Single Source Shortest Paths

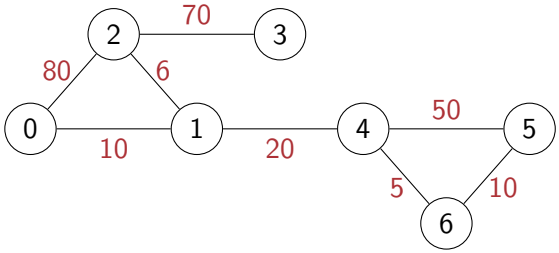
Madhavan Mukund

https://www.cmi.ac.in/~madhavan

Mathematics for Data Science 1
Week 12

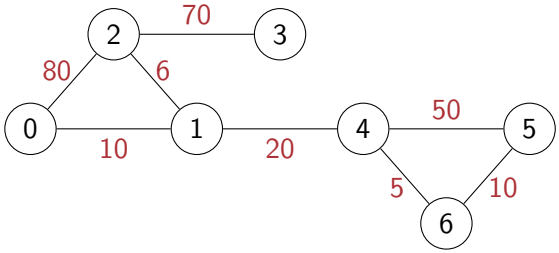
Single source shortest paths

- Weighted graph:
  - $G = (V, E)$
  - $W : E \to \mathbb{R}$



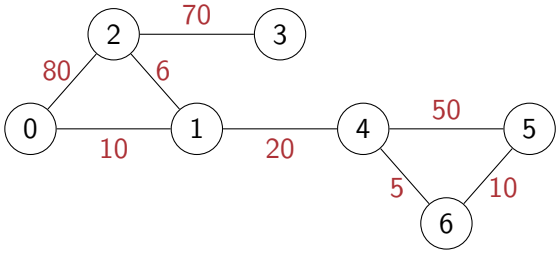
Single source shortest paths

- Weighted graph:
  - $G = (V, E)$
  - $W : E \to \mathbb{R}$
- Single source shortest paths
  - Find shortest paths from a fixed vertex to every other vertex



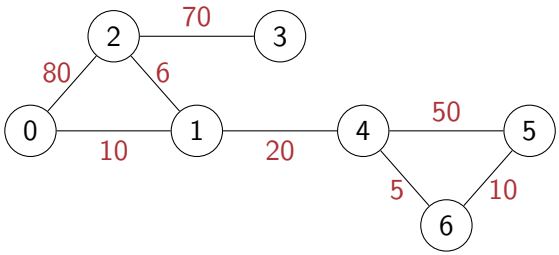
Single source shortest paths

- Weighted graph:
  - $G = (V, E)$
  - $W : E \to \mathbb{R}$
- Single source shortest paths
  - Find shortest paths from a fixed vertex to every other vertex
- Assume, for now, that edge weights are all non-negative



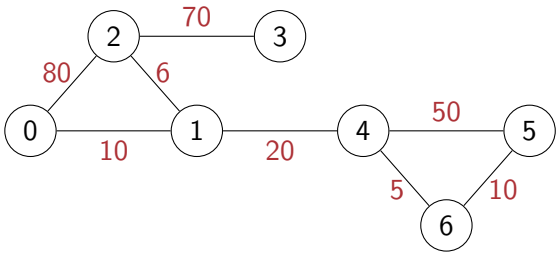
Single source shortest paths

- Compute shortest paths from 0 to all other vertices



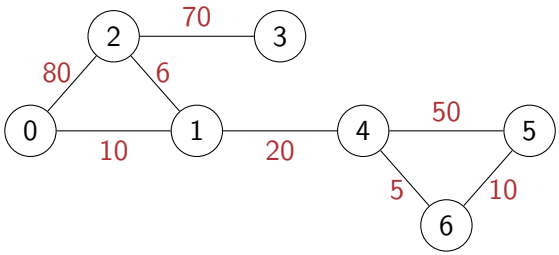
Single source shortest paths

- Compute shortest paths from 0 to all other vertices
- Imagine vertices are oil depots, edges are pipelines



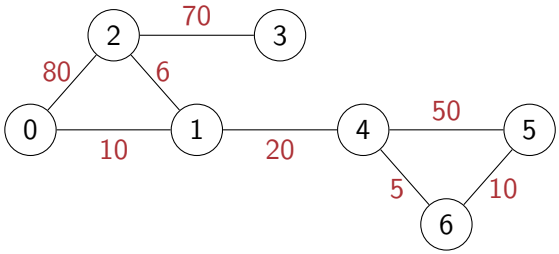
Single source shortest paths

- Compute shortest paths from 0 to all other vertices
- Imagine vertices are oil depots, edges are pipelines
- Set fire to oil depot at vertex 0



