Graph Algorithms

I. Shortest paths

- $D = (V, A)$ directed graph, $s, t \in V$.
- A *walk* is a sequence $P = (v_0, a_1, v_1, \dots, a_k, v_k), k \geq 0$, where a_i is an arc from v_{i-1} to v_i, for $i = 1, \dots, k$.
- P is a *path*, if $v_0, \dots, v_k$ are all different.
- If $s = v_0$ and $t = v_k$, P is a *s-t walk* resp. *s-t path* of *length k* (i.e., each arc has length 1).
- The *distance* from s to t is the minimum length of any s-t path (and $+\infty$ if no s-t path exists).

Shortest paths with unit lengths

Algorithm (Breadth-first search)

Initialization: $V_0 = \{s\}$
Iteration: $V_{i+1} = \{v \in V \setminus (V_0 \cup V_1 \cup \dots \cup V_i) \mid (u, v) \in A, \text{ for some } u \in V_i\}$,
until $V_{i+1} = \emptyset$.

Running time: $O(|A|)$

- V_i is the set of nodes with distance i from s.
- The algorithm computes shortest paths from s to all reachable nodes.
- Can be described by a directed tree $T = (V', A')$ with root s such that each u-v path in T is a shortest u-v path in D.

Shortest paths with non-negative lengths

- Length function $l : A \to \mathbb{Q}_+ = \{x \in \mathbb{Q} \mid x \geq 0\}$
- For a walk $P = (v_0, a_1, v_1, \dots, a_k, v_k)$ define $l(P) = \sum_{i=1}^{k} l(a_i)$.

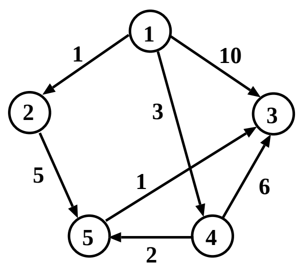
Algorithm (Dijkstra 1959)

Initialization: $U = V, f(s) = 0, f(v) = \infty$, for $v \in V \setminus \{s\}$
Iteration: Find $u \in U$ with $f(u) = \min\{f(v) \mid v \in U\}$.
For all $a = (u, v) \in A$ with $f(v) > f(u) + l(a)$ let $f(v) = f(u) + l(a)$.
Let $U \leftarrow U \setminus \{u\}$, until $U = \emptyset$.

Upon termination, $f(v)$ gives the length of a shortest path from s to v.

Running time: $O(|V|^2)$ (can be improved to $O(|A| + |V| \log |V|)$.)

Example



Iteration	u	U	$f[1]$	$f[2]$	$f[3]$	$f[4]$	$f[5]$
0	–	$\{1,2,4,3,5\}$	0	∞	∞	∞	∞
1	1	$\{2,3,4,5\}$	0	1	10	3	∞
2	2	$\{3,4,5\}$	0	1	10	3	6
3	4	$\{3,5\}$	0	1	9	3	5
4	5	$\{3\}$	0	1	6	3	5
5	3	$\{\}$	0	1	6	3	5

Application: Longest common subsequence

- Sequences $a = a_1, \dots, a_m$ and $b = b_1, \dots, b_n$
- Find the longest common subsequence of a and b (obtained by removing symbols in a or b).

Modeling as a shortest path problem

- Grid graph with nodes $(i,j), 0 \leq i \leq m, 0 \leq j \leq n$.
- Horizontal and vertical arcs of length 1.
- Diagonal arcs $((i-1,j-1),(i,j))$ of length 0, if $a_i = b_j$.

The diagonal arcs on a shortest path from $(0,0)$ to (m,n) define a longest common subsequence.

Circuits of negative length

- Consider arbitrary length functions $l : A \to \mathbb{Q}$.
- A *directed circuit* is a walk $P = (v_0, a_1, v_1, \dots, a_k, v_k)$ with $k \geq 1$ and $v_0 = v_k$ such that $v_1, \dots, v_k$ and $a_1, \dots, a_k$ are all different.
- If $D = (V,A)$ contains a directed circuit of negative length, there exist s-t walks of arbitrary small negative length.

