Algorithms Theory

11 – Shortest Paths

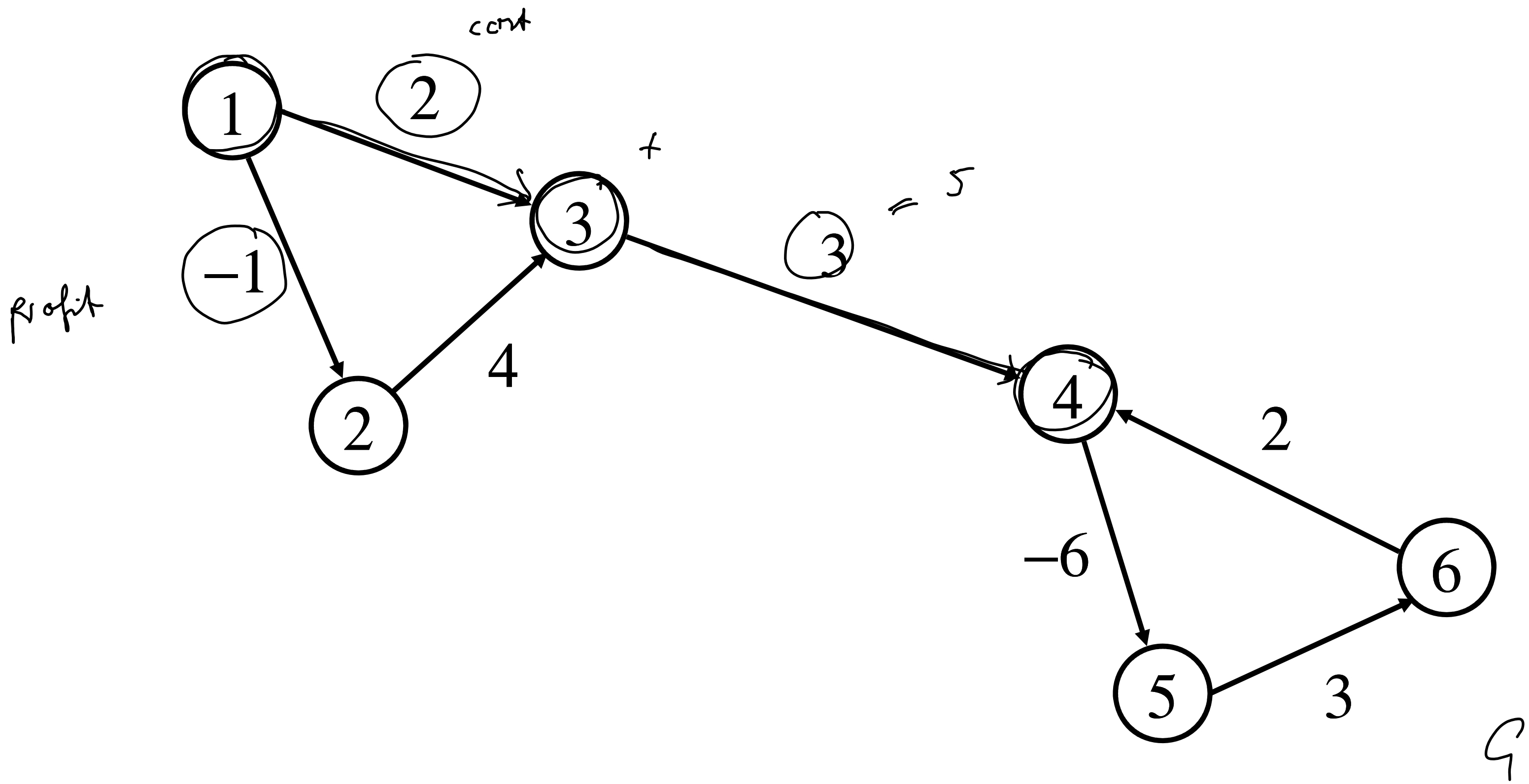
Dr. Alexander Souza

Winter term 11/12

1. Shortest-paths problem

Directed graph $G = (V, E)$

Cost function $c : E \rightarrow R$ maybe also negative



Distance between two vertices

$u \qquad\qquad v$
$\| \qquad\qquad \|$

Cost of a path $P = v_0, v_1, \ldots, v_l$ from u to v : $\qquad v_i v_{i+1} \in E$

