

# Coin Changing

Greed is good. Greed is right. Greed works.  
Greed clarifies, cuts through, and captures the  
essence of the evolutionary spirit.

- *Gordon Gecko (Michael Douglas)*



# Coin Changing

**Goal.** Given currency denominations: 1, 5, 10, 25, 100, devise a method to pay amount to customer using fewest number of coins.

**Ex:** 34¢.



**Cashier's algorithm.** At each iteration, add coin of the largest value that does not take us past the amount to be paid.

**Ex:** \$2.89.



# Coin-Changing: Analysis of Greedy Algorithm

**Observation.** Greedy algorithm is sub-optimal for US postal denominations: 1, 10, 21, 34, 70, 100, 350, 1225, 1500.

**Counterexample.** 140¢.

- Greedy: 100, 34, 1, 1, 1, 1, 1, 1.
- Optimal: 70, 70.



# Coin-Changing: Analysis of Greedy Algorithm

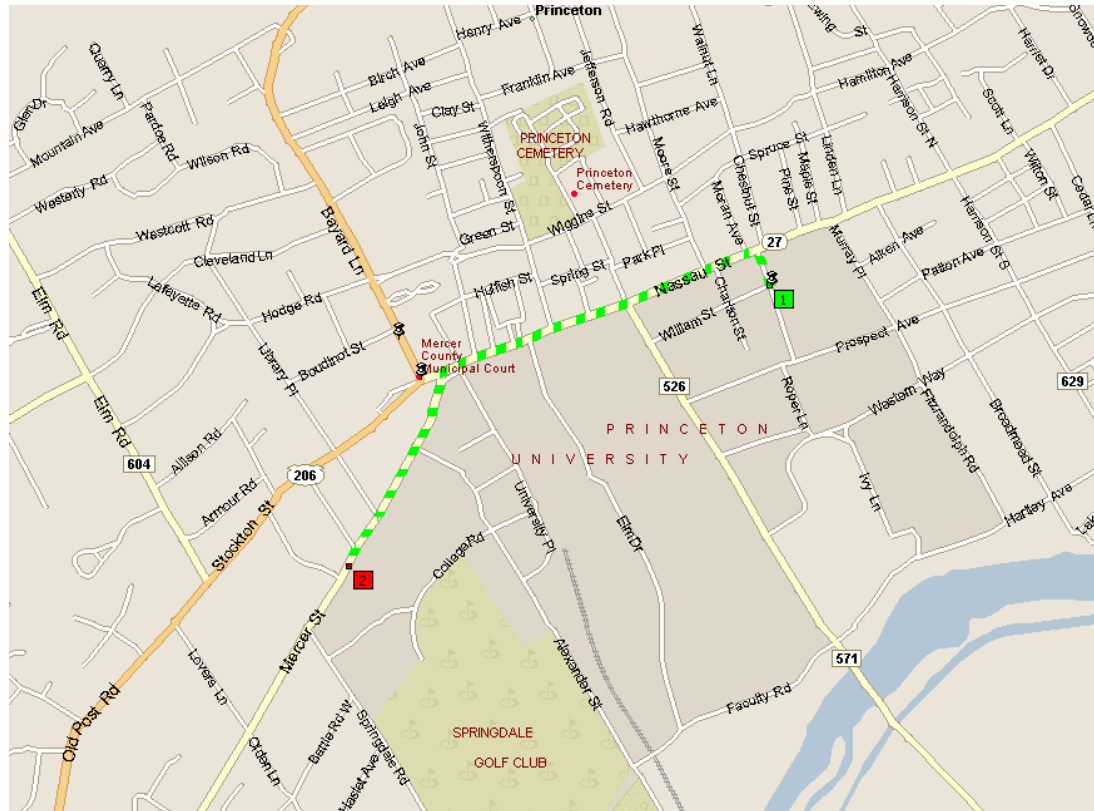
**Theorem.** Greedy is optimal for U.S. coinage: 1, 5, 10, 25, 100.

