

# CSE 326 Data Structures

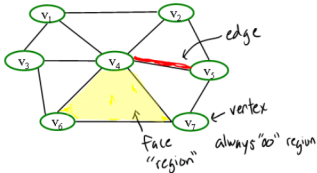
Dave Bacon

Dijkstra, Dijkstra, Dijkstra  
Minimal Spanning Trees  
Fun With Tablets

# Logisitics

- Homework 7 will be due Wed, March 7
- Read Chapter 9 of Weiss
- Monday: Huffman Coding, Class Evaluations
- Wednesday: Final Review
- Friday: Games and NP completeness
- **Final:** Thursday March 15, 8:30-10:20 MGH 231

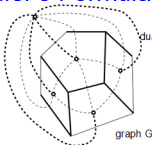
# Euler's Formula (Planar Graphs)



Euler's formula:

$$|V| - |E| + |F| = 2$$

# Euler's Formula (Planar Graphs)



Any cycle in  $G$  disconnects  $G^*$  and vice versa

A acyclic subgraph of  $G$  does not disconnect  $G^*$

Pick spanning tree of  $G$ , call it  $S$ .  $(G-S)$  is  $G$  with  $S$ 's edges removed.

Edges of  $(G-S)$  form a subgraph of dual graph  $G^*$ , call it  $R$

This subgraph is connected (since  $G$  is acyclic) and acyclic (since  $G$  is connected)

Number edges in  $S = |V|-1$ . Number of edges in  $R = |F|-1$ . But this should equal the total number of edges!

# Dijkstra's Alg: Implementation Optimized

Initialize the cost of each node to  $\infty$

Initialize the cost of the source to 0

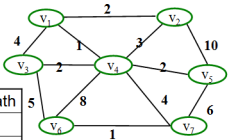
While there are unknown nodes left in the graph

- Select the unknown node  $b$  with the lowest cost

- Mark  $b$  as known

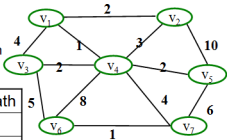
- For each node  $a$  adjacent to  $b$

  - $a$ 's cost =  $\min(a$ 's old cost,  $b$ 's cost + cost of  $(b, a)$ )



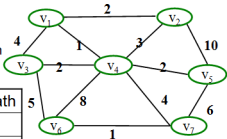
V	Kwn	Distance	path
v1	F	0	0
v2	F	$\infty$	0
v3	F	$\infty$	0
v4	F	$\infty$	0
v5	F	$\infty$	0
v6	F	$\infty$	0
v7	F	$\infty$	0

After  $v_1$  is declared known



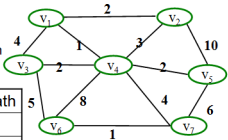
V	Kwn	Distance	path
v1			
v2			
v3			
v4			
v5			
v6			
v7			

After  $v_4$  is declared known



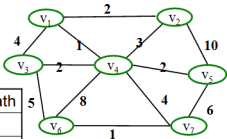
V	Kwn	Distance	path
v1			
v2			
v3			
v4			
v5			
v6			
v7			

After  $v_2$  is declared known



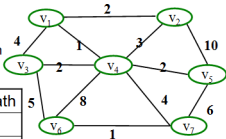
V	Kwn	Distance	path
v1			
v2			
v3			
v4			
v5			
v6			
v7			

After  $v_5$  and  $v_3$  is declared known



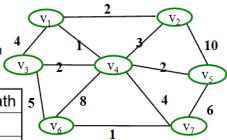
V	Kwn	Distance	path
v1			
v2			
v3			
v4			
v5			
v6			
v7			

After  $v_7$  is declared known



V	Kwn	Distance	path
v1			
v2			
v3			
v4			
v5			
v6			
v7			

After  $v_6$  is declared known



V	Kwn	Distance	path
v1			
v2			
v3			
v4			
v5			
v6			
v7			

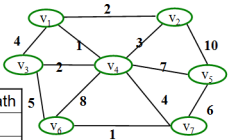
# Prim's Algorithm for MST

## A *node-based greedy algorithm*

Builds MST by greedily adding nodes

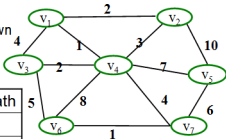
1. Select a node to be the "root"
  - mark it as *known*
  - Update cost of all its neighbors
2. While there are *unknown* nodes left in the graph
  - a. Select an *unknown* node *b* with the smallest cost from some *known* node *a*
  - b. Mark *b* as *known*
  - c. Add (*a*, *b*) to MST
  - d. Update cost of all nodes adjacent to *b*

