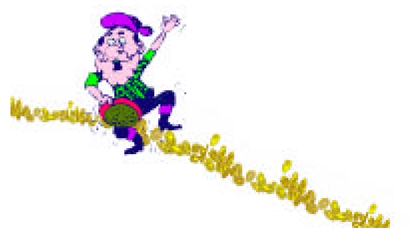
Algorithm Design Techniques

- Induction
- Divide and Conquer
- Dynamic Programming
- Reduction
- **Greedy**
  - Kruskal's algorithm for MST

Greedy Paradigm
Get what you can NOW!

But sometimes it's better to look around!

But sometimes it isn't …



I hate gas stations!

- I'm driving cross country and my route is fixed.
- My map tells me exactly where every gas station along the route is located.
- I want to minimize the number of times I stop for gas…
- … without running out!

Greedy

- First stop
  - I'll stop at the farthest gas station I can get to without running out.
- Then repeat

Greedy is Optimal!

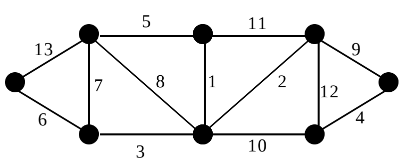
- Can the optimal make a first stop that is later?

Minimum Spanning Tree

- Input: Weighted graph G
- Output: Minimum weight spanning tree of G

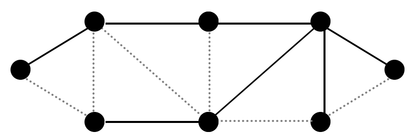
Weighted Graph

- G=(V,E) is a connected, weighted graph with n vertices and m edges.



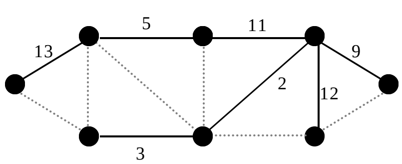
Spanning Tree

- A spanning tree of G is a connected, acyclic subgraph with vertex set V.



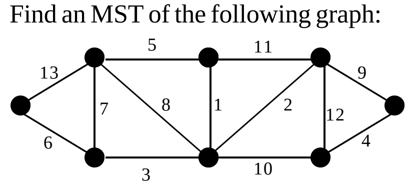
Weight of Spanning Tree

- The weight of spanning tree of G is the sum of the weights of its edges.



Minimum Spanning Tree

- A minimum spanning tree of G is one with smallest possible weight.
- Find an MST of the following graph:



Kruskal’s (Greedy) Algorithm

Let e_1, e_2, … , e_m be the edges of G sorted by increasing weight.

F=V (F is a forest of isolated vertices)

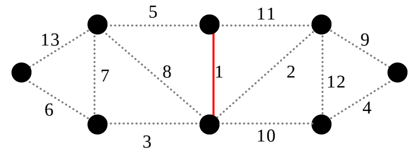
For i=1 to m

  If F+{e_i} is acyclic then F=F+{e_i}.

Return(F)

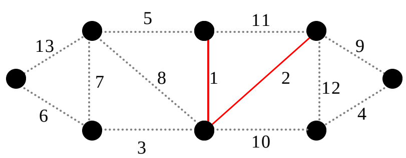
Kruskal’s Algorithm

- Order the edge weights. (In this graph the weights are unique.)
- 1,2,3,4,5,6,7,8,9,10,11,12,13



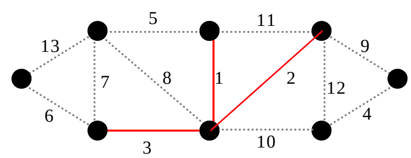
Kruskal’s Algorithm-cont.

- 1,2,3,4,5,6,7,8,9,10,11,12,13



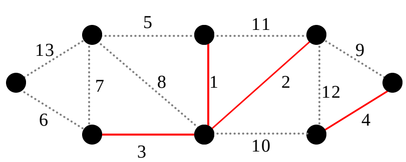
Kruskal’s Algorithm-cont.

- 1,2,3 ,4,5,6,7,8,9,10,11,12,13



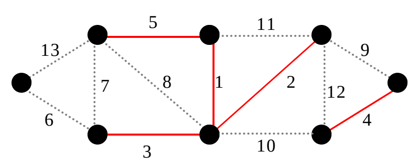
Kruskal’s Algorithm-cont.

- 1,2,3,4 ,5,6,7,8,9,10,11,12,13



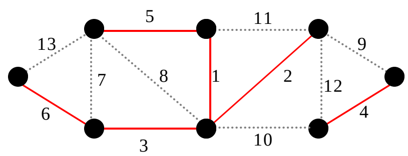
Kruskal’s Algorithm-cont.