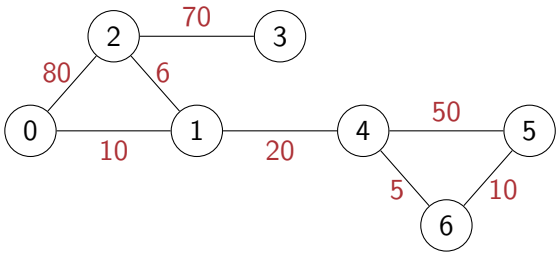
Single source shortest paths

- Compute shortest paths from 0 to all other vertices
- Imagine vertices are oil depots, edges are pipelines
- Set fire to oil depot at vertex 0
- Fire travels at uniform speed along each pipeline



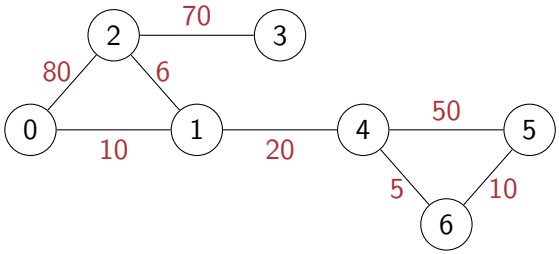
Single source shortest paths

- Compute shortest paths from 0 to all other vertices
- Imagine vertices are oil depots, edges are pipelines
- Set fire to oil depot at vertex 0
- Fire travels at uniform speed along each pipeline
- First oil depot to catch fire after 0 is nearest vertex



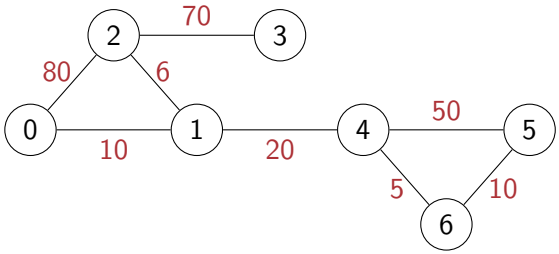
Single source shortest paths

- Compute shortest paths from 0 to all other vertices
- Imagine vertices are oil depots, edges are pipelines
- Set fire to oil depot at vertex 0
- Fire travels at uniform speed along each pipeline
- First oil depot to catch fire after 0 is nearest vertex
- Next oil depot is second nearest vertex



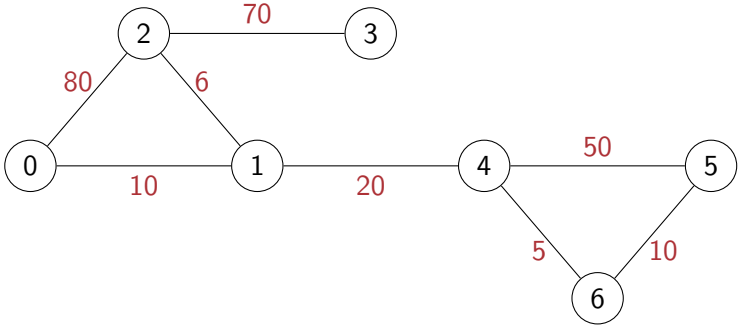
Single source shortest paths

- Compute shortest paths from 0 to all other vertices
- Imagine vertices are oil depots, edges are pipelines
- Set fire to oil depot at vertex 0
- Fire travels at uniform speed along each pipeline
- First oil depot to catch fire after 0 is nearest vertex
- Next oil depot is second nearest vertex
- ...



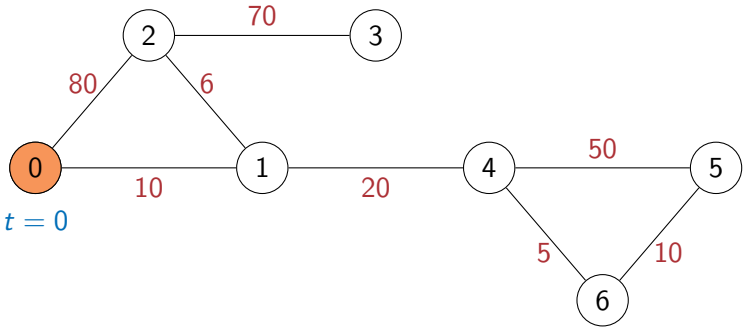
Single source shortest paths

- Set fire to oil depot at vertex 0
- Fire travels at uniform speed along each pipeline
- First oil depot to catch fire after 0 is nearest vertex
- Next oil depot is second nearest vertex
- ...



