Algorithms and Complexity

3. Shortest Paths

Marc Hellmuth

University of Stockholm

Shortest Path

A **path** in a (di)graph $G = (V,E)$ is a sequence $< v_0, \dots, v_k >$ of vertices in V such that $(v_i, v_{i+1}) \in E$, $0 \leq i \leq k-1$ (also called v_0-v_k-path)

If for a path $< v_0, \dots, v_k >$ it holds that $v_i \neq v_j$, $0 \leq i < j \leq k$, then the path is **simple**

Length of path $< v_0, \dots, v_k >$ is k (=number of edges).

If we have a weighting function $w\colon E \to \mathbb{R}$, then the length of a path $< v_0, \dots, v_k >$ is $\sum_{i=0}^{k-1} w(e_i)$ where $e_i = (v_i, v_{i+1})$

The **distance** δ between vertices $u, v \in V$ is

$$\delta(u,v) = \begin{cases} \text{length of shortest } u\text{-}v\text{-path} & \text{if it exists} \\ \infty & \text{else} \end{cases}$$

The aim is to find shortest paths, however, the complexity of this problem heavily depends on the weighting function w.

Difficult shortest path problems

Problem: Shortest-Simple-Path (SSP):
input: (di)graph $G = (V, E)$, weighting $w\colon E \to \mathbb{R}$, $k \in \mathbb{Z}$
question: Is there a simple path in G of length $\leq k$?

Theorem

SSP is NP-complete

WHITEBOARD: proof.

Now, simple shortest path problems.

Lemma 3.1

Let $G = (V,E)$ be a digraph with weighting function $w\colon E \to \mathbb{R}$. Let $< v_0, \ldots, v_k >$ be shortest $v_0 - v_k$ path in G. Then, for all $1 \leq i,j \leq k$ it holds that $< v_i, \ldots, v_j >$ is a shortest $v_i - v_j$ path in G.

WHITEBOARD: proof.

A (di)graph $G = (V,E)$ has a **conservative** weighting function $w\colon E \to \mathbb{R}$, if G contains no cycles of negative length.

$\Longrightarrow$ In this case, we can w.l.o.g. consider simple paths

WHITEBOARD: example.

Basic functions

PRINT_PATH(digraph G vertices s, v) runtime $O(|V(G)|)$

if $v = s$ **then** print "s"
else if $v.\pi = \mathrm{NIL}$ **then** print "$\nexists\ s - v$ path" // $x.\pi$ = predecessor of x in $s - x$ path
else
  PRINT_PATH($G, s, v.\pi$)
  print "v"

INIT_SINGLE_SOURCE(digraph G, sourse s) runtime $O(|V(G)|)$

for each vertex v of G **do**
  $v.d = \infty$ // $x.d$ = upper bound on distance from s to x
  $v.pi = \mathrm{NIL}$
$s.d = 0$

RELAX(vertices u, v, weight fct w) runtime $O(|1|)$

if $v.d > u.d + w(u, v)$ **then**
  $v.\pi = u$
  $v.d = u.d + w(uv)$

Properties of basic functions

digraph $G=(V,E)$, weight $w\colon E\to\mathbb{R}$, sourse s (proofs WHITEBOARD)

Lemma 3.2 Δ-inequality

$\delta(s,v)\le\delta(s.u)+w(u,v)$ *for all* $(u,v)\in E$.

Lemma 3.3 upper bound property

After call of INIT_SINGLE_SOURCE*(G, s) we always have* $v.d\ge\delta(s,v)\ \forall v\in V$ *and this property is maintained over any sequence of calls of* RELAX.
In particular, if $v.d=\delta(s,v)$, *then* $v.d$ *never changes again.*

Corollary 3.4 No-path-property

If there is no $s-v$ *path, then after call of* INIT_SINGLE_SOURCE*(G, s)* $v.d=\delta(s,v)=\infty$ *and and this property is maintained over any sequence of calls of* RELAX.

Properties of basic functions

digraph $G = (V, E)$, weight $w\colon E \to \mathbb{R}$, sourse s (proofs WHITEBOARD)

Lemma 3.5 Convergence-property

Let $P = s \rightsquigarrow u \to v$ shortest $s - v$ path in G for some $u, v \in V$. Suppose that G is initialized by INIT_SINGLE_SOURCE*(G, s) and then a sequence of calls of* RELAX *are applied that includes the call* RELAX*(u, v, w).*
If $u.d = \delta(s, u)$ at any time prior to this call, then $v.d = \delta(s, v)$ at all times after the call.

