

Shortest path algorithms

Data Structures and Algorithms for Computational Linguistics III
(ISCL-BA-07)

Çağrı Çöltekin
ccoltekin@uni-tuebingen.de

University of Tübingen
Seminar für Sprachwissenschaft

Winter Semester 2021/22

version: 09/05/2022 12:08

Weighted graphs

- A *weighted graph* is a graph, where each edge is associated with a weight
- Weights can be any numeric value, but for some algorithms require
 - Non-negative weights
 - 'Euclidean' weights: weights that are proper distance metrics
- Weights often indicate distance or cost, but they can also represent positive relations (e.g., affinity between nodes)
- Weight of a path is the sum of weights of the edges on the path



Ç. Çöltekin, SS / University of Tübingen

Winter Semester 2021/22 1 / 18

Shortest path

- Finding shortest paths on a weighted (directed) graph is one of the most common problems in many fields
- Applications include
 - Navigation
 - Routing in computer networks
 - Optimal construction of electronic circuits, VLSI chips
 - Robotics, transportation, finance, ...

Ç. Çöltekin, SS / University of Tübingen

Winter Semester 2021/22 2 / 18

Shortest paths on unweighted graphs

BFS

- A BFS search tree gives the shortest path from the source node to all other nodes
- The BFS is not enough on weighted graphs
- Shortest-cost path may be longer in terms of nodes visited



Ç. Çöltekin, SS / University of Tübingen

Winter Semester 2021/22 3 / 18

Shortest paths on weighted graphs

variations of the problem

- Different versions of the problem:
 - Single source shortest path: find shortest path from a source node to all others
 - Single target (sometimes called sink) shortest path: find shortest path from all nodes to a target node
 - Source to target: from a particular source node to a particular target node
 - All pairs: shortest paths between all pairs of nodes
- Restrictions on weights:
 - Euclidean weights
 - Non-negative weights
 - Arbitrary weights

Ç. Çöltekin, SS / University of Tübingen

Winter Semester 2021/22 4 / 18

Dijkstra's algorithm

intro

- Dijkstra's algorithm is a 'weighted' version of the BFS
- The algorithm finds shortest path from a single source node to all connected nodes
- Weights has to be non-negative
- It is a greedy algorithm that grows a 'cloud' of nodes for which we know the shortest paths from the source node
- The new nodes are included in the cloud in order of their shortest paths from the source node

Ç. Çöltekin, SS / University of Tübingen

Winter Semester 2021/22 5 / 18

Dijkstra's algorithm

the algorithm

```
1: D[s] ← 0
2: for each node v ≠ s do
3:   D[v] ← ∞
4: Q ← nodes
5: while Q is not empty do
6:   Remove node u with min D[u] from Q
7:   for each edge (u, v) do
8:     if D[u] + w(u, v) < D[v] then
9:       D[v] ← D[u] + w(u, v)
10: D contains the shortest distances from s
```

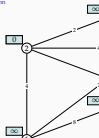
- We maintain a list D of minimum known distances to each node
- At each step
 - we take closest node out of Q
 - update the distances of all nodes
- Can be more efficient if Q is implemented using a (adaptable) priority queue