**Pf.** (by induction on  $x$ )

- Consider optimal way to change  $c_k \leq x < c_{k+1}$  : greedy takes coin  $k$ .
- We claim that any optimal solution must also take coin  $k$ .
  - if not, it needs enough coins of type  $c_1, \dots, c_{k-1}$  to add up to  $x$
  - table below indicates no optimal solution can do this
- Problem reduces to coin-changing  $x - c_k$  cents, which, by induction, is optimally solved by greedy algorithm. ■

$k$	$c_k$	All optimal solutions must satisfy	Max value of coins 1, 2, ..., $k-1$ in any OPT
1	1	$P \leq 4$	-
2	5	$N \leq 1$	4
3	10	$N + D \leq 2$	$4 + 5 = 9$
4	25	$Q \leq 3$	$20 + 4 = 24$
5	100	no limit	$75 + 24 = 99$

## 4.4 Shortest Paths in a Graph



shortest path from Princeton CS department to Einstein's house

# Edsger W. Dijkstra

The question of whether computers can think is like the question of whether submarines can swim.

Do only what only you can do.

In their capacity as a tool, computers will be but a ripple on the surface of our culture. In their capacity as intellectual challenge, they are without precedent in the cultural history of mankind.

The use of COBOL cripples the mind; its teaching should, therefore, be regarded as a criminal offence.

APL is a mistake, carried through to perfection. It is the language of the future for the programming techniques of the past: it creates a new generation of coding bums.



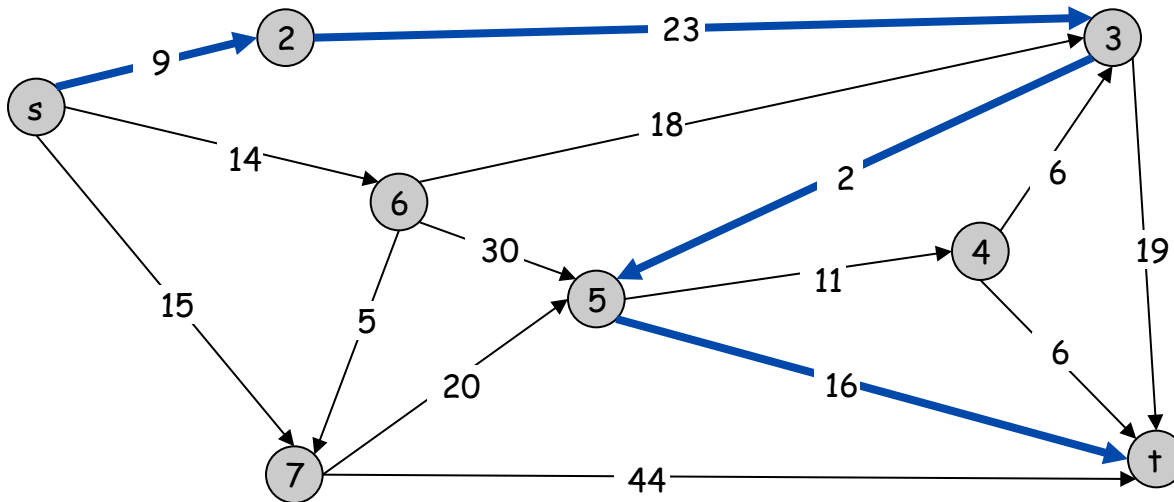
# Shortest Path Problem

## Shortest path network.

- Directed graph  $G = (V, E)$ .
- Source  $s$ , destination  $t$ .
- Length  $\ell_e$  = length of edge  $e$ .

Shortest path problem: find shortest directed path from  $s$  to  $t$ .

↑  
cost of path = sum of edge costs in path



Cost of path  $s-2-3-5-t$   
=  $9 + 23 + 2 + 16$   
= 48.