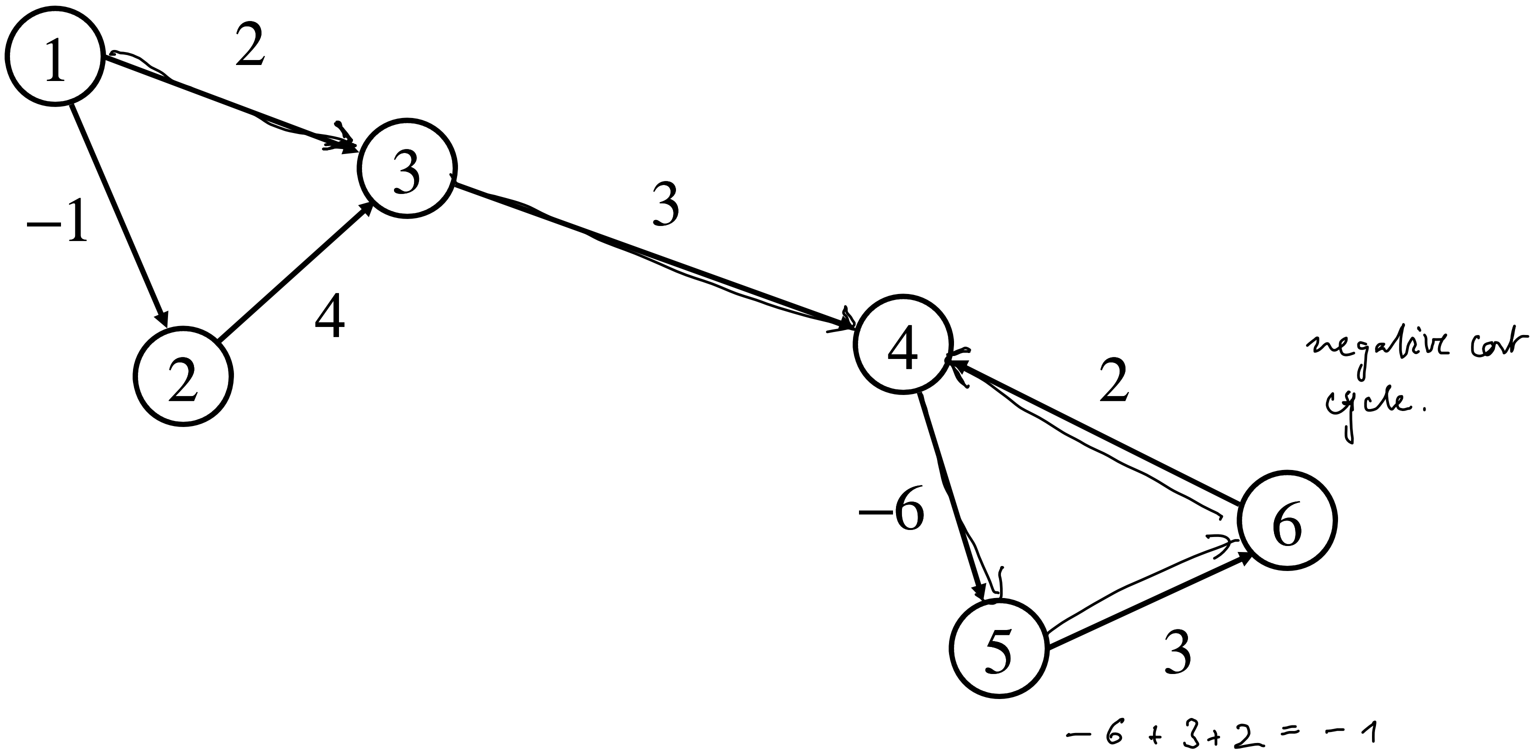
$$c(P) = \sum_{i=0}^{l-1} c(v_i, v_{i+1})$$

Distance between u and v (not always defined):

$dist(u,v) = \inf \{ c(P) \mid P \text{ is a path from } u \text{ to } v \}$

Example

In shortest paths problems, paths need not be simple, i.e. we may traverse an edge arbitrarily often



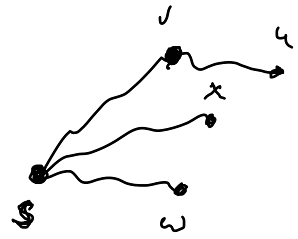
dist(1,2) = −1

dist(1,3) = 2

dist(3,1) = +∞

dist(3,4) = −∞

2. Single-source shortest paths problem



Input: network $G = (V, E, c)$, $c : E \rightarrow R$, vertex s

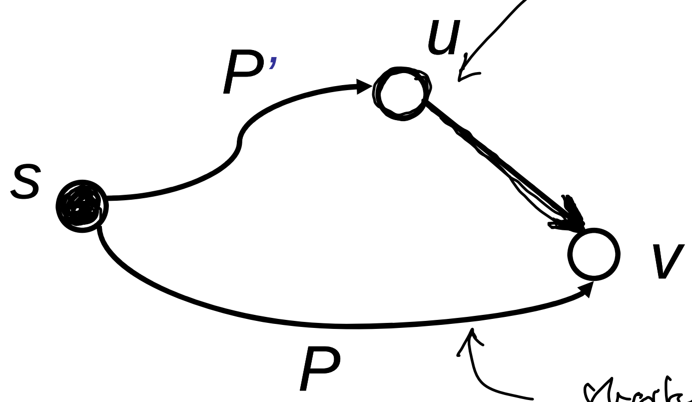
Output: $dist(s,v)$ for all $v \in V$

Observation: The function *dist* satisfies the triangle inequality.

For any edge $(u,v) \in E$:

$$dist(s,v) \leq dist(s,u) + c(u,v)$$

Same path from s to v



P = shortest path from s to v

P' = shortest path from s to u

shortest path from s to v.

