Datastructures and Algorithms

Single Source Shortest Paths Algorithms (A*, Bellman-Ford),
All Pairs Shortest Paths Algorithms (Floyd-Warshall)

Adel Gavranović — ETH Zürich — 2025

Overview

Learning Objectives
Recap Theory
- Shortest Paths
- All Pairs Shortest Paths

Code-Expert Example

n.ethz.ch/~agavranovic

Material

Webpage

Mail

1. Follow-up

Follow-up from last session

Follow-up from last session

- Welcome back!

2. Feedback regarding **code** expert

General things regarding **code** expert

General things regarding code expert

- If anyone needs more XP for unlocking the Bonus Exercises, let me know via (your ETH) mail!

Any questions regarding code expert on your part?

3. Learning Objectives

Objectives

Understand how and why…

- ☐ …the A* algorithm,
- ☐ …the Bellman-Ford algorithm,
- ☐ …and the Floyd-Warshall algorithm

work and when to use which.

4. Summary

Getting on the same page

Getting on the same page

- What did you cover in the lectures?
- Did you cover Minimum Spanning Trees?
- Did you cover Directed Acylic Graphs?
- Did you cover Topological Sorting?

5. Recap Theory

5. Recap Theory

5.1. Shortest Paths

A*-Algorithm($G, s, t, \hat{h}$)

Input: Positively weighted Graph $G = (V, E, c)$, starting point $s \in V$, end point $t \in V$, estimate $\widehat{h}(v) \leq \delta(v, t)$

Output: Existence and value of a shortest path from s to t

foreach $u \in V$ **do**
    $d[u] \leftarrow \infty$; $\widehat{f}[u] \leftarrow \infty$; $\pi[u] \leftarrow$ null

$d[s] \leftarrow 0$; $\widehat{f}[s] \leftarrow \widehat{h}(s)$; $U \leftarrow \{s\}$
while $U \neq \emptyset$ **do**
    $u \leftarrow$ ExtractMin$_{\widehat{f}}(U)$
    if $u = t$ **then return** success
    foreach $v \in N^+(u)$ with $d[v] > d[u] + c((u, v))$ **do**
        $d[v] \leftarrow d[u] + c((u, v))$; $\widehat{f}[v] \leftarrow d[v] + \widehat{h}(v)$; $\pi[v] \leftarrow u$
        $U \leftarrow U \cup \{v\}$
return failure

←

Properties

- The A*-Algorithm is an extension of the Dijkstra algortihm by a distance heuristic $\hat{h}$.

Properties

- The A*-Algorithm is an extension of the Dijkstra algortihm by a distance heuristic $\hat{h}$.
- A* is Dijkstra if $\hat{h} \equiv 0$

Properties

- The A*-Algorithm is an extension of the Dijkstra algortihm by a distance heuristic $\hat{h}$.
- A* is Dijkstra if $\hat{h} \equiv 0$
- Underestimation: $\forall v \in V : \hat{h}(v) \leq \delta(v, t)$
  If $\hat{h}$ underestimates the real distance, the algorithm works correctly.

Properties

- The A*-Algorithm is an extension of the Dijkstra algortihm by a distance heuristic $\hat{h}$.
- A* is Dijkstra if $\hat{h} \equiv 0$
- Underestimation: $\forall v \in V : \hat{h}(v) \leq \delta(v, t)$
  If $\hat{h}$ underestimates the real distance, the algorithm works correctly.
- Monotonicity: $\forall (u, v) \in E : \hat{h}(u) \leq c((u, v)) + \hat{h}(v)$
  If $\hat{h}$ is monotone in addition, then the algorithm works efficiently.

General Weighted Graphs

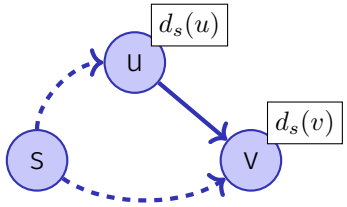
Relax(u, v) $(u, v \in V, (u, v) \in E)$

if $d_s(v) > d_s(u) + c(u, v)$ **then**

  $d_s(v) \leftarrow d_s(u) + c(u, v)$

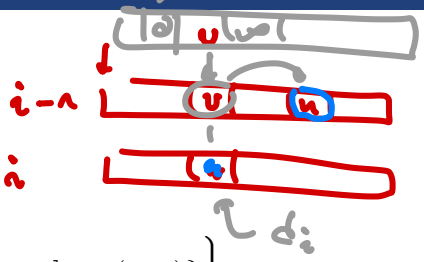
  return true

return false



Problem: cycles with negative weights can shorten the path, a shortest path is not guaranteed to exist.

