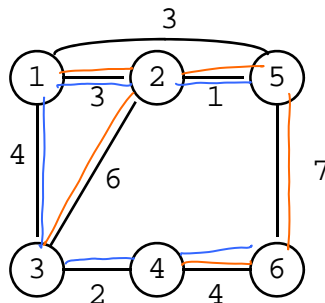


Minimum Cost Spanning Tree

$G = (N, E)$ Connected, undirected graph
real numbers

Cost function $c: E \rightarrow \mathbb{R}$ give edge weights

A spanning tree is a graph $S = (N, E')$
 where $E \supseteq E'$ and S is a tree



$1 + 3 + 6 + 7 + 4 = 21$
 $1 + 3 + 4 + 2 + 4 = 14$

Cost of a Spanning Tree

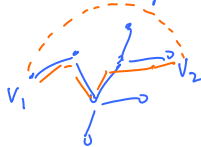
Cost of a spanning tree $S = (N, E')$ is

$$\sum_{e \in E'} c(e)$$

Want minimum cost

Lemma: Let $S = (N, E')$ be a spanning tree
 for $G = (N, E)$. Then

- a. for v_1, v_2 in N
there a single simple path between v_1 and v_2
- b. adding an edge from $E - E'$ to S
create a unique simple cycle



Spanning Forest

A set of trees

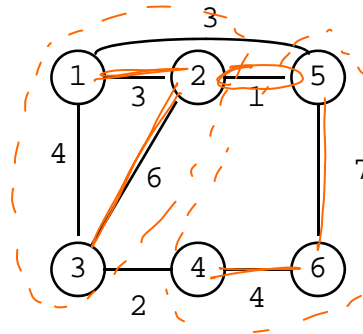
$$\{(N_1, E_1), (N_2, E_2), \dots, (N_k, E_k)\}$$

Where

$$N = N_1 \cup N_2 \cup \dots \cup N_k$$

$$E_i \subseteq E$$

$$E_i \cap E_j = \emptyset$$



Expanding a Spanning Forest

Lemma: Suppose we have a spanning forest of $k > 1$ trees. Let $E' = E_1 \cup E_2 \cup \dots \cup E_k$

Let $e = \{v, w\}$ be cheapest edge in $E - E'$ such that $v \in N_1, w \notin N_1$.

Then there is a spanning tree for G that includes $E' \cup \{v, w\}$ and has minimal cost among trees that include E' .

What kind of algorithm is this leading us towards?

Proof of Lemma

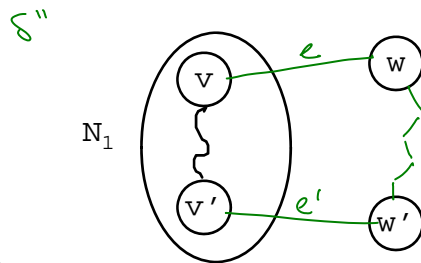
Suppose not. Let

$$S'' = (N, E''), \quad \underline{E'} \subseteq E'', \quad e \notin E''$$

have lower cost than any spanning tree

that includes E' and e

Adding e to S'' causes



Proof of Lemma 2

S'' must have an edge $e' = \{v', w'\}$
with $v' \in N_1, w' \notin N_1$

We know $c(e) \leq c(e')$

Let $S = S'' - \{e'\} \cup \{e\}$

S has no cycles because adding e creates
a unique cycle that removal of e' breaks

S is still a tree (still connected)

any path using e'

can go $v' \rightsquigarrow v \xrightarrow{e} w \rightsquigarrow w'$ instead

so $c(S) \leq c(S'')$

Greedy Strategy

Take least cost edge e_1 in E . By last lemma,

[Consider an initial spanning forest

$$\{ (\{v_1\}, \emptyset), (\{v_2\}, \emptyset), \dots, (\{v_k\}, \emptyset) \}$$

each node in a different tree

Iterative Step

Let

$$\{ (N_1, E_1), (N_2, E_2), \dots, (N_k, E_k) \}$$

be a spanning forest that is *part of a minimum spanning tree*

Add the lowest cost edge that *connects any two of the trees in this forest*

By the last lemma, the result is *a spanning forest that's part of a minimum spanning tree.*

