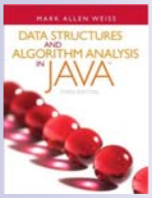


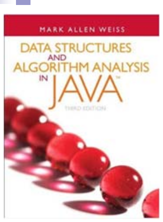

FLORIDA INTERNATIONAL UNIVERSITY

# Some Problems on Graphs (IV)

**Dr. Antonio L. Bajuelos**

FIU School of Computing & Information Sciences






## COP-3530 - Data Structures

**Module #7: Some Problems on Graphs (part IV)**

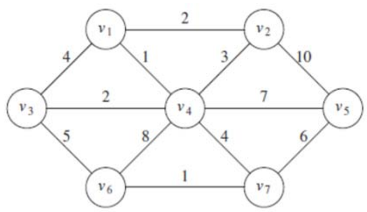
**Outline:**

- **Minimum Spanning Tree (MST):**
  - **Prim's Algorithm**
  - **Kruskal's Algorithm**

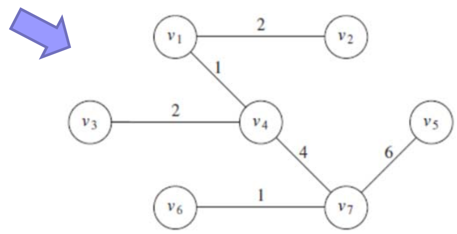
2

## The problem

- A **Minimum Spanning Tree (MST)** of an **undirected connected weighted graph** G is a **tree** formed from graph edges that connects <u>all the vertices of G at lowest total cost</u>.
- **Example:**



The original Graph

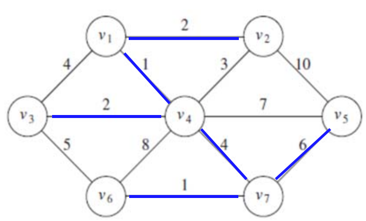


The Minimum Spanning Tree

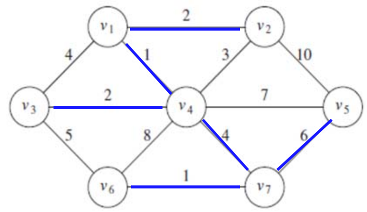


## The problem (more formally)

- Given an **undirected graph G = <V,E>**, find a **graph G*=<V,E*>** such that:
  - E* $\subseteq$ E
  - |E*| = |V| - 1
  - G* is connected
  - **G*** is a **ST** (or **Spanning Tree**)
- **Note that:**
  - For any spanning tree *T*, if an edge *e* that is not in *T* is added, a cycle is created.



## The problem (more formally)

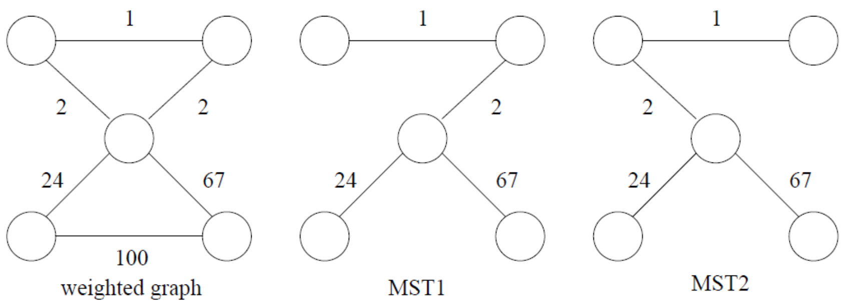



- If as a spanning tree is created, then
  - the edge that is added is the one of minimum cost that avoids creation of a cycle, then
    - the cost of the resulting spanning tree cannot be improved, because any replacement edge would have cost at least as much as an edge already in the spanning tree.



## The MST may be not unique!

- Example:



## Application of Minimum-cost Spanning Trees

Minimum-cost spanning trees have many applications.

**Examples:**

- Building cable networks that join n locations with minimum cost.
- Building a road network that joins n cities with minimum cost.
- Obtaining an independent set of circuit equations for an electrical network.
- In pattern recognition minimal spanning trees can be used to find noisy pixels.



## Prim's Algorithm*

- Based on expanding cloud of known vertices (**greedy strategy**)
- Basic ideas:
  - **Prim's algorithm** finds a **MST** by selecting edges from the graph one-by-one as follows:
    - It starts with a tree, T, consisting of the starting vertex, v, (v is **any vertex** in V)
    - Then, it adds the shortest edge (= edge with a **minimum-cost**) emanating from v that connects T to the rest of the graph.
    - It then moves to the added vertex and repeats the process.

