M+E+C: Shortest Path Algorithms

Keith O'Hara

01/15/2018

Shortest Path Algorithms for Directed Graphs

- Shortest path problem: find the minimum distance path from a source node to other nodes in a network.

1. **Dijkstra**: Begin at the source node and choose the unvisited vertex with minimal distance. Repeat.

   $$O(|\mathcal{X}|^2)$$

   Requires non-negative weights along edges.

   (Worst case cost can be reduced to $O(|\mathcal{A}| + |\mathcal{X}| \ln(|\mathcal{X}|))$.)

2. **Bellman-Ford**: Iterate over all nodes and adjacent vertices.

   $$O(|\mathcal{A}||\mathcal{X}|)$$

   Allows for negative weights along edges.

Dijkstra's Algorithm

Algorithm 1 Dijkstra

procedure DIJKSTRA($s, \mathcal{X}, \mathcal{A}$) ▷ Solve for $\mathbf{d}^*$
  Define the pair $(u_j, d_j) \in (\mathcal{X}, \mathbb{R})$, where $d_j := \text{dist}(u_s, u_j)$.
  $u_s \in \mathcal{X}$, $\mathcal{V} = \mathcal{X}$; $\mathbf{d}^* = \boldsymbol{\infty}$, $d_s^* = 0$ ▷ Initialization
  while $\mathcal{V} \neq \{\emptyset\}$ **do**
    Choose $u_i := \{u_j \in \mathcal{V} : d_j^* = \min_k \{d_k^*\}\}$.
    if $d_j^* = \infty$ **then**
      break;
    $\mathcal{V} = \mathcal{V} \backslash \{u_i\}$ ▷ Remove u_i
    Define the adjacent network to u_i by $N(u_i)$.
    For each $v_j \in N(u_i) \cap \mathcal{V}$, with arc weights $\{a_{i,j}\}$, calculate

$$c_j = d_i^* + a_{i,j}$$

    if $c_j < d_j^*$ **then**
      $d_j^* = c_j$ ▷ Update d_j^*
  return $\mathbf{d}^* = \{d_j^*\}_{j \in \mathcal{X}}$. ▷ Solution

Dijkstra in Words

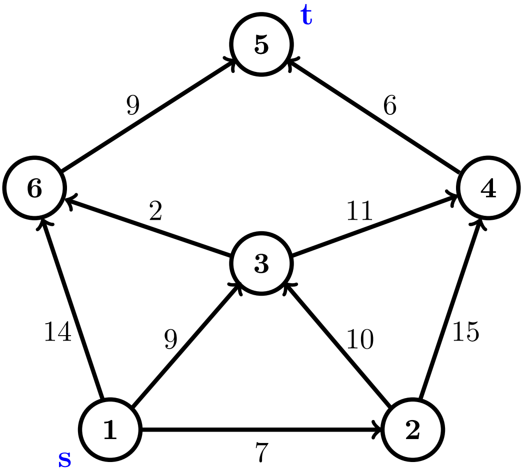
1 Begin at the source node u_s

- initialize the (minimum) distance vector: $\{d_j^*\}_{j=1}^{|\mathcal{X}|} = \infty$ and $d_s = 0$
- insert the source node into a vertex queue $\mathcal{V}$

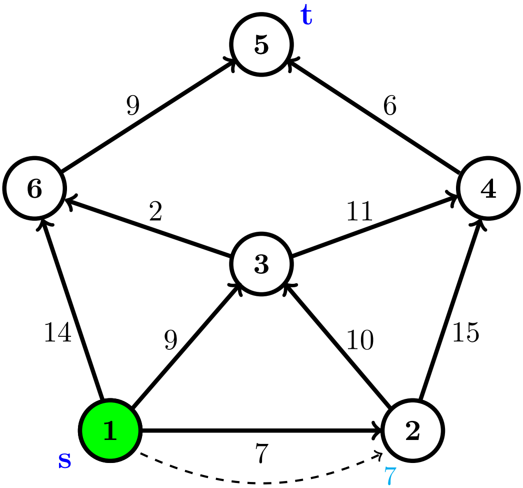
2 While the vertex queue is nonempty:

- select the node at the beginning of the vertex queue; call it u_i
- for every node adjacent to u_i, labelled $\{v_j\}_{j \in N(i)}$:
    - ★ calculate the distance to each v_j node through u_i: $\{c_j(u_i)\}_{j \in N(i)}$
    - ★ if $c_j(u_i)$ is less than the minimum distance observed thus far (d_j^*), set $d_j^* = c_j(u_i)$
- update the vertex queue from $N(i)$; remove u_i from the queue