Kruskal's Algorithm

PRIORITY QUEUE

- insert
- extract-min

```

KRUSKAL(G(N,E))
  S ← ∅  forest edges
  NS ← {{v1}, {v2}, ..., {vn}}
  Q ← PRIORITY queue of all edges by cost
  while |NS| > 1 do
    {v,w} ← extract-min(Q)
    if Find(v) ≠ Find(w) then
      S ← S ∪ {{v,w}}
      Union(Find(v), Find(w))
    
```

(S)

Example

Q	c	NS
{2,5}	1	{1} {2} {3} {4} {5} {6}
{3,4}	2	{1,2,5} {3,4} {5} {6}
{1,2}	3	{1,2,3,5}
{1,5}	3	{1,2,3,4,5,6}
{1,3}	4	
{4,6}	4	
{2,3}	6	
{5,6}	7	

Complexity

overall time complexity

Priority Queue: sort, make a list

Union/Find

$O(|E| \lg |E|)$

Union's = $|N| - 1$

Find's = $2 \cdot |E|$

$\leq 3 \cdot |E|$ ops

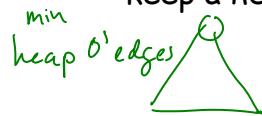
$O(|E| G(|E|))$

rest
 $O(|E|)$

if the graph is connected

Can do a little better—

keep a heap of edges by cost



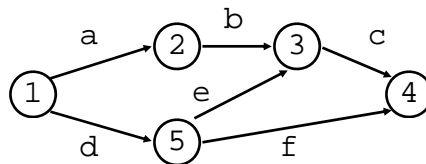
extract min + remake heap for each considered

If only d edges are considered, $O(d \cdot \lg |E|)$

Path Problems

Directed graphs with labels on edges

$(a \cdot b \cdot c) + (d \cdot e \cdot c) + (d \cdot f)$



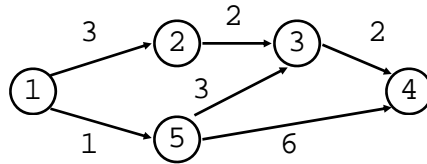
Shortest Path

Labels could be non-negative integers

Use for *distances*

want path of minimum distance

*• \rightsquigarrow +
+ \rightsquigarrow min*



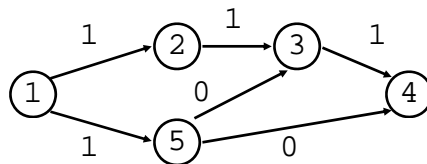
$(3 + 2 + 2) \min (1 + 3 + 2) \min (1 + 6)$

Connectivity

Labels could be Boolean values - *edge is available or not*

Use for *reachability*

*• \rightsquigarrow Boolean \wedge
+ \rightsquigarrow Boolean \vee*



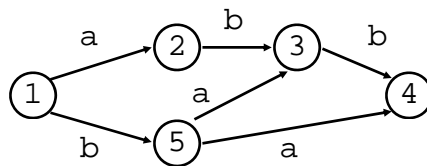
*$(1 \wedge 1 \wedge 1) \vee (1 \wedge 0 \wedge 1) \vee (1 \wedge 0)$
 $\underbrace{\hspace{1cm}}_1 \vee \underbrace{\hspace{1cm}}_0 \vee \underbrace{\hspace{1cm}}_0 = 1$*

Finite Automata

Labels could be symbols from an alphabet

(as in a finite automaton). Use for

patterns that take you from place to another



• \rightsquigarrow concat
 $+$ \rightsquigarrow union

$(abb) \cup (bab) \cup (ba) = \{abb, bab, ba\}$
 Cycles are tricky - need $*$ (Kleene closure)

Path Algorithms

Can produce generic algorithms, just need definitions for $\cdot, +$

Can take advantage of identities $|v \text{ any} = 1$

Two kinds of algorithms

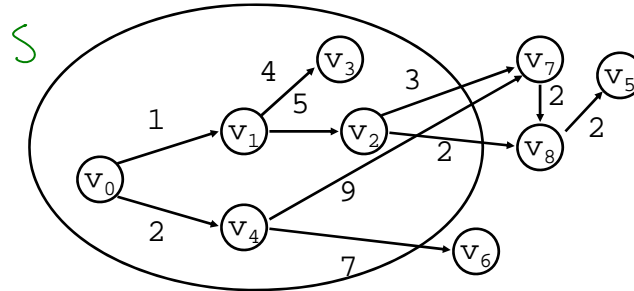
1. Single source
 from one node v to all the rest

