Definitions: Paths in graphs
Consider a digraph $G = (V, E)$ with edge-weight function $w : E \rightarrow \mathbb{R}$.
The weight of path $p = v_1 \rightarrow v_2 \rightarrow \cdots \rightarrow v_k$ is defined to be:
$$w(p) = \sum_{i=1}^{k-1} w(v_i, v_{i+1}).$$

Example:

```
\begin{tikzpicture}
    \node[shape=circle,draw=black] (u) at (0,0) {$v_1$};
    \node[shape=circle,draw=black] (v) at (1,0) {$v_2$};
    \node[shape=circle,draw=black] (w) at (2,0) {$v_3$};
    \node[shape=circle,draw=black] (x) at (3,0) {$v_4$};
    \node[shape=circle,draw=black] (y) at (4,0) {$v_5$};
    \draw[->, thick] (u) to node [above] {$4$} (v);
    \draw[->, thick] (v) to node [above] {$-2$} (w);
    \draw[->, thick] (w) to node [above] {$-5$} (x);
    \draw[->, thick] (x) to node [above] {$1$} (y);
    \end{tikzpicture}
```

$w(p) = -2$

Definitions: Shortest paths
A shortest path from $u$ to $v$ is a path of minimum weight from $u$ to $v$.
The shortest-path weight from $u$ to $v$ is defined as
$$\delta(u, v) = \min \{ w(p) : p \text{ is a path from } u \text{ to } v \}.$$ Note: $\delta(u, v) = \infty$ if no path from $u$ to $v$ exists.

Shortest path properties: 1. Well-definedness
If a graph $G$ contains a negative-weight cycle, then some shortest paths do not exist.

Example:

```
\begin{tikzpicture}
    \node[shape=circle,draw=black] (u) at (0,0) {$u$};
    \node[shape=circle,draw=black] (v) at (1,0) {$v$};
    \node[shape=circle,draw=black] (w) at (2,0) {$w$};
    \draw[->, thick] (u) to node [above] {$<0$} (v);
    \draw[->, thick] (v) to node [above] {$<0$} (w);
    \draw[->, thick] (w) to node [above] {$<0$} (u);
    \end{tikzpicture}
```

$<0$: Shortest path does not exist $>0$: Cycle can be removed $=0$: Choose no cycle, same cost, fewer edges, wlog

⇒ Only need to consider paths of $|V|-1$ edges!

Shortest path properties: 2. Triangle inequality
Theorem. For all $s, u, v \in V$, we have
$$\delta(s, v) \leq \delta(s, u) + w(u, v).$$

Proof. If optimal path $p$ uses $(u,v)$, then $\delta(s, v) = \delta(s, u) + w(u, v)$.
Else: proof by contradiction. If $w(p\rightarrow v) > \delta(s, u) + w(u, v)$, then can make shorter path $p'$ by going through $u$.

Shortest path properties: 3. Optimal Substructure
Theorem. Any subpath $i\rightarrow j$ of a shortest path $s\rightarrow k$ is a shortest path.

Proof. Cut and paste:

```
\begin{tikzpicture}
    \node[shape=circle,draw=black] (s) at (0,0) {$s$};
    \node[shape=circle,draw=black] (i) at (1,0) {$i$};
    \node[shape=circle,draw=black] (j) at (2,0) {$j$};
    \node[shape=circle,draw=black] (k) at (3,0) {$k$};
    \draw[->, thick] (s) to node [above] {$\delta(s, v)$} (v);
    \draw[->, thick] (v) to node [above] {$\delta(v, u)$} (u);
    \draw[->, thick] (u) to node [above] {$w(u, v)$} (v);
    \draw[->, thick] (s) to node [above] {$\delta(s, u)$} (i);
    \draw[->, thick] (i) to node [above] {$\delta(i, j)$} (j);
    \draw[->, thick] (j) to node [above] {$\delta(j, k)$} (k);
    \end{tikzpicture}
```

If not, make shorter path by $s\rightarrow i\rightarrow j\rightarrow k$
Shortest paths: Problem settings

<table>
<thead>
<tr>
<th>Setting</th>
<th>Weights</th>
<th>Greedy/DP</th>
<th>Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single source</td>
<td>=1</td>
<td>Greedy</td>
<td>BFS</td>
</tr>
<tr>
<td>Single source</td>
<td>≥0</td>
<td>Greedy</td>
<td>Dijkstra</td>
</tr>
<tr>
<td>Single source</td>
<td>General</td>
<td>Dyn. Prog.</td>
<td>Bellman-Ford</td>
</tr>
<tr>
<td>All pairs</td>
<td>General</td>
<td>Dyn. Prog.</td>
<td>MX-mult, Floyd-Warshall, Johnson</td>
</tr>
</tbody>
</table>