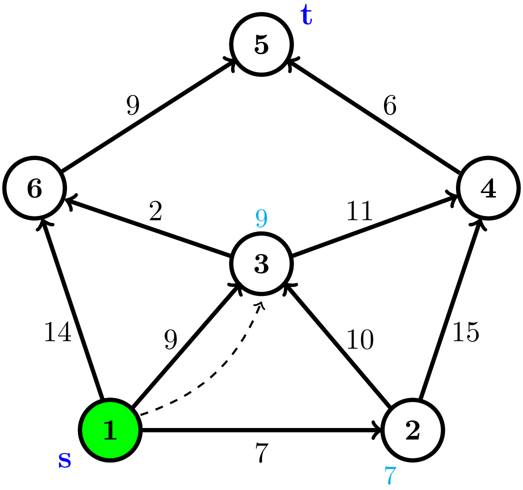
Dijkstra in Action



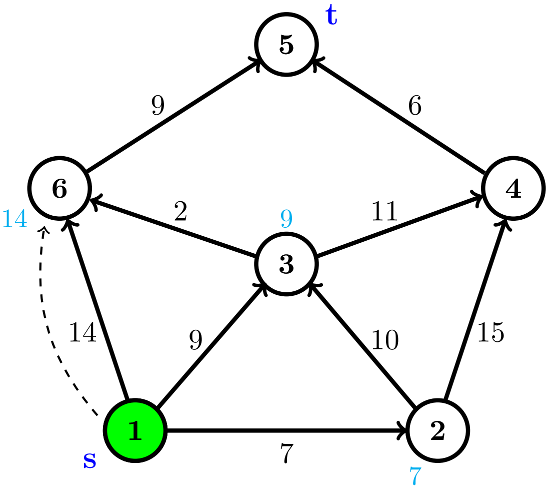
Dijkstra in Action



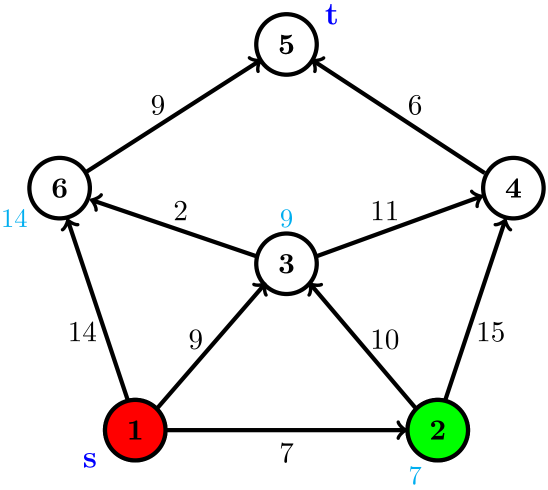
Dijkstra in Action



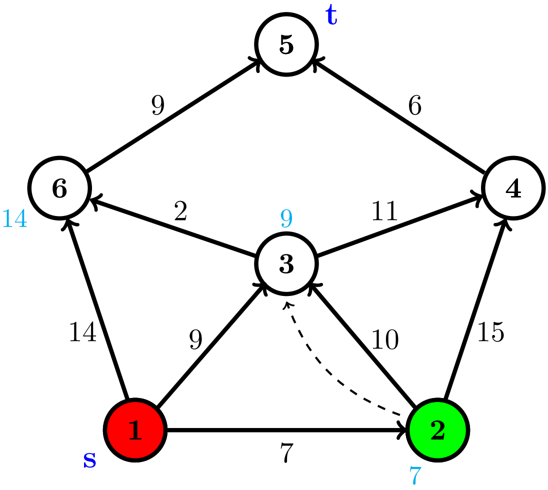
Dijkstra in Action



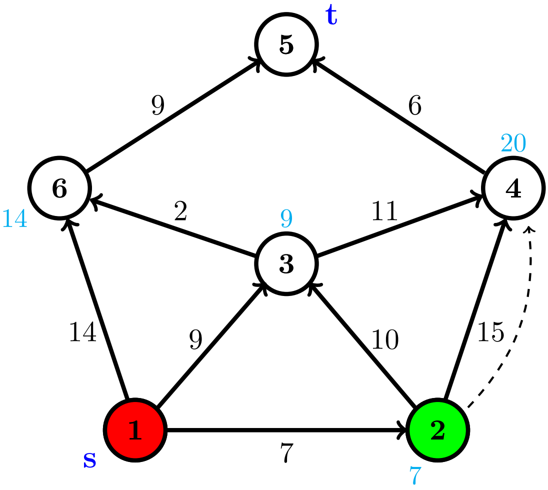
Dijkstra in Action



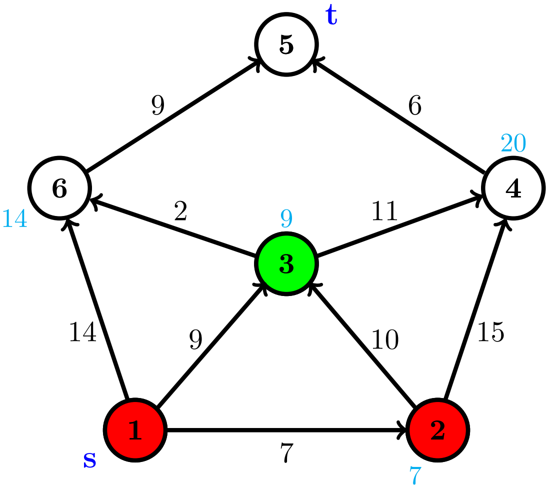
Dijkstra in Action



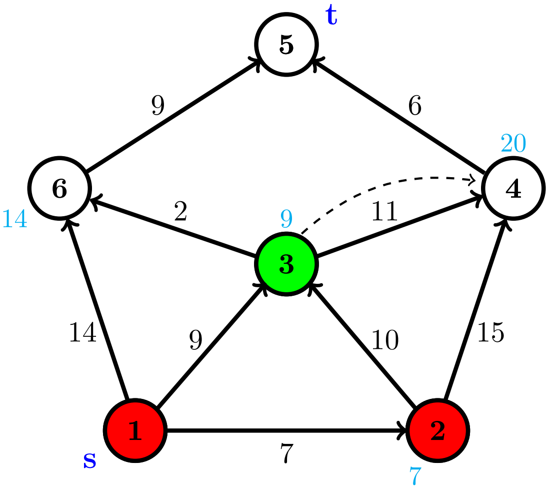
Dijkstra in Action



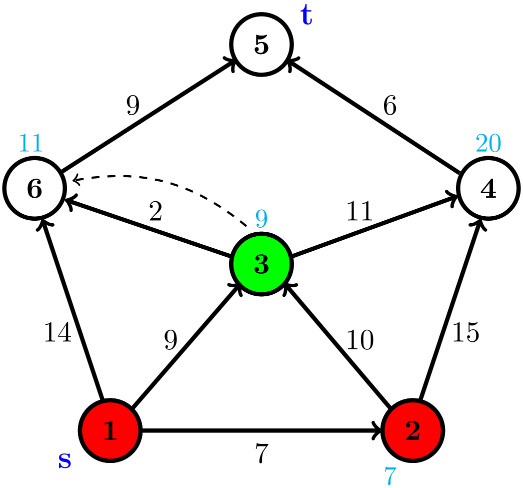
Dijkstra in Action



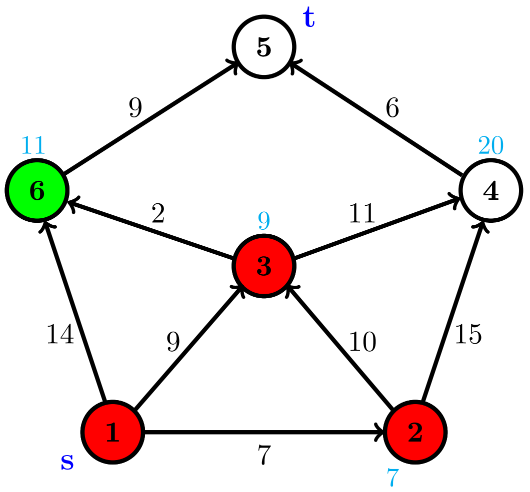
Dijkstra in Action



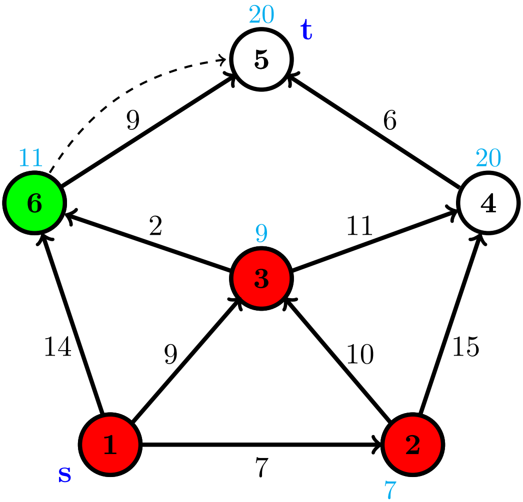
Dijkstra in Action



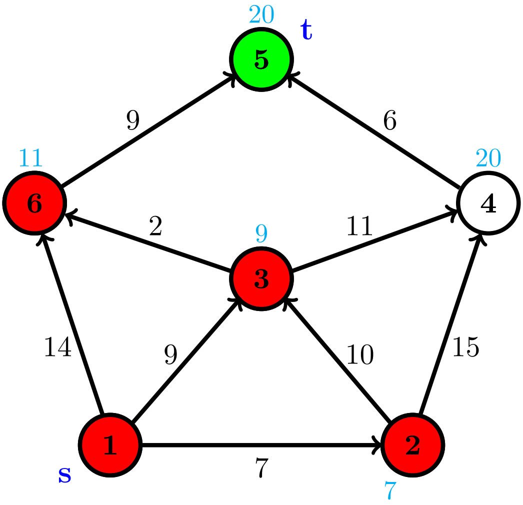
Dijkstra in Action



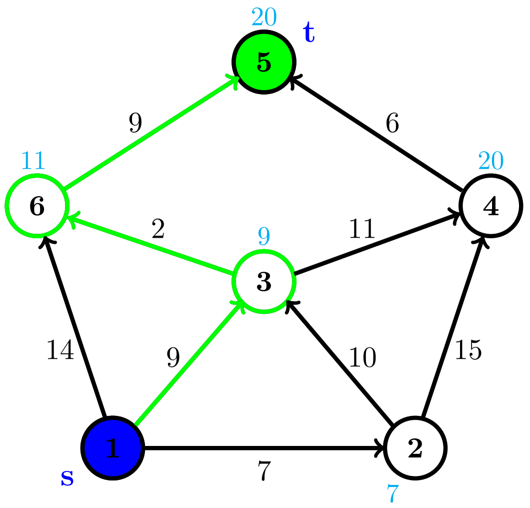
Dijkstra in Action



