Debugging order.
  ↳ topological [████] sort   B→A
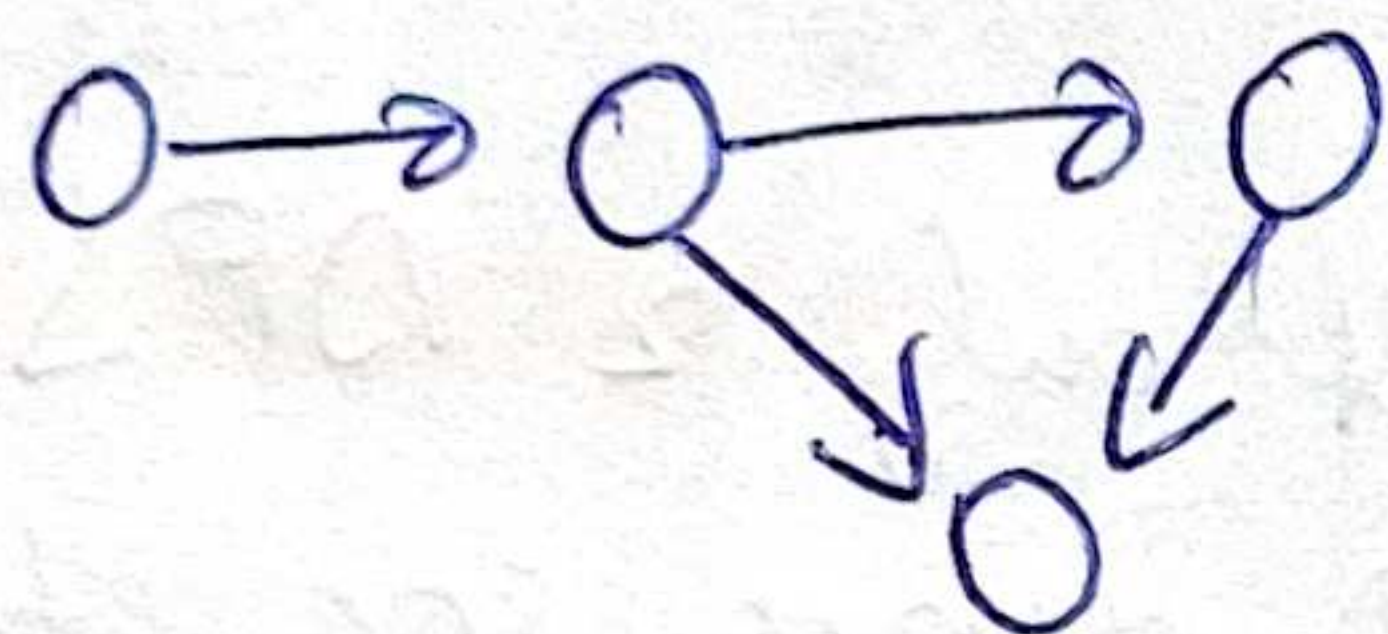
  ↳ DFS (decr. postorder)

SCC : Strongly Connected Components

Source   ◯ indeg 0
                          ↓
sink   ↓ outdeg 0
                          ◯

need   source SCC
       sink SCC ⇐ DFS

lemma [██] if we start a DFS at a vertex in a sink SCC
         ↳ get sink SCC before jumping

problem: find a vertex in a sink SCC
  – run DFS, smallest postorder → FALSE

lemma [██] largest postorder or in the source SCC
      proof by contradiction: assume postorder(v) < postorder(u)

case 1   u is on stack before v
         contradiction

case 2   v is on stack before u
         there is a path from
         v to u
         contradiction

lemma   reverse the graph ⇒
        sinks become sources
        sources become sinks

① take G, form G^R

② Do a BFS on G^R
   find source vertex in G^R, sink vertex in G

③ Run BFS on sink vertex in G
   and remove it

$O(2v) = O(v)$ DFS
       $O(E)$ per DFS

get tighter bound:

$O(EV)$

But only 2 DFSs!

