PRIM'S MINIMUM SPANNING TREE ALGORITHM

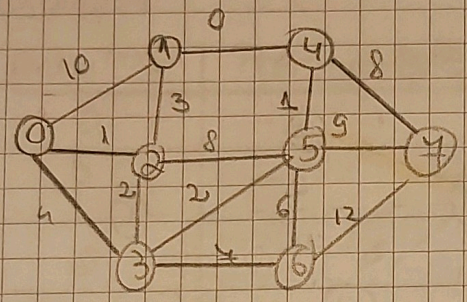
Lazy version: $O(E * \log(E))$ ↳ works well on dense graphs

Eager version: $O(E * \log(V))$

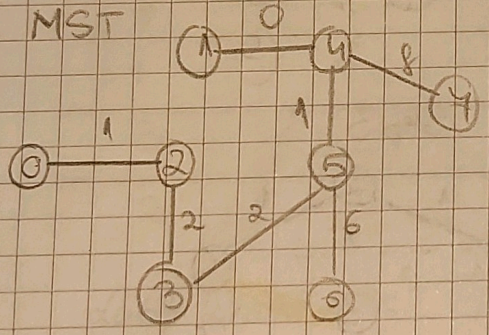
Lazy Prim Overview

↳ PQ (priority queue) that sorts edges based on min edge cost (used to determine the next node to visit and the edge to get there)

Start the algorithm on any node s. Mark s as visited and iterate over all edges of s, adding them to the PQ. While the PQ is not empty and a MST has not been formed, dequeue the next cheapest edge from the PQ. If the dequeued edge is outdated (the node it points to has already been visited) then skip it and poll again. Else, mark the current node as visited and add the selected edge to the MST



MST



Edges in PQ
start, end, cost

(0,1,10)
(0,2,1)
(0,3,4)
(2,1,3)
(2,5,8)
(2,3,2)
(2,0,1) → 0 is visited already
(3,5,2)
(3,6,7)
(5,4,1)
(5,6,6)
(5,7,9)
(4,1,8)
(4,7,8)
(6,7,12)

The algorithm

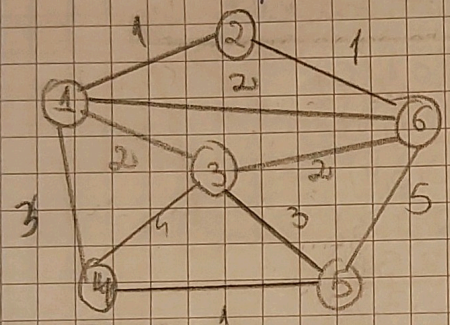
Input:
G: directed graph with costs

Output:
edges: a collection of edges forming a minimum cost spanning tree

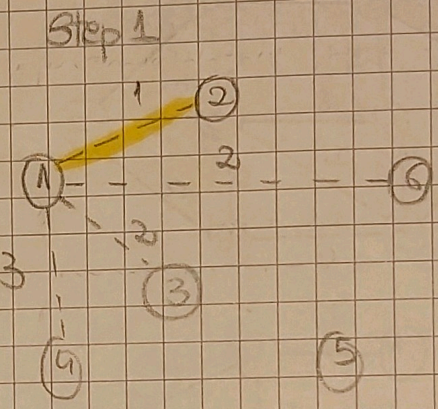
Algorithm
```
PriorityQueue Q
Dictionary prev, dist
edges = ∅

choose in X arbitrarily:
vertices = {s}
for x in N(s) do
    dist[x] = cost(x,s)
    prev[x] = s
    Q.enqueue(x, d[x]) //2nd argument is priority
```

```
while not Q.isEmpty() do
    x = Q.dequeue()   //dequeue the element with minimum
                      //value of priority
    if x ∉ vertices then // not a part of MST yet
        edges.add({x, prev[x]})
        vertices.add(x)
        for y in N(x) do
            if y not in dist.keys() or cost(x,y) < dist[y] then
                dist[y] = cost(x,y)
                Q.enqueue(y, dist[y])
                prev[y] = x
```