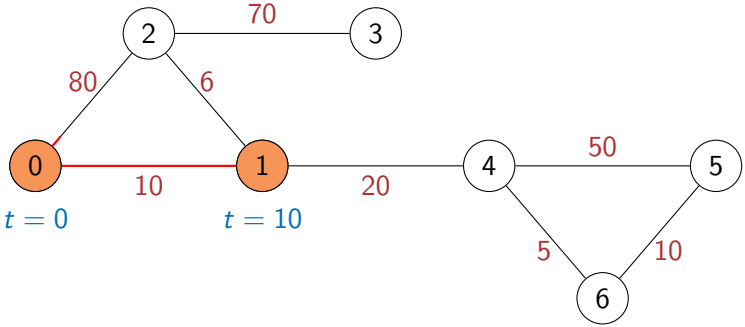
Single source shortest paths

- Set fire to oil depot at vertex 0
- Fire travels at uniform speed along each pipeline
- First oil depot to catch fire after 0 is nearest vertex
- Next oil depot is second nearest vertex
- ...



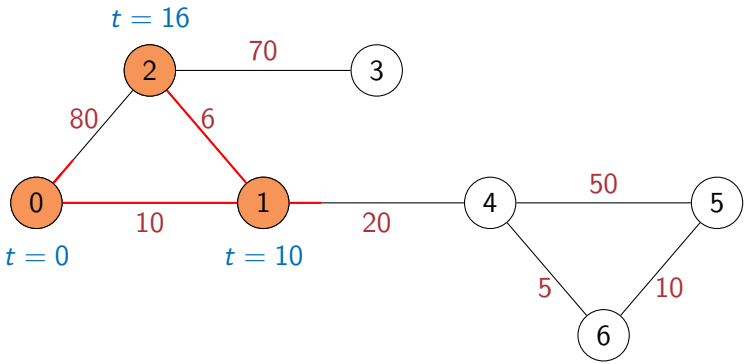
Single source shortest paths

- Set fire to oil depot at vertex 0
- Fire travels at uniform speed along each pipeline
- First oil depot to catch fire after 0 is nearest vertex
- Next oil depot is second nearest vertex
- ...



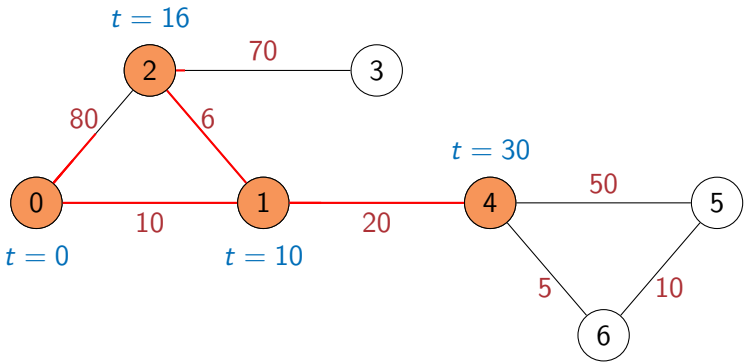
Single source shortest paths

- Set fire to oil depot at vertex 0
- Fire travels at uniform speed along each pipeline
- First oil depot to catch fire after 0 is nearest vertex
- Next oil depot is second nearest vertex
- ...



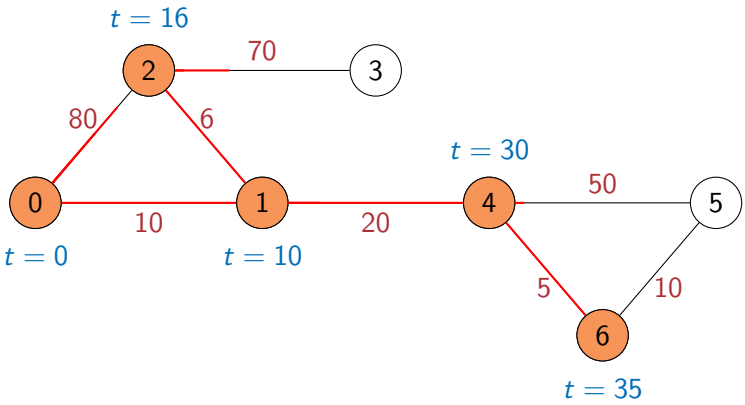
Single source shortest paths

- Set fire to oil depot at vertex 0
- Fire travels at uniform speed along each pipeline
- First oil depot to catch fire after 0 is nearest vertex
- Next oil depot is second nearest vertex
- ...