Greedy approach to an algorithm

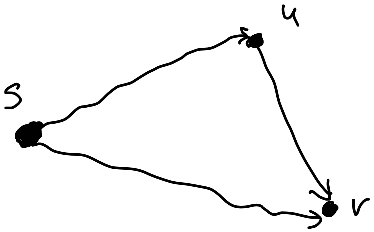
1. Overestimate the function *dist*

$$dist'(s,v) = \begin{cases} 0 & \text{if } v = s \\ \infty & \text{if } v \neq s \end{cases}$$

2. While there exists an edge $e = (u,v)$ with

$$dist'(s,v) > dist'(s,u) + c(u,v)$$

set $dist'(s,v) \leftarrow dist'(s,u) + c(u,v)$



Correctness? Termination?

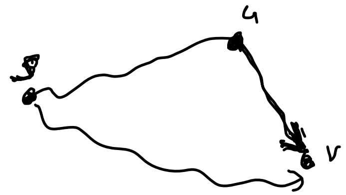
Generic algorithm

array storing our estimates for dist(s,v) ~ DIST[v]

1. DIST[s] $\leftarrow 0$;
2. **for all** $v \in V \setminus \{s\}$ **do** DIST[v] $\leftarrow \infty$ **endfor;**
3. **while** $\exists\, e = (u,v) \in E$ with DIST[v] > DIST[u] + $c(u,v)$ **do**
4. Choose such an edge $e = (u,v)$;
5. DIST[v] $\leftarrow$ DIST[u] + $c(u,v)$;
6. **endwhile;**

edge (u,v) violates △-inequality with our current estimate DIST

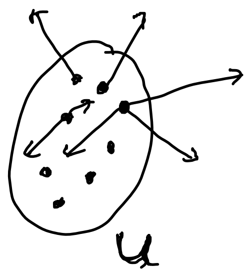
edge relaxation



Questions:

1. How can we efficiently check in line 3 if the triangle inequality is violated?
2. Which edge shall we choose in line 4?

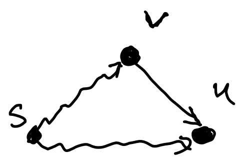
Solution



Maintain a set *U* of all those vertices that might have an outgoing edge violating the triangle inequality.

- Initialize $U = \{s\}$
- Add vertex v to U whenever DIST[v] decreases.

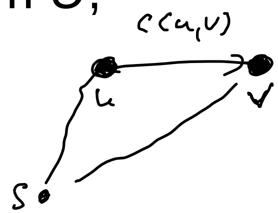
$$\mathrm{DIST}[u] \le \mathrm{DIST}[v] + c(v,u)$$



1. Check if the triangle inequality is violated: $U \neq \emptyset$?
2. Choose a vertex from U and restore the triangle inequality for all outgoing edges (edge relaxation).

Refined algorithm

```
DIST[s] ← 0;
for all v ∈ V \ {s} do DIST[v] ← ∞ endfor;
U ← {s};
while U ≠ ∅ do
    Choose a vertex u ∈ U and delete it from U;
    for all e = (u,v) ∈ E do
        if DIST[v] > DIST[u] + c(u,v) then
            DIST[v] ← DIST[u] + c(u,v);
            U ← U ∪ {v};
        endif;
    endfor;
endwhile;
```



Invariant for the DIST values

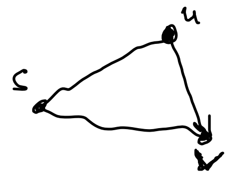
Lemma 1: For each vertex $v \in V$ we have DIST[v] $\geq$ *dist*(s,v).

DIST remains an overestimate.

Proof: (by contradiction)

Let v be the first vertex for which the relaxation of an edge (u,v) yields DIST[v] $<$ *dist*(s,v).

DIST[u] ≥ dist(s,u)



Then:

$$\mathrm{DIST}[u] + \cancel{c(u,v)} = \mathrm{DIST}[v] < \underbrace{dist(s,v)}_{\text{shortest path length from } s \text{ to } v} \leq dist(s,u) + \cancel{c(u,v)} \quad \Big|\, - c(u,v)$$

DIST[u] < dist(s,u) ↯

Contradiction that v is the first vertex with DIST[v] < dist(s,v)

Important properties

Lemma 2: (Running time + Correctness)

a) If $v \notin U$, then for all $(v,w) \in E$: DIST[w] $\leq$ DIST[v] + $c(v,w)$

b) Let $s = v_0, v_1, \ldots, v_l = v$ be a shortest path from s to v.
If DIST[v] $>$ $dist(s,v)$, then there exists v_i, $0 \leq i \leq l-1$, with
$v_i \in U$ and DIST[v_i] = $dist(s,v_i)$. $\rightsquigarrow$ $U \neq \emptyset \Rightarrow$ Alg. has not terminated.
$\sim$ Correctness.

c) If G has no negative-cost cycles and DIST[v] > $dist(s,v)$ for any $v \in V$, then there exists a $u \in U$ with DIST[u] = $dist(s,u)$.

d) If in line 5 we always choose $u \in U$ with DIST[u] = $dist(s,u)$, then the while-loop is executed only once per vertex.