Prim’s Minimum Spanning Tree implemented with a MinHeap

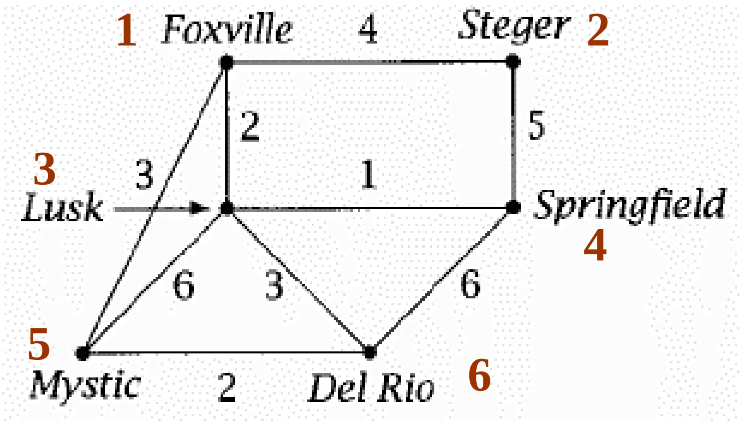
C++ Object Oriented Programming

Pei-yih Ting

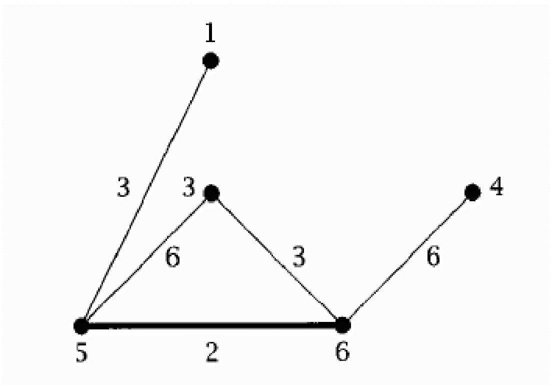
NTOU CS

Prim’s MST

- In JohnsonBaugh’s “Algorithms”
  Minimum Spanning Tree starting with vertex 5 (Mystic):
  Prim’s algorithm

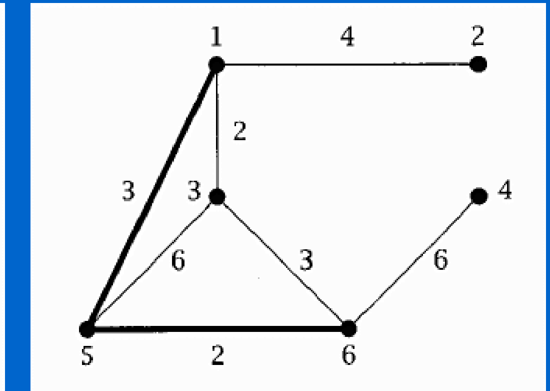


➊



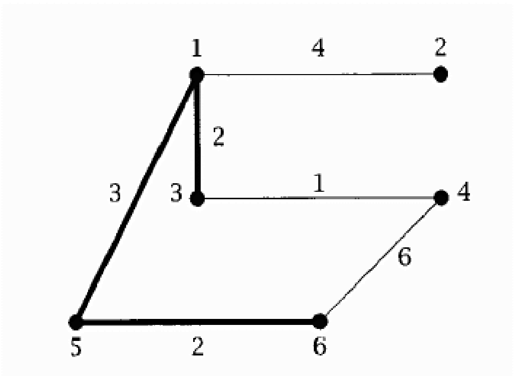
(a)

➋



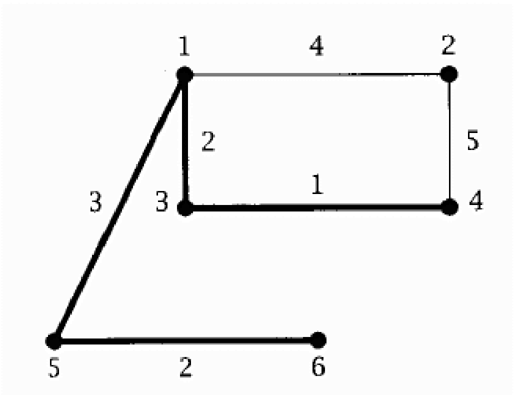
(b)