② remove scc, then
   get largest post order
   from remaining

③ go through each edge
              once

$$O(E + U)$$

     (notion of distance)
   BFS ⟺ queue

Breadth First Search
pick start vertex s
$dist[s] := 0$

$O(1)$ inject (q, s)
        placed $(s) := 1$

$O(v)$ while $(size(q) > 0)$
          $v := pop(q)$

$O(E)$ for $(v, w) \in E$
total   if placed $(w) = 0$
             inject (q, w) ⟵
             placed $(w) := 1$
             $dist[w] := dist[v] + 1$

$\{s\}$
$\{A, C\}$
$\{C, B, D, E\}$
$\{B, D, E, F\}$
$\{E, F, G\}$

$$O(E + V)$$

pop
min ⟵ only 2 Levels

level
by level

of edges distance
→ very basic
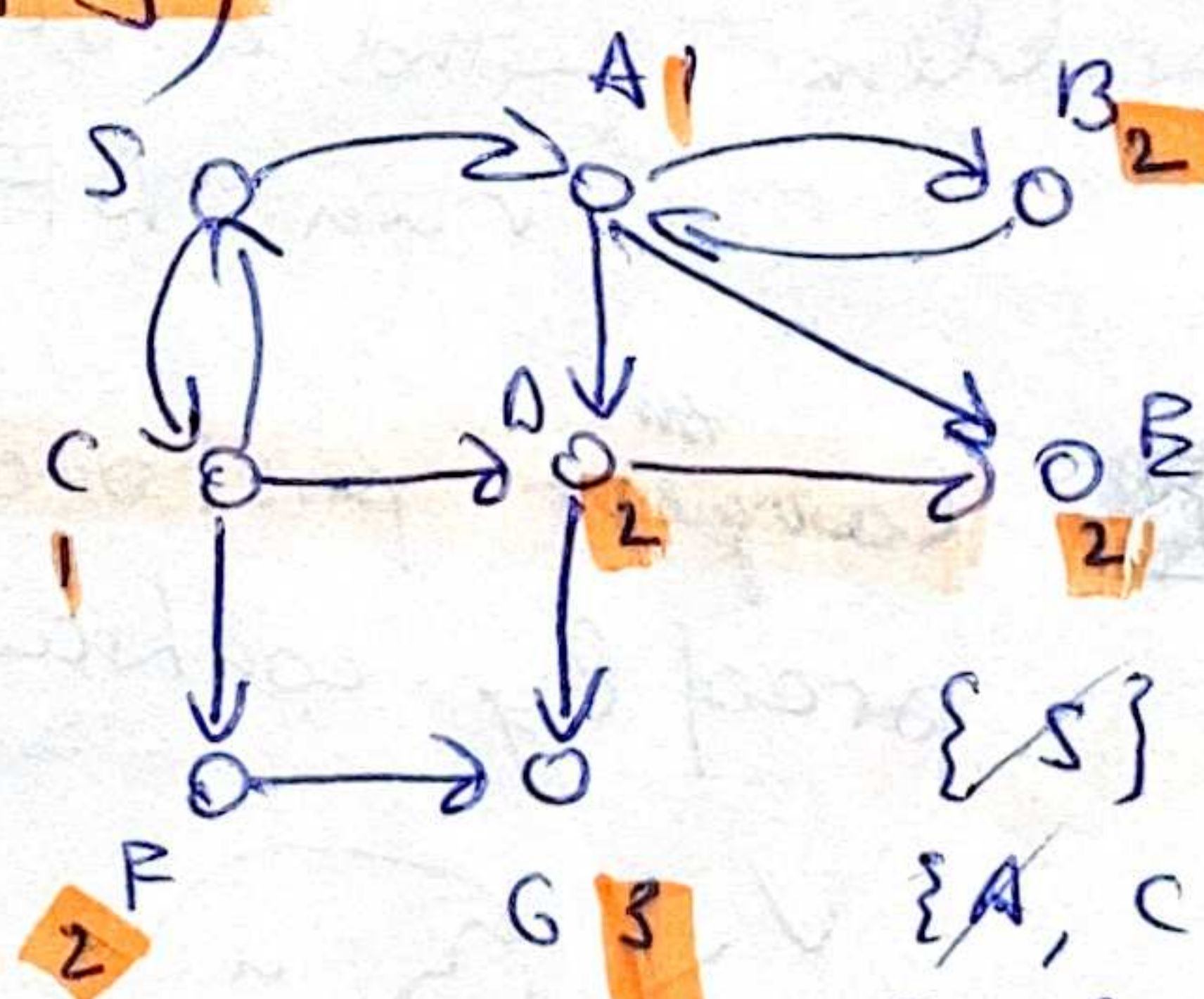
queue pop