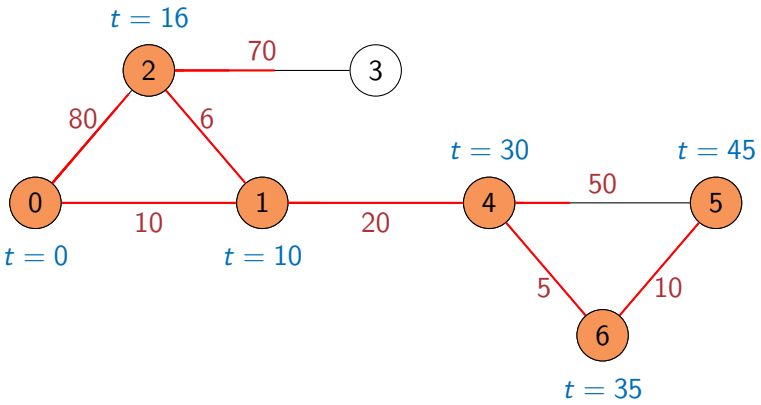
Single source shortest paths

- Set fire to oil depot at vertex 0
- Fire travels at uniform speed along each pipeline
- First oil depot to catch fire after 0 is nearest vertex
- Next oil depot is second nearest vertex
- ...



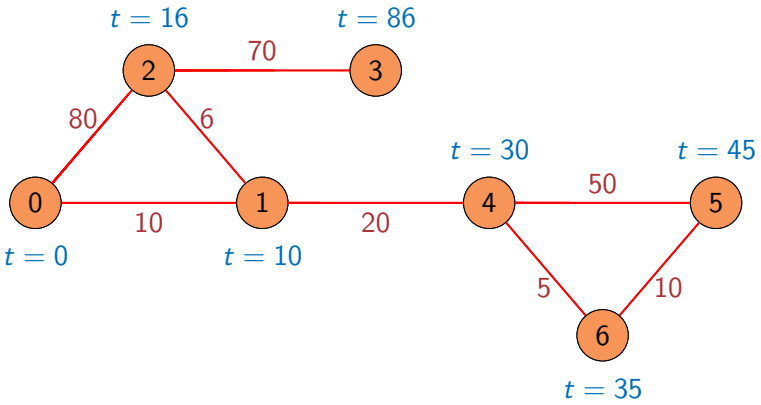
Single source shortest paths

- Set fire to oil depot at vertex 0
- Fire travels at uniform speed along each pipeline
- First oil depot to catch fire after 0 is nearest vertex
- Next oil depot is second nearest vertex
- ...



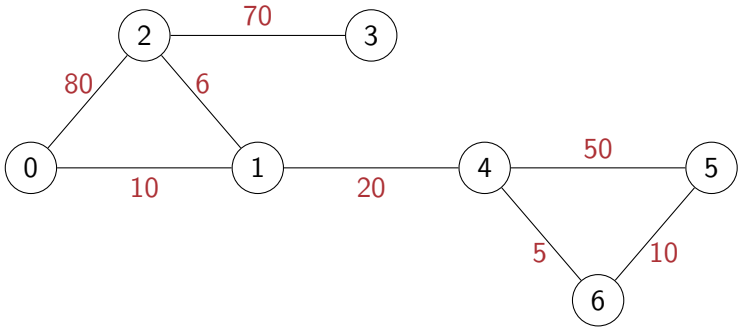
Single source shortest paths

- Set fire to oil depot at vertex 0
- Fire travels at uniform speed along each pipeline
- First oil depot to catch fire after 0 is nearest vertex
- Next oil depot is second nearest vertex
- ...



