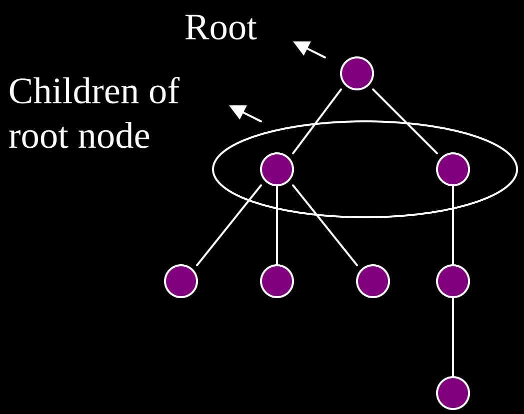
Minimum Spanning Trees

What is a MST (Minimum Spanning Tree) and how to find it with Prim's algorithm and Kruskal's algorithm

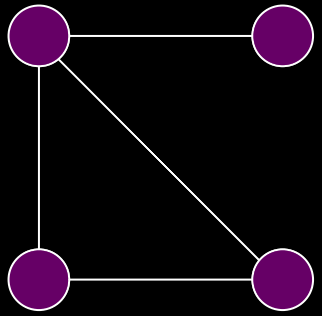
Tree

What is a tree?



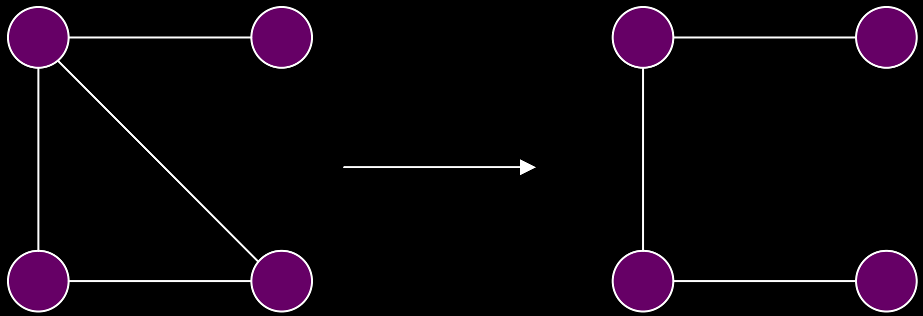
Spanning Tree

- Find a subgraph with minimum amount of edges.
- There must be a path between every pair of vertices.



Spanning Tree

Answer:



- This is called a spanning tree

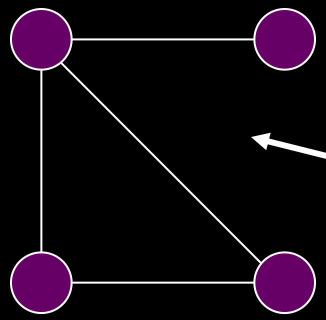
Spanning Tree

What is a spanning tree?

- Contains all the vertices of the graph and some or all of the edges
- Path from any node to any other node

Spanning Tree

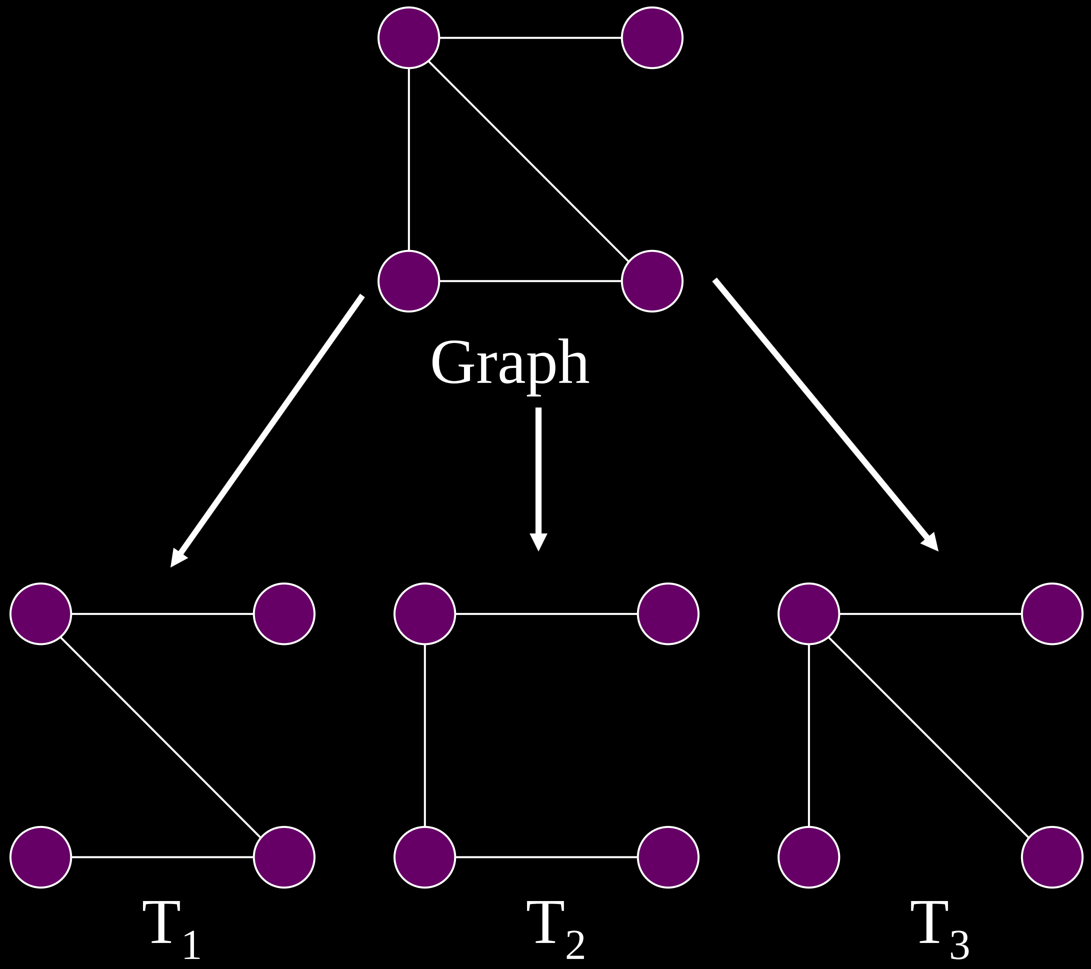
- A graph can have lots of spanning trees



