

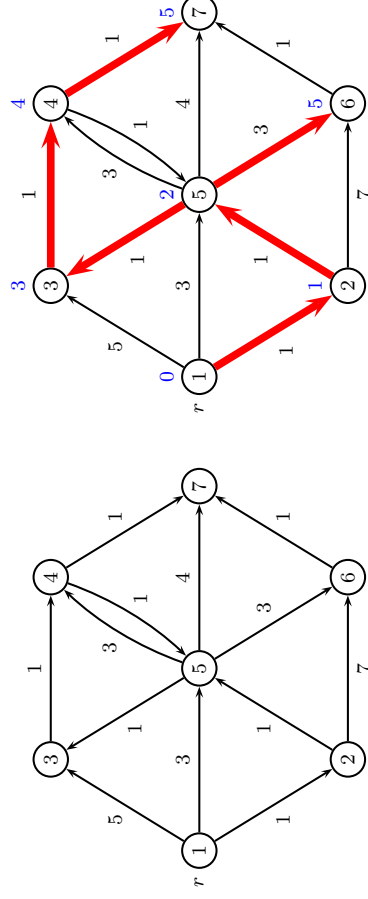
The Shortest Path Problem

Bellman-Ford and Dijkstra's algorithm

Jesper Larsen, Jens Clausen, David Pisinger
jesla, jenc1, pisinger@man.dtu.dk

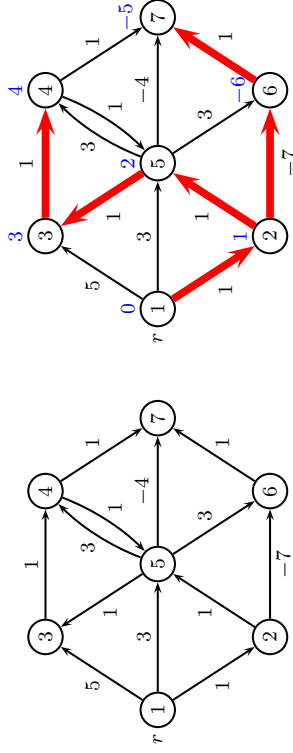
DTU Management
Technical University of Denmark

Example

Shortest path tree rooted at r

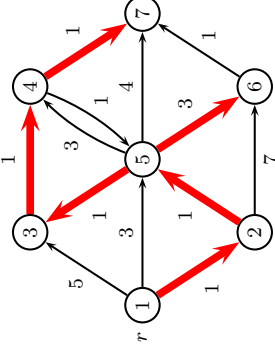
The Shortest Path Problem

- Given a **directed** network $\mathcal{G} = (V, E, w)$ where $w \in \mathbb{R}$ (n nodes, m edges)
- Furthermore, a **source vertex** r is given.
- **Objective:** Find for each $v \in V$ a shortest (with respect to w) directed path from r to v (if such one exists).
- **No negative cycles** reachable from r



Integer Programming Formulation

- Tree rooted at r . Look at the **number** of paths leaving a vertex vs. the **number** of paths entering a vertex.
- Vertex r must have $n - 1$ paths leaving the vertex.
- For any other vertex, the number of paths entering the vertex must be exactly 1 larger than the number of paths leaving the vertex.
- Let x_e denote the **number** of paths using edge $e \in E$.



Mathematical Model

This gives the following IP model:

$$\begin{aligned} \min \quad & \sum_{(i,j) \in E} w_{ij} x_{ij} \\ \text{s.t.} \quad & \sum_{i \in V} x_{ri} - \sum_{i \in V} x_{ir} = n - 1 \\ & \sum_{i \in V} x_{ij} - \sum_{i \in V} x_{ji} = 1 \quad j \neq r \\ & x_{ij} \in \mathbb{Z}_+ \quad (i,j) \in E \end{aligned}$$

Can be solved with CPLEX. Show more efficient algorithm

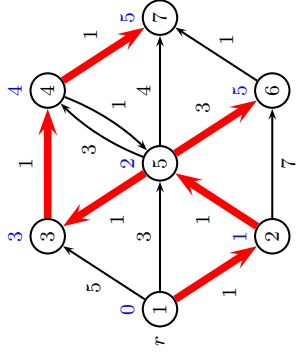
Feasible potentials

- Consider an n -vector $y = y_1, \dots, y_n$ (potentials)
- If y satisfies that $y_r = 0$ and

$$\forall (i,j) \in E : y_i + w_{ij} \geq y_j \quad (\text{no shortcuts})$$

then y is called a **feasible potential**.

