Unweighted directed graphs
BFS and DFS in trees

Solve problems by making a tree of the state space

max

min

max
BFS and DFS in trees

Often times, fully exploring the state space is too costly (takes forever)

Chess: $10^{47}$ states (tree about $10^{123}$)
Go: $10^{171}$ states (tree about $10^{360}$)

At 1 million states per second...
Chess: $10^{109}$ years (past heat death
Go: $10^{346}$ years of universe)
BFS and DFS in trees

BFS prioritizes “exploring”
DFS prioritizes “exploiting”

White to move	Black to move
BFS and DFS in trees

BFS benefits?

DFS benefits?
BFS and DFS in trees

BFS benefits?
- can evaluate best path

DFS benefits?
- uses less memory on complete search
BFS and DFS in graphs

BFS: shortest path from origin to any node

DFS: find graph structure

Both running time of $O(V+E)$
Breadth first search

BFS(G,s) // to find shortest path from s
for all v in V
    v.color=white, v.d=∞, v.π=NIL
s.color=grey, v.d=0
Enqueue(Q,s)

while(Q not empty)
    u = Dequeue(Q,s)
    for v in G.adj[u]
        if v.color == white
            v.color=grey, v.d=u.d+1, v.π=u
            Enqueue(Q,v)
    u.color=black
Breadth first search

Let $\delta(s,v)$ be the shortest path from $s$ to $v$

After running BFS you can find this path as: $v.\pi$ to $(v.\pi).\pi$ to ... $s$

(pseudo code on p. 601, recursion)
BFS correctness

Proof: contradiction
Assume $\delta(s,v) \neq v.d$
$v.d \geq \delta(s,v)$ (Lemma 22.2, induction)
Thus $v.d > \delta(s,v)$
Let $u$ be previous node on $\delta(s,v)$
Thus $\delta(s,v) = \delta(s,u)+1$
and $\delta(s,u) = u.d$
Then $v.d > \delta(s,v) = \delta(s,u)+1 = u.d+1$
BFS correctness

\[ v.d > \delta(s,v) = \delta(s,u) + 1 = u.d + 1 \]

Cases on color of \( v \) when \( u \) dequeue,

all cases invalidate top equation

Case white: alg sets \( v.d = u.d + 1 \)

Case black: already removed

thus \( v.d \leq u.d \) (corollary 22.4)

Case grey: exists \( w \) that dequeued \( v \),

\( v.d = w.d + 1 \leq u.d + 1 \) (corollary 22.4)
Depth first search

DFS(G)
for all v in V
  v.color=white, v.π=NIL
time=0
for each v in V
  if v.color==white
    DFS-Visit(G,v)
Depth first search

DFS-Visit(G,u)

time=time+1

u.d=time, u.color=grey

for each v in G.adj[u]
   if v.color == white
      v.π=u
      DFS-Visit(G,v)

u.color=black, time=time+1, u.f=time
Depth first search

Edge markers:

Consider edge u to v

C = Edge to black node \( (u.d > v.f) \)

B = Edge to grey node \( (u.f < v.f) \)

F = Edge to black node \( (u.f > v.f) \)
Depth first search

DFS can do topographical sort

Run DFS, sort in decreasing finish time
DFS can find strongly connected components
Depth first search

Let $G^T$ be $G$ with edges reversed

Then to get strongly connected:
1. DFS($G$) to get finish times
2. Compute $G^T$
3. DFS($G^T$) on vertex in decreasing finish time
4. Each tree in forest SC component