This graph has

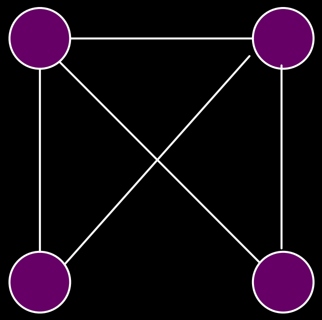
3 different spanning trees

Spanning Tree



Spanning Tree

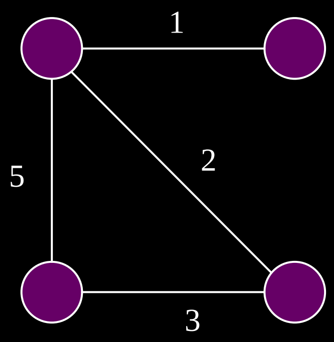
- How many spanning trees does this graph have?



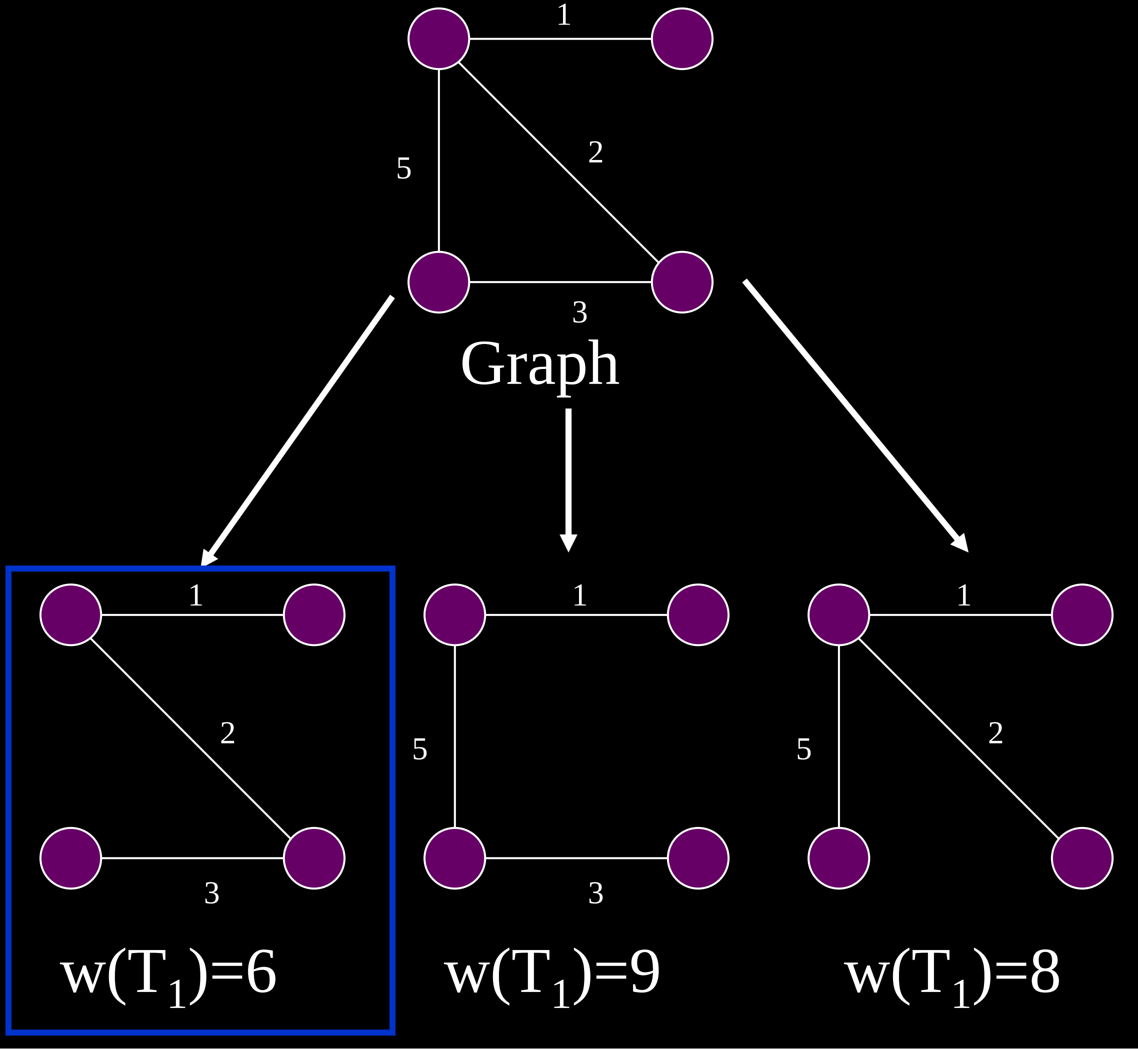
Answer: 8

Minimum spanning tree

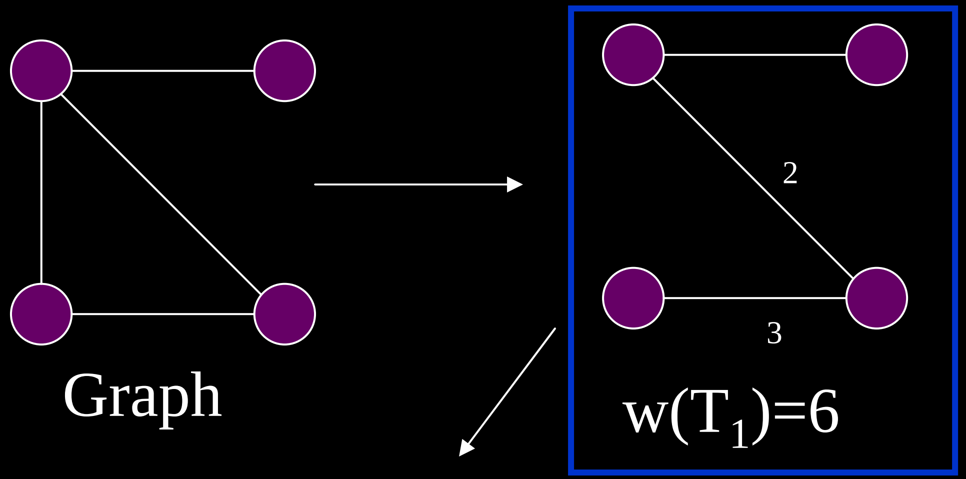
- Suppose we add weights to the graph
- Find the spanning tree with the minimum sum cost



Minimum spanning Tree



Minimum spanning Tree



- This is the minimum spanning tree of the graph

MST applications

- MST's can be applied to problems like phone networks, computer networks and trail networks

Sample problem

- Farmer John has ordered a high speed internet connection and is going to share his connectivity with the other farmers.
- To minimize cost, he wants to minimize the length of optical fiber to connect his farm to all the other farms

How to solve a MST

- One way to solve a MST, is to find all the spanning trees of the graph and find the minimum one, but
  - the number of spanning trees grows exponentially with the graph size
  - Generating all spanning trees for a weighted graph is not easy

Prim’s algorithm

- One way to solve a MST is with Prim’s algorithm
- Prim’s algorithm builds a tree one vertex at a time
- Start by selecting a vertex randomly
- On each iteration, simply add the nearest vertex not in the tree connected to a vertex in the tree
- The algorithm stops when all the graph’s vertices has been included in the tree being constructed
- This is a *greedy algorithm*

Prim’s algorithm

- The algorithm