Greedy choice property
(only if all weights are \( ≥ 0 \))

- Maintain set \( S \) of nodes whose shortest path distances are minimal. At each step, include the vertex \( v \) whose current distance estimate from \( S \) is minimum, through edge \((u,v)\).
- Then edge \((u,v)\) must be part of a shortest path, since any other indirect path to \( v \) must go through a longer-distance intermediate (because path costs never decrease, all \( w \geq 0 \)).

Note: Full proof also shows that our estimates are correct.

Single-source shortest paths
(non-negative edge weights)

**Problem.** Assume that \( w(u,v) \geq 0 \) for all \((u,v) \in E \). (Hence, all shortest-path weights must exist.) From a given source vertex \( s \in V \), find the shortest-path weights \( \delta(s,v) \) for all \( v \in V \).

**Algorithm:** Greedy choice.
1. Maintain a set \( S \) of vertices whose shortest-path distances from \( s \) are known.
2. At each step, add to \( S \) the vertex \( v \in V - S \) whose distance estimate from \( s \) is minimum.
3. Update the distance estimates of vertices adjacent to \( v \).

Example of Dijkstra's algorithm

Graph with nonnegative edge weights:

Dijkstra's algorithm

1. \( d[s] \leftarrow 0 \)
2. For each \( v \in V - \{s\} \) do \( d[v] \leftarrow \infty \)
3. \( S \leftarrow \emptyset \)
4. \( Q \leftarrow V \quad \text{(} Q \text{ is a priority queue maintaining } V - S, \text{ keyed on } d[v]\text{)} \)
5. While \( Q \neq \emptyset \) do
   - \( u \leftarrow \text{EXTRACT-MIN}(Q) \)
   - \( S \leftarrow S \cup \{u\} \)
   - For each \( v \in \text{Adj}[u] \) do
     - If \( d[v] > d[u] + w(u,v) \) then \( d[v] \leftarrow d[u] + w(u,v) \)

Implicit DECREASE-KEY
Example of Dijkstra’s algorithm

Initialize:

```
Q: A B C D E
0 ∞ ∞ ∞ ∞

S: {}
```

```
Example of Dijkstra’s algorithm

```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
**Example of Dijkstra’s algorithm**

Relax all edges leaving $E$:

- $Q$: $A$ $B$ $C$ $D$ $E$
  - $S$: $\{A, C, E\}$

“$B$” $\leftarrow$ **Extract-Min**(Q):

- $Q$: $A$ $B$ $C$ $D$ $E$
  - $S$: $\{A, C, E\}$

Relax all edges leaving $B$:

- $Q$: $A$ $B$ $C$ $D$ $E$
  - $S$: $\{A, C, E\}$

“$D$” $\leftarrow$ **Extract-Min**(Q):

- $Q$: $A$ $B$ $C$ $D$ $E$
  - $S$: $\{A, C, E, B\}$

### Correctness proof: Part 1: Upper Bound (Th. 24.11)

**Lemma.** Initializing $d[s] \leftarrow 0$ and $d[v] \leftarrow \infty$ for all $v \in V - \{s\}$ establishes $d[v] \geq \delta(s, v)$ for all $v \in V$, and this invariant is maintained over any sequence of relaxation steps. (Katherine’s Lemma)

**Proof.** Prove invariant $d[v] \geq \delta(s, v)$ for all vertices by induction over number of relaxation steps.

- Base case upon $d[s]=0$ and $d[v]=\infty$ otherwise.
- Inductive step upon relaxation of edge $(u,v)$; by inductive hypothesis $d[x] \geq \delta(s,x)$ for all $x \in V$.

The only $d$ value changing is $d[v]$, and it becomes:

$$d[v] = d[u] + w(u,v) \geq \delta(s,u) + w(u,v)$$

by inductive hypothesis

$$\geq \delta(s,v) \text{ by the triangle inequality}.$$
**Correctness proof: Part 2: Convergence (Th. 24.16)**

**Lemma.** Let \( u \) be \( v \)'s predecessor on a shortest path from \( s \) to \( v \). Then, if \( d[u] = \delta(s, u) \) and edge \((u, v)\) is relaxed, we have \( d[v] = \delta(s, v) \) after the relaxation.