➌



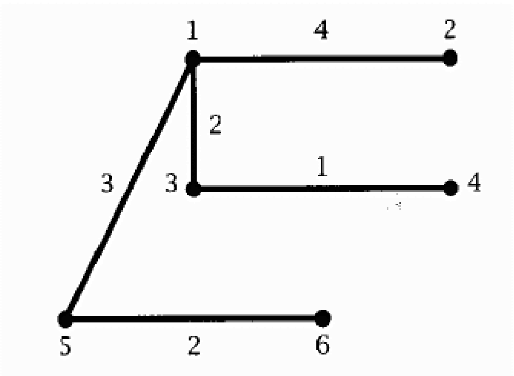
(c)

➍



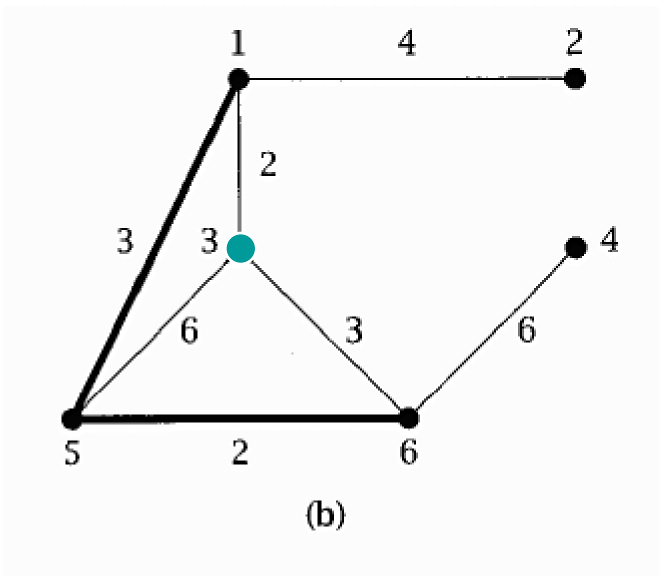
(d)

➎



(e)

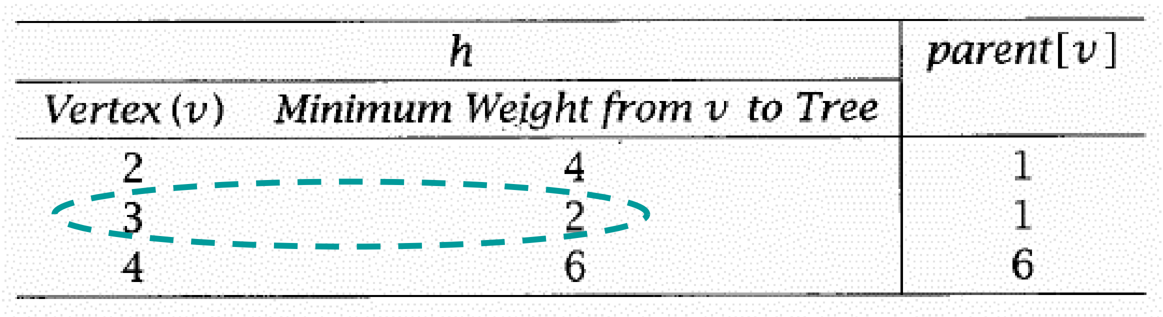
Prim’s MST (cont’d)



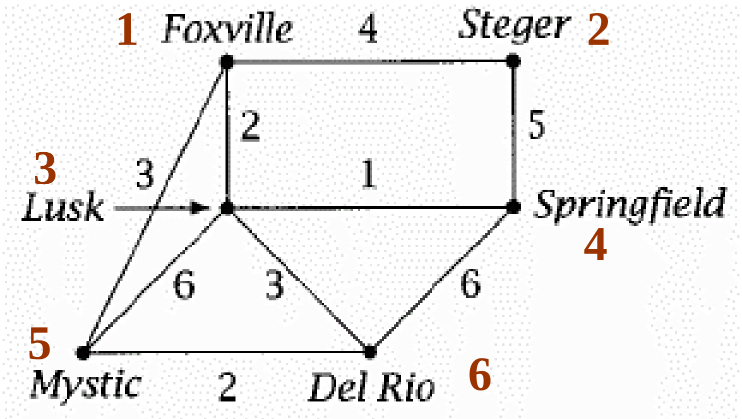
(b)