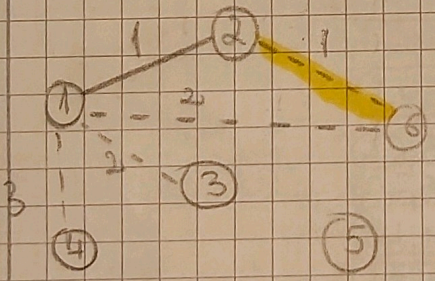
Example



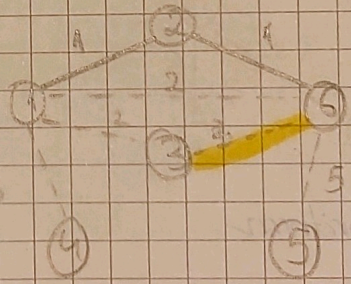
Step 1



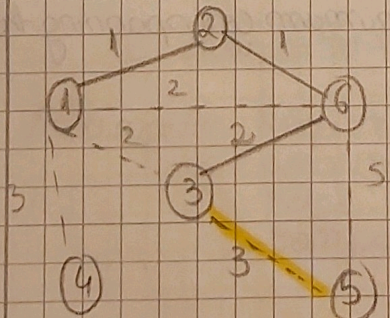
Step 2



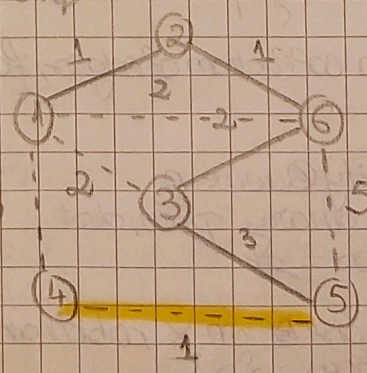
Step 3



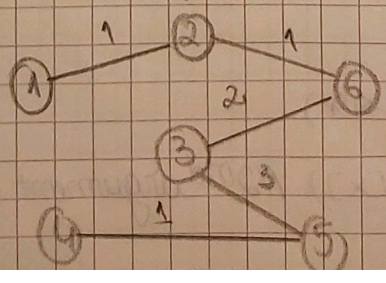
Step 4



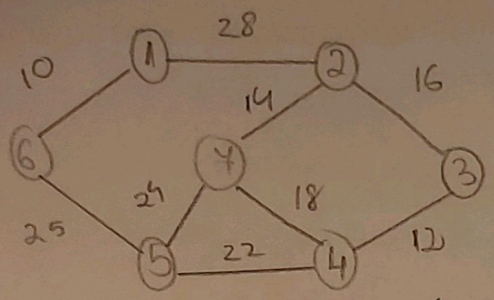
Step 5



MST



MORE PRIM



Start from vertex 1 → (1,2): 28
(1,6): 10 ✓

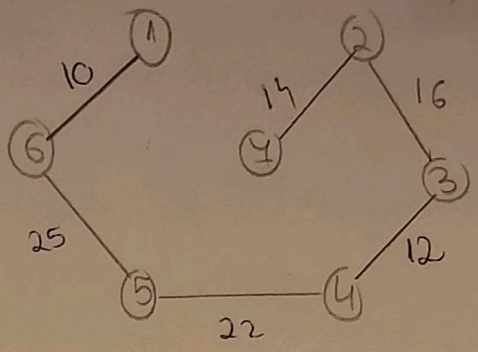
Vertex 6: (6,5): 25 ✓

Vertex 5: (5,7): 24 ✓
(5,4): 22 ✓

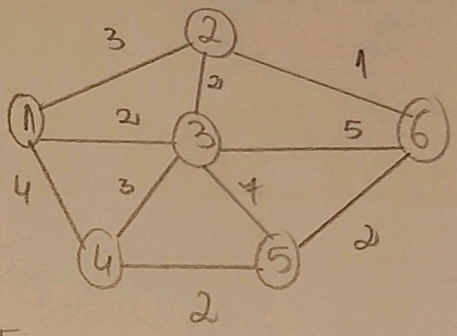
Vertex 4: (4,3): 12 ✓
(4,7): 18

Vertex 3: (3,2): 16 ✓

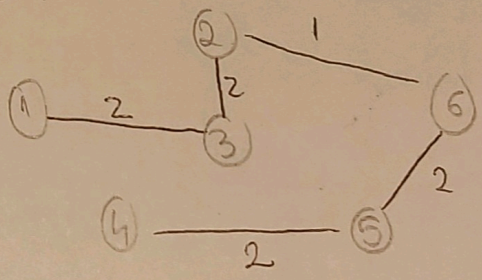
Vertex 2: (2,7): 14 ✓
(2,1): 28 ✗



MST of cost
10+25+22+12+16+14=99



MST:



Start from vertex 1

PQ: (3,2), (2,3), (4,4)
dist: 2→3, 3→2, 4→4
prev: 2:1, 3:1, 4:1 ; vertices: 1 ; edges:

x = 3 → not in vertices
edges: (1,3) ; vertices: 1, 3

y = 2 ✗ y = 4 ✗
y = 1 ✗ ~~y = 5~~

- Vertex 1:
PQ: ((1,3),2), ((1,2),3), ((1,4),4)

Vertex 3:
PQ: ((3,2),2), ((1,2),3), ((3,4),3), ((1,4),4), ((3,6),5), ((3,5),7)

Vertex 2: PQ: ((2,6),1), ((1,2),3), ((3,4),3), ((1,4),4), ((3,6),5), ((3,5),7)

Vertex 6: PQ ((6,5),2), ((1,2),3), ((3,4),3), ((1,4),4), ((3,6),5), ((3,5),7)

Vertex 5: PQ: ((4,5),2), ((1,2),3) . . .