Ç. Çöltekin, SS / University of Tübingen

Winter Semester 2021/22 6 / 18

Dijkstra's algorithm

demonstration

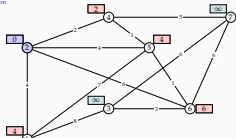


Ç. Çöltekin, SS / University of Tübingen

Winter Semester 2021/22 7 / 18

Dijkstra's algorithm

demonstration

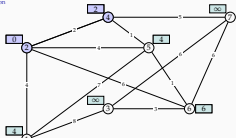


Ç. Çöltekin, SS / University of Tübingen

Winter Semester 2021/22 7 / 18

Dijkstra's algorithm

demonstration

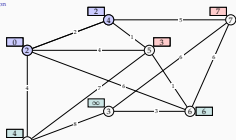


Ç. Çöltekin, SS / University of Tübingen

Winter Semester 2021/22 7 / 18

Dijkstra's algorithm

demonstration

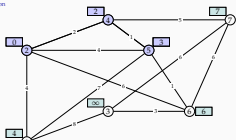


Ç. Çöltekin, SS / University of Tübingen

Winter Semester 2021/22 7 / 18

Dijkstra's algorithm

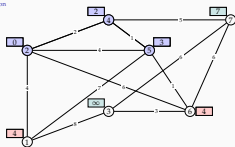
demonstration



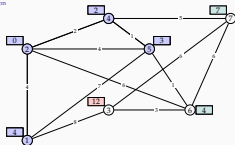
Ç. Çöltekin, SS / University of Tübingen

Winter Semester 2021/22 7 / 18

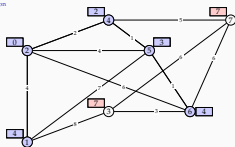
Dijkstra's algorithm demonstration



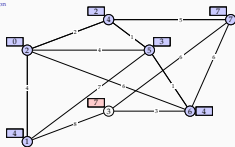
Dijkstra's algorithm demonstration



Dijkstra's algorithm demonstration



Dijkstra's algorithm demonstration



Dijkstra's algorithm and negative weights



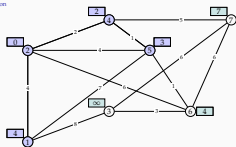
Shortest-path tree

- The way we introduced, the Dijkstra's algorithm does not give the shortest-path tree
 - Similar to traversal algorithms, we can extract it from distances D
 - Running time is $O(n^2)$ (or $O(n + m)$)
- ```

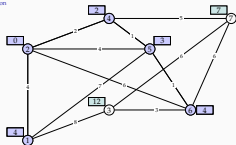
1: T ← ∅
2: for u ∈ D ← {s} do
3: for each edge (v, u) do
4: if D[v] == D[u] + w(v, u) then
5: T ← T ∪ (v, u)

```

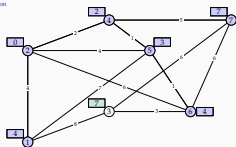
## Dijkstra's algorithm demonstration



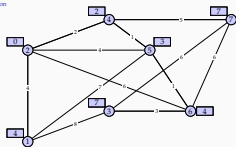
## Dijkstra's algorithm demonstration



## Dijkstra's algorithm demonstration



## Dijkstra's algorithm demonstration



## Dijkstra's algorithm the algorithm

- In general, complexity is  $O(\text{find\_min} + \text{update\_key})m$
  - With list-based implementation of Q:  $O(m + n^2) = O(n^2)$
  - With a priority queue:  $O((m + n) \log n)$
- ```

1: D[s] ← 0
2: for each node v ≠ s do
3:   D[v] ← ∞
4: Q ← nodes
5: while Q is not empty do
6:   Remove node u with min D[u] from Q
7:   for each edge (u, v) do
8:     if D[u] + w(u, v) < D[v] then
9:       D[v] ← D[u] + w(u, v)
10: D contains the shortest distances from s

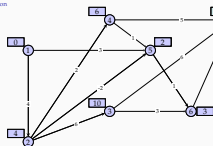
```

Shortest-paths on DAGs

- The shortest path can be found more efficiently, if the graph is a DAG
- The algorithm is similar to Dijkstra's, but simpler and faster
- Only difference is we follow a topological order
- The algorithm will also work with negative edge weights

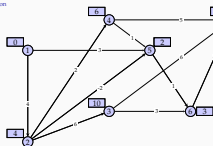
Shortest-paths on DAGs

demonstration



Shortest-paths on DAGs

demonstration

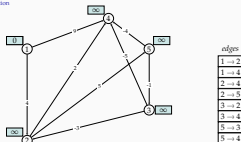


Shortest-paths on directed graphs with negative weights - without negative cycles

- Single-source shortest path problem can also be solved efficiently for any directed graph
 - including cycles (no DAG requirement)
 - including negative weights
 - excluding negative cycles
- The algorithm is known as Bellman-Ford algorithm
 - Similar to earlier algorithms, initialize $D[s] = 0, D[v] = \infty$
 - Make n passes over the edges
 - Update distances for each edge (relax edges)
 - Stop if there were no changes at the end of a pass