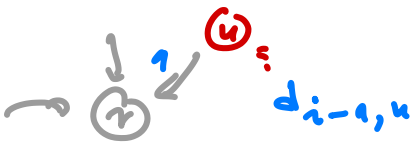
Dynamic Programming Approach (Bellman)



Induction over number of edges $d_s[i, v]$:
Shortest path from s to v via maximally i edges.

$$d_s[i, v] = \min\left\{ d_s[i-1, v], \min_{(u,v)\in E} \{d_s[i-1, u] + c(u, v)\} \right\}$$

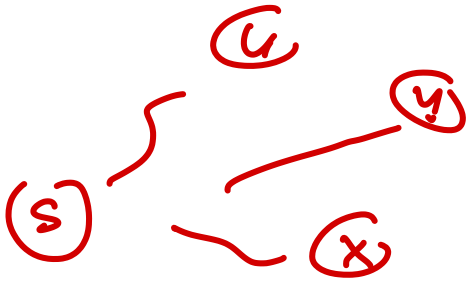
$$d_s[0, s] = 0, d_s[0, v] = \infty \; \forall v \neq s.$$



Algorithm Bellman-Ford(G, s)

Input: Graph $G = (V, E, c)$, starting point $s \in V$
Output: If return value true, minimal weights d for all shortest paths from s, otherwise no shortest path.

foreach $u \in V$ **do**
    $d_s[u] \leftarrow \infty$; $\pi_s[u] \leftarrow$ null
$d_s[s] \leftarrow 0$;
for $i \leftarrow 1$ **to** $|V|$ **do**
    $f \leftarrow$ false
    foreach $(u, v) \in E$ **do**
        $f \leftarrow f \vee \mathrm{Relax}(u, v)$
    if $f =$ false **then return** true
return false;



Quiz: Single Source Shortest Paths

$n := |V|, m := |E|$

problem	method	runtime	dense $m \in \mathcal{O}(n^2)$	sparse $m \in \mathcal{O}(n)$
$c \equiv 1$				
DAG[1] ($c \in \mathbb{R}$)				
$c \geq 0$				
general ($c \in \mathbb{R}$)				

[1] Directed Acyclic Graph

Quiz: Single Source Shortest Paths

$n := |V|, m := |E|$

problem	method	runtime	dense $m \in \mathcal{O}(n^2)$	sparse $m \in \mathcal{O}(n)$
$c \equiv 1$	BFS			
DAG[1] ($c \in \mathbb{R}$)				
$c \geq 0$				
general ($c \in \mathbb{R}$)				

[1] Directed Acyclic Graph

Quiz: Single Source Shortest Paths

$n := |V|, m := |E|$

problem	method	runtime	dense $m \in \mathcal{O}(n^2)$	sparse $m \in \mathcal{O}(n)$
$c \equiv 1$	BFS			
DAG[1] ($c \in \mathbb{R}$)	Top-Sort			
$c \geq 0$				
general ($c \in \mathbb{R}$)				

[1] Directed Acyclic Graph

Quiz: Single Source Shortest Paths

$n := |V|, m := |E|$

topological sort (in $\mathcal{O}(n+m)$)

problem	method	runtime	dense $m \in \mathcal{O}(n^2)$	sparse $m \in \mathcal{O}(n)$
$c \equiv 1$	BFS			
DAG[1] ($c \in \mathbb{R}$)	Top-Sort			
$c \geq 0$	Dijkstra			
general ($c \in \mathbb{R}$)				

[1] Directed Acyclic Graph

Quiz: Single Source Shortest Paths

$n := |V|, m := |E|$

problem	method	runtime	dense $m \in \mathcal{O}(n^2)$	sparse $m \in \mathcal{O}(n)$
$c \equiv 1$	BFS			
DAG[1] ($c \in \mathbb{R}$)	Top-Sort			
$c \geq 0$	Dijkstra			
general ($c \in \mathbb{R}$)	Bellman-Ford			

[1] Directed Acyclic Graph

Quiz: Single Source Shortest Paths

$n := |V|, m := |E|$

problem	method	runtime	dense $m \in \mathcal{O}(n^2)$	sparse $m \in \mathcal{O}(n)$
$c \equiv 1$	BFS	$\mathcal{O}(m+n)$		
DAG[1] ($c \in \mathbb{R}$)	Top-Sort			
$c \geq 0$	Dijkstra			
general ($c \in \mathbb{R}$)	Bellman-Ford			

[1] Directed Acyclic Graph

Quiz: Single Source Shortest Paths

$n := |V|, m := |E|$

problem	method	runtime	dense $m \in \mathcal{O}(n^2)$	sparse $m \in \mathcal{O}(n)$
$c \equiv 1$	BFS	$\mathcal{O}(m+n)$		
DAG[1] ($c \in \mathbb{R}$)	Top-Sort	$\mathcal{O}(m+n)$		
$c \geq 0$	Dijkstra			
general ($c \in \mathbb{R}$)	Bellman-Ford			