*The algorithm was developed in 1930 by Czech mathematician **Vojtěch Jarník** and later rediscovered and republished by computer scientists **Robert C. Prim** in 1957 and **Edsger W. Dijkstra** in 1959.

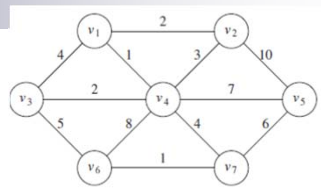
## Prim's Algorithm

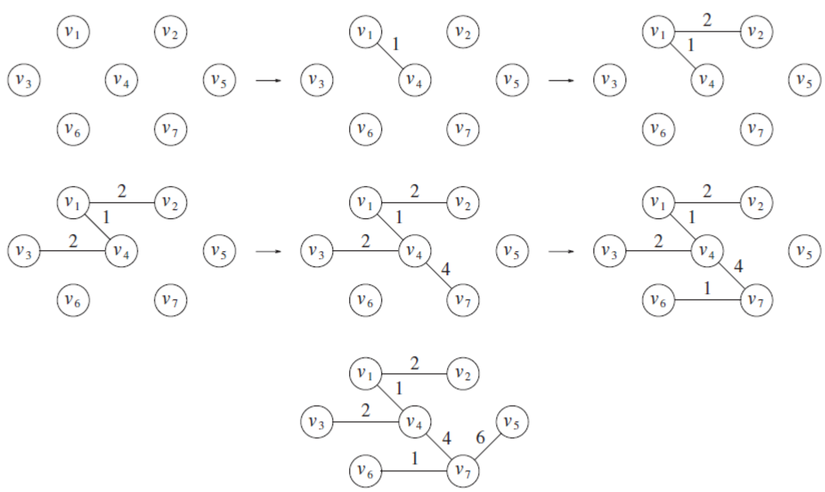
- General Pseudocode:

```
Consider a graph G = (V,E,W);
T: a tree consisting of only the starting vertex v;
while (T has fewer than |V| vertices)
{
    e ← find a smallest edge connecting T to G\T;
    add e to T;
}
```

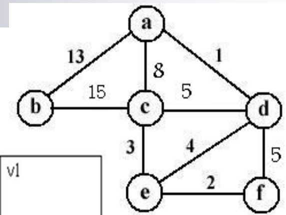


## Prim's Algorithm. Example



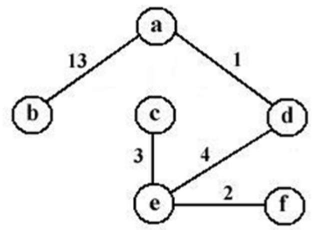
## Prim’s Algorithm. Example




- Starting at vertex **a**

| Pass: | initially | 1 | 2 | 3 | 4 | 5 | 6 | weight | vl |
|---|---|---|---|---|---|---|---|---|---|
| Active vertex: | | a | d | e | f | c | b | | |
| a | 0 | | | | | | | 0 | - |
| b | ∞ | 13 | 13 | 13 | 13 | 13 | | 13 | a |
| c | ∞ | 8 | 5 | 3 | 3 | | | 3 | e |
| d | ∞ | 1 | | | | | | 1 | a |
| e | ∞ | ∞ | 4 | | | | | 4 | d |
| f | ∞ | ∞ | 5 | 2 | | | | 2 | e |

- The resulting MST is:




11

## Pseudocode of Prim’s Algorithm.

1. For each vertex **v**, set:
   **v.cost = ∞** and **v.known = false**
1. Choose **any** vertex **v**
   a) Mark **v** as known
   b) For each edge **(v,u)** with weight **w**, set:
      **u.cost=w** and **u.prev=v**
2. **While** there are unknown vertices in G
   a) Select the unknown vertex **v** with lowest cost
   b) Mark **v** as known and add the edge **(v, v.prev)** to output
   c) For each edge **(v,u)** with weight **w**,

```
if(w < u.cost)
{
    u.cost = w;
    u.prev = v;
}
```

The **running time** is ***O*(|*V*|²)** without **heaps**, which is optimal for **dense graphs**, and ***O*(|*E*|log|*V*|)** using binary heaps, which is good for **sparse graphs**.