# Dijkstra's Algorithm

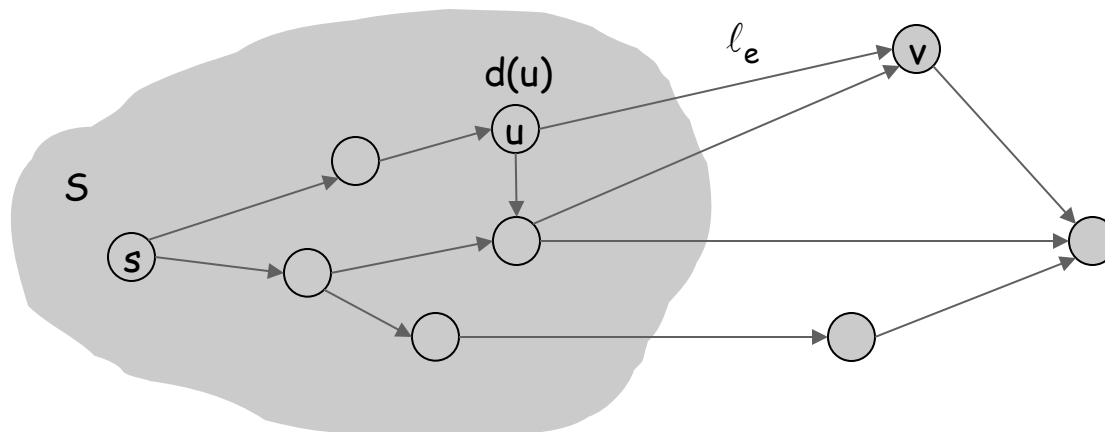
## Dijkstra's algorithm.

- Maintain a set of **explored nodes**  $S$  for which we have determined the shortest path distance  $d(u)$  from  $s$  to  $u$ .
- Initialize  $S = \{s\}$ ,  $d(s) = 0$ .
- Repeatedly choose unexplored node  $v$  which minimizes

$$\pi(v) = \min_{e = (u,v) : u \in S} d(u) + \ell_e,$$

add  $v$  to  $S$ , and set  $d(v) = \pi(v)$ .

← shortest path to some  $u$  in explored part, followed by a single edge  $(u, v)$





# Dijkstra's Algorithm

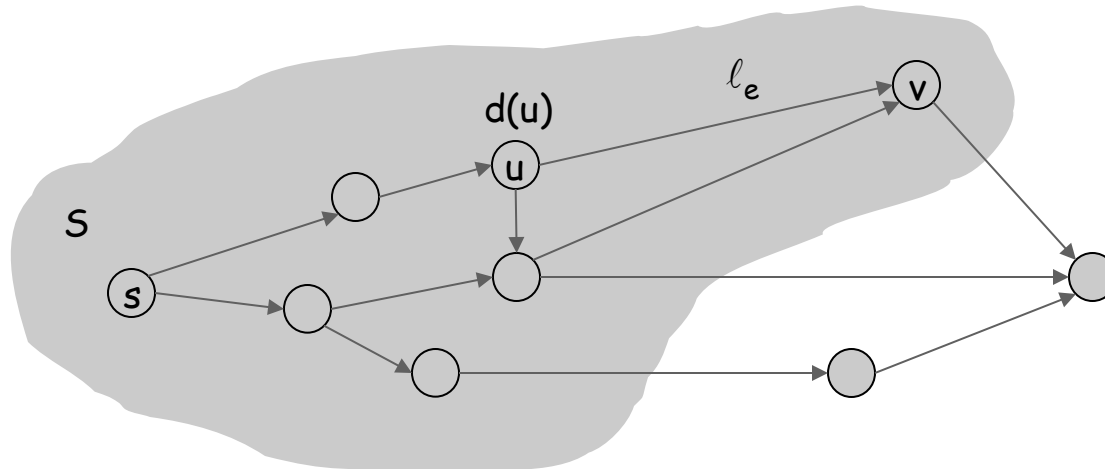
## Dijkstra's algorithm.