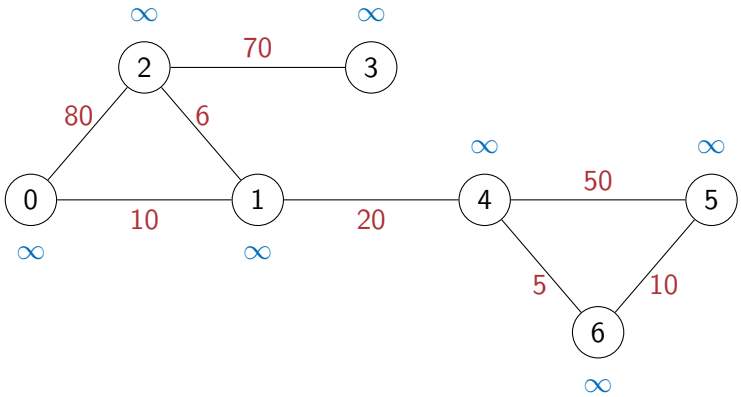
Single source shortest paths

- Compute expected burn time for each vertex
- Each time a new vertex burns, update the expected burn times of its neighbours



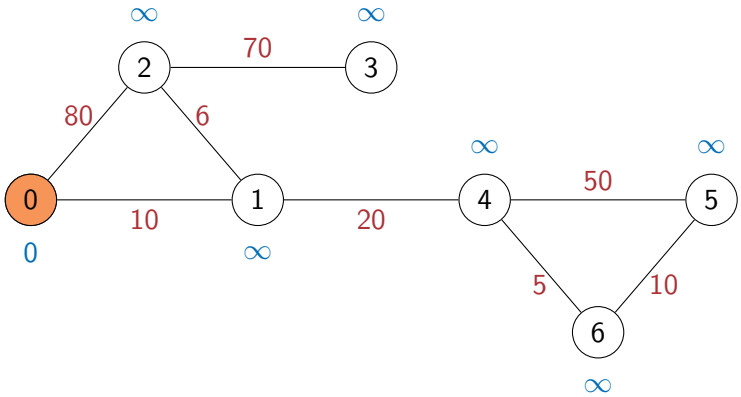
Single source shortest paths

- Compute expected burn time for each vertex
- Each time a new vertex burns, update the expected burn times of its neighbours



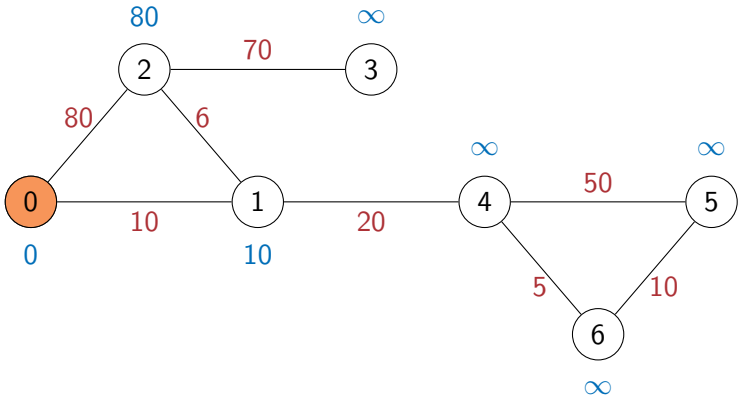
Single source shortest paths

- Compute expected burn time for each vertex
- Each time a new vertex burns, update the expected burn times of its neighbours



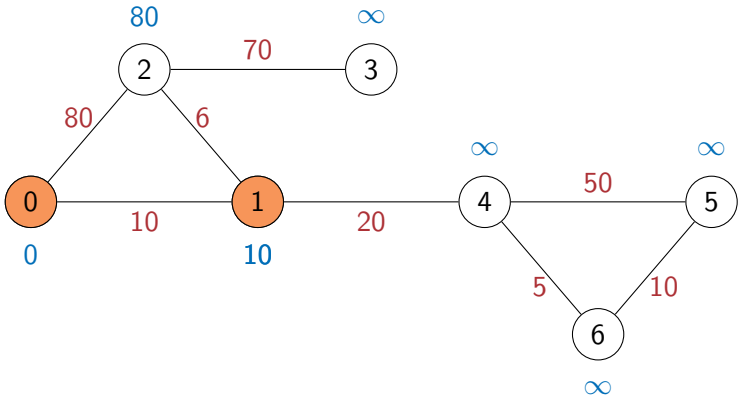
Single source shortest paths

- Compute expected burn time for each vertex
- Each time a new vertex burns, update the expected burn times of its neighbours



