Solution

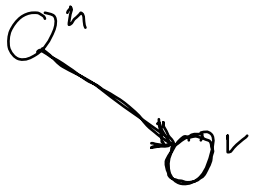
Maintain a set U of all those vertices that might have an outgoing edge violating the triangle inequality.

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

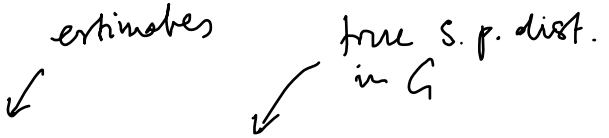
1. Check if the triangle inequality is violated: $U \neq \emptyset$?
2. Choose a vertex from U and restore the triangle inequality for all outgoing edges (edge relaxation).

Refined algorithm

1. DIST[s] $\leftarrow 0$;
2. **for all** $v \in V \setminus \{s\}$ **do** DIST[v] $\leftarrow \infty$ **endfor;**
3. $U \leftarrow \{s\}$;
4. **while** $U \neq \emptyset$ **do**
5. Choose a vertex $u \in U$ and delete it from U;
6. **for all** $e = (u,v) \in E$ **do**
7. **if** DIST[v] > DIST[u] + $c(u,v)$ **then**
8. DIST[v] $\leftarrow$ DIST[u] + $c(u,v)$;
9. $U \leftarrow U \cup \{v\}$;
10. **endif;**
11. **endfor;**
12. **endwhile;**



Invariant for the DIST values



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

Proof: (by contradiction)

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

Then:

$$\text{DIST}[u] + c(u,v) = \text{DIST}[v] < dist(s,v) \leq dist(s,u) + c(u,v)$$

Important properties

Lemma 2:

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

U serves its purpose

b) Let $s = v_0, v_1, \ldots, v_l = v$ be a shortest path from s to v.
If $\text{DIST}[v] > dist(s,v)$, then there exists v_i, $0 \leq i \leq l-1$, with $v_i \in U$ and $\text{DIST}[v_i] = dist(s,v_i)$.

Correctness

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

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

Running time

Proof of Lemma 2

a) Induction on the number *i* of executions of while-loop

i = 0:

Proof of Lemma 2

i > 0:

Proof of Lemma 2

b)

Efficient implementations

Line 5: How can we find a vertex $u \in U$ with DIST[u] = $dist(s,u)$?

This is not known in general, but for some important special cases.

- Non-negative networks (only non-negative edge costs)
  Dijkstra's algorithm
- Networks without negative-cost cycles
  Bellman-Ford algorithm
- Acyclic networks

3. Non-negative networks

5'. Choose a vertex $u \in U$ with minimum DIST[u] and delete it from U.

Lemma 3: Using 5' we have DIST[u] = $dist(s,u)$.

DIST[u] > dist(s,u)

Proof: By Lemma 2b) there is a vertex $v \in U$ on the shortest path from s to u with DIST[v] = $dist(s,v)$.

$$DIST[u] \leq DIST[v] = dist(s,v) \leq dist(s,u) \Rightarrow DIST[u] \leq dist(s,u)$$

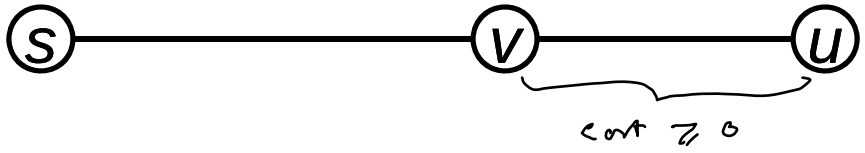
Alg. chooses vertex with min. DIST

Lemma 2b

non-neg. edge-cost

Lemma 1: $DIST[u] \geq dist(s,u)$

$\Rightarrow DIST[u] = dist(s,u)$



Implementing *U* as priority queue

The elements of the form (*key*, *inf*) are the pairs (DIST[*v*], *v*).

Empty(*Q*): Is *Q* empty?

Insert(*Q*, *key*, *inf*): Inserts (*key*,*inf*) into *Q*.