- Maintain a set of **explored nodes**  $S$  for which we have determined the shortest path distance  $d(u)$  from  $s$  to  $u$ .
- Initialize  $S = \{s\}$ ,  $d(s) = 0$ .
- Repeatedly choose unexplored node  $v$  which minimizes

$$\pi(v) = \min_{e = (u,v) : u \in S} d(u) + \ell_e,$$

add  $v$  to  $S$ , and set  $d(v) = \pi(v)$ .

shortest path to some  $u$  in explored part, followed by a single edge  $(u, v)$



# Dijkstra's Algorithm: Proof of Correctness

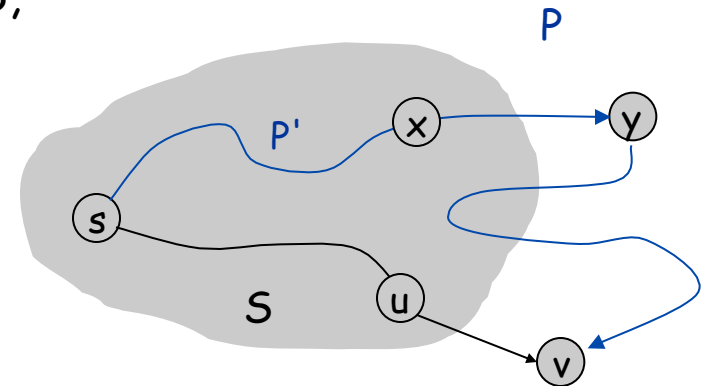
**Invariant.** For each node  $u \in S$ ,  $d(u)$  is the length of the shortest  $s$ - $u$  path.

**Pf.** (by induction on  $|S|$ )

**Base case:**  $|S| = 1$  is trivial.

**Inductive hypothesis:** Assume true for  $|S| = k \geq 1$ .

- Let  $v$  be next node added to  $S$ , and let  $u$ - $v$  be the chosen edge.
- The shortest  $s$ - $u$  path plus  $(u, v)$  is an  $s$ - $v$  path of length  $\pi(v)$ .
- Consider any  $s$ - $v$  path  $P$ . We'll see that it's no shorter than  $\pi(v)$ .
- Let  $x$ - $y$  be the first edge in  $P$  that leaves  $S$ , and let  $P'$  be the subpath to  $x$ .
- $P$  is already too long as soon as it leaves  $S$ .



$$\ell(P) \geq \ell(P') + \ell(x, y) \geq d(x) + \ell(x, y) \geq \pi(y) \geq \pi(v)$$

↑  
nonnegative  
weights

↑  
inductive  
hypothesis

↑  
defn of  $\pi(y)$

↑  
Dijkstra chose  $v$   
instead of  $y$

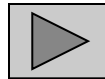
# Dijkstra's Algorithm: Implementation

For each unexplored node, explicitly maintain  $\pi(v) = \min_{e=(u,v): u \in S} d(u) + \ell_e$ .

- Next node to explore = node with minimum  $\pi(v)$ .
- When exploring  $v$ , for each incident edge  $e = (v, w)$ , update

$$\pi(w) = \min \{ \pi(w), \pi(v) + \ell_e \}.$$

**Efficient implementation.** Maintain a priority queue of unexplored nodes, prioritized by  $\pi(v)$ .



PQ Operation	Dijkstra	Array	Binary heap	d-way Heap	Fib heap <sup>†</sup>
Insert	$n$	$n$	$\log n$	$d \log_d n$	1
ExtractMin	$n$	$n$	$\log n$	$d \log_d n$	$\log n$
ChangeKey	$m$	1	$\log n$	$\log_d n$	1
IsEmpty	$n$	1	1	1	1
Total		$n^2$	$m \log n$	$m \log_{m/n} n$	$m + n \log n$