```
let T be a single vertex
while (T has fewer than n vertices)
{
   find the smallest edge connecting a vertex
        not in the tree to a vertex in the tree
   add it to T
}
```

Prim's algorithm

- The algorithm with more detail:

```
Initialize MST to vertex 0.
priority[0] = 0
For all other vertices, set priority[i] = infinity
Initialize prioritySet to all vertices;
while prioritySet.notEmpty()
     v = remove minimal-priority vertex from
         prioritySet;
     for each neighbor u of v
         // Explore the edge.
         w = weight of edge (v, u)
         if w < priority[u]
             priority[u] = w
         endif
     endfor
endwhile
```

Running time: $O(n^2)$

Prim's algorithm

- $O(V^2)$ is too slow when finding the MST of a very large graph
- Some data structures can be used to speed it up
- Use a heap to remember, for each vertex, the smallest edge connecting T with that vertex.

Prim's algorithm (heap)

- Prim's algorithm with a heap:

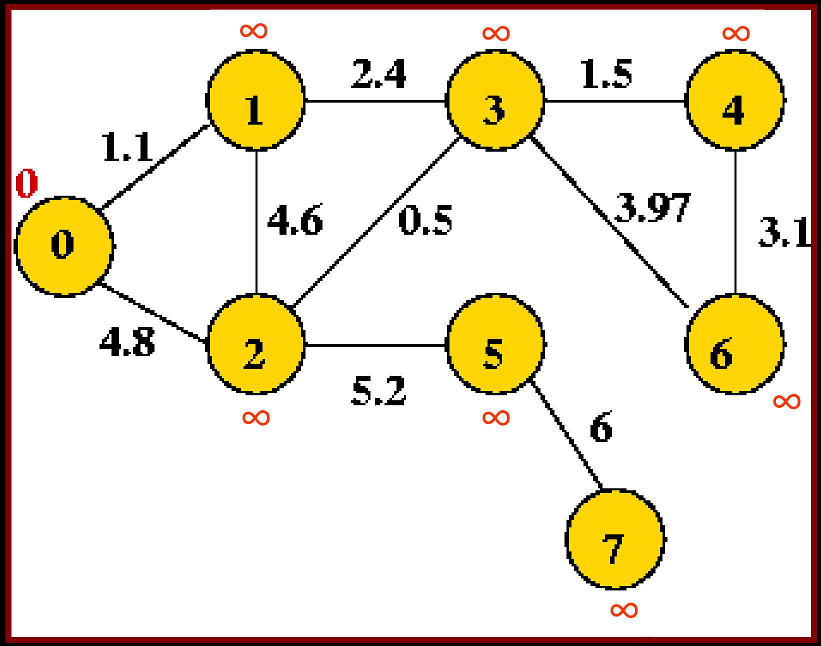
```
make a heap of values (vertex,edge,weight(edge))
  initially (v,-,infinity) for each vertex
let T be a single vertex x
for each edge f=(u,v)
  add (u,f,weight(f)) to heap

while (T has fewer than n vertices)
  let (v,e,weight(e)) be the edge  with the
                      smallest weight on the heap
  remove (v,e,weight(e)) from the heap
  add v and e to T
  for each edge f=(u,v)
    if u is not already in T
      find value (u,g,weight(g)) in heap
      if weight(f) < weight(g)
        replace (u,g,weight(g)) with
                (u,f,weight(f))
```