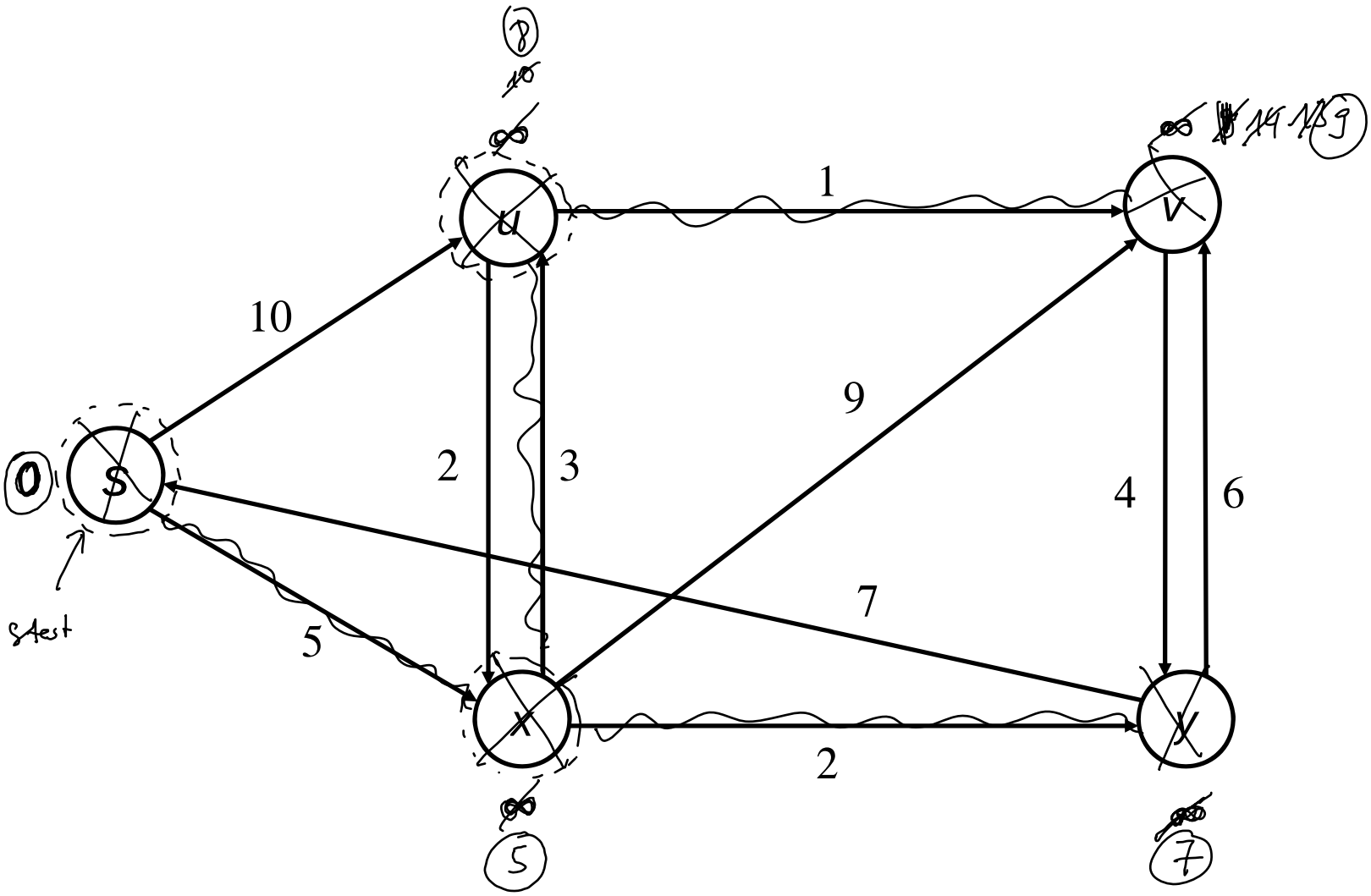
DeleteMin(*Q*): Returns the element with minimum key and deletes it from *Q*.

DecreaseKey(*Q*, *element*, *j*): Decreases the value of *element*'s key to the new value *j*, provided that *j* is less than the former key.

