1) start w/ any vertex $s$; set $w(s)=0$
2) set $w(\neq s) = \infty$ & put all in pr. queue
3) while pr. queue not empty
   \[ x: \text{extract-min} \ & \text{add edge to } T \]
   mark $x \rightarrow$ in $T$.
   for each unmarked neighbor $v$ of $x$
     if $w(v) > w(v,x)$ then decrease.
Remember PRIM'S ALGORITHM for MST?

(after initializing)
while pr.queue not empty
    x: extract-min & add edge to T
    mark x -> in T.
    for each unmarked neighbor v of x
        if w(v) > w(v,x) then decrease.
(after initializing)
while pr.queue not empty
  x: extract-min & add edge to T
  mark x -> in T.
  for each unmarked neighbor v of x
    if w(v) > w(v,x) then decrease.
    RELAX(x,v)
Dijkstra's Algorithm for SSSP
(1956 - 1959)

(after initializing)
while pr.queue not empty
  x: extract-min & add edge to T
  mark x → in T.
  for each neighbor v of x
    relax(x, v)
Dijkstra’s Algorithm for SSSP

(after initializing)

while pr.queue not empty

  x: extract-min & add edge to T

  mark x→ in T.

  for each neighbor v of x
  RELAX(x,v)
Dijkstra’s Algorithm for SSSP

(after initializing)
while pr.queue not empty
  x: extract-min & add edge to T
  mark x → in T.
  for each neighbor v of x
    RELAX(x, v)

extract source & relax two edges
Dijkstra's Algorithm for SSSP

(after initializing)

while pr.queue not empty
    x: extract-min & add edge to T
    mark x→ in T.
    for each neighbor v of x
        RELAX(x,v)

extract min: 10
Dijkstra's Algorithm for SSSP

(after initializing)
while pr.queue not empty
    x: extract-min & add edge to T
    mark x → in T.
    for each neighbor v of x
        RELAX(x,v)

Relax neighbors of 10
DIJKSTRA'S ALGORITHM for SSSP

(after initializing)
while pr.queue not empty
  x: extract-min & add edge to T
  mark x→ in T.
  for each neighbor v of x
    RELAX(x,v)
Dijkstra's Algorithm for SSSP

(after initializing)

while pr.queue not empty

    x: extract-min & add edge to T

    mark x→ in T.

    for each neighbor v of x

        RELAX(x,v)

No update from relaxing
DIJKSTRA'S ALGORITHM for SSSP

(after initializing)
while pr.queue not empty
x: extract-min & add edge to T
mark x → in T.
for each neighbor v of x
RELAX(x, v)
Dijkstra's Algorithm for SSSP

(after initializing)

while pr.queue not empty
    x: extract-min & add edge to T
    mark x -> in T.
    for each neighbor v of x
        RELAX(x, v)
DIJKSTRA'S ALGORITHM for SSSP

(after initializing)

while pr.queue not empty
x: extract-min & add edge to T
mark x → in T.
for each neighbor v of x
RELAX(x,v)
DIJKSTRA’S ALGORITHM for SSSP

(after initializing)

while priority_queue not empty

x: extract-min & add edge to T
mark x → in T.
for each neighbor v of x
RELAX(x, v)
Dijkstra's Algorithm for SSSP

(after initializing)

while pr.queue not empty

\[ \text{x: extract-min \& add edge to T} \]

mark \( x \rightarrow \) in \( T \).

for each neighbor \( v \) of \( x \)

RELAX(\( x, v \))
DIJKSTRA'S ALGORITHM for SSSP

(after initializing)
while pr.queue not empty
    x: extract-min & add edge to T
    mark x -> in T.
    for each neighbor v of x
        RELAX(x, v)
DIJKSTRA'S ALGORITHM for SSSP

(after initializing)
while pr.queue not empty
  x: extract-min & add edge to T
  mark x -> in T.
  for each neighbor v of x
    RELAX(x,v)

etc

time?
Dijkstra's Algorithm for SSSP

(after initializing)

while priority queue not empty

    x: extract-min & add edge to T
    mark x → in T.

    for each neighbor v of x
        RELAX(x,v)

    etc

time: $O(V^2)$ or $O(E \log V)$

(like Prim's algo) // for fancier see CLRS
Correctness:

assume
we have shortest path to a set of red vertices
Correctness:

Assume we have shortest path to a set of red vertices.

Somewhere outside this set is a vertex $v$ with shortest path $\equiv [\text{a path in known set}] + \text{black edge}$. 
Correctness:

Assume we have shortest path to a set of red vertices.

Somewhere outside this set is a vertex $v$ with shortest path $=$

= [a path in known set] + black edge

Any other path $s \rightarrow v$ will cost more.
Correctness:

assume we have shortest path to a set of red vertices

Somewhere outside this set is a vertex \( v \) with shortest path =

\[ [\text{a path in known set}] + \text{black edge} \]

Any other path \( s \rightarrow v \) will cost more