[1] Directed Acyclic Graph

Quiz: Single Source Shortest Paths

$n := |V|, m := |E|$

problem	method	runtime	dense $m \in \mathcal{O}(n^2)$	sparse $m \in \mathcal{O}(n)$
$c \equiv 1$	BFS	$\mathcal{O}(m+n)$		
DAG[1] ($c \in \mathbb{R}$)	Top-Sort	$\mathcal{O}(m+n)$		
$c \geq 0$	Dijkstra	$\mathcal{O}((m+n)\log n)$		
general ($c \in \mathbb{R}$)	Bellman-Ford			

[1] Directed Acyclic Graph

Quiz: Single Source Shortest Paths

$n := |V|, m := |E|$

problem	method	runtime	dense $m \in \mathcal{O}(n^2)$	sparse $m \in \mathcal{O}(n)$
$c \equiv 1$	BFS	$\mathcal{O}(m+n)$		
DAG[1] ($c \in \mathbb{R}$)	Top-Sort	$\mathcal{O}(m+n)$		
$c \geq 0$	Dijkstra	$\mathcal{O}((m+n)\log n)$		
general ($c \in \mathbb{R}$)	Bellman-Ford	$\mathcal{O}(m \cdot n)$		

[1] Directed Acyclic Graph

Quiz: Single Source Shortest Paths

$n := |V|, m := |E|$

problem	method	runtime	dense $m \in \mathcal{O}(n^2)$	sparse $m \in \mathcal{O}(n)$
$c \equiv 1$	BFS	$\mathcal{O}(m+n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$
DAG[1] ($c \in \mathbb{R}$)	Top-Sort	$\mathcal{O}(m+n)$		
$c \geq 0$	Dijkstra	$\mathcal{O}((m+n)\log n)$		
general ($c \in \mathbb{R}$)	Bellman-Ford	$\mathcal{O}(m \cdot n)$		

[1] Directed Acyclic Graph

Quiz: Single Source Shortest Paths

$n := |V|, m := |E|$

problem	method	runtime	dense $m \in \mathcal{O}(n^2)$	sparse $m \in \mathcal{O}(n)$
$c \equiv 1$	BFS	$\mathcal{O}(m+n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$
DAG[1] ($c \in \mathbb{R}$)	Top-Sort	$\mathcal{O}(m+n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$
$c \geq 0$	Dijkstra	$\mathcal{O}((m+n)\log n)$		
general ($c \in \mathbb{R}$)	Bellman-Ford	$\mathcal{O}(m \cdot n)$		

[1] Directed Acyclic Graph

Quiz: Single Source Shortest Paths

$n := |V|, m := |E|$

problem	method	runtime	dense $m \in \mathcal{O}(n^2)$	sparse $m \in \mathcal{O}(n)$
$c \equiv 1$	BFS	$\mathcal{O}(m+n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$
DAG[1] ($c \in \mathbb{R}$)	Top-Sort	$\mathcal{O}(m+n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$
$c \geq 0$	Dijkstra	$\mathcal{O}((m+n)\log n)$	$\mathcal{O}(n^2 \log n)$	$\mathcal{O}(n \log n)$
general ($c \in \mathbb{R}$)	Bellman-Ford	$\mathcal{O}(m \cdot n)$		

[1] Directed Acyclic Graph

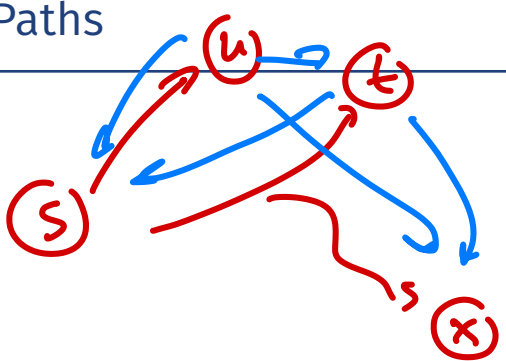
Quiz: Single Source Shortest Paths

$n := |V|, m := |E|$

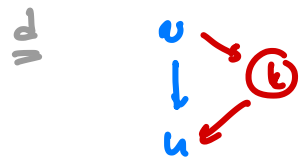
problem	method	runtime	dense $m \in \mathcal{O}(n^2)$	sparse $m \in \mathcal{O}(n)$
$c \equiv 1$	BFS	$\mathcal{O}(m+n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$
DAG[1] ($c \in \mathbb{R}$)	Top-Sort	$\mathcal{O}(m+n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$
$c \geq 0$	Dijkstra	$\mathcal{O}((m+n)\log n)$	$\mathcal{O}(n^2 \log n)$	$\mathcal{O}(n \log n)$
general ($c \in \mathbb{R}$)	Bellman-Ford	$\mathcal{O}(m \cdot n)$	$\mathcal{O}(n^3)$	$\mathcal{O}(n^2)$

[1] Directed Acyclic Graph

5.2. All Pairs Shortest Paths



DP Algorithm Floyd-Warshall(G)