Running time: $O(m + n \log n)$

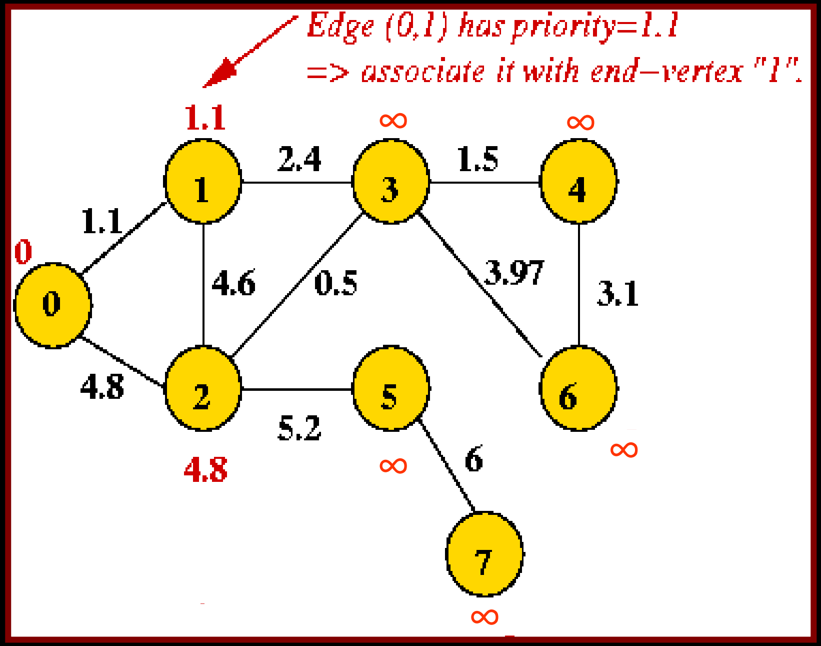
Prim's algorithm demonstration

- Initially, place vertex 0 in the MST and set the "priority" of each vertex to infinity.



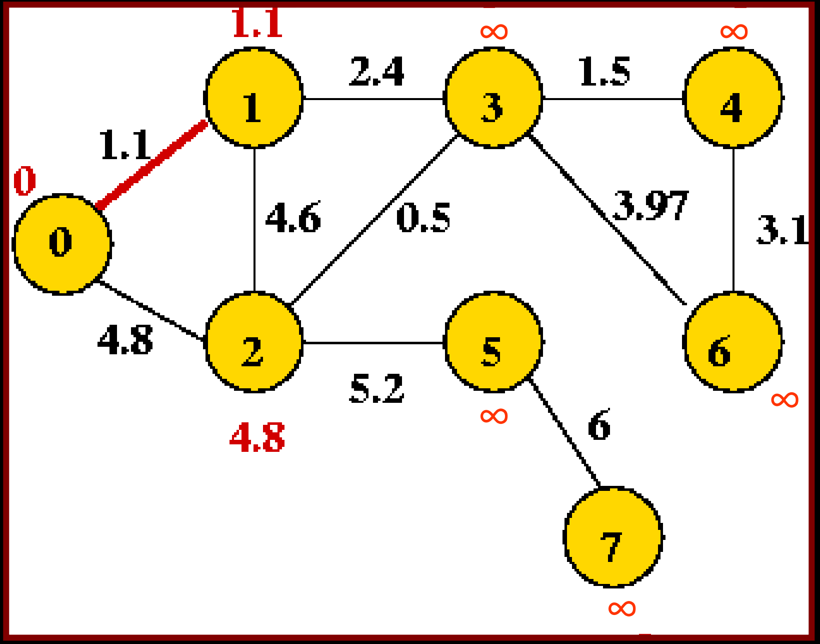
Prim's algorithm demonstration

- Explore edges from current MST: (0, 1) and (0, 2)



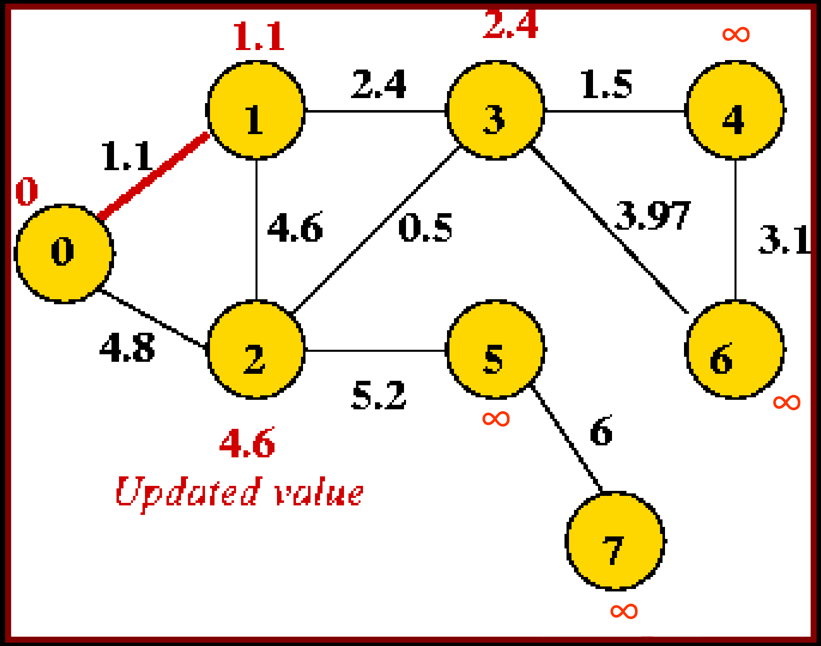
Prim's algorithm demonstration

- Pick lowest-weight edge (0, 1) to add => same as selecting lowest-priority vertex (vertex 1)



