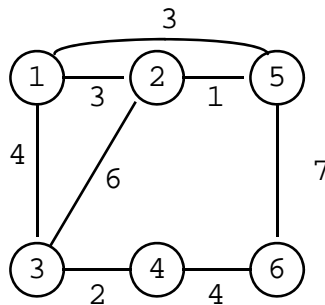


Minimum Cost Spanning Tree

$G = (N, E)$

Cost function c give edge
weights

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



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
- b. adding an edge from $E - E'$ to S

Spanning Forest

A set of trees

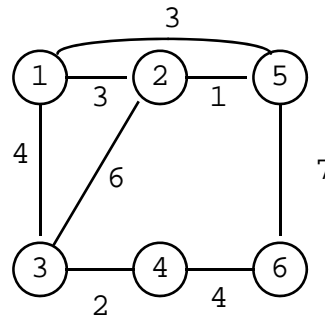
$$\{(N_1, E_1), (N_2, E_2), \dots, (N_k, E_k)\}$$

Where

$$N =$$

$$\subseteq E$$

$$E_i \quad E_j$$



Expanding a Spanning Forest

Lemma: Suppose we have a spanning forest of $k > 1$ trees. Let $E' =$

Let $e = \{v, w\}$ be

such that $v \in N_1, w \notin N_1$.

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

What kind of algorithm is this leading us towards?

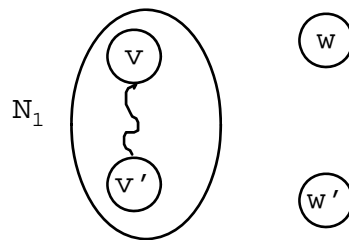
Proof of Lemma

Suppose not. Let

$$S'' = (N, E''), \quad E' \subseteq E'',$$

have lower cost than any spanning tree

Adding e to S'' causes



Proof of Lemma 2

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

We know $c(e) < c(e')$

Let $S = S''$

S has no cycles

S is still a tree

Greedy Strategy

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

[Consider an initial spanning forest

$$\{(\{ \}, \emptyset), (\{ \}, \emptyset), \dots, (\{ \}, \emptyset)\}]$$

Iterative Step

Let

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

be a spanning forest that is

Add the lowest cost edge that

By the last lemma, the result is

Kruskal's Algorithm

```

KRUSKAL(G(N,E))
  S ← ∅
  NS ← {{v1}, {v2}, ..., {vn}}
  Q ←
  while |NS| > 1 do
    {v,w}
    if Find(v) ≠ Find(w) then
      S ← S
      Union(
  
```

Example

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

Complexity

Priority Queue: sort, make a list

Union/Find

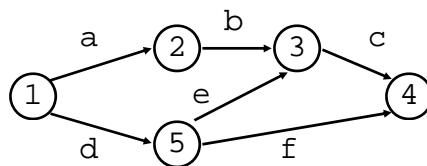
Union's =

Find's =

Can do a little better—
keep a heap of edges by cost

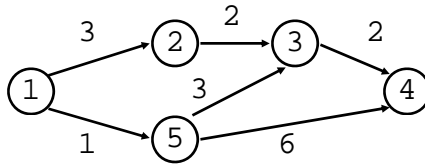
Path Problems

Directed graphs with labels on edges



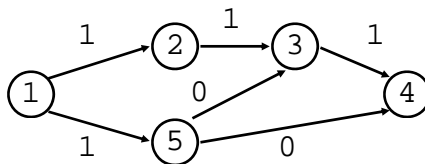
Shortest Path

Labels could be non-negative integers
Use for



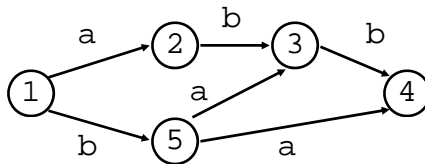
Connectivity

Labels could be Boolean values
Use for



Finite Automata

Labels could be symbols from an alphabet
(as in a finite automaton). Use for



Path Algorithms

Can produce generic algorithms, just need definitions for

Can take advantage of identities

Two kinds of algorithms

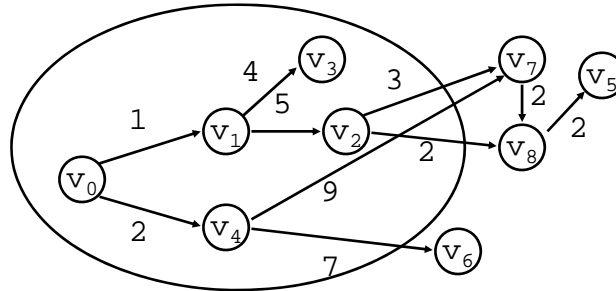
1.