Proposition
Let $D = (V,A)$ be a directed graph without circuits of negative length.
For any $s,t \in V$ for which there exists at least one s-t walk, there exists a shortest s-t walk, which is a path.

Shortest paths with arbitrary lengths

$D = (V,A), n = |V|, l : A \to \mathbb{Q}$.

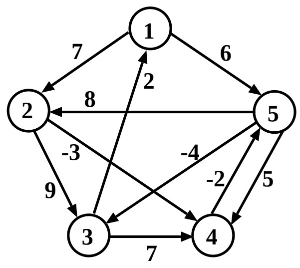
Algorithm (Bellman-Ford 1956/58)

Compute $f_0, \dots, f_n : V \to \mathbb{R} \cup \{\infty\}$ in the following way:

Initialization: $f_0(s) = 0$, $f_0(v) = \infty$, for $v \in V \setminus \{s\}$
Iteration: For $k = 1, \dots, n$ and all $v \in V$:
$$f_k(v) = \min\{f_{k-1}(v), \min_{(u,v)\in A}(f_{k-1}(u) + l(u,v))\}$$

Running time: $O(|V||A|)$

Example



Iteration k	$f_k[1]$	$f_k[2]$	$f_k[3]$	$f_k[4]$	$f_k[5]$
0	0	∞	∞	∞	∞
1	0	7	∞	∞	6
2	0	7	2	4	6
3	0	7	2	4	2
4	0	7	-2	4	2

Properties

- For each $k = 0, \dots, n$ and each $v \in V$:

  $$f_k(v) = \min\{l(P) \mid P \text{ is an } s\text{-}v \text{ walk traversing at most } k \text{ arcs}\}$$

  (by induction)

- If D contains no circuits of negative length, $f_{n-1}(v)$ is the length of a shortest path from s to v.

Finding an explicit shortest path

- When computing $f_0, \dots, f_n$ determine a predecessor function $p : V \to V$ by setting $p(v) = u$ whenever $f_{k+1}(v) = f_k(u) + l(u, v)$.
- At termination, $v, p(v), p(p(v)), \dots, s$ gives the reverse of a shortest s-v path.

Theorem
Given $D = (V, A), s, t \in V$ and $l : A \to \mathbb{Q}$ such that D contains no circuit of negative length, a shortest s-t path can be found in time $O(|V||A|)$.

Remark
D contains a circuit of negative length reachable from s if and only if $f_n(v) \neq f_{n-1}(v)$, for some $v \in V$.

NP-completeness

For directed graphs containing circuits of negative length, the problem becomes NP-complete:

Theorem
The decision problem

> *Input:* Directed graph $D = (V, A)$, $s, t \in V$, $l : A \to \mathbb{Z}$, $L \in \mathbb{Z}$
> *Question:* Does there exist an s-t path P with $l(P) \leq L$?

is NP-complete.

Corollary
The shortest path problem with arbitrary lengths is NP-complete.
The longest path problem with non-negative lengths is NP-complete.

Application: Knapsack problem

- Knapsack, volume 8, 5 articles

Article i	Volume a_i	Value c_i
1	5	4
2	3	7
3	2	3
4	2	5
5	1	4

- Objective: Select articles fitting into the knapsack and maximizing the total value.

Possible models

- *Shortest path model*
  - Directed graph with nodes $(i,x), 0 \leq i \leq 6,\ 0 \leq x \leq 8$.
  - Arcs from $(i-1,x)$ to (i,x) resp. $(i,x+a_i)$ of length 0 resp. $-c_i$, for $0 \leq i \leq 5$.
  - Arcs from $(5,x)$ to $(6,8)$ of length 0, for $0 \leq x \leq 6$.
  - A shortest path from $(0,0)$ to $(6,8)$ gives an optimal solution.

  $\rightsquigarrow$ *pseudo-polynomial algorithm*

- *Linear 0-1 model*

$$\max\{4x_1+7x_2+3x_3+5x_4+4x_5 \mid 5x_1+3x_2+2x_3+2x_4+x_5 \leq 8, x_1,\ldots,x_5 \in \{0,1\}\}$$