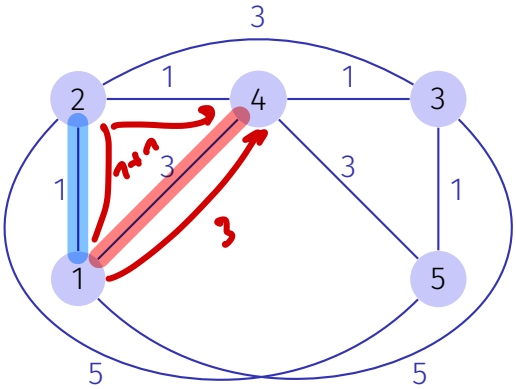
Input: Acyclic Graph $G = (V, E, c)$
Output: Minimal weights of all paths d

$d^0 \leftarrow c$
for $k \leftarrow 1$ **to** $|V|$ **do**
  for $i \leftarrow 1$ **to** $|V|$ **do**
    for $j \leftarrow 1$ **to** $|V|$ **do**
      $d^k(v_i, v_j) = \min\{d^{k-1}(v_i, v_j), d^{k-1}(v_i, v_k) + d^{k-1}(v_k, v_j)\}$

Runtime: $\Theta(|V|^3)$

Remark: Algorithm can be executed with a single matrix d (in place).