- 1,2,3,4,5,6,7,8,9,10,11,12,13



Kruskal’s Algorithm-cont.

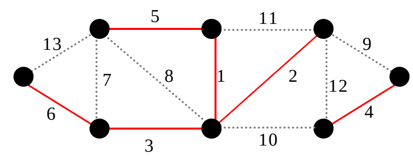
- 1,2,3,4,5,6 ,7,8,9,10,11,12,13



11/1/00 CS140(15)

Kruskal’s Algorithm-cont.

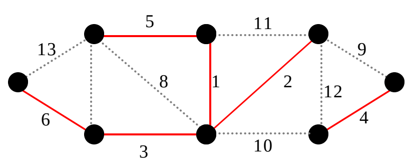
- 1,2,3,4,5,6,7,8,9,10,11,12,13
- Can we add the edge with weight 7?



11/1/00 CS140(15)

Kruskal’s Algorithm-cont.

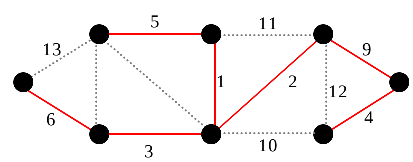
- 1,2,3,4,5,6,7 ,8,9,10,11,12,13
- Can we add the edge with weight 8?



11/1/00 CS140(15)

Kruskal’s Algorithm-cont.

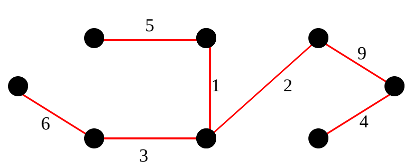
- 1,2,3,4,5,6,7,8,9,10,11,12,13



11/1/00 CS140(15)

Kruskal’s Algorithm-cont.

MST of G with cost _______



11/1/00 CS140(15)

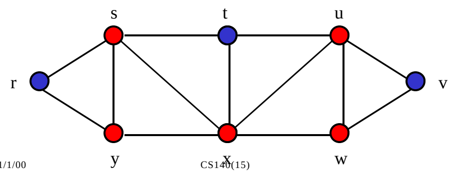
Kruskal’s Algorithm

- Does it work in general?
- Prove it.

11/1/00 CS140(15)

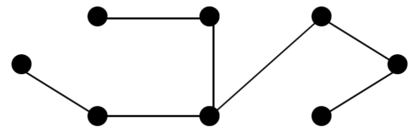
Cut

- A **cut** is a partition of the vertices of G into two sets (R,B).
- An edge e crosses the cut if it has an endpoint in each set of the cut.
- Which edges cross the (R,B) cut?



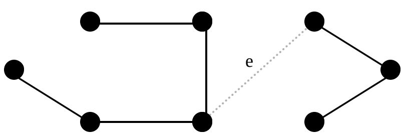
Tree Facts

- A tree on n nodes has n -1 edges.



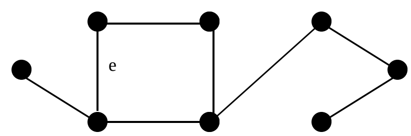
Tree Facts

- If e is an edge of T then T-{e} is a forest consisting of two trees.



Tree Facts

- If e is an edge of G but not of T then T+{e} contains exactly one cycle.



Tree Facts

1. A tree on n nodes has n -1 edges.
2. If e is an edge of T then T-{e} is a forest consisting of two trees.
3. If e is an edge of G but not of T then T+{e} contains exactly one cycle.

Kruskal’s Algorithm Proof of Correctness

Claim:

At each stage of the algorithm F is a subgraph of some MST of G.

Kruskal's Algorithm

Let $e_1, e_2, \ldots, e_m$ be the edges of G sorted by increasing weight.

F=V (F is a forest of isolated vertices)

Claim is true here

For i=1 to m

If F+{e_i} is acyclic then F=F+{e_i}.

Return(F)

11/1/00 CS140(15)

Loop Invariant

Let $e_1, e_2, \ldots, e_m$ be the edges of G sorted by increasing weight.

F=V (F is a forest of isolated vertices) Claim is true here

If Claim is true here

For i=1 to m

If F+{e_i} is acyclic then F=F+{e_i}.

Then Claim is true here

Return(F)

11/1/00 CS140(15)

Kruskal's Algorithm Proof of Correctness

Loop Invariant:

F is a subgraph of some MST of G .