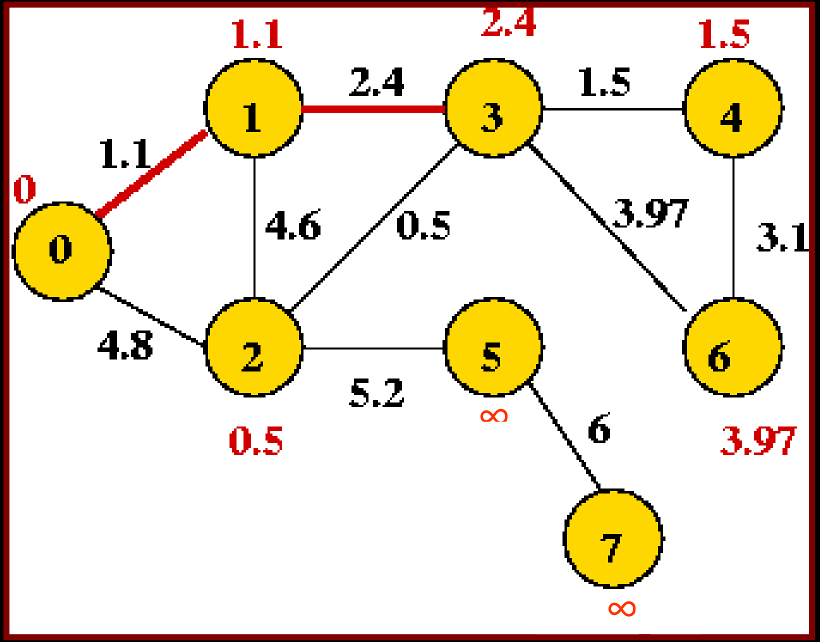
Prim’s algorithm demonstration

- Explore edges from newly-added vertex: (1,3), (1,2)



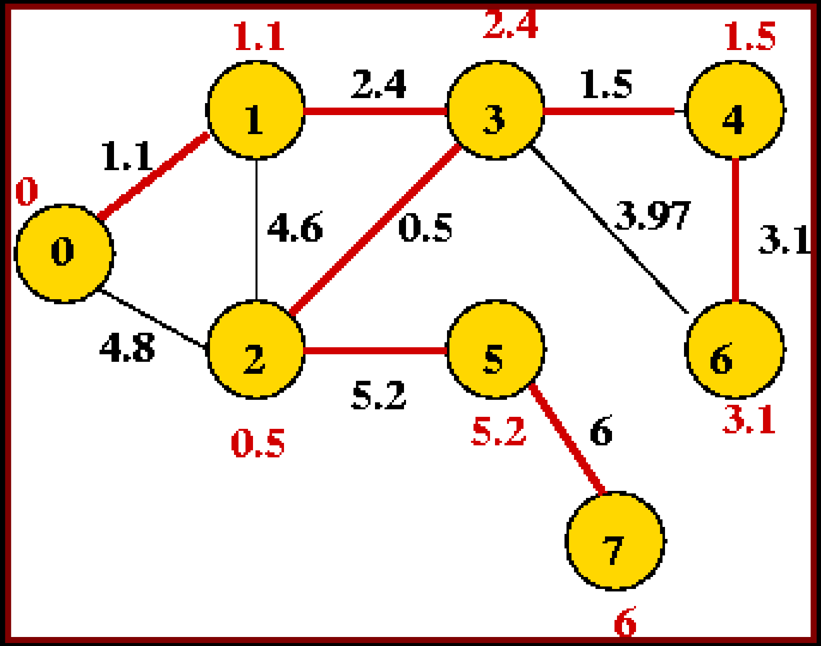
Prim's algorithm demonstration

- Pick vertex with lowest priority (vertex 3) and explore its edges:



Prim's algorithm demonstration

- Continuing, we add vertices 2, 4, 6, 5 and 7:



Kruskal’s algorithm

- Another way to solve a MST is with Kruskal’s algorithm
- Kruskal is easier to code and easier to understand
- This is a *greedy algorithm*
- Basics of algorithm:
  - Sort edges in order of increasing weight.
  - Process edges in sort-order.
  - For each edge, add it to the MST if it does not cause a cycle.

Kruskal's algorithm

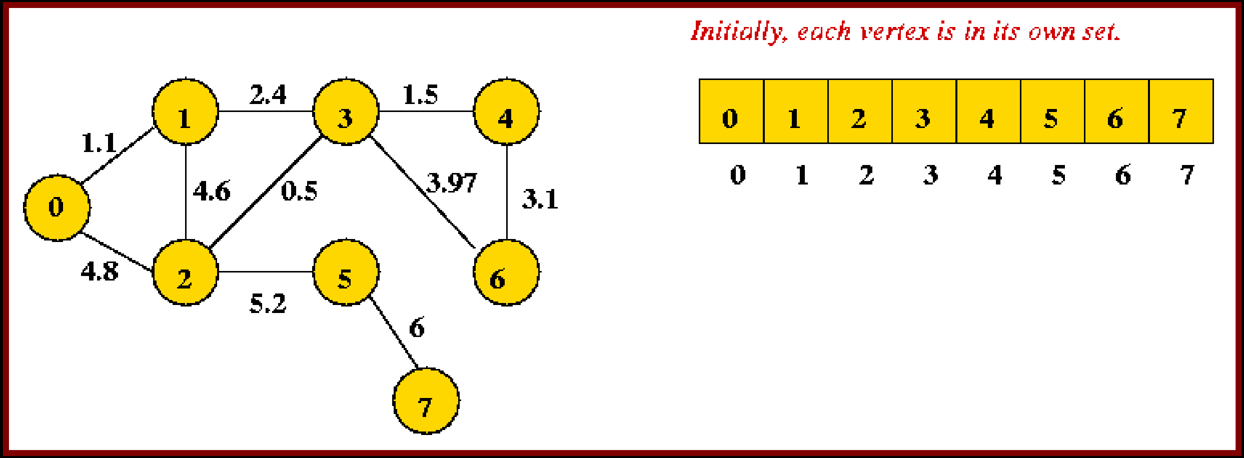
- More advanched algorithm:

```
Initialize MST to be empty;
Place each vertex in its own set;
Sort edges of G in increasing-order;
for each edge e = (u,v) in order
     if u and v are not in the same set
        Add e to MST;
        Compute the union of the two sets;
     endif
endfor
return MST
```

Running time: O(m log m)