Example



adjacency matrix $M = c$

	1	2	3	4	5
1	0	1	5	3	∞
2	1	0	3	1	5
3	5	3	0	1	1
4	3	1	1	0	3
5	∞	5	1	3	0

Example

$k = 1$

0	1	5	3	∞
1	0	3	1	5
5	3	0	1	1
3	1	1	0	3
∞	5	1	3	0

d^0

0	1	5	3	∞
1	0	3	1	5
5	3	0	1	1
3	1	1	0	3
∞	5	1	3	0

d^1

Example

$k = 1$

0	1	5	3	∞
1	0	3	1	5
5	3	0	1	1
3	1	1	0	3
∞	5	1	3	0

d^0

$k = 2$

0	1	5	3	∞
1	0	3	1	5
5	3	0	1	1
3	1	1	0	3
∞	5	1	3	0

d^1

4 5

0	1	5	3	∞
1	0	3	1	5
5	3	0	1	1
3	1	1	0	3
∞	5	1	3	0

d^1

0	1	5	3	∞
1	0	3	1	5
5	3	0	1	1
3	1	1	0	3
∞	5	1	3	0

d^2

←

Example

$k = 1$

0	1	5	3	∞
1	0	3	1	5
5	3	0	1	1
3	1	1	0	3
∞	5	1	3	0

d^0

$k = 2$ 2

0	1	5	4	∞
1	0	3	1	5
5	3	0	1	1
3	1	1	0	3
∞	5	1	3	0

d^1

0	1	5	3	∞
1	0	3	1	5
5	3	0	1	1
3	1	1	0	3
∞	5	1	3	0

d^1

0	1	4	3	∞
1	0	3	1	5
5	3	0	1	1
3	1	1	0	3
∞	5	1	3	0

d^2

Example

$k = 1$

0	1	5	3	∞
1	0	3	1	5
5	3	0	1	1
3	1	1	0	3
∞	5	1	3	0

d^0

$k = 2$

0	1	5	3	∞
1	0	3	1	5
5	3	0	1	1
3	1	1	0	3
∞	5	1	3	0

d^1

0	1	5	3	∞
1	0	3	1	5
5	3	0	1	1
3	1	1	0	3
∞	5	1	3	0

d^1

0	1	**4**	**2**	∞
1	0	3	1	5
5	3	0	1	1
3	1	1	0	3
∞	5	1	3	0

d^2

Example

$k = 1$

0	1	5	3	∞
1	0	3	1	5
5	3	0	1	1
3	1	1	0	3
∞	5	1	3	0

d^0

$k = 2$

0	1	5	3	∞
1	0	3	1	5
5	3	0	1	1
3	1	1	0	3
∞	5	1	3	0

d^1

0	1	5	3	∞
1	0	3	1	5
5	3	0	1	1
3	1	1	0	3
∞	5	1	3	0

d^1

0	1	**4**	**2**	**6**
1	0	3	1	5
5	3	0	1	1
3	1	1	0	3
∞	5	1	3	0

d^2

Example

$k = 1$

0	1	5	3	∞
1	0	3	1	5
5	3	0	1	1
3	1	1	0	3
∞	5	1	3	0

d^0

$k = 2$

0	1	5	3	∞
1	0	3	1	5
5	3	0	1	1
3	1	1	0	3
∞	5	1	3	0

d^1

0	1	5	3	∞
1	0	3	1	5
5	3	0	1	1
3	1	1	0	3
∞	5	1	3	0

d^1

0	1	**4**	**2**	**6**
1	0	3	1	5
4	3	0	1	1
2	1	1	0	3
6	5	1	3	0

d^2

←

Example

$k = 1$

0	1	5	3	∞
1	0	3	1	5
5	3	0	1	1
3	1	1	0	3
∞	5	1	3	0

d^0

$k = 2$

0	1	5	3	∞
1	0	3	1	5
5	3	0	1	1
3	1	1	0	3
∞	5	1	3	0

d^1

$k = 3$

0	1	4	2	6
1	0	3	1	5
4	3	0	1	1
2	1	1	0	3
6	5	1	3	0

d^2

0	1	5	3	∞
1	0	3	1	5
5	3	0	1	1
3	1	1	0	3
∞	5	1	3	0

d^1

0	1	**4**	**2**	**6**
1	0	3	1	5
4	3	0	1	1
2	1	1	0	3
6	5	1	3	0

d^2

←

Example

$k = 1$

0	1	5	3	∞
1	0	3	1	5
5	3	0	1	1
3	1	1	0	3
∞	5	1	3	0

d^0

$k = 2$

0	1	5	3	∞
1	0	3	1	5
5	3	0	1	1
3	1	1	0	3
∞	5	1	3	0

d^1

$k = 3$

0	1	4	2	6
1	0	3	1	5
4	3	0	1	1
2	1	1	0	3
6	5	1	3	0

d^2

0	1	5	3	∞
1	0	3	1	5
5	3	0	1	1
3	1	1	0	3
∞	5	1	3	0

d^1

0	1	**4**	**2**	**6**
1	0	3	1	5
4	3	0	1	1
2	1	1	0	3
6	5	1	3	0

d^2

0	1	4	2	**5**
1	0	3	1	**4**
4	3	0	1	1
2	1	1	0	**2**
5	**4**	1	**2**	0

d^3

←

Example

$k = 1$

0	1	5	3	∞
1	0	3	1	5
5	3	0	1	1
3	1	1	0	3
∞	5	1	3	0

d^0

$k = 2$

0	1	5	3	∞
1	0	3	1	5
5	3	0	1	1
3	1	1	0	3
∞	5	1	3	0

d^1

$k = 3$

0	1	4	2	6
1	0	3	1	5
4	3	0	1	1
2	1	1	0	3
6	5	1	3	0

d^2

$k = 4$

0	1	4	2	5
1	0	3	1	4
4	3	0	1	1
2	1	1	0	2
5	4	1	2	0

d^3

0	1	5	3	∞
1	0	3	1	5
5	3	0	1	1
3	1	1	0	3
∞	5	1	3	0

d^1

0	1	**4**	**2**	**6**
1	0	3	1	5
4	3	0	1	1
2	1	1	0	3
6	5	1	3	0

d^2

0	1	4	2	**5**
1	0	3	1	**4**
4	3	0	1	1
2	1	1	0	**2**
5	**4**	1	**2**	0

d^3

←

Example

$k = 1$

0	1	5	3	∞
1	0	3	1	5
5	3	0	1	1
3	1	1	0	3
∞	5	1	3	0

d^0

$k = 2$

0	1	5	3	∞
1	0	3	1	5
5	3	0	1	1
3	1	1	0	3
∞	5	1	3	0

d^1

$k = 3$

0	1	4	2	6
1	0	3	1	5
4	3	0	1	1
2	1	1	0	3
6	5	1	3	0

d^2

$k = 4$

0	1	4	2	5
1	0	3	1	4
4	3	0	1	1
2	1	1	0	2
5	4	1	2	0

d^3

0	1	5	3	∞
1	0	3	1	5
5	3	0	1	1
3	1	1	0	3
∞	5	1	3	0

d^1

0	1	**4**	**2**	**6**
1	0	3	1	5
4	3	0	1	1
2	1	1	0	3
6	5	1	3	0

d^2

0	1	4	2	**5**
1	0	3	1	**4**
4	3	0	1	1
2	1	1	0	**2**
5	**4**	1	**2**	0

d^3

0	1	**3**	2	**4**
1	0	**2**	1	**3**
3	**2**	0	1	1
2	1	1	0	2
4	**3**	1	2	0

d^4

←

Example

$k = 1$

0	1	5	3	∞
1	0	3	1	5
5	3	0	1	1
3	1	1	0	3
∞	5	1	3	0

d^0

$k = 2$

0	1	5	3	∞
1	0	3	1	5
5	3	0	1	1
3	1	1	0	3
∞	5	1	3	0

d^1

$k = 3$

0	1	4	2	6
1	0	3	1	5
4	3	0	1	1
2	1	1	0	3
6	5	1	3	0

d^2

$k = 4$

0	1	4	2	5
1	0	3	1	4
4	3	0	1	1
2	1	1	0	2
5	4	1	2	0

d^3

$k = 5$

0	1	3	2	4
1	0	2	1	3
3	2	0	1	1
2	1	1	0	2
4	3	1	2	0

d^4

0	1	5	3	∞
1	0	3	1	5
5	3	0	1	1
3	1	1	0	3
∞	5	1	3	0

d^1

0	1	**4**	**2**	**6**
1	0	3	1	5
4	3	0	1	1
2	1	1	0	3
6	5	1	3	0

d^2

0	1	4	2	**5**
1	0	3	1	**4**
4	3	0	1	1
2	1	1	0	**2**
5	**4**	1	**2**	0

d^3

0	1	**3**	2	**4**
1	0	**2**	1	**3**
3	**2**	0	1	1
2	1	1	0	2
4	**3**	1	2	0

d^4

0	1	3	2	4
1	0	2	1	3
3	2	0	1	1
2	1	1	0	2
4	3	1	2	0

d^5

←

Shortest Path for each Pair?