2.

Can do 2. with

Dijkstra's Algorithm

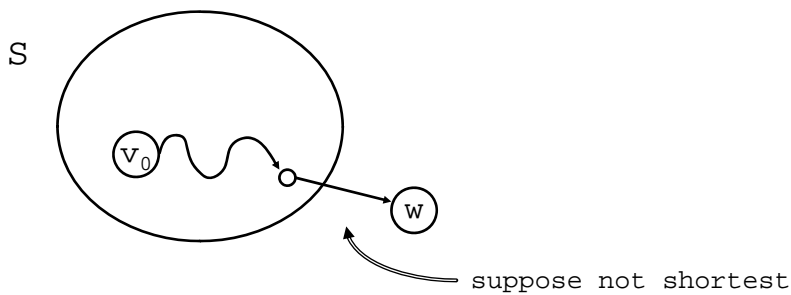
Single-source, shortest path in a digraph



S—
D—array of distances

General Step

Add node with minimum D value to S.
Update distances to nodes in N-S
Is S still correct?

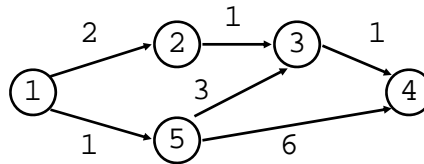


Full Algorithm

```

G = (N, E)
lab(v, w) - label
  represent with matrix
S ← {v0}
D[v0] ← 0
D[v] ← lab(v0, v) for rest
while S ≠ N do
  choose w
  S ←
  for all v in N-S do
    D[v] ←
    
```

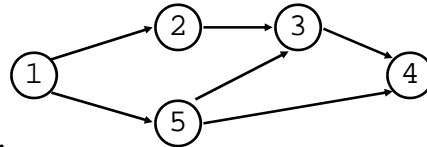
Example



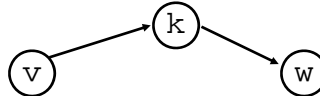
S	D[1]	D[2]	D[3]	D[4]	D[5]
{1}	0	2	∞	∞	1

Warshall's Algorithm

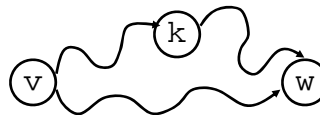
Transitive Closure



Basic op



Consider each k only once



Algorithm

```

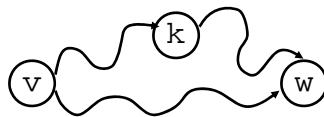
for k = 1 to n do
  for v = 1 to n do
    if a[v,k] then
      for w = 1 to n do
        if a[k,w] then
  
```

Example

		1	2	3	4	5
1	(1	1	0	0	1
2		0	1	1	0	0
3		0	0	1	1	0
4		0	0	0	1	0
5		0	0	1	1	1
)					

Floyd's Algorithm

All-pairs shortest path



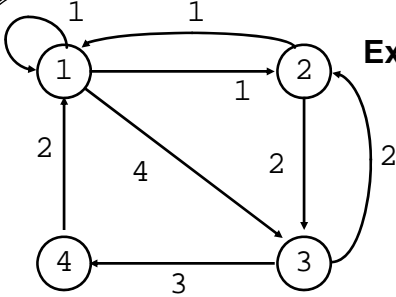
```

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] ←

```

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

Algorithm Design & Analysis



Example

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

Lecture Notes 4 David Maier **25**

Algorithm Design & Analysis

Example 2

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

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

Lecture Notes 4 David Maier **26**

Example 3

		1	2	3	4	$d[i, j]^1$
1	(0	1	4	∞)
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	∞)
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	∞)
2		1	0	2	∞	
3		3	2	0	3	
4	(2	3	5	0)

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