Dijkstra's algorithm

1. DIST[s] $\leftarrow 0$; Insert(U,0,s);
2. **for all** $v \in V \setminus \{s\}$ **do** DIST[v] $\leftarrow \infty$; Insert(U, ∞, v); **endfor;**
3. **while** $\neg$Empty(U) **do**
4. (d,u) $\leftarrow$ DeleteMin(U);
5. **for all** $e = (u,v) \in E$ **do**
6. **if** DIST[v] > DIST[u] + c(u,v) **then**
7. DIST[v] $\leftarrow$ DIST[u] + c(u,v);
8. DecreaseKey(U, v, DIST[v]);
9. **endif;**
10. **endfor;**
11. **endwhile;**

Example



Running time

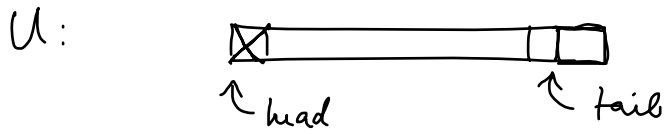
→ $O(n \cdot (T_{\text{Insert}} + T_{\text{Empty}} + T_{\text{DeleteMin}}) + m \cdot T_{\text{DecreaseKey}} + m + n)$

Fibonacci heaps:

T_{Insert} :	O(1)
$T_{\text{DeleteMin}}$:	O(log n) amortized
$T_{\text{DecreaseKey}}$:	O(1) amortized

$O(n \log n + m)$

4. Networks without negative-cost cycles



Implement *U* as a queue.

Lemma 4: Each vertex *v* is inserted into *U* at most *n* times.

⇒ Thus also deleted at most n times.

Proof: Suppose that DIST[*v*] > *dist*(*s*,*v*) and *v* is appended at *U* for the *i*-th time. Then, by Lemma 2c) there exists $u_i \in U$ with DIST[u_i] = *dist*(*s*,u_i).

→ Vertex u_i is deleted from *U* before *v* and will never be appended at *U* again. By Lemma 1. DIST[u_i] = dist(s,u_i).

Vertices u_1, u_2, u_3, ... are pairwise distinct.

v is appended to U for at most n-1 times when DIST[v] > dist(s,v).

v is appended once more when DIST[v] = dist(s,v).