inject

shortest # of edges distance from
s to any vertex that can be reached
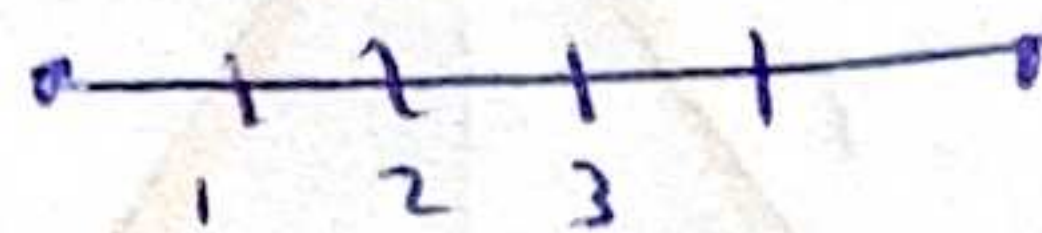by s by induction
$O(pop)$

BFS invariant

- queue is sufficient to pop minimums
- each node in the queue has shortest path
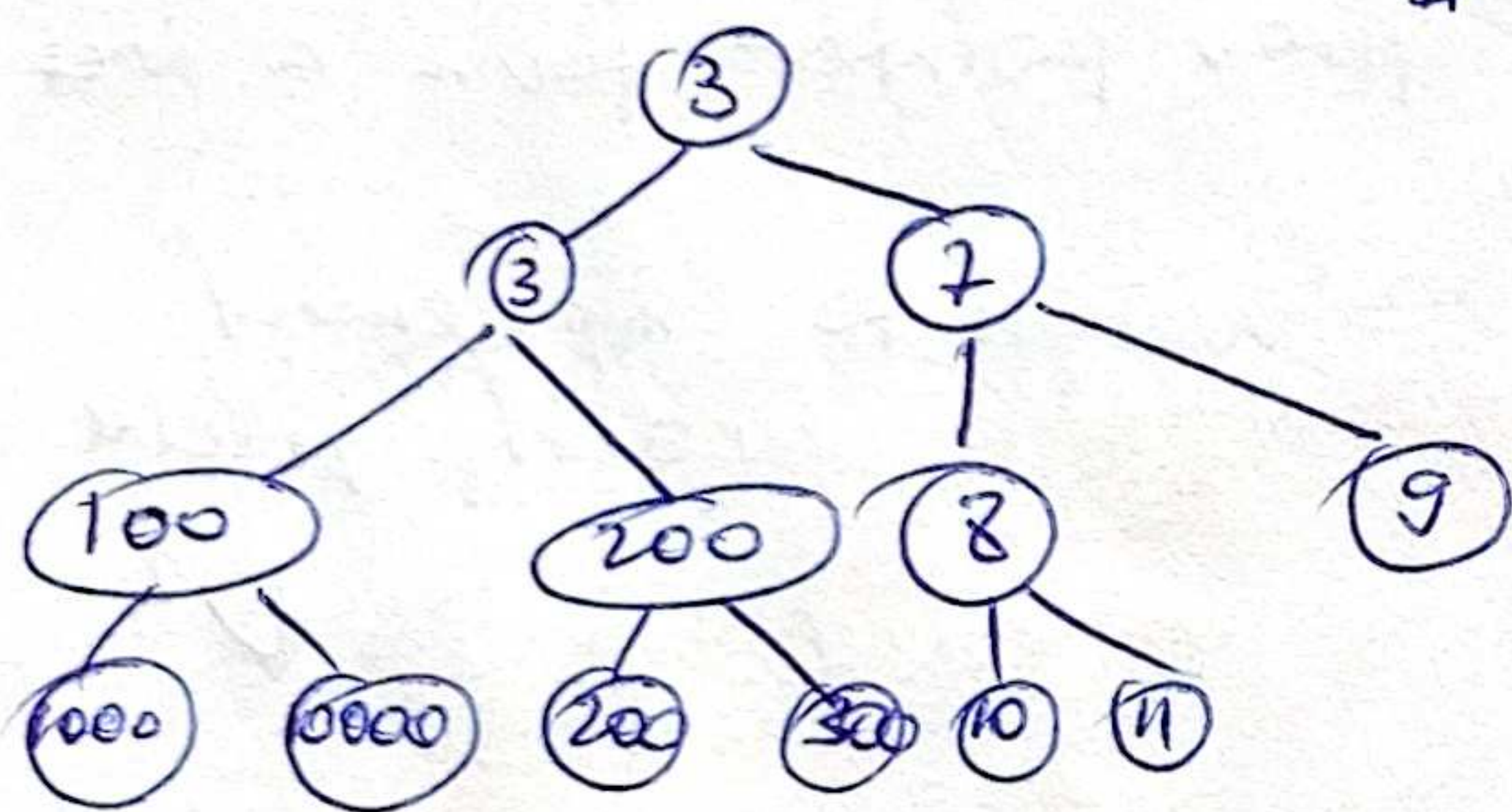
Add weights