- Informally: potentials y_v are possible distances from r to v . Potentials are feasible when no more shortcuts are possible



Mathematical Model for $r \rightarrow t$

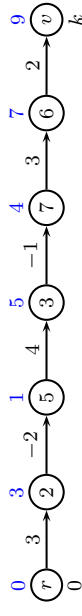
Shortest path from r to t

$$\begin{aligned} \min \quad & \sum_{(i,j) \in E} w_{ij} x_{ij} \\ \text{s.t.} \quad & \sum_{i \in V} x_{ri} - \sum_{i \in V} x_{ir} = 1 \\ & \sum_{i \in V} x_{ti} - \sum_{i \in V} x_{it} = -1 \\ & \sum_{i \in V} x_{ij} - \sum_{i \in V} x_{ji} = 0 \quad j \neq \{r, t\} \\ & x_{ij} \in \{0, 1\} \quad (i,j) \in E \end{aligned}$$

Feasible potentials II

- If P is a path from r to $v \in V$, then if y is a feasible potential, $w_P \geq y_v$:

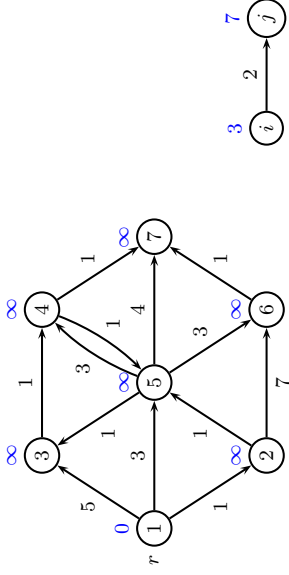
$$\begin{aligned} w_P &= \sum_{i=1}^k w_{e_i} \\ &\geq \sum_{i=1}^k (y_{v_i} - y_{v_{i-1}}) \\ &= y_{v_k} - y_{v_0} \\ &= y_v \end{aligned}$$



- y_v is smaller or equal to length of any path r to v

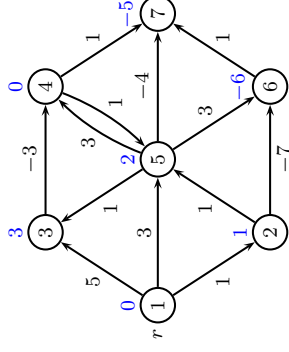
Basic algorithmic idea

- 🔴 Start with the potential $y_r = 0$ and $y_i = \infty, i \in V \setminus \{r\}$
- 🔴 Check for each edge if the potential is feasible
- 🔴 If YES - Stop - the potentials identify shortest paths
- 🔴 If an edge (i, j) violates feasibility condition, update y_j . This is sometimes called “correct (i, j) ” or “relax (i, j) ”



Problem with Ford's Algorithm

- 🔴 **Complexity!** Beware of *negative length circuits* - these may lead to infinite computation.
- 🔴 **Solution:** Use the same sequence for the edges in each iteration.

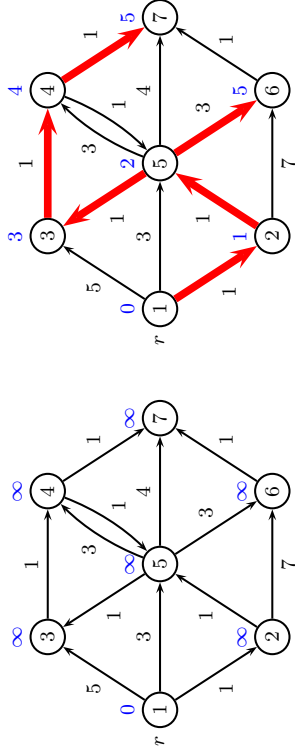


Ford's Shortest Path Algorithm

- ```

1 $y_r \leftarrow 0, y_i \leftarrow \infty$ for all other i ;
2 $p_r \leftarrow 0, p_i \leftarrow \text{NIL}$ for all other i ;
3 while an edge exists $(i, j) \in E$ such that $y_i + w_{ij} < y_j$ do
4 $y_j \leftarrow y_i + w_{ij}$;
5 $p_j \leftarrow i$;