M					$D := d^5$				
0	1	5	3	∞	0	1	3	2	4
1	0	3	1	5	1	0	2	1	3
5	3	0	1	1	3	2	0	1	1
3	1	1	0	3	2	1	1	0	2
∞	5	1	3	0	4	3	1	2	0

Question: Can we use the computed matrix D to determine the shortest path between each pair of nodes?

Shortest Path for each Pair?

M					$D := d^5$									
0	1	5	3	∞	0	1	3	2	4	0	1	3	2	4
1	0	3	1	5	1	0	2	1	3	1	0	2	1	3
5	3	0	1	1	3	2	0	1	1	3	2	0	1	1
3	1	1	0	3	2	1	1	0	2	2	1	1	0	2
∞	5	1	3	0	4	3	1	2	0	4	3	1	2	0

Question: Can we use the computed matrix D to determine the shortest path between each pair of nodes?

Direct connections $i \to j$ where $M[i,j] = D[i,j]$ (cf markings in D' above)

Shortest Path for each Pair?

M					$D := d^5$										D''				
0	1	5	3	∞	0	1	3	2	4	0	1	3	2	4	0	1	3	2	4
1	0	3	1	5	1	0	2	1	3	1	0	2	1	3	1	0	2	1	3
5	3	0	1	1	3	2	0	1	1	3	2	0	1	1	3	2	0	1	1
3	1	1	0	3	2	1	1	0	2	2	1	1	0	2	2	1	1	0	2
∞	5	1	3	0	4	3	1	2	0	4	3	1	2	0	4	3	1	2	0

Question: Can we use the computed matrix D to determine the shortest path between each pair of nodes?

Direct connections $i \to j$ where $M[i,j] = D[i,j]$ (cf markings in D' above)

Could try to run the algorithm backwards. Example $1 \to 3$ above in D''. Find, with decreasing k, the first fitting candidate.

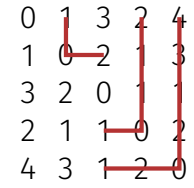
Shortest Path for each Pair?

M				
0	1	5	3	∞
1	0	3	1	5
5	3	0	1	1
3	1	1	0	3
∞	5	1	3	0

$D := d^5$				
0	1	3	2	4
1	0	2	1	3
3	2	0	1	1
2	1	1	0	2
4	3	1	2	0

0	1	3	2	4
1	0	2	1	3
3	2	0	1	1
2	1	1	0	2
4	3	1	2	0

D''				
0	1	3	2	4
1	0	2	1	3
3	2	0	1	1
2	1	1	0	2
4	3	1	2	0



Question: Can we use the computed matrix D to determine the shortest path between each pair of nodes?

Direct connections $i \to j$ where $M[i,j] = D[i,j]$ (cf markings in D' above)

Could try to run the algorithm backwards. Example $1 \to 3$ above in D''. Find, with decreasing k, the first fitting candidate.

Complicated and inefficient.

Idea

Idea

Memorize the best k in the algorithm for each node pair (i, j) in a matrix M.

Idea

Memorize the best k in the algorithm for each node pair (i, j) in a matrix M.
Start with matrix of existing direct connections (edges)

Example

$k = 1$

D

0	1	5	3	∞
1	0	3	1	5
5	3	0	1	1
3	1	1	0	3
∞	5	1	3	0

M

1	2	3	4	-
1	2	3	4	5
1	2	3	4	5
1	2	3	4	5
-	2	3	4	5

$k = 2$

D

0	1	**4**	**2**	**6**
1	0	3	1	5
4	3	0	1	1
2	1	1	0	3
6	5	1	3	0

M

1	2	**2**	**2**	**2**
1	2	3	4	5
2	2	3	4	5
2	2	3	4	5
2	2	3	4	5

$k = 3$

D

0	1	4	2	**5**
1	0	3	1	**4**
4	3	0	1	1
2	1	1	0	**2**
5	**4**	1	**2**	0

M

1	2	2	2	**3**
1	2	3	4	**3**
2	2	3	4	5
2	2	3	4	**3**
3	**3**	**3**	**3**	5

$k = 4$

D

0	1	**3**	2	**4**
1	0	**2**	1	**3**
3	**2**	0	1	1
2	1	1	0	2
4	**3**	1	2	0

M

1	2	**4**	2	**4**
1	2	**4**	**4**	**4**
4	**4**	3	4	5
2	2	3	4	3
4	**4**	3	3	5

$k = 5$

D

0	1	3	2	4
1	0	2	1	3
3	2	0	1	1
2	1	1	0	2
4	3	1	2	0

M

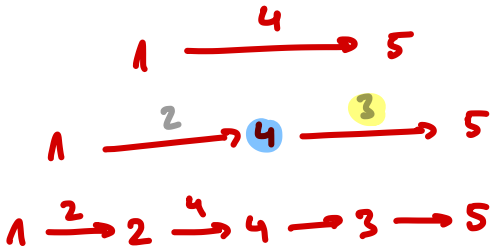
1	2	4	2	4
1	2	4	4	4
4	4	3	4	5
2	2	3	4	3
4	4	3	3	5