Bellman-Ford algorithm

counts number of times vertex v has been deleted from U

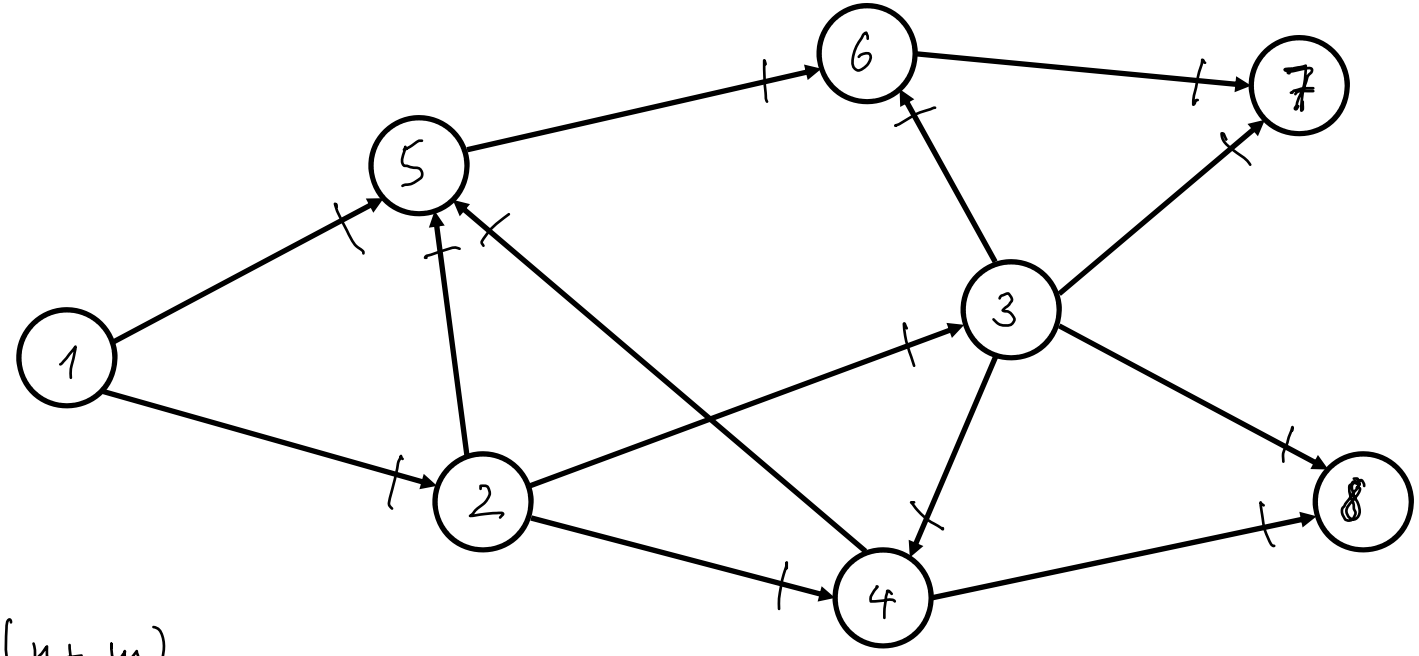
1. DIST[s] $\leftarrow 0$; A[s] $\leftarrow 0$;
2. **for all** $v \in V \setminus \{s\}$ **do** DIST[v] $\leftarrow \infty$; A[v] $\leftarrow 0$; **endfor;**
3. append s to U;
4. **while** $U \neq \emptyset$ **do**
5. Choose the first vertex u in U and delete it from U; A[u] $\leftarrow$ A[u]+1;
6. **if** A[u] > n **then** return „negative-cost cycle“;
7. **for all** $e = (u,v) \in E$ **do**
8. **if** DIST[v] > DIST[u] + $c(u,v)$ **then**
9. DIST[v] $\Leftarrow$ DIST[u] + $c(u,v)$;
10. append v to U;
11. **endif;**
12. **endfor;**
13. **endwhile;**

Running time: $O(n + n^2 + n \cdot m) = O(n^2 + n \cdot m) = O(n \cdot m)$

5. Acyclic networks

Topological sorting: num: $V \rightarrow \{1, \ldots, n\}$

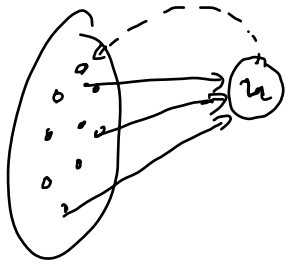
such that for all $(u,v) \in E$: num(u) < num(v)