```



where  $p_i$  marks entering edge  $(i, j)$  [red edges]

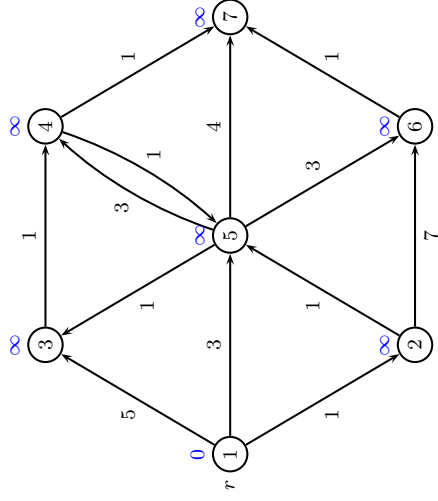
## Bellman-Ford's Shortest Path Algorithm

- ```

1  $y_r \leftarrow 0; y_i \leftarrow \infty$  for all other  $i$ ;
2  $p_r \leftarrow 0; p_i \leftarrow \text{NIL}$  for all other  $i$ ;
3 for  $k = 1$  to  $n - 1$  do
4   for  $(i, j) \in E$  do
5     if  $y_i + w_{ij} < y_j$  then
6        $y_j \leftarrow y_i + w_{ij}$ ;
7        $p_j \leftarrow i$ ;
8   for  $(i, j) \in E$  do /* check if negative cycle */
9     if  $y_i + w_{ij} < y_j$  then
10      negative cycle found, follow  $p_i$  to find the cycle

```

Example



The Shortest Path Problem – p. 13

Complexity of Bellman-Ford's Algorithm

A worst-case time complexity analysis leads to the following conclusions:

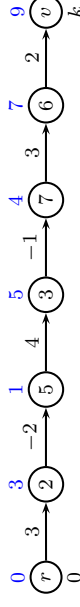
- Initialization: $O(n)$.
- Outer loop: $(n - 1)$ times.
- In the loop: each edge is considered one time: $O(m)$.
- All in all: $O(nm)$.

The Shortest Path Problem – p. 14

Correctness of Bellman-Ford's Algorithm

Proof by induction

- The induction hypothesis is: After iteration k of the main loop, y_i contains the length of any shortest path with at most k edges from 1 to i for any $i \in V$.
- For the base case $k = 0$ the induction hypothesis is trivially fulfilled. ($y_r = 0 =$ shortest path from r to r).
- For the inductive step, we assume that $y_{v_{k-1}} =$ shortest path from r to v_{k-1} after the $(k - 1)$ 'st pass. Then (v_{k-1}, v_k) is corrected in the k 'th iteration. So $y_{v_k} =$ shortest path from r to v_k .



The Shortest Path Problem – p. 15

Correctness of Bellman-Ford's Algorithm

- If all distances are non-negative, a shortest path containing at most $(n - 1)$ edges exists for each $v \in V$. If negative edge lengths are present, the algorithm still works. If a negative length circuit exists, this can be discovered by an extra iteration in the main loop. If at least one y_i changes, there is a negative length circuit.

The Shortest Path Problem – p. 16

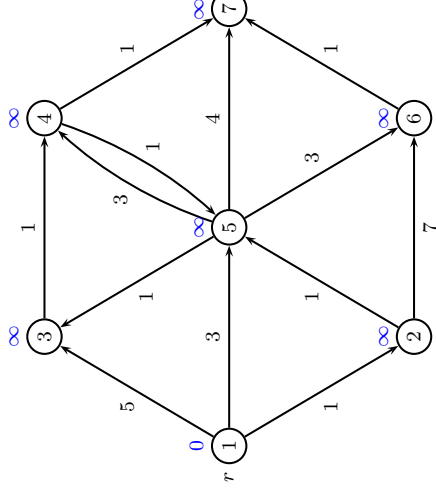
Introduction to Dijkstra's Algorithm

If we assume all edge weights $w_e \geq 0$ are non-negative we can derive a more efficient algorithm.

