Dijkstra's algorithm

- **Dijkstra's algorithm**: finds shortest (minimum weight) path between a particular pair of vertices in a *weighted* directed graph with nonnegative edge weights
  - solves the "one vertex, shortest path" problem
  - basic algorithm concept: create a table of information about the currently known best way to reach each vertex (distance, previous vertex) and improve it until it reaches the best solution

- in a graph where:
  - vertices represent cities,
  - edge weights represent driving distances between pairs of cities connected by a direct road,
  Dijkstra's algorithm can be used to find the shortest route between one city and any other
Dijkstra pseudocode

\[\text{Dijkstra}(v_1, v_2):\]
\[
\text{for each vertex } v:\quad // \text{Initialization}
\]
\[
\quad v's\ distance := \text{infinity}.\quad v's\ previous := \text{none}.\quad v_1's\ distance := 0.\quad \text{List := \{all vertices\}.}
\]
\[
\text{while List is not empty:}\quad
\]
\[
\quad v := \text{remove List vertex with minimum distance.}\quad \text{mark } v \text{ as known.}\quad
\]
\[
\quad \text{for each unknown neighbor } n \text{ of } v:\quad
\quad \quad \text{dist := } v's\ distance + \text{edge } (v, n)'s\ weight.\quad
\]
\[
\quad \quad \text{if dist is smaller than } n's\ distance:\quad
\quad \quad \quad n's\ distance := \text{dist.}\quad n's\ previous := v.\quad
\]
\[
\text{reconstruct path from } v_2 \text{ back to } v_1,\quad \text{following previous pointers.}\]
Example: Initialization

Distance(source) = 0

Distance (all vertices but source) = $\infty$

Pick vertex in List with minimum distance.
Example: Update neighbors' distance

Distance(B) = 2
Distance(D) = 1
Example: Remove vertex with minimum distance

Pick vertex in List with minimum distance, i.e., D
Example: Update neighbors

Distance(C) = 1 + 2 = 3
Distance(E) = 1 + 2 = 3
Distance(F) = 1 + 8 = 9
Distance(G) = 1 + 4 = 5
Example: Continued...

Pick vertex in List with minimum distance (B) and update neighbors

Note: distance(D) not updated since D is already known and distance(E) not updated since it is larger than previously computed
Example: Continued...

Pick vertex List with minimum distance (E) and update neighbors.
Example: Continued...

Pick vertex List with minimum distance (C) and update neighbors

Distance(F) = 3 + 5 = 8
Example: Continued...

Pick vertex List with minimum distance (G) and update neighbors

Distance(F) = min (8, 5+1) = 6
Example (end)

Pick vertex not in S with lowest cost (F) and update neighbors
Correctness

• Dijkstra’s algorithm is a greedy algorithm
  – make choices that currently seem the best
  – locally optimal does not always mean globally optimal

• Correct because maintains following two properties:
  – for every known vertex, recorded distance is shortest distance to that vertex from source vertex
  – for every unknown vertex $v$, its recorded distance is shortest path distance to $v$ from source vertex, considering only currently known vertices and $v$
“Cloudy” Proof: The Idea

- If the path to v is the next shortest path, the path to v' must be at least as long. Therefore, any path through v' to v cannot be shorter!