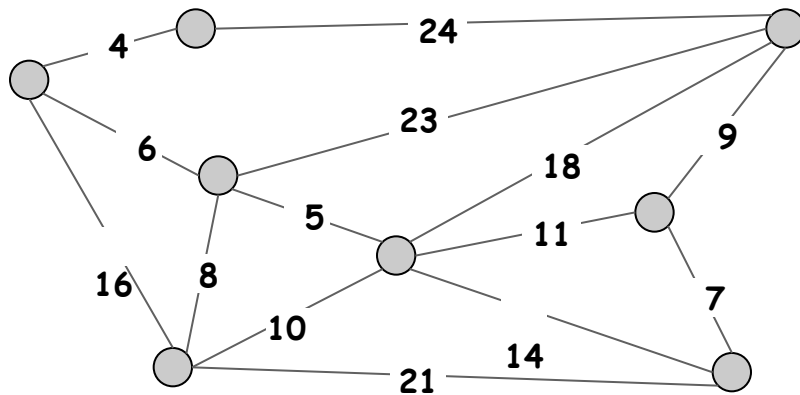
<sup>†</sup> Individual ops are amortized bounds

## 4.5 Minimum Spanning Tree

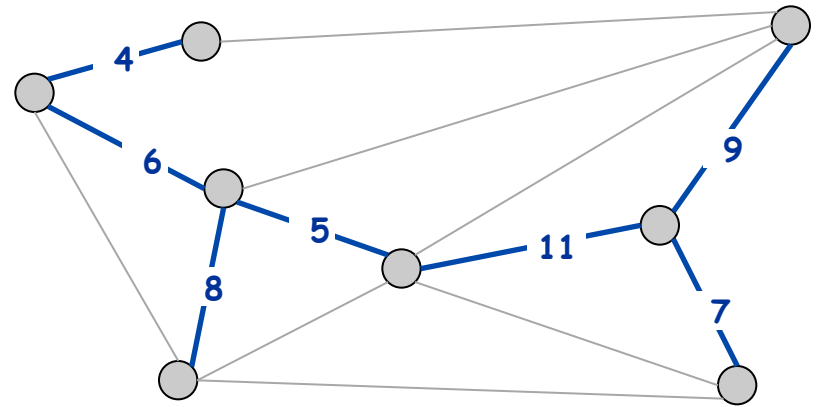
---

# Minimum Spanning Tree

**Minimum spanning tree.** Given a connected graph  $G = (V, E)$  with real-valued edge weights  $c_e$ , an MST is a subset of the edges  $T \subseteq E$  such that  $T$  is a spanning tree whose sum of edge weights is minimized.



$G = (V, E)$



$T, \sum_{e \in T} c_e = 50$

**Cayley's Theorem.** There are  $n^{n-2}$  spanning trees of  $K_n$ .

↑  
can't solve by brute force

# Applications

MST is fundamental problem with diverse applications.

- Network design.
  - telephone, electrical, hydraulic, TV cable, computer, road
- Approximation algorithms for NP-hard problems.
  - traveling salesperson problem, Steiner tree
- Indirect applications.
  - max bottleneck paths
  - LDPC codes for error correction
  - image registration with Renyi entropy
  - learning salient features for real-time face verification
  - reducing data storage in sequencing amino acids in a protein
  - model locality of particle interactions in turbulent fluid flows
  - autoconfig protocol for Ethernet bridging to avoid cycles in a network
- Cluster analysis.

# Greedy Algorithms

**Kruskal's algorithm.** Start with  $T = \emptyset$ . Consider edges in ascending order of cost. Insert edge  $e$  in  $T$  unless doing so would create a cycle.

**Reverse-Delete algorithm.** Start with  $T = E$ . Consider edges in descending order of cost. Delete edge  $e$  from  $T$  unless doing so would disconnect  $T$ .

**Prim's algorithm.** Start with some root node  $s$  and greedily grow a tree  $T$  from  $s$  outward. At each step, add the cheapest edge  $e$  to  $T$  that has exactly one endpoint in  $T$ .

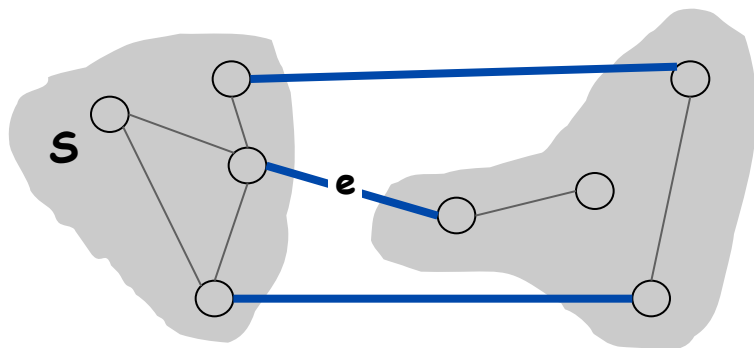
**Remark.** All three algorithms produce an MST.

