# Dijkstra’s Algorithm

Adapted from the CLRS book slides

# DIJKSTRA'S ALGORITHM

No negative-weight edges!

Have two sets of vertices:

- $S$ = vertices whose final shortest-path weights are determined,
- $Q$ = priority queue = $V - S$.

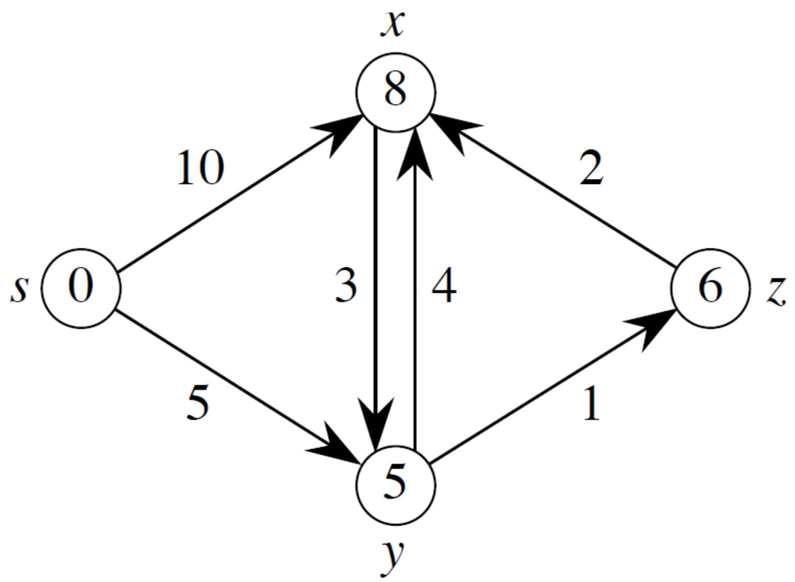
# DIJKSTRA'S ALGORITHM (continued)

```
DIJKSTRA(G, w, s)
INITIALIZE-SINGLE-SOURCE(G, s)
S = Ø
Q = Ø
for each vertex u ∈ G.V
    INSERT(Q, u)
while Q ≠ Ø
    u = EXTRACT-MIN(Q)
    S = S ∪ {u}
    for each vertex v in G.Adj[u]
        RELAX(u, v, w)
        if the call of RELAX decreased v.d
            DECREASE-KEY(Q, v, v.d)
```

# DIJKSTRA'S ALGORITHM (continued)

- Looks a lot like Prim's algorithm, but computing $v.d$, and using shortest-path weights as keys.
- Dijkstra's algorithm can be viewed as greedy, since it always chooses the "lightest" ("closest"?) vertex in $V - S$ to add to $S$.

# EXAMPLE




Order of adding to $S$: $s, y, z, x$.

***Correctness***

The algorithm extracts vertices from the heap in order of shortest distance from the source. Inductively, if the algorithm has found the shortest paths to some set *S*, the shortest path to the closest vertex in *V-S* can be found by appending a single edge to a path to some vertex in *S*.

# ANALYSIS

$|V|$ INSERT and EXTRACT-MIN operations.
$\leq |E|$ DECREASE-KEY operations.

Like Prim's algorithm, depends on implementation of priority queue.

- If binary heap, each operation takes $O(\lg V)$ time $\Rightarrow O(E \lg V)$.
- If a Fibonacci heap:
  - Each DECREASE-KEY takes $O(1)$ amortized time.
  - There are $\Theta(V)$ INSERT and EXTRACT-MIN operations, taking $O(\lg V)$ amortized time each.
  - Therefore, time is $O(V \lg V + E)$.