idea reduce to BFS

$O(\sum \text{length all edges})$

Priority queue: different implementations

Heap   min-heap (Binary tree(pts same level))
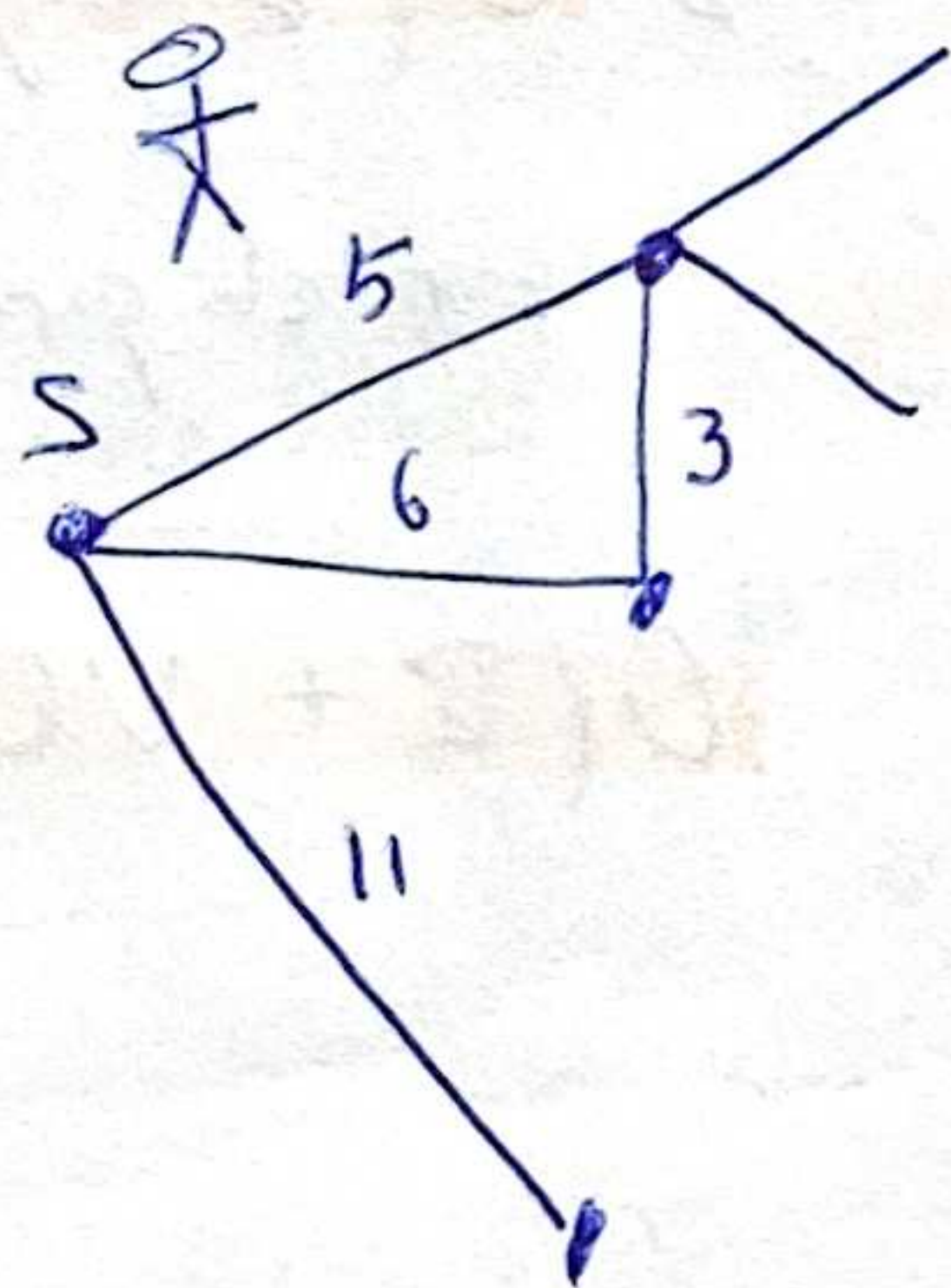# increases as go down any path