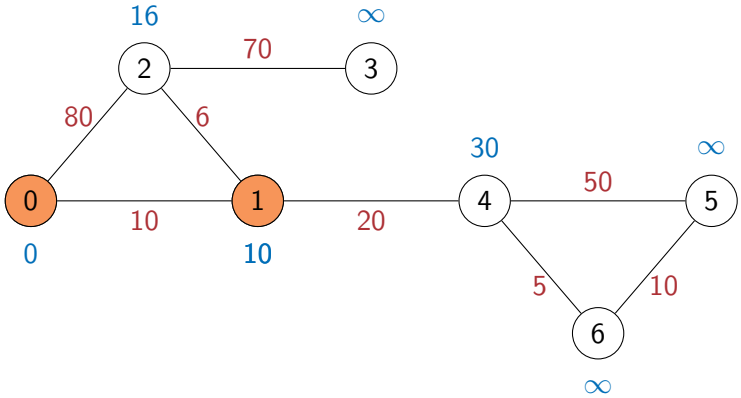
Single source shortest paths

- Compute expected burn time for each vertex
- Each time a new vertex burns, update the expected burn times of its neighbours



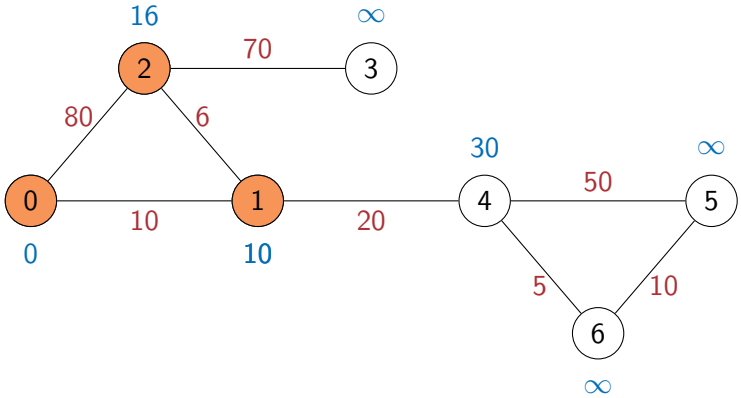
Single source shortest paths

- Compute expected burn time for each vertex
- Each time a new vertex burns, update the expected burn times of its neighbours



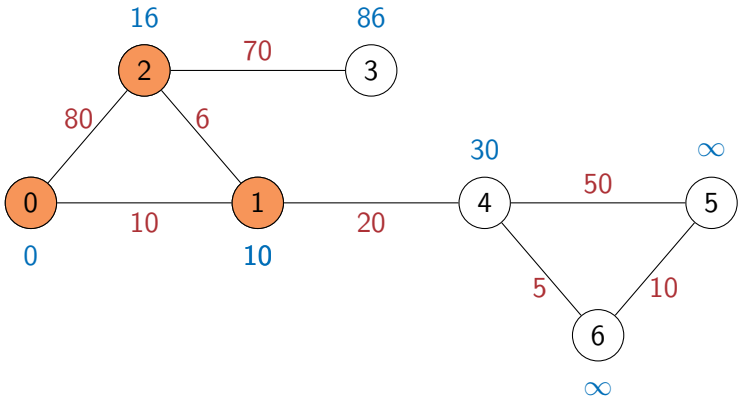
Single source shortest paths

- Compute expected burn time for each vertex
- Each time a new vertex burns, update the expected burn times of its neighbours



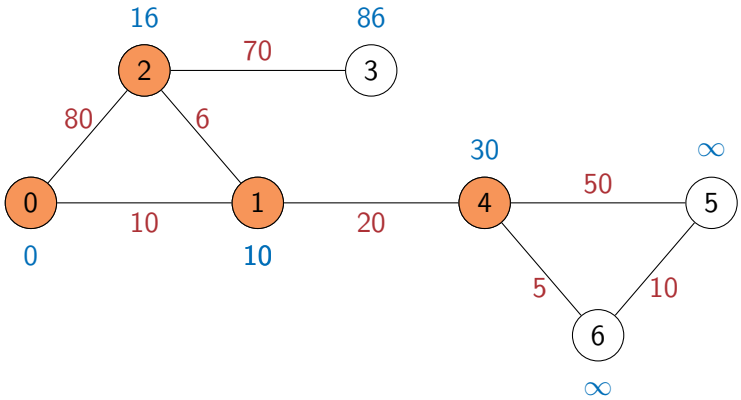
Single source shortest paths

- Compute expected burn time for each vertex
- Each time a new vertex burns, update the expected burn times of its neighbours