Dijkstra in Action



Dijkstra in Action



Dijkstra in Code

- We've written a lightweight **R** package for this: SPR. It acts as a wrapper to C++ code.

```
library(SPR)

arcs <- rbind(c(1,2,7),
              c(1,3,9),
              c(1,6,14),
              c(2,3,10),
              c(2,4,15),
              c(3,4,11),
              c(3,6,2),
              c(4,5,6),
              c(6,5,9))

nbNodes <- max(max(arcs[,1]),max(arcs[,2]))

sol <- dijkstra(nbNodes,1,arcs)

sp <- get_shortest_path(5,sol$path_list)
```

Bellman-Ford Algorithm

Algorithm 2 Bellman-Ford

procedure BELLMANFORD$(s, \mathcal{X}, \mathcal{A})$ $\triangleright$ Solve for $\mathbf{d}^*$

  Define the pair $(u_j, d_j) \in (\mathcal{X}, \mathbb{R})$, where $d_j := \mathrm{dist}(u_s, u_j)$.

  $u_s \in \mathcal{X}$; $\mathbf{d}^* = \boldsymbol{\infty}$, $d_s^* = 0$ $\triangleright$ Initialization

  for $i = 1, \ldots, |\mathcal{X}| - 1$ **do**

    for $j = 1, \ldots, |\mathcal{X}|$ **do**

      for $v_k \in N(j)$ **do**

        Calculate

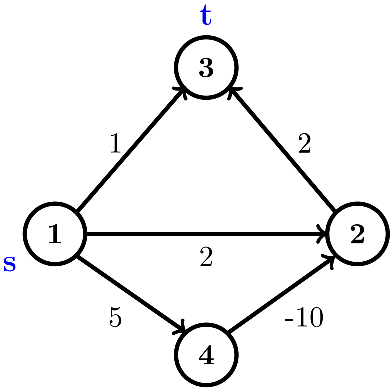
$$c_k = d_j^* + a_{j,k}$$

        if $c_k < d_k^*$ **then**

          $d_k^* = c_k$ $\triangleright$ Update d_k^*

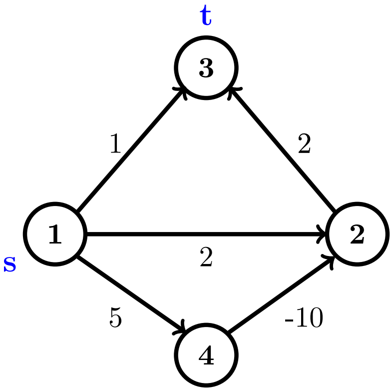
  return $\mathbf{d}^* = \{d_j^*\}_{j \in \mathcal{X}}$. $\triangleright$ Solution

Bellman-Ford Example



- What would Dijkstra do?

Bellman-Ford Example



- What would Dijkstra do?
- 1: $\{0, 2, 1, 5\}$. 3: $\{0, 2, 1, 5\}$. 2: $\{0, 2, 1, 5\}$. 4: $\{0, -5, 1, 10\}$. End.