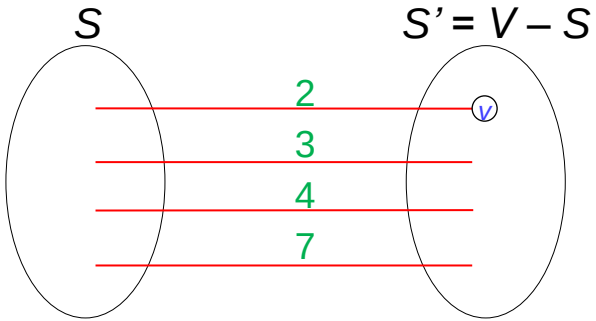
Prim's Algorithm

Algorithm

- What we have to maintain for Prim's algorithm?

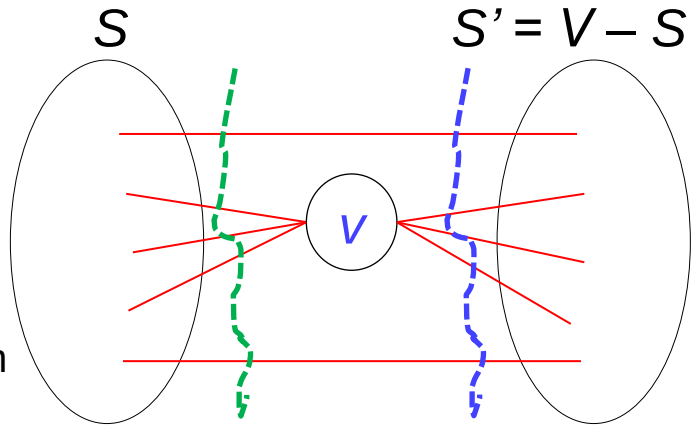
- At every time instance, we will be having two sets in the cut: **(*S, S'*)**
  - ***S*** will contain the vertices which we have reached already from the ***root***
  - ***S'*** will contain the remaining vertices
  - This can be maintained by keeping one bit with every vertex (0 or 1)



- What we have to do at each step of Prim's algorithm?
  - We have to investigate all the ***edges of the cut***
  - Find the minimum cost
  - How to do this?

Algorithm

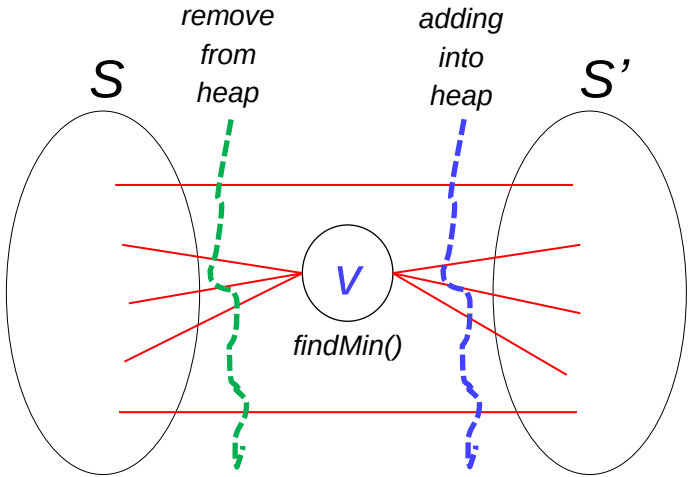
- When ***v*** will move from ***S'*** to ***S***, there are two types of ***edges in the cut***



- At this time instance, we know the minimum cost among the LHS edges (green cross)
  - The minimum has to be an edge incident to vertex ***v***
- After moving ***v*** from ***S*** to ***S'***, we want to know the minimum cost among the RHS edges (blue cross)
- What data structure to use? Heap?
  - Heap is the data structure in which each of the elements has certain priority or certain key value
  - We can use an *deleteMin()* operation which will remove the minimum element
  - We insert element into the heap, can remove elements from the heap
  - All operations take *log N* time
  - The *findMin()* operation takes *O(1)* time

Algorithm

- Why Heap?
  - Suppose we want to keep a heap which contains the edges of green cross (***edges of the cut***)
  - Then, using *findMin()*, we can find out what is the minimum edge
  - Suppose ***v*** is chosen
  - The edges which are going in ***S***, we have to remove them from this heap
  - The edges which are going from ***v*** to ***S'***, we have to add them into the heap



- So at any point, what does the heap contain?
  - Exactly the ***edges of the cut***