Kruskal's algorithm example

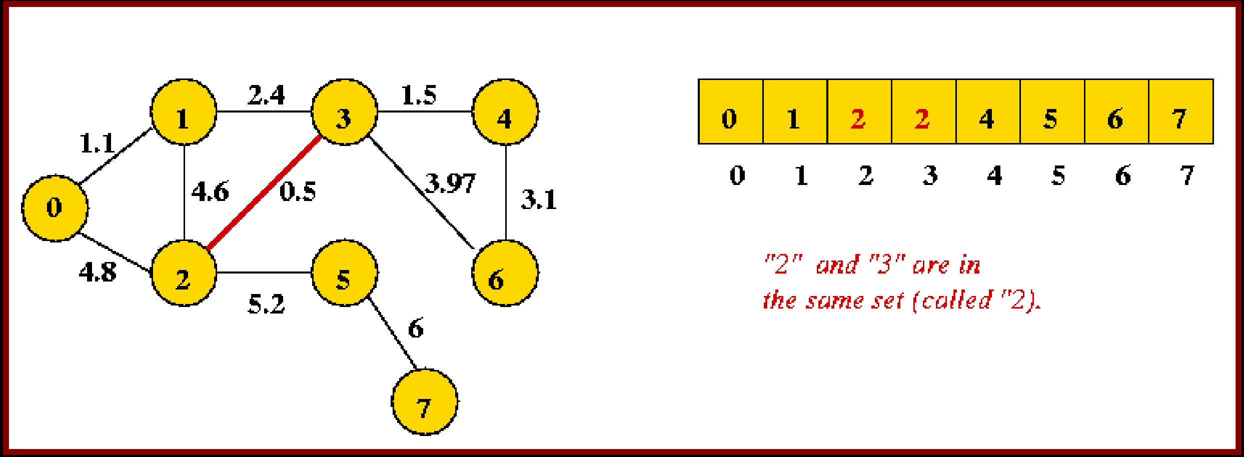
- Initially:



- Sort order of edges: **(2, 3), (0, 1), (3, 4), (1, 3), (4, 6), (3, 6), (1, 2), (0, 2), (2, 5), (5, 7)**

Kruskal's algorithm example

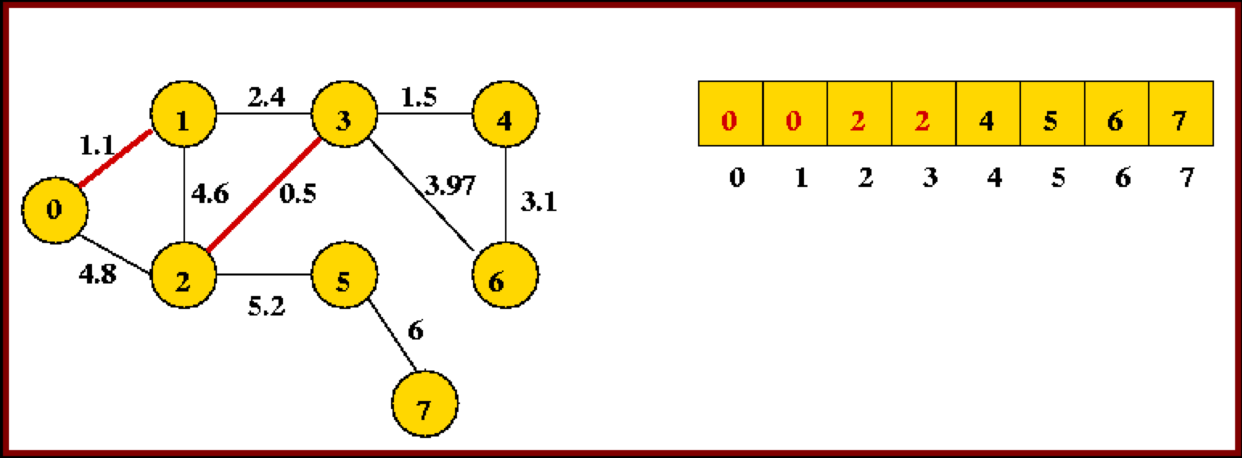
- First edge to add joins "2" and "3" (no cycle):



- Sort order of edges: <u>(2, 3)</u>, **(0, 1), (3, 4), (1, 3), (4, 6), (3, 6), (1, 2), (0, 2), (2, 5), (5, 7)**

Kruskal’s algorithm example

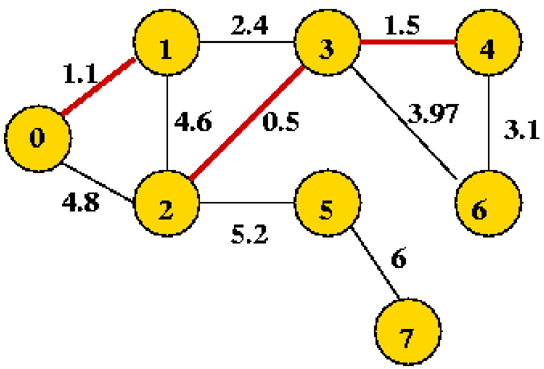
- Next edge in sort order: (0, 1):



- Sort order of edges: ~~(2, 3),~~ **(0, 1)**, **(3, 4), (1, 3), (4, 6), (3, 6), (1, 2), (0, 2), (2, 5), (5, 7)**

Kruskal's algorithm example

- Next edge in sort order: (3, 4):