# Greedy Algorithms

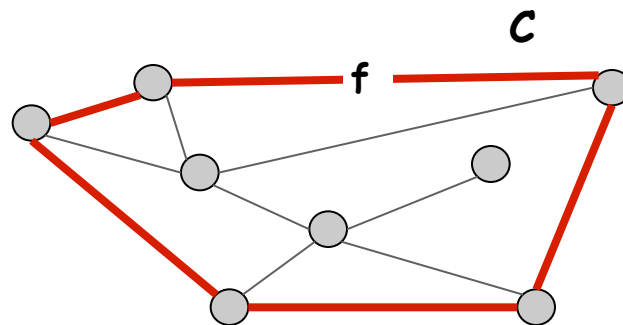
**Simplifying assumption.** All edge costs  $c_e$  are distinct.

**Cut property.** Let  $S$  be any subset of nodes, and let  $e$  be the min cost edge with exactly one endpoint in  $S$ . Then the MST contains  $e$ .

**Cycle property.** Let  $C$  be any cycle, and let  $f$  be the max cost edge belonging to  $C$ . Then the MST does not contain  $f$ .



**e is in the MST**

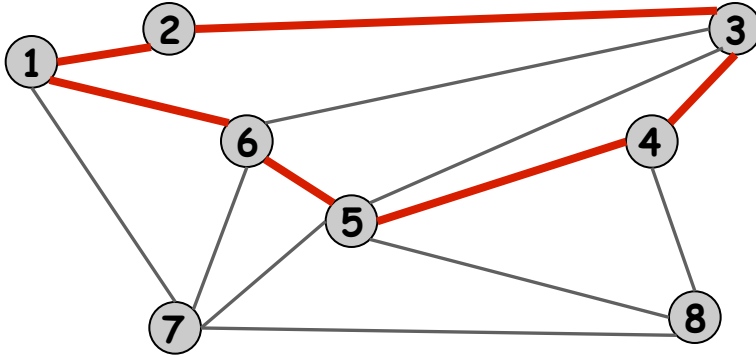


**f is not in the MST**



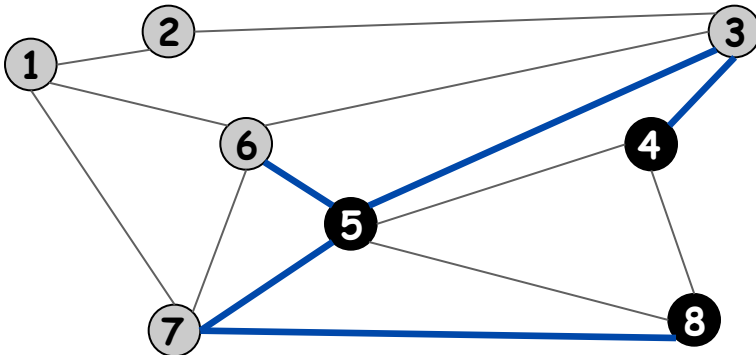
## Cycles and Cuts

**Cycle.** Set of edges the form  $a-b, b-c, c-d, \dots, y-z, z-a$ .



**Cycle C = 1-2, 2-3, 3-4, 4-5, 5-6, 6-1**

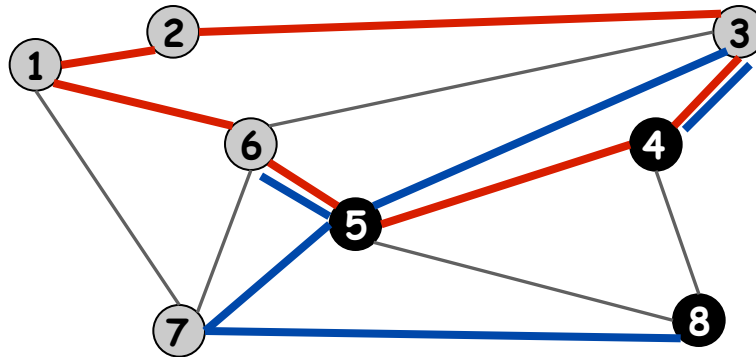
**Cutset.** A cut is a subset of nodes  $S$ . The corresponding cutset  $D$  is the subset of edges with exactly one endpoint in  $S$ .