## Kruskal's Algorithm*.

- As a **Prim's algorithm** this method is based on expanding cloud of known vertices (**greedy strategy**)
- The Basic idea of the **Kruskal's algorithm** is:
  - Continually to select the edges in order of smallest weight
  - Accept an edge if it does not cause a cycle.

*This algorithm first appeared in *Proceedings of the American Mathematical Society*, pp. 48–50 in 1956, and was written by **Joseph Kruskal**.

13

## Kruskal's Algorithm.

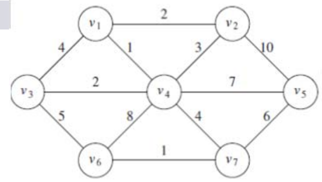
- General Pseudocode:

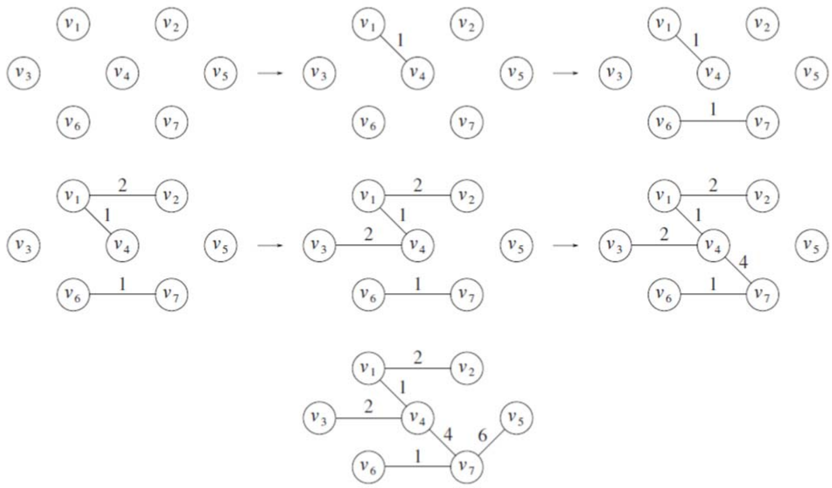
```
Sort edges by weight (better: put in min-heap)
T = empty
While output size < |V|-1
{
   Consider next smallest edge, e = (u,v)
   if(e does not create a cycle with edges in T)
       add e to T
  }
```

The worst-case running time of this algorithm is ***O*(|*E*|log|*E*|)**, which is dominated by the heap operations.

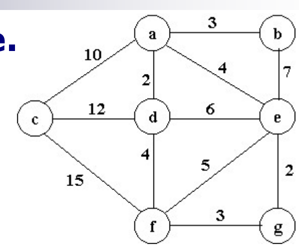
14

## Kruskal's Algorithm. Example.



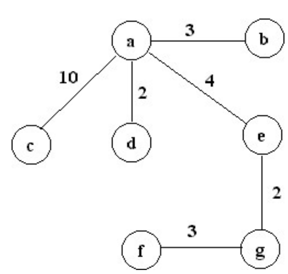


## Kruskal's Algorithm. Example.



| edge | ad | eg | ab | fg | ae | df | ef | de | be | ac | cd | cf |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| weight | 2 | 2 | 3 | 3 | 4 | 4 | 5 | 6 | 7 | 10 | 12 | 15 |
| insertion status | √ | √ | √ | √ | √ | x | x | x | x | √ | x | x |
| insertion order | 1 | 2 | 3 | 4 | 5 | | | | | 6 | | |

- The resulting **MST** is:



16

## Kruskal's Algorithm. Check for cycles.

- How to **check** for **cycles**?
- **Observations:**
  - At each step of the **Kruskal's algorithm** T is acyclic.
  - If u and v are previously in T, then adding the edge (u,v) to T creates a cycle.
- **Question:**
  - How to test whether u and v are in the same set?
- **Answer:**
  - Trivial algorithm: **O(|V|)**.
  - Non-trivial:
    - Use a **disjoint-set data structure!** Vertices in T are considered to be in same set.
    - Test if **Find-set(u) = Find-set(v)**??
    - **Find-set(u)** is **O(log n)**, where n is the size of the set.

17

## Java implementations.

- For a full version of the Prim's algorithm implementation please consult:

  http://algs4.cs.princeton.edu/43mst/PrimMST.java.html

  **(Authors: Robert Sedgewick and Kevin Wayne)**

- For a full version of the Kruskal's algorithm implementation please consult:

  http://algs4.cs.princeton.edu/43mst/KruskalMST.java.html

  **(Authors: Robert Sedgewick and Kevin Wayne)**

18