delete min — return object w/ smallest value and restructure

insert (x, y, h)
— insert item x with value y in the heap

change (x, y, h)
— if the current value for x is $> y$, change its value

Dijkstra's shortest path alg

dist, prev. arrays, heap H   { priority queue may be heap

$H := \{S, 0\}$
for $v \in V$ do
  $dist[v] := \infty$
  $prev[v] := null$
$dist[S] := 0$
while $H \neq \emptyset$
  $v := delete min (H)$
  for $(v, w) \in E$
    if $dist[w] > dist[v] + length(v, w)$
      $dist[w] := dist[v] + length(v, w)$
      $prev[w] := v$
      change / insert $(w, dist[w], h)$

induction on popped vertices
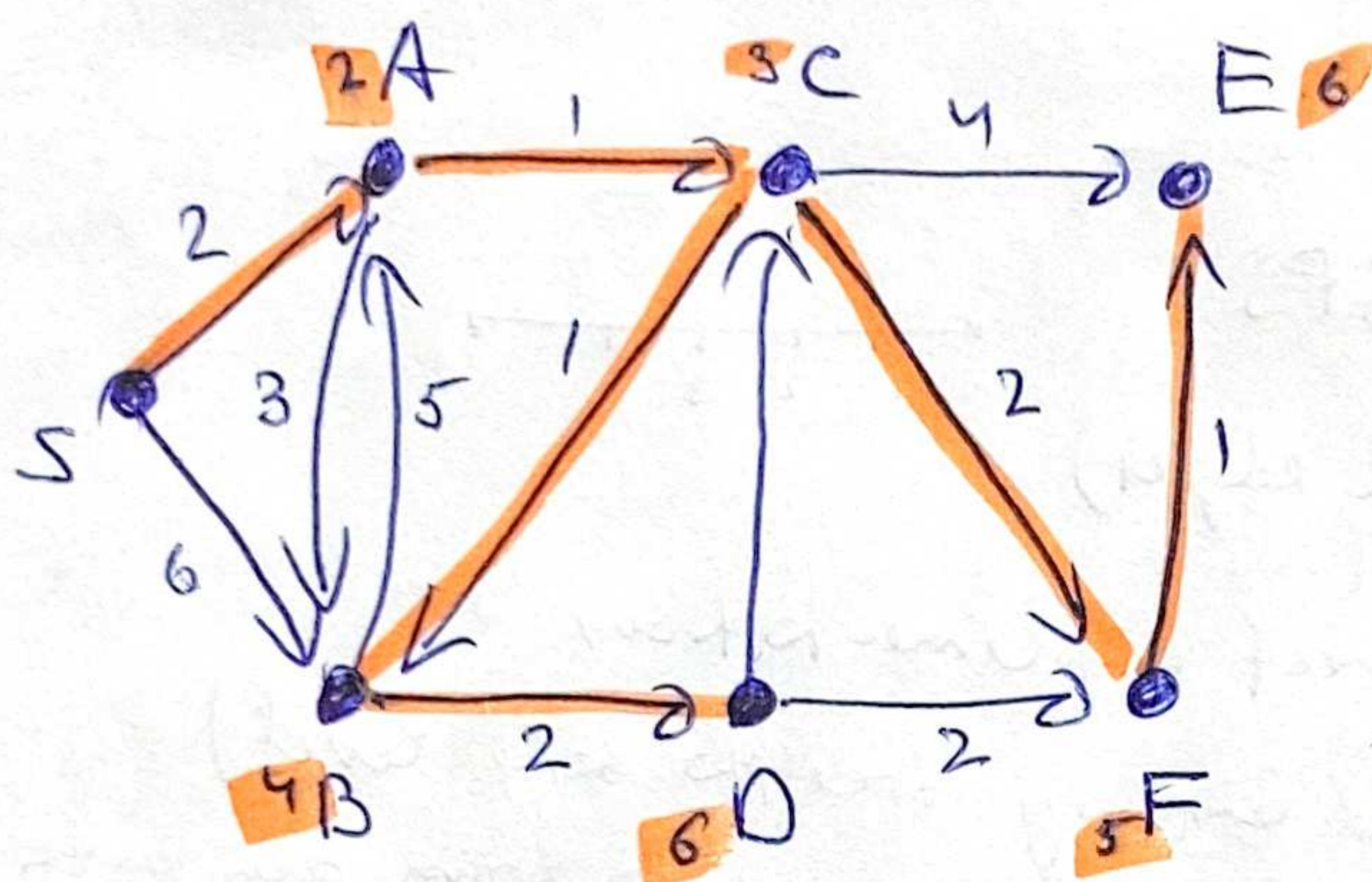
→ each vertex deleted,
⟹ a shortest path to it is
     found. All positive
distances, paths to other
fringe nodes are longer/as long

invariant:

▓▓ min node in priority queue has shortest path ▓▓▓▓▓▓

Graph with nodes: A (2), C (3), E (6), S, B (4), D (6), F (5)