Lemma 3.6 Path-Relaxation-property

Let $P = < s = v_0, \ldots, v_k >$ any shortest $s - v_k$ path in G. If G is initialized by INIT_SINGLE_SOURCE*(G, s) and then a sequence of calls of* RELAX *are applied that includes, in order, the calls* RELAX*(v_0, v_1, w), ...,* RELAX*(v_{k-1}, v_k, w), then $v_k.d = \delta(s, v_k)$ at all times after the call.*

Bellmann-Ford-Algorithm

Solves single sourse shortest path problem

BELLMANN-FORD(digraph G, weight fct w, sourse s)

INIT_SINGLE_SOURCE(G, s)
for $i = 1, \ldots, |V| - 1$ **do**
    for every edge $(u, v) \in E$ **do**
        RELAX(u, v, w)

Bellmann-Ford-Algorithm

Solves single sourse shortest path problem

BELLMANN-FORD(digraph G, weight fct w, sourse s)

INIT_SINGLE_SOURCE(G, s)
for $i = 1, \ldots, |V| - 1$ **do**
    for every edge $(u, v) \in E$ **do**
        RELAX(u, v, w)

Theorem 3.7

BELLMANN-FORD($G = (V, E)$, w, s) with conservative weighting function $w : E \to \mathbb{R}$ correctly computes all distances from s to all $v \in V$ in $O(|V||E|)$ time.
[a shortest path can then be printed with PRINT_PATH(digraph G vertices s, v) for all $v \in V$]

WHITEBOARD: proof

Dijkstra's-Algorithm

Solves single sourse shortest path problem

DIJKSTRA(digraph G, weight fct w, sourse s)

INIT_SINGLE_SOURCE(G, s)
$S = \emptyset$
$Q = V(G)$
while $Q \neq \emptyset$ **do**
    $u =$ EXTRACT_MIN(Q)
    $S = S \cup \{u\}$
    for all $v \in N^+(u)$ **do**
        RELAX(u, v, w)

Dijkstra's-Algorithm

Solves single sourse shortest path problem

DIJKSTRA(digraph G, weight fct w, sourse s)

INIT_SINGLE_SOURCE(G, s)
$S = \emptyset$
$Q = V(G)$
while $Q \neq \emptyset$ **do**
    $u =$ EXTRACT_MIN(Q)
    $S = S \cup \{u\}$
    for all $v \in N^+(u)$ **do**
        RELAX(u, v, w)

Theorem 3.8

DIJKSTRA($G = (V, E)$, w, s) with nonnegative weighting function $w : E \to \mathbb{R}_{\geq 0}$ correctly computes all distances from s to all $v \in V$ in $O(|V| f(|V|) + |E|)$ time where $f(|V|)$ is runtime of EXTRACT_MIN*(Q).*
[a shortest path can then be printed with PRINT_PATH*(digraph G vertices s, v) for all $v \in V$]*

WHITEBOARD: proof

Dijkstra's-Algorithm

Solves single sourse shortest path problem

DIJKSTRA(digraph G, weight fct w, sourse s)

INIT_SINGLE_SOURCE(G, s)
$S = \emptyset$
$Q = V(G)$
while $Q \neq \emptyset$ **do**
    $u =$ EXTRACT_MIN(Q)
    $S = S \cup \{u\}$
    for all $v \in N^+(u)$ **do**
        RELAX(u, v, w)

Theorem 3.8

DIJKSTRA($G = (V, E)$, w, s) with nonnegative weighting function $w : E \to \mathbb{R}_{\geq 0}$ correctly computes all distances from s to all $v \in V$ in $O(|V| f(|V|) + |E|)$ time where $f(|V|)$ is runtime of EXTRACT_MIN(Q).
[a shortest path can then be printed with PRINT_PATH(digraph G vertices s, v) for all $v \in V$]

Trivially EXTRACT_MIN(Q) runs in $O(|V|)$ time. However, using efficient datastructures (Min-priority queue and Fibonacci Heap) this runtime can be improved to $O(\log_2(|V|))$ time, in which case Dikstra is faster than Bellmann-Ford

Floyd-Warshall-Algorithm

Solves many sourses shortest path problem

Later in Part “Dynamic Programming”