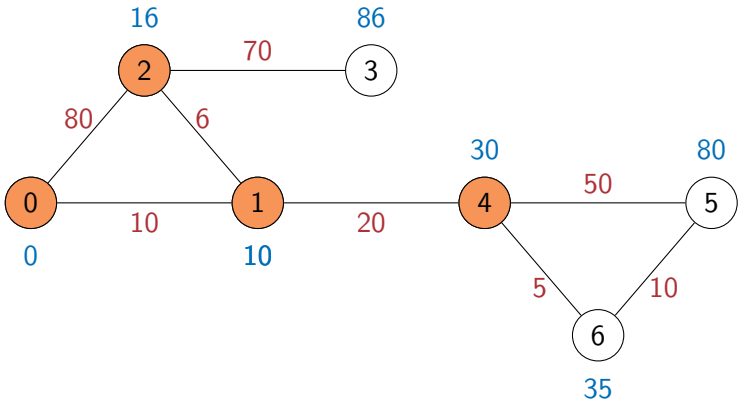
Single source shortest paths

- Compute expected burn time for each vertex
- Each time a new vertex burns, update the expected burn times of its neighbours



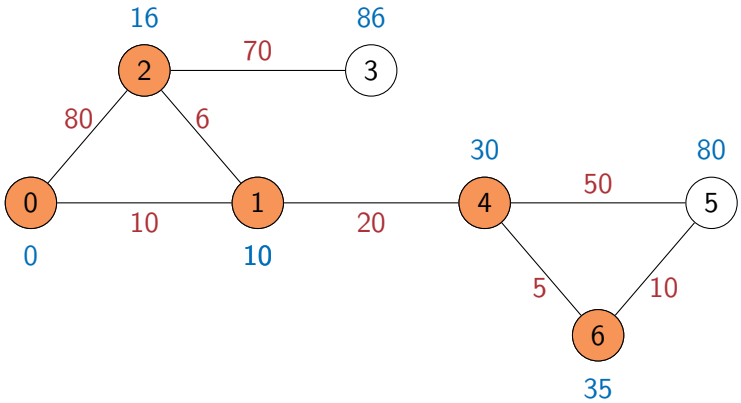
Single source shortest paths

- Compute expected burn time for each vertex
- Each time a new vertex burns, update the expected burn times of its neighbours



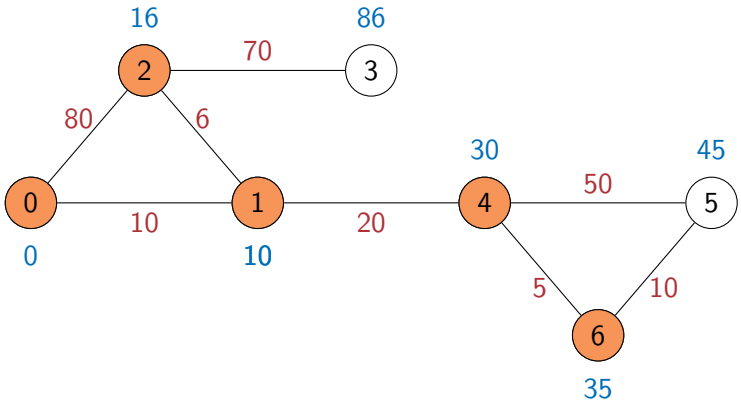
Single source shortest paths

- Compute expected burn time for each vertex
- Each time a new vertex burns, update the expected burn times of its neighbours



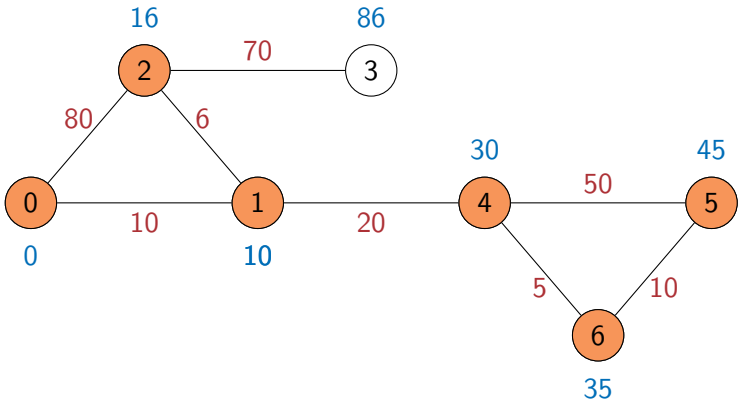
Single source shortest paths

- Compute expected burn time for each vertex
- Each time a new vertex burns, update the expected burn times of its neighbours



