

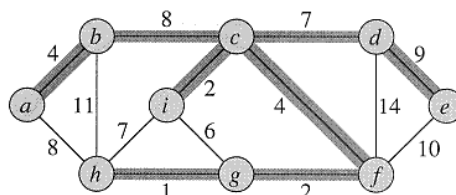
# CSC373

## Announcements

- Assignment 1, problem 1:
- The graph is connected.
- In A we are looking for a set of edges connecting the three nodes.
- In A we are looking to minimize the **total** weight of the edges connecting the three nodes. The paths from  $s$  to  $t_1$  and from  $s$  to  $t_2$  do not have to be the shortest.

## Minimum Spanning Trees

- Example Problem
  - You are planning a new terrestrial telecommunications network to connect a number of remote mountain villages in a developing country.
  - The cost of building a link between pairs of neighboring villages  $(u,v)$  has been estimated:  $w(u,v)$ .
  - You seek the minimum cost design that ensures each village is connected to the network.
  - The solution is called a *minimum spanning tree*.



## Properties of a Minimum Spanning Tree

- $|V|-1$  edges
- Acyclic
- Unique???

## A greedy strategy for MST – Kruskal's algorithm

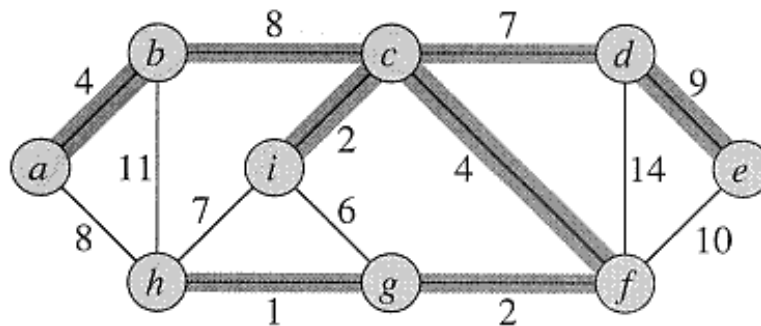
- The idea:

*expand the network at minimal cost*

- Throw in the lightest edge that does not close a cycle

## Example:

### Kruskal's Algorithm



### Kruskal's Algorithm for computing MST

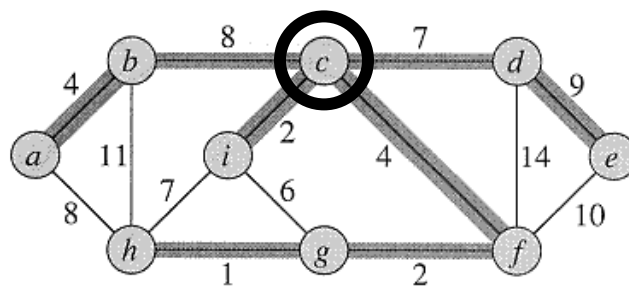
MST-KRUSKAL( $G, w$ )

```
1  $A \leftarrow \emptyset$  Running Time =  $O(E \log E)$ 
2 for each vertex  $v \in V[G]$   $= O(E \log V)$ 
3   do MAKE-SET( $v$ )
4 sort the edges of  $E$  into nondecreasing order by weight  $w$ 
5 for each edge  $(u, v) \in E$ , taken in nondecreasing order by weight
6   do if FIND-SET( $u$ )  $\neq$  FIND-SET( $v$ )
7     then  $A \leftarrow A \cup \{(u, v)\}$ 
8         UNION( $u, v$ )
9 return  $A$ 
```

## Kruskal's algorithm – correctness proof plan

- The algorithm obviously terminates;
- show that at each step the partial solution the algorithm holds is promising:
  - can be extended to an MST;
- when the algorithm terminates, show that the solution it holds is a tree;
- it is promising, hence it must be an MST.

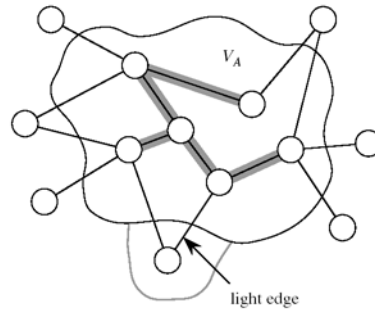
## Prim's Algorithm



- A different greedy strategy.
- Build the net by choosing the cheapest link to connect next.

## Prim's Algorithm – cont'd

- Build one tree
- Start from arbitrary root  $r$
- At each step, add lightest edge crossing cut  $(V_A, V - V_A)$ , where  $V_A =$  vertices processed so far.



[Edges of  $A$  are shaded.]

## Finding light edges quickly

- Use priority queue
- Each object in the queue is a vertex  $v$  that is still waiting to be connected (in  $V - V_A$ )
- Key of  $v$  is minimum weight of any edge  $(u, v)$ ,  $u \in V_A$ .
- Each vertex knows its parent in tree by  $\pi[v]$ .

## Prim's Algorithm

MST-PRIM( $G, w, r$ ) Running Time =  $O(E \log(V))$

```

1  for each  $u \in V[G]$ 
2      do  $key[u] \leftarrow \infty$ 
3      do  $\pi[u] \leftarrow \text{NIL}$ 
4   $key[r] \leftarrow 0$ 
5   $Q \leftarrow V[G]$ 
6  while  $Q \neq \emptyset$ 
7      do  $u \leftarrow \text{EXTRACT-MIN}(Q)$ 
8          for each  $v \in \text{Adj}[u]$ 
9              do if  $v \in Q$  and  $w(u, v) < key[v]$ 
10                 then  $\pi[v] \leftarrow u$ 
11                     $key[v] \leftarrow w(u, v)$ 

```

Annotations for running time analysis:

- Lines 2, 3, and 4 are grouped by a brace and labeled  $O(V)$ .
- Line 6 is labeled "Executed  $|V|$  times".
- Line 7 is labeled  $O(\log V)$ .
- Line 8 is labeled "Executed  $|E|$  times".
- Line 11 is labeled  $O(\log V)$ .

## Prim's algorithm – correctness proof plan

- The algorithm obviously terminates;
- show that at each step the partial solution the algorithm holds is promising:
  - can be extended to an MST;
- when the algorithm terminates, show that the solution it holds is a tree;
- it is promising, hence it must be an MST.

## Three Knapsack Examples

- Fractional knapsack with variable value/weight
- 0-1 knapsack with variable value/weight – the (general knapsack)
- 0-1 knapsack with fixed value/weight

## The (General) Knapsack Problem

Input: information  $(w_i, g_i)$  about  $n$  objects, where  
 $w_i$  = integer weight of object  $i$   
 $g_i$  = real value of object  $i$   
 $C$  = integer weight capacity of knapsack

Let  $S \subseteq \{1, \dots, n\}$  = set of objects selected.

$S$  is feasible if  $K(S) = \sum_{i \in S} w_i \leq C$

Output: A feasible set  $S$  with maximum profit:  $P(S) = \sum_{i \in S} g_i$