2. All pairs
 from v to w for every v, w pair

Can do 2. with 1. applied $\#nodes$ times
 \searrow but other methods are faster

Dijkstra's Algorithm

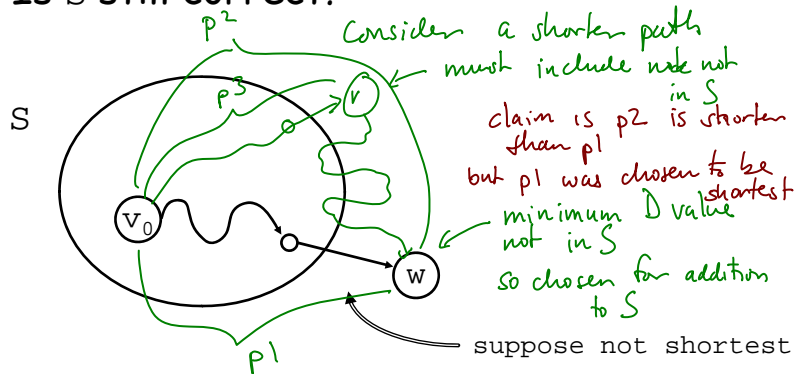
Single-source, shortest path in a digraph
non-negative edge weights



S — nodes whose shortest distance from v_0 is known
D — array of distances along paths that only use nodes in *S*

General Step

→ not already in S
 Add node with minimum *D* value to *S*.
 Update distances to nodes in $N-S$
 Is *S* still correct?



CS 410/584, Algorithm Design & Analysis, Lecture Notes 4

Algorithm Design & Analysis

Full Algorithm $|N|=n$ $|E|=e$

$G = (N, E)$

$\text{lab}(v, w)$ - label of edge (v, w)
 represent with matrix assume $\text{lab}(v, w) = +\infty$
if $(v, w) \in E$

$S \leftarrow \{v_0\}$

$D[v_0] \leftarrow 0$

$D[v] \leftarrow \text{lab}(v_0, v)$ for rest

while $S \neq N$ **do**

- choose w in $N-S$ than minimized $D[w]$

$S \leftarrow S \cup \{w\}$

- **for** all v in $N-S$ **do**

$D[v] \leftarrow \min(D[v], D[w] + \text{lab}(w, v))$

$O(n^2)$ dense

$O(e \lg n)$ - min heap
 sparse

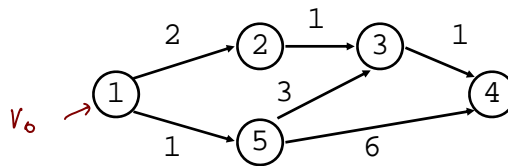
Lecture Notes 4

David Maier

19

Algorithm Design & Analysis

Example



S	\checkmark D[1]	\checkmark D[2]	\checkmark D[3]	\checkmark D[4]	\checkmark D[5]
{1}	0	2	∞	∞	1
{1,5}	0	2	4	7	1
{1,2,5}	0	2	3	7	1
{1,2,3,5}	0	2	3	4	1
{1,2,3,4,5}	min distances				

Lecture Notes 4

David Maier

20

Algorithm Design & Analysis

Warshall's Algorithm

all pairs

Transitive Closure

Basic op

Consider each k only once
consider 1, 2, 3, ... in turn as pivot

Lecture Notes 4 David Maier **21**

Algorithm Design & Analysis

Algorithm

edges are in an adjacency matrix a

```

for k = 1 to n do
  for v = 1 to n do
    if  $a[v,k]$  then
      for w = 1 to n do
        if  $a[k,w]$  then  $a[v,w] \leftarrow 1$ 
    
```

$O(n^3)$

Lecture Notes 4 David Maier **22**

Example

a

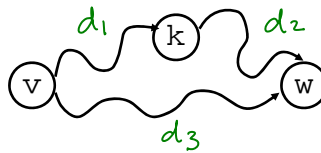
	1	2	3	4	5
1	1	1	∅ 1	∅ 1	1
2	0	1	1	∅ 1	0
3	0	0	1	1	0
4	0	0	0	1	0
5	0	0	1	1	1

pivot on 1 no (v,1) type edges ← initialize diagonal to 1
 pivot on 2 1 → 2 2 → 3 get 1 → 3
 pivot on 3 1 → 3 2 → 3 3 → 4
 pivot on 4 5 → 3 any → 4 4 → none
 pivot on 5 1 → 5 nothing new