- Only consider each node once
- Add node v to set T when considered
- Always select the node $v \notin T$ with smallest potential
- Correct all outgoing edges from v

Greedy method

Example



Dijkstra's Algorithm for Shortest Path

```

1  $T \leftarrow \emptyset$ ;
2  $y_r \leftarrow 0$ ;  $y_i \leftarrow \infty$  for all other  $i$ ;
3 while  $V \setminus T \neq \emptyset$  do
4   select a  $i \in V \setminus T$  minimizing  $y_i$ ;
5   for  $\{(i, j) \in E : j \notin T\}$  do
6     if  $y_i + w_{ij} < y_j$  then
7        $y_j \leftarrow y_i + w_{ij}$ ;
8        $p_j \leftarrow i$ ;
9    $T \leftarrow T \cup \{i\}$ ;
```

Complexity of Dijkstras Algorithm

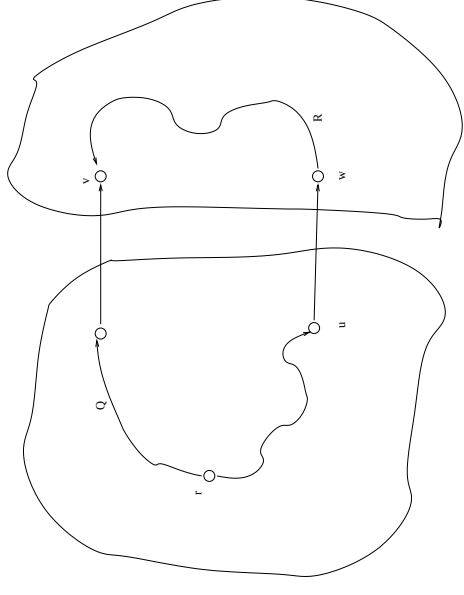
- The only difference to Prim's algorithm for Minimum Spanning Trees is the update step in the inner loop, and this step takes — like in the MST algorithm — $O(1)$.
- Hence the complexity of the algorithm is $O(n^2)$ if a list representation of the y vector is used, and a complexity of $O(m \log n)$ can be obtained if heap or balanced binary tree data structure is used for the y vector.
 - n numbers $\{y_i\}$ in data structure
 - n times find minimum in time $O(\log n)$
 - m times correct edge (i, j) in time $O(1)$
 - m times update y_j by delete/insert in time $O(\log n)$

Correctness of Dijkstras Algorithm

- This proof is made by induction:
- Suppose that before an iteration of the `while` loop it holds that
 1. for each vertex $u \in T$, the shortest path from r to u has been found and is of length y_u , and
 2. for each vertex $u \notin T$, y_u is the shortest path from r to u with all vertices except u belonging to T .
- This is obviously true initially.

The Shortest Path Problem – p. 21

Correctness of Dijkstras Algorithm III



$$y_w \geq y_v (= \text{length}(Q)) \Rightarrow \text{length}(R) \geq \text{length}(Q)$$

The Shortest Path Problem – p. 23

Correctness of Dijkstras Algorithm II

- Let v be the element with least y value picked initially in the inner loop of iteration k .
- y_v is the length of a path Q from r to v passing only through vertices in T .
- Suppose that this is not the shortest path from r to v . Then another path R from r to v is shorter.

The Shortest Path Problem – p. 22

Correctness of Dijkstras Algorithm IV

- Look at R : R starts in r , which is in T . Since v is not in T , R has an edge from a vertex in P to a vertex not in T .
- Let (u, w) be the first edge of this type. w is a candidate for vertex choice in the current iteration, where v was picked. Hence $y_w \geq y_v$.
- If all edge lengths are non-negative, the length of the part of R from w to v is non-negative, and hence the total length of R is at least the length of the path Q .

The Shortest Path Problem – p. 24

Correctness of Dijkstras Algorithm V

- This is a contradiction – hence Q is a shortest path from r to v .
- Furthermore, the update step in the inner loop ensures that after the current iteration it again holds for u not in P (which is now the “old” P augmented with v) that y_u is the shortest path from from r to u with all vertices except u belonging to P .

Dijkstra’s Algorithm, negative edges ??

Find the most negative edge weight w_m and add $|w_m|$ to all edge weights. Now all $w_e \geq 0$.