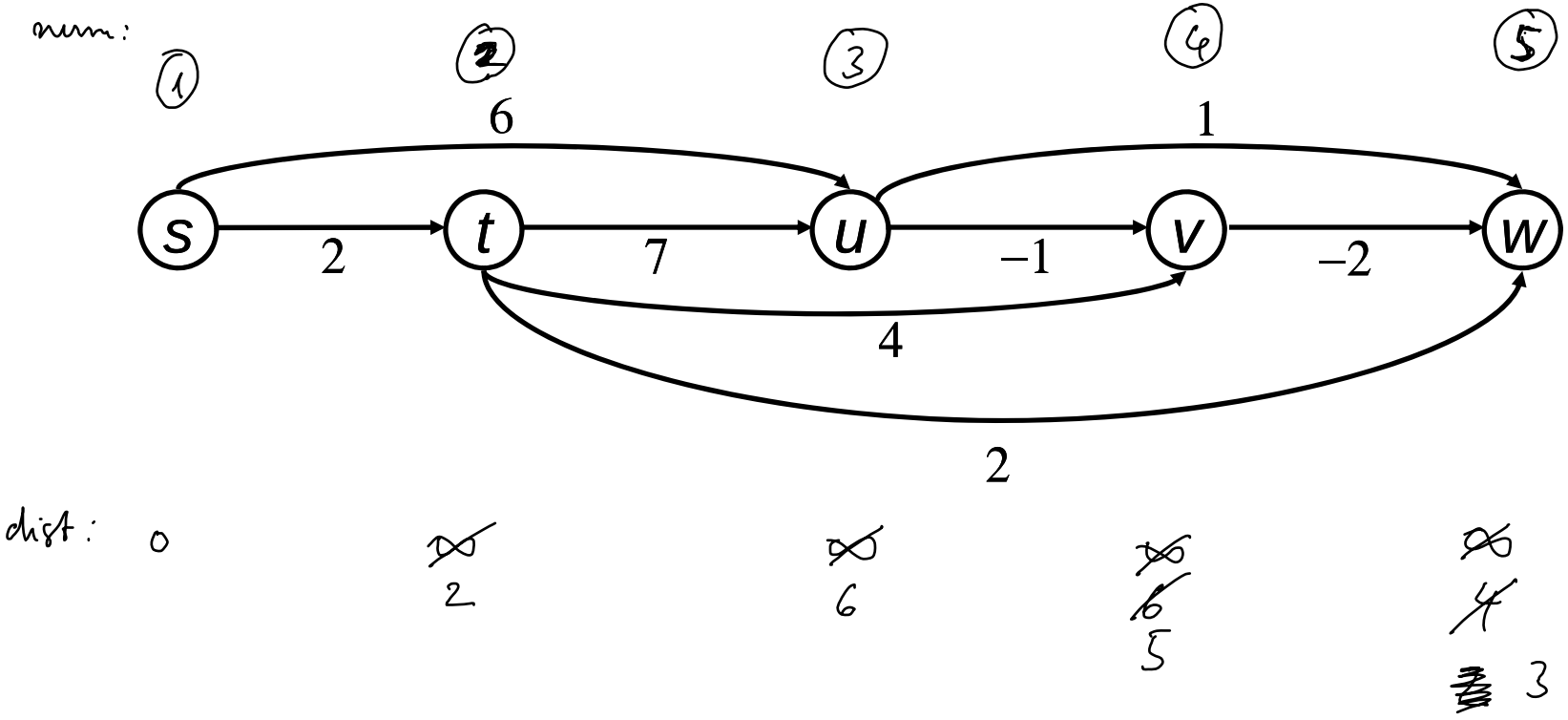
$O(n+m)$

Algorithm for acyclic graphs

1. Sort $G = (V, E, c)$ topologically;
2. DIST[s] $\leftarrow 0$;
3. **for all** $v \in V \setminus \{s\}$ **do** DIST[v] $\leftarrow \infty$; **endfor;**
4. $U \leftarrow \{ v \mid v \in V$ with num(v) < $n\}$;
5. **while** $U \neq \emptyset$ **do**
6. Choose the vertex $u \in U$ with minimum num;
7. **for all** $e = (u,v) \in E$ **do**
8. **if** DIST[v] > DIST[u] + $c(u,v)$ **then**
9. DIST[v] $\leftarrow$ DIST[u] + $c(u,v)$;
10. **endif;**
11. **endfor;**
12. **endwhile;**



Example



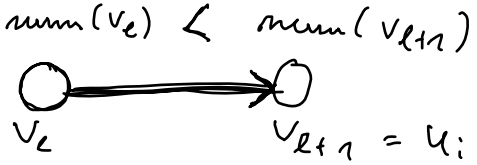
Correctness

Lemma 5: When the *i*-th vertex u_i is deleted from U, then
$\mathrm{DIST}[u_i] = dist(s,u_i)$.

Proof: Induction on *i*.

i = 1: ok

i > 1: Let $s = v_1, v_2, \ldots, v_l, v_{l+1} = u_i$ be a shortest path from *s* to u_i.



v_l is deleted from U before u_i.

Then, by induction hypothesis: $\mathrm{DIST}[v_l] = dist(s,v_l)$.

After (v_l, u_i) has been relaxed:

$\mathrm{DIST}[u_i] \leq \mathrm{DIST}[v_l] + c(v_l, u_i) = dist(s,v_l) + c(v_l, u_i) = dist(s,u_i)$

$\mathrm{DIST}[u_i] \leq dist(s,u_i)$
$\mathrm{DIST}[u_i] \geq dist(s,u_i)$