Example

M

	1	2	3	4	5
1	1	2	4	2	4
2	1	2	4	4	4
3	4	4	3	4	5
4	2	2	3	4	3
5	4	4	3	3	5

How to read this matrix M?



Example

M

	1	2	3	4	5
1	1	2	4	2	4
2	1	2	4	4	4
3	4	4	3	4	5
4	2	2	3	4	3
5	4	4	3	3	5

How to read this matrix M? Example path $1 \to 5$:

←

Example

M

	1	2	3	4	5
1	1	2	4	2	4
2	1	2	4	4	4
3	4	4	3	4	5
4	2	2	3	4	3
5	4	4	3	3	5

How to read this matrix M? Example path $1 \to 5$:

- Path $1 \to 5$ goes via node 4.

Example

M

	1	2	3	4	5
1	1	2	4	2	4
2	1	2	4	4	4
3	4	4	3	4	5
4	2	2	3	4	3
5	4	4	3	3	5

How to read this matrix M? Example path $1 \to 5$:

- Path $1 \to 5$ goes via node 4.
- Path $1 \to 4$ goes via node 2.

Example

M

	1	2	3	4	5
1	1	2	4	2	4
2	1	2	4	4	4
3	4	4	3	4	5
4	2	2	3	4	3
5	4	4	3	3	5

How to read this matrix M? Example path $1 \to 5$:

- Path $1 \to 5$ goes via node 4.
- Path $1 \to 4$ goes via node 2.
- Path $4 \to 5$ goes via node 3.

Example

M

	1	2	3	4	5
1	1	2	4	2	4
2	1	2	4	4	4
3	4	4	3	4	5
4	2	2	3	4	3
5	4	4	3	3	5

How to read this matrix M? Example path $1 \to 5$:

- Path $1 \to 5$ goes via node 4.
- Path $1 \to 4$ goes via node 2.
- Path $4 \to 5$ goes via node 3.
- Paths $1 \to 2$ and $2 \to 4$ are direct.

Example

M

	1	2	3	4	5
1	1	2	4	2	4
2	1	2	4	4	4
3	4	4	3	4	5
4	2	2	3	4	3
5	4	4	3	3	5

How to read this matrix M? Example path $1 \to 5$:

- Path $1 \to 5$ goes via node 4.
- Path $1 \to 4$ goes via node 2.
- Path $4 \to 5$ goes via node 3.
- Paths $1 \to 2$ and $2 \to 4$ are direct.
- Paths $4 \to 3$ and $3 \to 5$ are direct.

Example

M

	1	2	3	4	5
1	1	2	4	2	4
2	1	2	4	4	4
3	4	4	3	4	5
4	2	2	3	4	3
5	4	4	3	3	5

How to read this matrix M? Example path $1 \to 5$:

- Path $1 \to 5$ goes via node 4.
- Path $1 \to 4$ goes via node 2.
- Path $4 \to 5$ goes via node 3.
- Paths $1 \to 2$ and $2 \to 4$ are direct.
- Paths $4 \to 3$ and $3 \to 5$ are direct.

Example

M

	1	2	3	4	5
1	1	2	4	2	4
2	1	2	4	4	4
3	4	4	3	4	5
4	2	2	3	4	3
5	4	4	3	3	5

How to read this matrix M? Example path $1 \to 5$:

- Path $1 \to 5$ goes via node 4.
- Path $1 \to 4$ goes via node 2.
- Path $4 \to 5$ goes via node 3.
- Paths $1 \to 2$ and $2 \to 4$ are direct.
- Paths $4 \to 3$ and $3 \to 5$ are direct.

Overall

$$1 \overset{4}{\to} 5 \quad \Rightarrow \quad 1 \overset{2}{\to} 4 \overset{3}{\to} 5 \quad \Rightarrow \quad 1 \to 2 \to 4 \to 3 \to 5$$

Example

M

	1	2	3	4	5
1	1	2	4	2	4
2	1	2	4	4	4
3	4	4	3	4	5
4	2	2	3	4	3
5	4	4	3	3	5

How to read this matrix M? Example path $1 \to 5$:

- Path $1 \to 5$ goes via node 4.
- Path $1 \to 4$ goes via node 2.
- Path $4 \to 5$ goes via node 3.
- Paths $1 \to 2$ and $2 \to 4$ are direct.
- Paths $4 \to 3$ and $3 \to 5$ are direct.