**Proof.** Observe that \( \delta(s, v) = \delta(s, u) + w(u, v) \). Suppose that \( d[v] > \delta(s, v) \) before the relaxation. (Otherwise, we’re done.) Then, the test \( d[v] > d[u] + w(u, v) \) succeeds, because \( d[v] > \delta(s, v) = \delta(s, u) + w(u, v) \), and the algorithm sets \( d[v] = d[u] + w(u, v) = \delta(s, v) \).

**Correctness proof (Th. 24.6): Part 3: Correctness of Dijkstra**

**Theorem.** Dijkstra’s algorithm terminates with \( d[v] = \delta(s, v) \) for all \( v \in V \).

**Proof.** It suffices to show that \( d[v] = \delta(s, v) \) for every \( v \in V \) when \( v \) is added to \( S \). Suppose \( u \) is the first vertex added to \( S \) for which \( d[u] > \delta(s, u) \). Let \( y \) be the first vertex in \( V - S \) along a shortest path from \( s \) to \( u \), and let \( x \) be its predecessor:

\[ S, \text{ just before adding } u. \]

---

**Dijkstra's Runtime analysis**

**Analysis of Dijkstra**

(continued)

\[ \text{Time} = \Theta(V^2)T_{\text{EXTRACT-MIN}} + \Theta(E)T_{\text{DECREASE-KEY}} \]

<table>
<thead>
<tr>
<th>Array</th>
<th>( O(V) )</th>
<th>( O(1) )</th>
<th>( O(V^2) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary heap</td>
<td>( O(\lg V) )</td>
<td>( O(\lg V) )</td>
<td>( O(E \lg V) )</td>
</tr>
<tr>
<td>Fibonacci heap</td>
<td>( O(\lg V) )</td>
<td>( O(1) )</td>
<td>( O(E + V \lg V) )</td>
</tr>
</tbody>
</table>

**Note:** Same formula as in the analysis of Prim’s minimum spanning tree algorithm.
Unweighted graphs: BFS

(all edges have positive, unit weights)

$w(u,v) = 1$ for all $(u,v)$

Unweighted graphs

Suppose that $w(u,v) = 1$ for all $(u,v) \in E$. Can Dijkstra’s algorithm be improved?

• Use a simple FIFO queue instead of a priority queue.

**Breadth-first search**

```
while Q ≠ ∅
    do u ← DEQUEUE(Q)
    for each v ∈ Adj[u]
        do if $d[v] = \infty$
            then $d[v] ← d[u] + 1$
        ENQUEUE(Q, v)
```

**Analysis:** Time = $O(V + E)$.

Example of breadth-first search

```
Q:  a  b  d  c  e
```

```
0 1 1 0 1
1 1 1 1 1
Q:  a b d
```

```
0 a d f h
b g i
c e
```

```
0 0
Q:  a
```

```
0
Q:  a
```

```
0 a d f h
b g i
c e
```

```
0 a d f h
b g i
c e
```

```
0 a d f h
b g i
c e
```

```
0 a d f h
b g i
c e
```

```
0 a d f h
b g i
c e
```

```
0 a d f h
b g i
c e
```

```
0 a d f h
b g i
c e
```

© Leiserson, Demaine, Kellis
Example of breadth-first search

Q: a b d c e

Q: a b d c e

Q: a b d c e g i

Q: a b d c e g i f

Q: a b d c e g i f h
Example of breadth-first search

Q: a b d e g i f h

Correctness of BFS

while Q ≠ ∅
do u ← DEQUEUE(Q)
   for each v ∈ Adj[u]
do if d[v] = ∞
   then d[v] ← d[u] + 1
   ENQUEUE(Q, v)

Key idea:
The FIFO Q in breadth-first search mimics the priority queue Q in Dijkstra.


Shortest paths: Summary

<table>
<thead>
<tr>
<th>Setting</th>
<th>Weights</th>
<th>Gr./DP</th>
<th>Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single source</td>
<td>=1</td>
<td>Greedy</td>
<td>BFS: O(V+E)</td>
</tr>
<tr>
<td>Single source</td>
<td>≥0</td>
<td>Greedy</td>
<td>Dijkstra: O(E+VlgV)</td>
</tr>
<tr>
<td>Single source</td>
<td></td>
<td></td>
<td>Bellman-Ford</td>
</tr>
<tr>
<td>All pairs</td>
<td></td>
<td></td>
<td>MX-mult, Floyd-Warshall, Johnson</td>
</tr>
</tbody>
</table>