

Shortest path

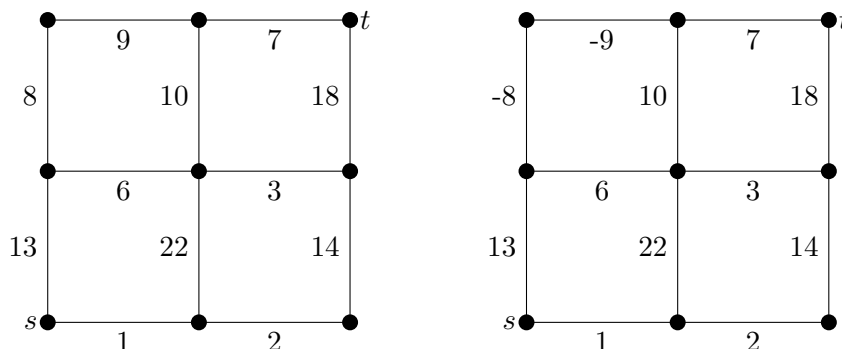
Source: Chapter 2.2 (Bills), Chapter 7 of Combinatorial Optimization (Korte)

Shortest path

Input: Graph $G = (V, E)$, costs $c : E \rightarrow \mathbb{R}$, and $s, t \in V$.

Output: s - t -path P , where $\sum_{e \in P} c(e)$ is minimized.

1: Find shortest (lowest cost) s - t -paths in the following graphs



Cost c is called *conservative* if there is no circuit of negative total weight.

Bellman's principle: Let s, \dots, v, w be the least cost s - w -path of length k . The s, \dots, v is the least cost s - v -path of length $k - 1$.

2: Prove Bellman's principle.

Notice: This gives a recursion for computing the shortest path.

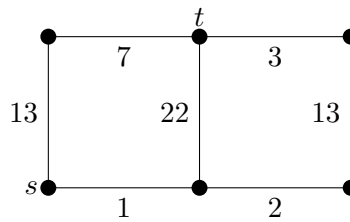
Dijkstra's algorithm

$c : E \rightarrow \mathbb{R}_+$, computes shortest s - t -path from s to ALL other vertices $t \in V$.

1. $l(s) := 0; \forall v \neq s \ l(v) = +\infty$
2. $R = \emptyset$
3. while $R \neq V$
4. find $v \in V - R$ with minimum $l(v)$
5. $R := R \cup \{v\}$
6. $\forall vw \in E, \ l(w) = \min\{l(w), l(v) + c(v, w)\}$

R ... vertices with final number; l ... upper bound on the cost;
 Running time $O(n^2)$ easily, $O(m + n \log n)$ with Fibonacci heaps.

3: Run Dijkstra's algorithm on the following graph



4: How to get shortest s - v -path?

Remember previous vertex. In step 6. of the algorithm, remember why the value was changed. So called *predecessor*.

5: Why is the algorithm correct? (show that if $v \in R$, then $l(v) = \text{cost for } s\text{-}t\text{-path.}$)

6: Why Dijkstra's algorithm does not work for negative costs?

Moore-Bellman-Ford Algorithm

$c : E \rightarrow \mathbb{R}$, computes shortest s - t -path from s to ALL other vertices $t \in V$ **OR** finds a cycle of negative cost. Assume $|V(G)| = n$.

1. $l(s) := 0; \forall v \neq s \ l(v) = +\infty$
2. repeat $n - 1$ times: // computes the costs
3. $\forall vw \in E,$
4. if $l(w) > l(v) + c(v, w)$
5. $l(w) := l(v) + c(v, w); p(w) = v$
6. $\forall vw \in E, //$ check for a negative cycle
7. if $l(w) > l(v) + c(v, w)$ then found negative cycle

Note: l gives the least cost, while p gives the **previous** vertex / **predecessor** on the shortest path from s .

7: What is the time complexity of the algorithm if G has m edges and n vertices?

8: Why the algorithm detects a negative cycle and why the algorithm works?