Edge weights shown: S–A: 2, S–B: 6, A–C: 1, A–B: 3, A–?: 5, C–E: 4, C–?: 1, C–F: 2, C–D: ?, E–F: 1, B–D: 2, D–F: 2

S : 0

A : 2 , B : 6

C : 3 , B : 5

B : 4 , E : 7 , F : 5

F : 5 , D : 6 , E : 7

B : 6 , D : 6

prev pointer from a tree

ways to implement priority queue

letter of $E = V^2$

$O(V \text{ deletemin} + E \text{ insert})$
$\quad\quad\quad\quad\quad\quad\quad\quad$ change

Linked List

$O(V)$ $\quad\quad\quad$ $O(1)$ ← array of
$\quad\quad\quad\quad\quad\quad\quad\quad$ pointers to change $\;\Rightarrow O(V^2)$

Binary heap

$O(\log V)$ $\quad\quad$ $O(\log V)$ $\quad\quad\quad\quad\quad \Rightarrow O(E \log V)$
$\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad$ ↑
$\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad$ rich graph

d-ary heap

Fibonacci heap

$O(\log V)$ $\quad\quad$ $O(1)$ amortized $\quad\quad\quad\quad$ $O(E + V \log V)$
$\quad\quad\quad\quad\quad\quad\quad\quad$ ↑
$\quad\quad\quad\quad\quad\quad\quad$ on average over
$\quad\quad\quad\quad\quad\quad\quad$ the course of all ops