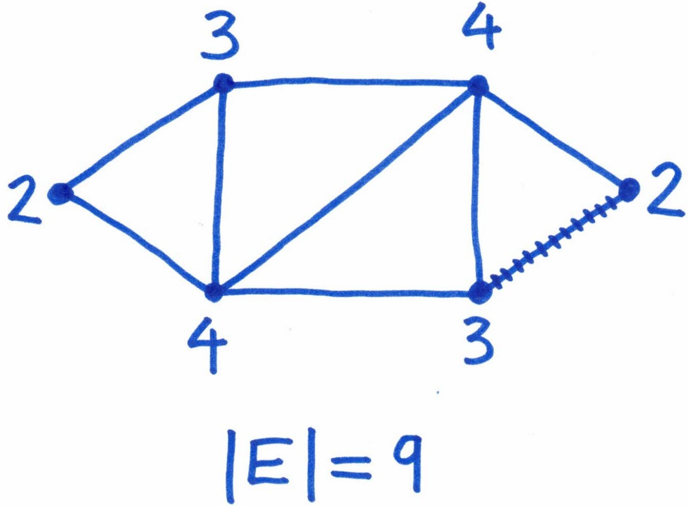
length of adjacency list of v: $deg(v)$

$$\sum_{v \in V} deg(v) = 2|E|$$

adjacency lists use memory $O(|V| + |E|)$

algorithms with running time $O(|V| + |E|)$ are optimal

Degrees



Running time of BFS

lines 1 - 9 : $O(|V|)$

lines 11 - 18 : $O(deg(v))$

while loop : $O(\sum_{v \in V} deg(v)) = O(|E|)$

running time: $O(|V| + |E|)$

BFS and distances

$\delta(s, v)$: distance of s and v

v reachable from s: there is a path from s to v

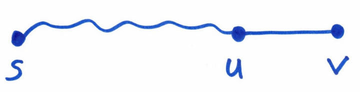
Theorem

A vertex is reachable from s iff it is discovered. If v is reachable then at the end $v.d = \delta(s, v)$. The backward path from v to s through predecessors is a shortest path from s to v. Edges $(v.\pi, v)$ form a tree, the BFS tree.

note: d-values in queue: $5, 5, 5, 6, 6, 6, 6, 6$

Proof outline

reachable $\rightarrow$ discovered and $v.d = \delta(s, v)$
assume false: let v be a counterexample with $\delta(s, v)$ as small as possible, and consider a shortest path from s to v



$\delta(s, v) = \delta(s, u) + 1$ (optimal substructure!)
$u.d = \delta(s, u)$
when u is removed from the queue:

- v is white: $v.d = u.d + 1$ so $v.d = \delta(s, v)$, contradiction
- v is gray or black: it became grey when some w was removed from the queue before u, but then $v.d = w.d + 1 \leq u.d + 1 = \delta(s, u) + 1 = \delta(s, v)$, again contradiction