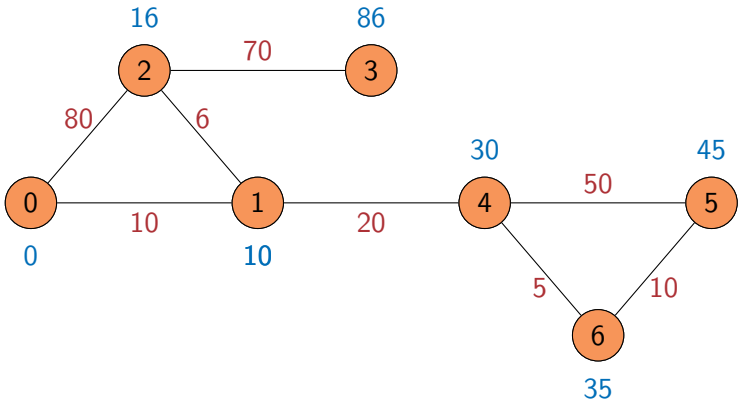
Single source shortest paths

- Compute expected burn time for each vertex
- Each time a new vertex burns, update the expected burn times of its neighbours



Single source shortest paths

- Compute expected burn time for each vertex
- Each time a new vertex burns, update the expected burn times of its neighbours
- Algorithm due to Edsger W Dikjstra



Dijkstra's algorithm: Proof of correctness

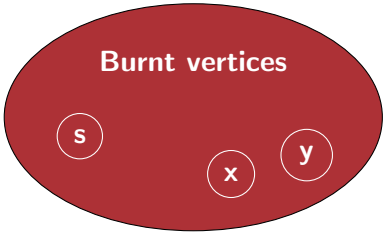
- Each new shortest path we discover extends an earlier one

Dijkstra's algorithm: Proof of correctness

- Each new shortest path we discover extends an earlier one
- By induction, assume we have found shortest paths to all vertices already burnt

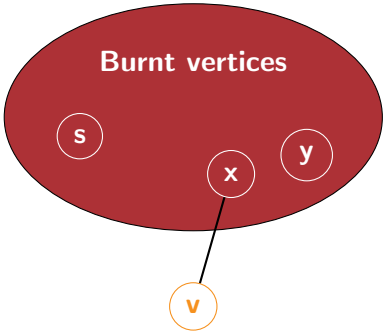
Dijkstra's algorithm: Proof of correctness

- Each new shortest path we discover extends an earlier one
- By induction, assume we have found shortest paths to all vertices already burnt



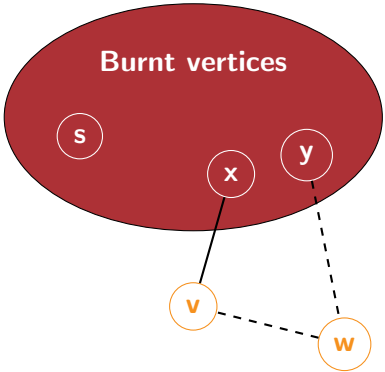
Dijkstra's algorithm: Proof of correctness

- Each new shortest path we discover extends an earlier one
- By induction, assume we have found shortest paths to all vertices already burnt
- Next vertex to burn is **v**, via **x**



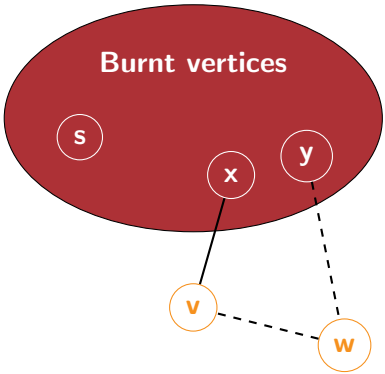
Dijkstra's algorithm: Proof of correctness

- Each new shortest path we discover extends an earlier one
- By induction, assume we have found shortest paths to all vertices already burnt
- Next vertex to burn is **v**, via **x**
- Cannot find a shorter path later from **y** to **v** via **w**
  - Burn time of **w** $\geq$ burn time of **v**
  - Edge from **w** to **v** has weight ≥ 0



Dijkstra's algorithm: Proof of correctness

- Each new shortest path we discover extends an earlier one
- By induction, assume we have found shortest paths to all vertices already burnt
- Next vertex to burn is **v**, via **x**
- Cannot find a shorter path later from **y** to **v** via **w**
  - Burn time of **w** $\geq$ burn time of **v**
  - Edge from **w** to **v** has weight ≥ 0
- This argument breaks down if edge (**w**,**v**) can have negative weight
  - Can't use Dijkstra's algorithm with negative edge weights



Summary

- Dijkstra's algorithm computes single source shortest paths
- Use fire analogy
  - Keep track of expected burn times for each vertex
  - Update burn times of neighbours each time a vertex burns
- Correctness requires edge weights to be non-negative