Note: cost from some *a*,  
not from root



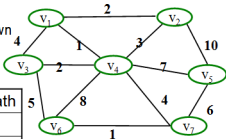
V	Kwn	Distance	path
v1	F	0	0
v2	F	$\infty$	0
v3	F	$\infty$	0
v4	F	$\infty$	0
v5	F	$\infty$	0
v6	F	$\infty$	0
v7	F	$\infty$	0

After  $v_1$  is declared known



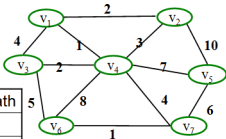
V	Kwn	Distance	path
v1			
v2			
v3			
v4			
v5			
v6			
v7			

After  $v_4$  is declared known



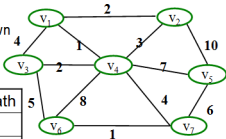
V	Kwn	Distance	path
v1			
v2			
v3			
v4			
v5			
v6			
v7			

After  $v_2$  and  $v_3$  are declared known



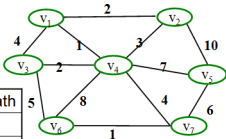
V	Kwn	Distance	path
v1			
v2			
v3			
v4			
v5			
v6			
v7			

After  $v_7$  is declared known



V	Kwn	Distance	path
v1			
v2			
v3			
v4			
v5			
v6			
v7			

After  $v_6$  and  $v_5$  are declared known



V	Kwn	Distance	path
v1			
v2			
v3			
v4			
v5			
v6			
v7			

# Prim's Algorithm Analysis

## Running time:

Same as Dijkstra's:  $O(|E| \log |V|)$

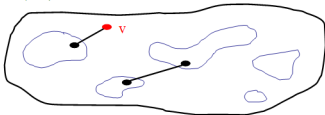
## Correctness:

Proof is similar to Dijkstra's

# Kruskal's MST Algorithm

**Idea:** Grow a **forest** out of edges that do not create a cycle. Pick an **edge with the smallest weight**.

$G=(V,E)$



# Kruskal's Algorithm for MST

## An edge-based greedy algorithm

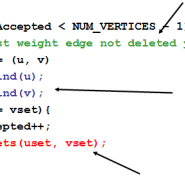
Builds MST by greedily adding edges

1. Initialize with
  - empty MST
  - all vertices marked unconnected
  - all edges **unmarked**
2. While there are still **unmarked** edges
  - a. Pick the lowest cost edge  $(u, v)$  and mark it
  - b. If  $u$  and  $v$  are not already connected, add  $(u, v)$  to the MST and mark  $u$  and  $v$  as connected to each other

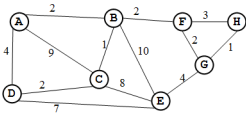
*Doesn't it sound familiar?*

# Kruskal code

```
void Graph::kruskal() {  
    int edgesAccepted = 0;  
    DisjSet s(NUM_VERTICES);  
  
    while (edgesAccepted < NUM_VERTICES - 1) {  
        e = smallest weight edge not deleted yet;  
        // edge e = (u, v)  
        uset = s.find(u);  
        vset = s.find(v);  
        if (uset != vset) {  
            edgesAccepted++;  
            s.unionSets(uset, vset);  
        }  
    }  
}
```



# Find MST using Kruskal's



# Kruskal's Algorithm: Correctness

It clearly generates a spanning tree. Call it  $T_K$ .

Suppose  $T_K$  is *not* minimum:

Pick another spanning tree  $T_{min}$  with *lower cost* than  $T_K$

Pick the smallest edge  $e_1=(u,v)$  in  $T_K$  that is not in  $T_{min}$

$T_{min}$  already has a path  $p$  in  $T_{min}$  from  $u$  to  $v$

⇒ Adding  $e_1$  to  $T_{min}$  will create a cycle in  $T_{min}$

Pick an edge  $e_2$  in  $p$  that Kruskal's algorithm considered *after* adding  $e_1$  (must exist:  $u$  and  $v$  unconnected when  $e_1$  considered)

⇒  $\text{cost}(e_2) \geq \text{cost}(e_1)$

⇒ can replace  $e_2$  with  $e_1$  in  $T_{min}$  without increasing cost!

Keep doing this until  $T_{min}$  is identical to  $T_K$

⇒  $T_K$  must also be minimal – contradiction!