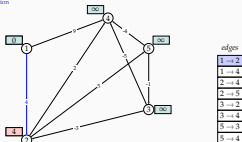
Bellman-Ford algorithm

demonstration



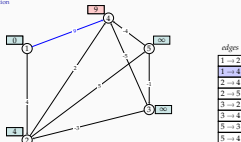
Bellman-Ford algorithm

demonstration



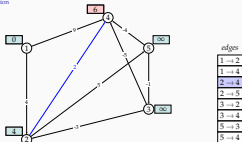
Bellman-Ford algorithm

demonstration



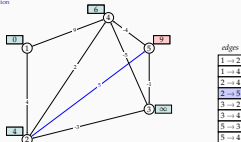
Bellman-Ford algorithm

demonstration



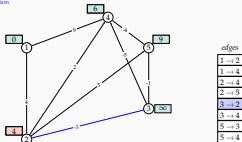
Bellman-Ford algorithm

demonstration



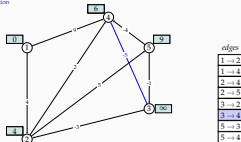
Bellman-Ford algorithm

demonstration



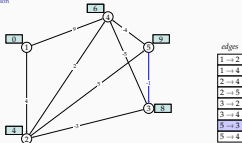
Bellman-Ford algorithm

demonstration



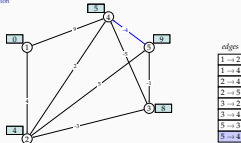
Bellman-Ford algorithm

demonstration



Bellman-Ford algorithm

demonstration



demonstration



Wissenschaften November 2021/22 14 / 19

demonstration



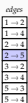
Winter Semester 2021/22 14 / 19

demonstration



Winter Semester 2021/22 14 / 19

demonstration



Winter, November 2021 (22) 14 / 19

demonstration



© 2004 Blackwell Publishing Ltd *Journal of Internal Medicine* 255: 333–340

demonstration



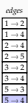
DOI: 10.1002/anie.201100000

demonstration



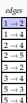
© 2004 Blackwell Publishing Ltd *Journal of Internal Medicine* 255: 333–340

demonstration



DOI: 10.1002/anie.201100000

demonstration



Winter Semester 2021/22 14 / 19

demonstration



Winter Semester 2021/22 14 / 19

demonstration

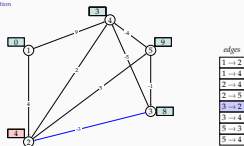
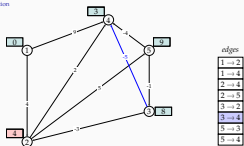
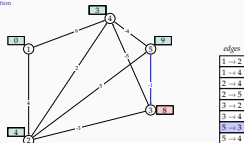
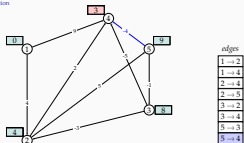


© 2004 Blackwell Publishing Ltd *Journal of Internal Medicine* 255: 333–340

demonstration



DOI: 10.1002/anie.200500000

Bellman-Ford algorithm
demonstrationBellman-Ford algorithm
demonstrationBellman-Ford algorithm
demonstrationBellman-Ford algorithm
demonstration

Summary

- Shortest path algorithms are one of the most applied graph algorithms
- We revised three algorithms
 - Dijkstra's: non-negative weights, general algorithm
 - For DAGs: unrestricted weights, following topological order
 - Bellman-Ford: no negative cycles, digraphs

Reading: Goodrich, Tamassia, and Goldwasser (2013, chapter 14)

Next:

- Minimum spanning trees
- Reading: Goodrich, Tamassia, and Goldwasser (2013, chapter 14)

Acknowledgments, credits, references

- Goodrich, Michael T., Roberto Tamassia, and Michael H. Goldwasser (2013). *Data Structures and Algorithms in Python*. John Wiley & Sons, Incorporated. isbn: 9781118476734.