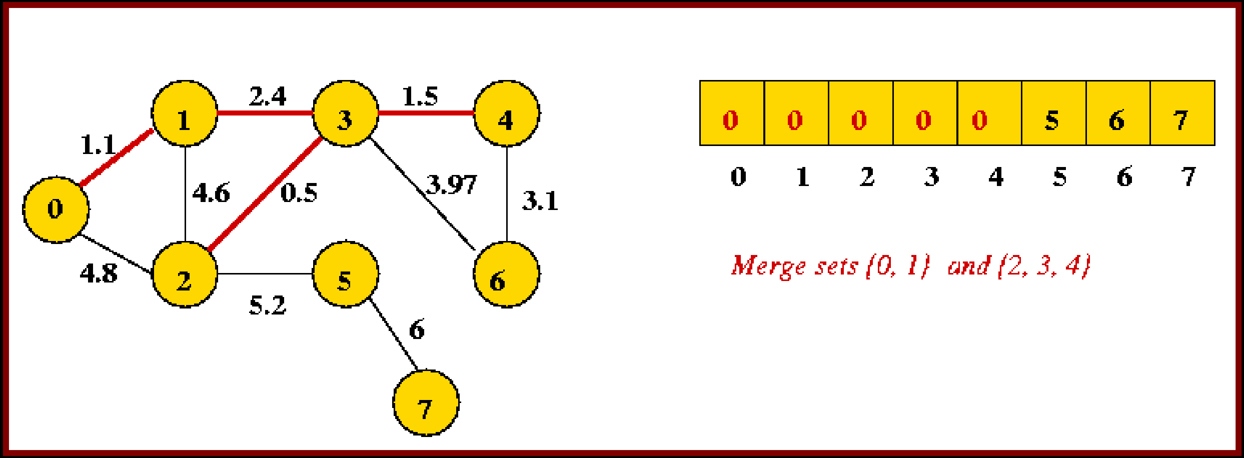


0	0	2	2	2	5	6	7
0	1	2	3	4	5	6	7

- Sort order of edges: ~~(2, 3), (0, 1)~~, <u>(3, 4)</u>, **(1, 3), (4, 6), (3, 6), (1, 2), (0, 2), (2, 5), (5, 7)**

Kruskal's algorithm example

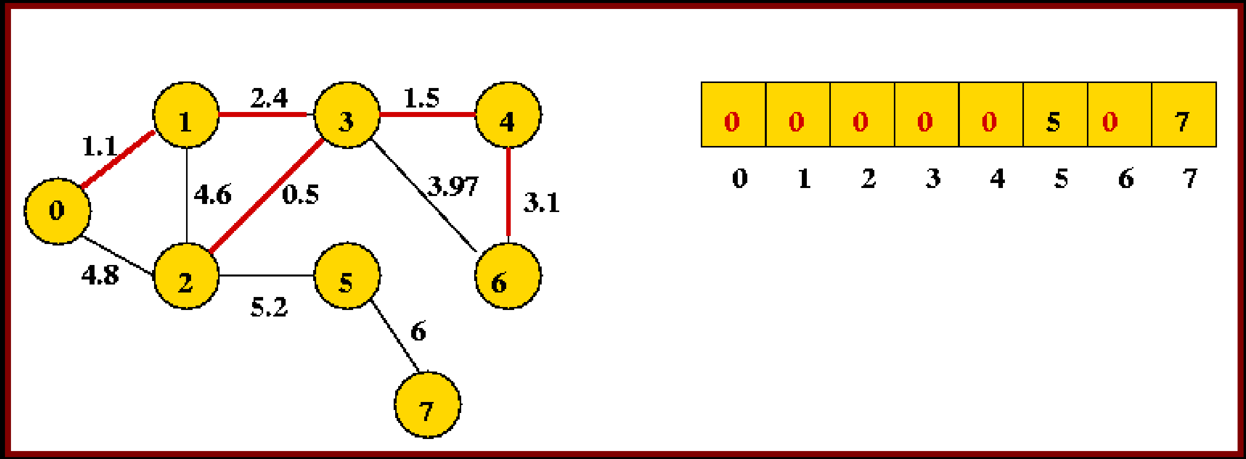
- Next edge in sort order: (1, 3): merges two sets (union)



- Sort order of edges: ~~(2, 3), (0, 1), (3, 4)~~, <u>(1, 3)</u>, **(4, 6), (3, 6), (1, 2), (0, 2), (2, 5), (5, 7)**

Kruskal’s algorithm example

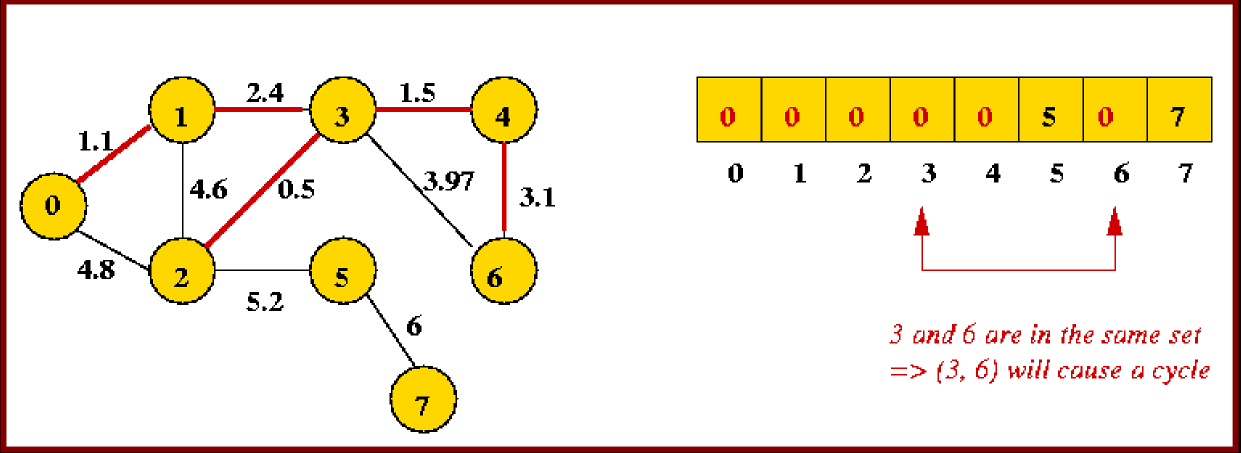
- Next edge in sort order: (4, 6):