Proof:

Consider the k^{th} execution of the loop. Let T be a MST of G containing F. What can happen during the loop?

1. e_k is not added to F
2. e_k is added to F

11/1/00 CS140(15)

Kruskal's Algorithm Proof of Correctness

Loop Invariant:

F is a subgraph of some MST of G.

Proof:

1. e_k is not added to F

   In this case F does not change so the claim holds when execution of loop concludes

11/1/00 CS140(15)

Kruskal's Algorithm Proof of Correctness

Loop Invariant:

F is a subgraph of some MST of G.

Proof:

2. e_k is added to F

   We know that T is a MST of G and T contains F.

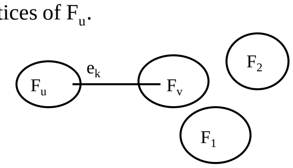
   Need to show there is a MST of G that contains F+{ e_k }

   If e_k is an edge of T we are done. So assume not.

11/1/00 CS140(15)

What do we know?

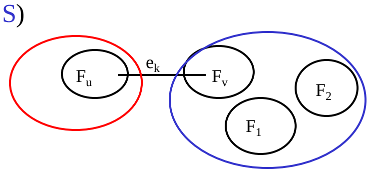
Assume e_k=(u,v). The vertices u and v are in separate connected components. Let S be the vertices of F_u.



11/1/00 CS140(15)

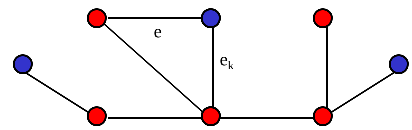
What do we know?

e_k is a minimum weight edge spanning (S,V-S)



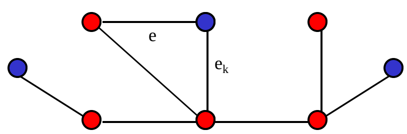
Using our tree facts

- The graph T+{e_k} contains exactly one cycle.
- This cycle contains e_k and at least one additional edge e that spans (S,V-S).
- T+{e_k}-{e} is an MST of G.



Moreover

- T+{e_k}-{e} is an MST of G **that contains the edges of F+{e_k}.**

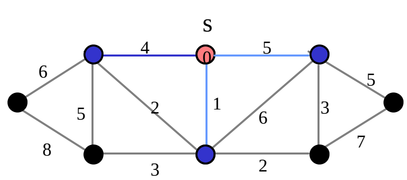


Running Time

- We'll save that for later…

Dykstra's Algorithm

Number in node u indicate $d_G(s,u)$



Prim's Algorithm (Another Greedy Algorithm)

Choose a vertex w ε V

F={w}

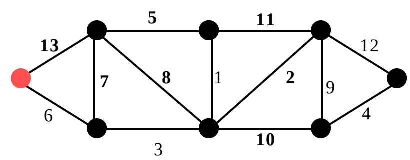
While V-F ≠ ϕ

Let e be a minimum weight edge that emerges from F

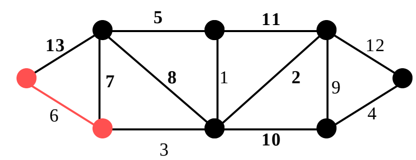
F=F+{e}

Prim's example

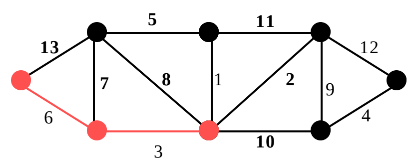
Start with red vertex



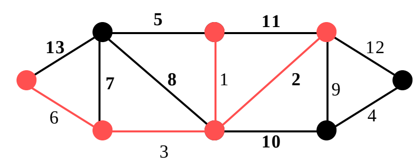
Prim's example



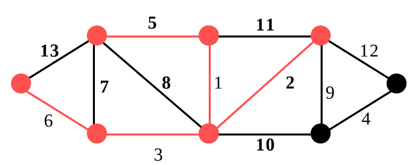
Prim's example



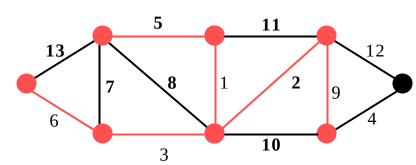
Prim's example



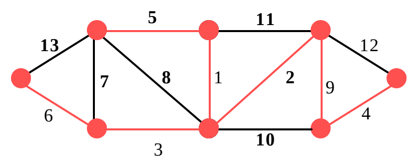
Prim's example



Prim's example



Prim's example



Prim's Algorithm

- Is it correct?
- Is it efficient?