Floyd's Algorithm

All-pairs shortest path

$d_3 > d_1 + d_2$?



uses pivots
 shortest distances
 using paths
 through nodes
 with # < k

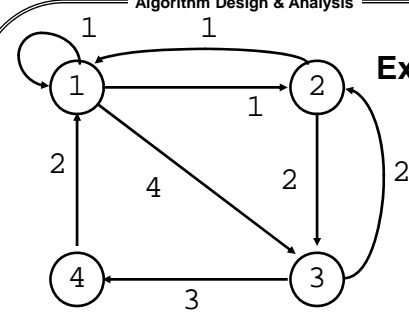
```

for k = 1 to n do
  for v = 1 to n do
    if d[v,k] ≠ ∞ then
      for w = 1 to n do
        d[v,w] ← min (d[v,w], d[v,k] + d[k,w])
    
```

initially
 $d[v,w]$ - weight of
 edge $v \rightarrow w$
 or ∞ if no edge

$O(N^3)$

Algorithm Design & Analysis



Example

		1	2	3	4		$d[i, j]^0$
1	(0	1	4	∞)	
2		1	0	2	∞		
3		∞	2	0	3		
4		2	∞	∞	0)	

0's on diagonal

Lecture Notes 4 David Maier **25**

Algorithm Design & Analysis

Example 2

		1	2	3	4		$d[i, j]^0$
1	(0	1	4	∞)	
2		1	0	2	∞		$2 \rightarrow 1 \quad 1$
3		∞	2	0	3		$4 \rightarrow 1 \quad 2 \quad 1 \rightarrow 2 \quad 1$
4		2	∞	∞	0)	$2 \rightarrow 3 \quad 4$

		1	2	3	4		$d[i, j]^1$
1	(0	1	4	∞)	
2		1	0	2	∞		$\leftarrow k$
3		∞	2	0	3		pivot m
4		2	3	6	0)	1

Lecture Notes 4 David Maier **26**

Example 3

	1	2	3	4	$d[i, j]^1$
1	(0	1	4	∞)	$\begin{array}{c} \underbrace{\quad\quad\quad}_3 \\ 1 \xrightarrow{1} 2 \xrightarrow{2} 3 \\ 3 \xrightarrow{2} 2 \xrightarrow{1} 1 \\ \underbrace{\quad\quad\quad}_3 \end{array}$
2	(1	0	2	∞)	
3	(∞	2	0	3)	
4	(2	3	6	0)	
	1	2	3	4	$d[i, j]^2$
1	(0	1	3	∞)	$\begin{array}{c} \underbrace{\quad\quad\quad}_5 \\ 4 \xrightarrow{3} 2 \xrightarrow{2} 3 \\ 4 \xrightarrow{3} 2 \xrightarrow{1} 1 \text{ no improvement} \\ \underbrace{\quad\quad\quad}_4 \end{array}$
2	(1	0	2	∞)	
3	(3	2	0	3)	
4	(2	3	5	0)	

Example 3

	1	2	3	4	$d[i, j]^2$
1	(0	1	3	∞)	$\begin{array}{c} \underbrace{\quad\quad\quad}_6 \\ 1 \xrightarrow{3} 3 \xrightarrow{3} 4 \\ 2 \xrightarrow{2} 3 \xrightarrow{3} 4 \\ \underbrace{\quad\quad\quad}_5 \end{array}$
2	(1	0	2	∞)	
3	(3	2	0	3)	
4	(2	3	5	0)	
	1	2	3	4	$d[i, j]^3$
1	(0	1	3	6)	$\begin{array}{c} \parallel \\ d[i, j]^4 \\ 2 \xrightarrow{5} 4 \xrightarrow{5} 3 \text{ no improvement} \\ \underbrace{\quad\quad\quad}_{10} \end{array}$
2	(1	0	2	5)	
3	(3	2	0	3)	
4	(2	3	5	0)	