Cut S = { 4, 5, 8 }  
Cutset D = 5-6, 5-7, 3-4, 3-5, 7-8

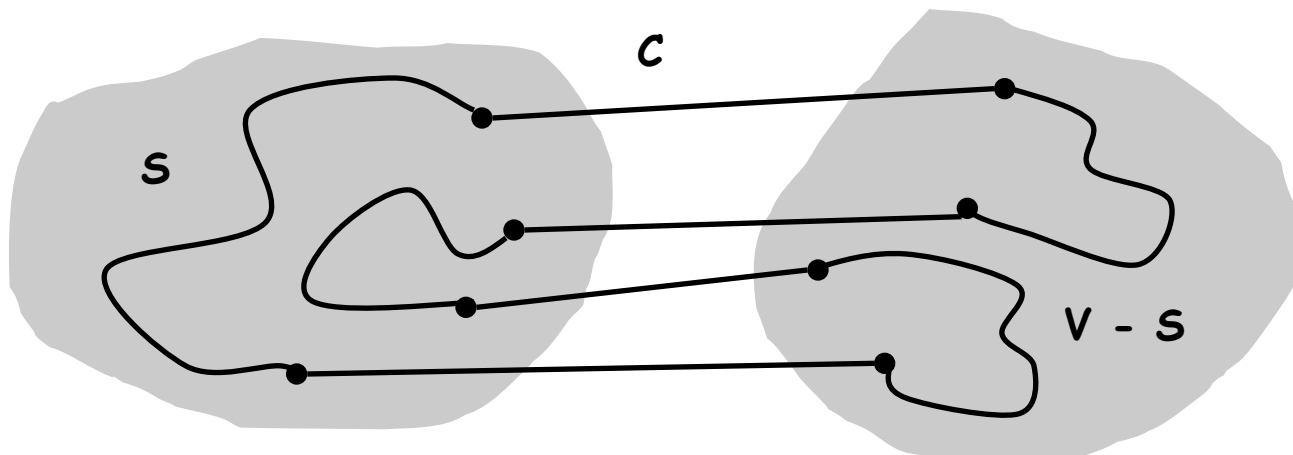
# Cycle-Cut Intersection

**Claim.** A cycle and a cutset intersect in an even number of edges.



Cycle  $C = 1-2, 2-3, 3-4, 4-5, 5-6, 6-1$   
Cutset  $D = 3-4, 3-5, 5-6, 5-7, 7-8, 8-4$   
Intersection =  $3-4, 5-6$

**Pf.** (by picture)



# Greedy Algorithms

**Simplifying assumption.** All edge costs  $c_e$  are distinct.

**Cut property.** Let  $S$  be any subset of nodes, and let  $e$  be the min cost edge with exactly one endpoint in  $S$ . Then the MST  $T^*$  contains  $e$ .

Pf. (exchange argument)

- Suppose  $e$  does not belong to  $T^*$ , and let's see what happens.
- Adding  $e$  to  $T^*$  creates a cycle  $C$  in  $T^*$ .
- Edge  $e$  is both in the cycle  $C$  and in the cutset  $D$  corresponding to  $S$   
 $\Rightarrow$  there exists another edge, say  $f$ , that is in both  $C$  and  $D$ .
- $T' = T^* \cup \{e\} - \{f\}$  is also a spanning tree.
- Since  $c_e < c_f$ ,  $\text{cost}(T') < \text{cost}(T^*)$ .
- This is a contradiction. ■