- Sort order of edges: ~~(2, 3), (0, 1), (3, 4), (1, 3),~~ **(4, 6)**, **(3, 6), (1, 2), (0, 2), (2, 5), (5, 7)**

Kruskal's algorithm example

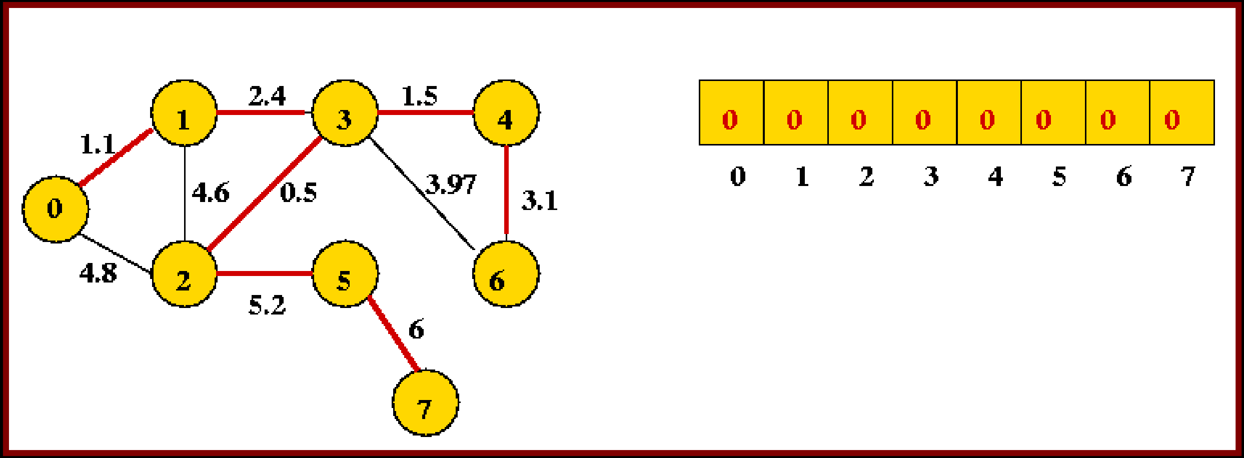
- Next edge in sort order: (3, 6): cannot be added



- Next two edges also cannot be added: (0, 2) and (1, 2).
- Sort order of edges: ~~(2, 3), (0, 1), (3, 4), (1, 3), (4, 6),~~ (3, 6), (1, 2), (0, 2), (2, 5), (5, 7)

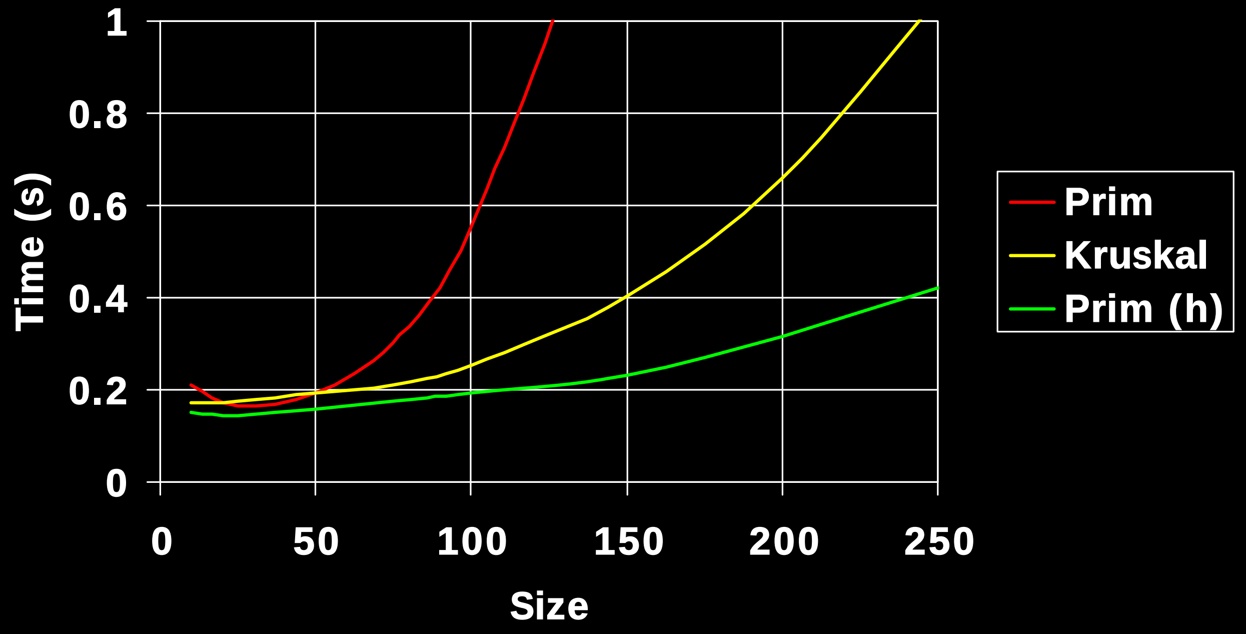
Kruskal’s algorithm example

- Finally, add (2, 5) and (5, 7):



- Sort order of edges: ~~(2, 3), (0, 1), (3, 4), (1, 3), (4, 6), (3, 6), (1, 2), (0, 2), (2, 5), (5, 7)~~

Time comparison



Comparison

- Prim with a heap is faster than Kruskal, but Kruskal is easier to code.
- Code both and choose the one you prefer.

Building Roads
(USACO Silver Dec. '07 competition)

Building Roads

- Farmer John had acquired several new farms!
- He wants to connect the farms with roads so that he can travel from any farm to any other farm via a sequence of roads
- Roads already connect some of the farms.

Building Roads
(USACO Silver Dec. '07 competition)

- This can be done by using any algorithm to find a MST.
- The edge weights are the Euclidean distances between the farms
- The easiest option is to use the $O(n^2)$ version of Prim's algorithm.