Overall

$$1 \overset{4}{\to} 5 \quad \Rightarrow \quad 1 \overset{2}{\to} 4 \overset{3}{\to} 5 \quad \Rightarrow \quad 1 \to 2 \to 4 \to 3 \to 5$$

Reconstruction via Recursion.

Example

M

	1	2	3	4	5
1	1	2	4	2	4
2	1	2	4	4	4
3	4	4	3	4	5
4	2	2	3	4	3
5	4	4	3	3	5

How to read this matrix M? Example path $1 \to 5$:

- Path $1 \to 5$ goes via node 4.
- Path $1 \to 4$ goes via node 2.
- Path $4 \to 5$ goes via node 3.
- Paths $1 \to 2$ and $2 \to 4$ are direct.
- Paths $4 \to 3$ and $3 \to 5$ are direct.

Overall

$$1 \overset{4}{\to} 5 \quad \Rightarrow \quad 1 \overset{2}{\to} 4 \overset{3}{\to} 5 \quad \Rightarrow \quad 1 \to 2 \to 4 \to 3 \to 5$$

Reconstruction via Recursion. Alternative?

Example

M

	1	2	3	4	5
1	1	2	4	2	4
2	1	2	4	4	4
3	4	4	3	4	5
4	2	2	3	4	3
5	4	4	3	3	5

How to read this matrix M? Example path $1 \to 5$:

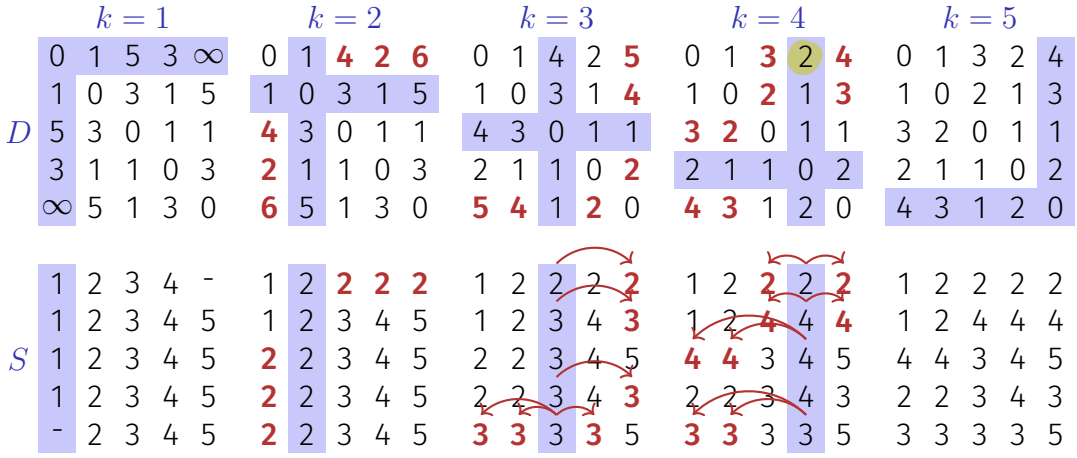
- Path $1 \to 5$ goes via node 4.
- Path $1 \to 4$ goes via node 2.
- Path $4 \to 5$ goes via node 3.
- Paths $1 \to 2$ and $2 \to 4$ are direct.
- Paths $4 \to 3$ and $3 \to 5$ are direct.

Overall

$$1 \overset{4}{\to} 5 \quad \Rightarrow \quad 1 \overset{2}{\to} 4 \overset{3}{\to} 5 \quad \Rightarrow \quad 1 \to 2 \to 4 \to 3 \to 5$$

Reconstruction via Recursion. Alternative? Store successor in the algorithm

Example



Comparison of the approaches

Algorithm			Runtime
Dijkstra (Heap)	$c_v \geq 0$	1:n	$\mathcal{O}(\lvert E\rvert \log \lvert V\rvert)$
Dijkstra (Fibonacci-Heap)	$c_v \geq 0$	1:n	$\mathcal{O}(\lvert E\rvert + \lvert V\rvert \log \lvert V\rvert)$ *
Bellman-Ford		1:n	$\mathcal{O}(\lvert E\rvert \cdot \lvert V\rvert)$
Floyd-Warshall		n:n	$\Theta(\lvert V\rvert^3)$
Johnson		n:n	$\mathcal{O}(\lvert V\rvert \cdot \lvert E\rvert \cdot \log \lvert V\rvert)$
Johnson (Fibonacci-Heap)		n:n	$\mathcal{O}(\lvert V\rvert^2 \log \lvert V\rvert + \lvert V\rvert \cdot \lvert E\rvert)$ *

* amortized

Johnson (not explained this year) is better than Floyd-Warshall only for sparse graphs ($|E| \approx \Theta(|V|)$).

6. Code-Expert Example

Code-Example

under "code examples" on

'Sliding Puzzle' on Code-Expert

7. Outro

General Questions?

See you next time!

Have a nice week!