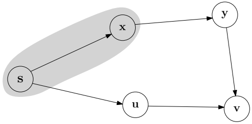
Single Source Shortest Path

- Weighted Graphs and Shortest Paths
- Dijkstra Algorithm
- Proof of Correctness
- Runtime
  - Basic Implementation
  - Vertex-Centric Implementation
  - Heap Based Implementation

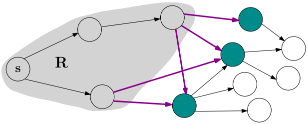
Imdad ullah Khan

Dijkstra Algorithm



$d[1 \ldots n] \leftarrow [\infty \ldots \infty]$
$d[s] \leftarrow 0 \quad R \leftarrow \{s\}$
while $R \neq V$ **do**
  Select $v \in \overline{R}$
  $R \leftarrow R \cup \{v\}$
  $d[v] \leftarrow d(s, v)$

- Which vertex from $\overline{R}$ to add to R?
- The vertex $v \in \overline{R}$ that is closest to s
- Such a v must be at the **"frontier"** of $\overline{R}$



Restrict search to **"single edge extensions"** of paths to $u \in R$

- Dijkstra assigns a score to each crossing edge

$$score(u, v) \ = \ d[u] + w(uv) \quad \text{for} \quad (u, v) \in E, u \in R, v \notin R$$

- Add a frontier vertex adjacent through minimum scoring edge

Dijkstra Algorithm

Algorithm Dijkstra's Algorithm for distances from s to all vertices

$d[1 \dots n] \leftarrow [\infty \dots \infty]$

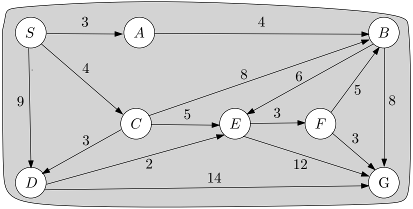
$d[s] \leftarrow 0 \qquad R \leftarrow \{s\}$

while $R \neq V$ **do**

  Select $e = (u, v)$, $u \in R, v \notin R$, with minimum $d[u] + w(uv)$

  $R \leftarrow R \cup \{v\}$

  $d[v] \leftarrow d[u] + w(uv)$



	d
S	0
A	3
B	7
C	4
D	9
E	9
F	12
G	15

Dijkstra Algorithm with paths

Record predecessor relationships (sources of used edges)

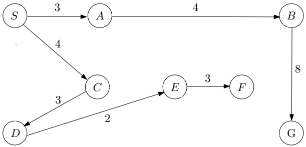
Implicitly builds a tree (shortest path tree)

Algorithm Dijkstra's Algorithm for Shortest Paths from s to all vertices

---

$d[1 \dots n] \leftarrow [\infty \dots \infty]$
$prev[1 \dots n] \leftarrow [null \;\dots\; null]$
$d[s] \leftarrow 0 \qquad R \leftarrow \{s\}$
while $R \neq V$ **do**
  Select $e = (u, v)$, $u \in R, v \notin R$, with minimum $d[u] + w(uv)$
  $R \leftarrow R \cup \{v\}$
  $d[v] \leftarrow d[u] + w(uv)$
  $prev[v] \leftarrow u$ ▷ predecessor is the vertex whose path is single-edge extended

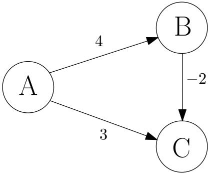
---



	d	$prev$
S	0	Nil
A	3	S
B	7	A
C	4	S
D	9	C
E	9	D
F	12	E
G	15	B

Dijkstra Algorithm: Proof of Correctness

One example (or millions) doesn't prove correctness



With source A, algorithm greedily selects C then B

Selected paths are $A - C$ and $A - B$

The shortest path from A to C is $A - B - C$

Where does the algorithm use the non-negative weights

Need a proof!

Dijkstra Algorithm: Proof of Correctness

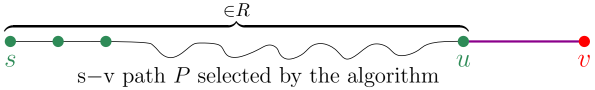
In every iteration $i = |R|, \quad \forall\, u \in R, \quad d[u] \;=\; d(s,u)$

- Proof by induction on i
- Base case: $i = 0$
- $R = \{s\}$
- $d[s] = 0$
- $d[s] = 0 = d(s,s)$, because all weights are ≥ 0

Dijkstra Algorithm: Proof of Correctness

In every iteration $i = |R|, \quad \forall\, u \in R, \quad d[u] \;=\; d(s,u)$

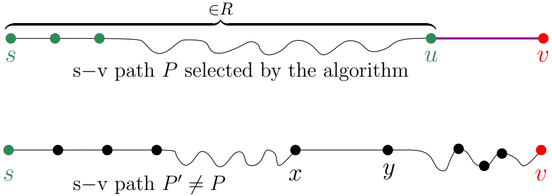
- Assume the statement is true from $i \leq k-1$
- Suppose v is added to R in the kth iteration using edge (u,v)
- Let the path made for v be $\quad P \;=\; s,\ldots,u,v$
- We show $d[v] = w(P) \leq w(P')$ for any other $s-v$ path P'



Dijkstra Algorithm: Proof of Correctness

In every iteration $i = |R|$, $\forall\ u \in R, d[u] = d(s,u)$

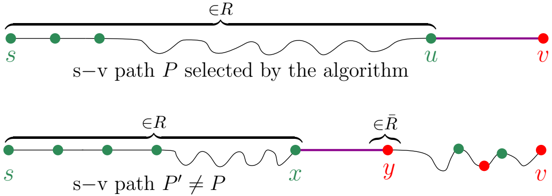
- Assume the statement is true from $i \leq k-1$
- Suppose v is added to R in the kth iteration using edge (u, v)
- Let the path made for v be $P = s, \ldots, u, v$
- We show $d[v] = w(P) \leq w(P')$ for any other $s - v$ path P'



Dijkstra Algorithm: Proof of Correctness

In every iteration $i = |R|$, $\forall\ u \in R, d[u] = d(s, u)$

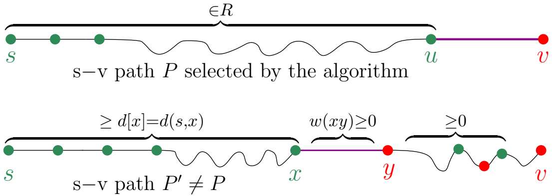
- Assume the statement is true from $i \leq k - 1$
- Suppose v is added to R in the kth iteration using edge (u, v)
- Let the path made for v be $P = s, \ldots, u, v$
- We show $d[v] = w(P) \leq w(P')$ for any other $s - v$ path P'



Dijkstra Algorithm: Proof of Correctness

In every iteration $i = |R|$, $\forall\ u \in R, d[u] = d(s,u)$

- Assume the statement is true from $i \leq k-1$
- Suppose v is added to R in the kth iteration using edge (u,v)
- Let the path made for v be $P = s, \ldots, u, v$
- We show $d[v] = w(P) \leq w(P')$ for any other $s-v$ path P'



$$w(P') \ \geq\ d[x] + w(xy) \ \geq\ d[u] + w(uv) \ =\ w(P)$$