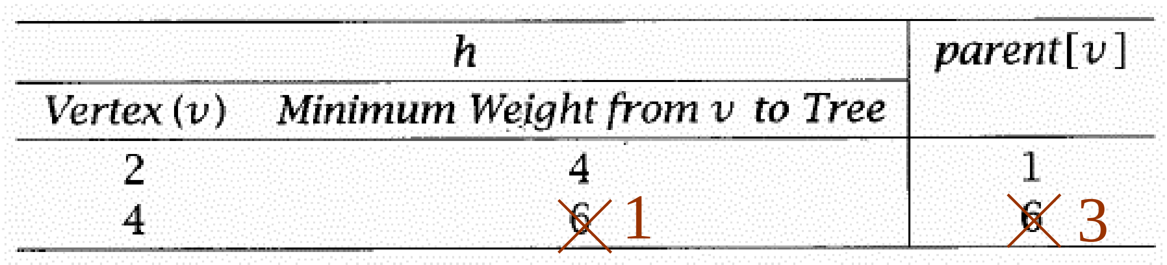
h: is a list of vertices *v* not in the tree and the minimum weight of an edge from *v* to a vertex *parent*[*v*] in the tree

parent: which edges give minimum weights

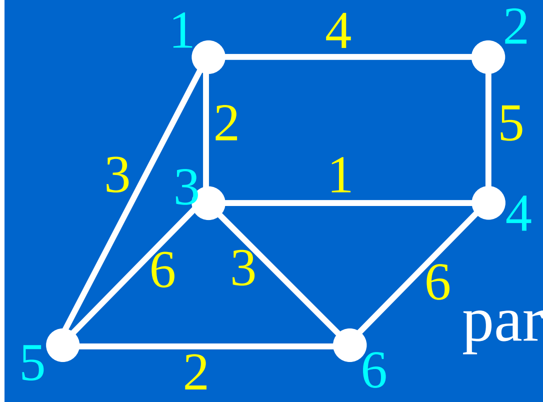


h		*parent*[*v*]
Vertex (*v*)	*Minimum Weight from v to Tree*	
2	4	1
3	2	1
4	6	6





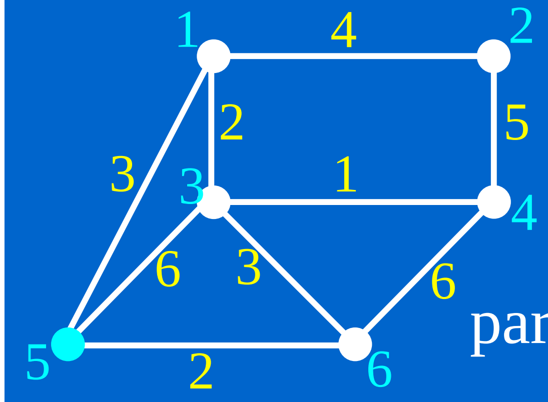
h		*parent*[*v*]
Vertex (*v*)	*Minimum Weight from v to Tree*	
2	4	1
4	~~6~~ 1	~~6~~ 3



h:{(1,∞),(2,∞),(3,∞),
(4,∞),(5,0),(6,∞)}

parent:

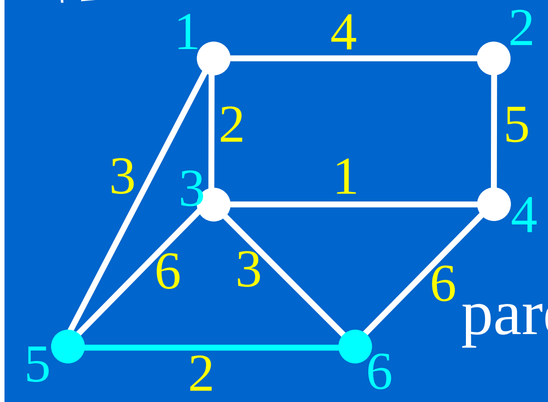
				0	



h: {(1,3),(2,∞),(3,6),
(4,∞),(6,2)}

parent:

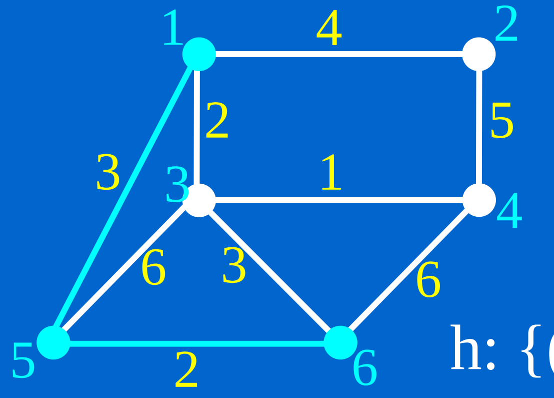
5		5		0	5



h: {(1,3),(2,∞),
(3,3),(4,6)}

parent:

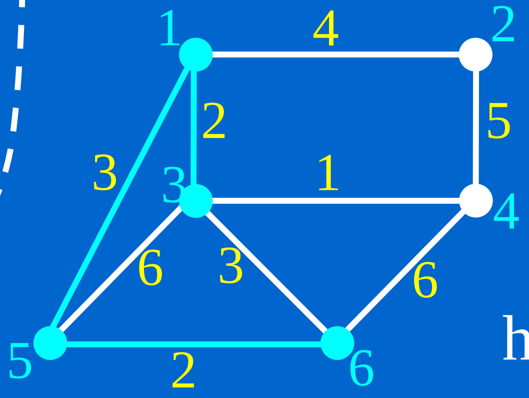
5		6	6	0	5



h: {(2,4),
(3,2),(4,6)}

parent:

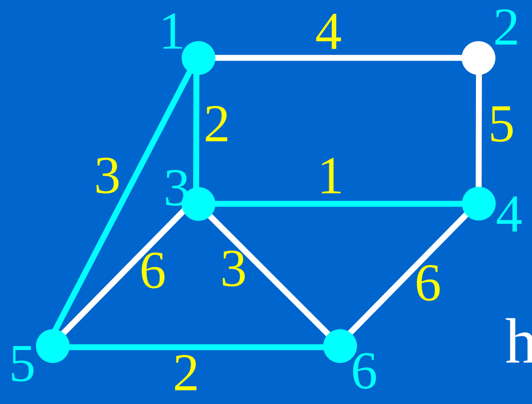
5	1	1	6	0	5



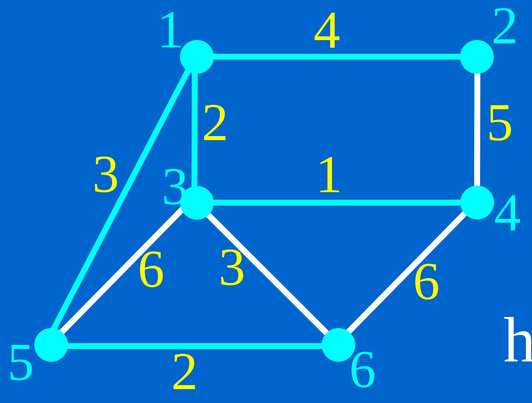
h: {(2,4),
(4,1)}

parent:

5	1	1	3	0	5



h: {(2,4)}



h: { }

parent:

5	1	1	3	0	5

parent:

5	1	1	3	0	5

adj

1:	2	3	5	
2:	1	4		
3:	1	4	5	6
4:	2	3	6	
5:	1	3	6	
6:	3	4	5	

Prim’s MST (cont’)

```
prim(adj, start, parent) {
    n = adj.last
    for i = 1 to n
        key[i] = ∝
    key[start] = 0
    parent[start] = 0
    h.init(key, n)
    for i = 1 to n {
        v = h.del()
        ref = adj[v]
        while (ref != null) {
            w = ref.ver
            if (h.isin(w) &&
                ref.weight < h.keyval(w)) {
                parent[w] = v
                h.decrease(w, ref.weight)
            }
            ref = ref.next
        }
    }
}
```

h is an **abstract data type** that supports the following operations

h.**init**(key, n): initializes h to the values in key

h.**del**(): deletes the item in h with the smallest weight and returns the vertex

h.**isin**(w): returns true if vertex w is in h

h.**keyval**(w): returns the weight corresponding to vertex w

h.**decrease**(w, new_weight): changes the weight of w to new_weight (smaller)