- Time taken by heap operations?
  - $\sum_v deg(v)\ log\ V + V.\ O(1)$

PseudoCode

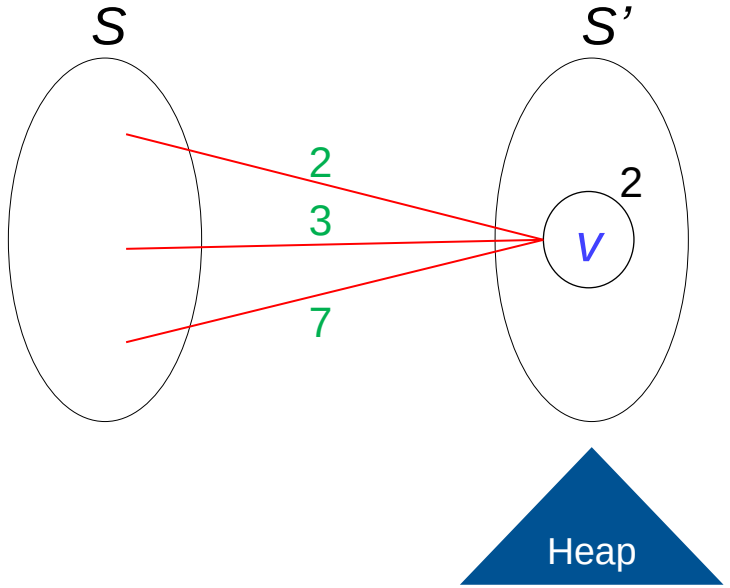
```
Prims(G, c, root)

for all v ∈ G
    S[v] = 0
    S[root] = 1
for all edge e incident to root do
    H.insert[e]
while ! H.empty() do {
    f = H.findMin()
    Let w be the end vertex of f such that S[w] = 0
    for all e = (w, x) adjacent to w do
        if S[x] == 0 then H.insert[e]
        else H.delete[e]
    S[w] = 1
}
```

- S[ ] is an array, H is the heap
- S[***v***] = 1 if ***v*** ∈ S
  = 0 if ***v*** ∈ S'
- How many times the loops are going to be executed?
  - While loop?
    - ***V – 1*** times
  - For loop?
    - $\sum_v$ ***deg(v) log V*** = ***E log V*** times
- Each *H.findMin()* takes how much time?
  - ***O(1)*** time

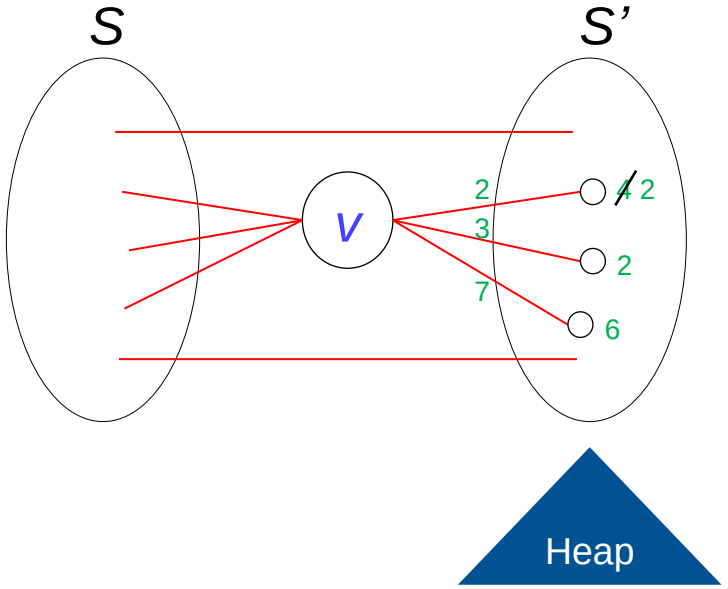
PseudoCode: Another Version

- We have set ***S*** and ***S'***
- Each vertex in ***S'*** has a label on it
  - The minimum of the costs of the edges from ***v*** to any vertex at ***S*** set
  - We are going to maintain a heap of vertices in ***S'***
    - One element corresponding to each vertex
    - The priority of the element will be the minimum of the costs of the edges from that vertex to any vertex at ***S*** set
  - What will be the minimum element in the heap?
    - The vertex which has the minimum edge incident at it among all the edges in the cut
  - If there is a vertex which has no edge going across the cut, its label will be ∞



PseudoCode: Another Version

- We have set ***S*** and ***S'***
- Each vertex in ***S'*** has a label on it
  - When we move a vertex across the cut, we have to change the label of the vertices in ***S'*** which are adjacent to this recently moved vertex
  - H.findMin()
  - H.decresePriority()



PseudoCode: Another Version

```
Prims(G, c, root)

for all v ∈ G
    S[v] = 0
    H.insert[v, ∞]
    H.decresePriority[root, 0]

while ! H.empty() do {
    v = H.findMin()
    for all w adjacent to v do
        if S[w] == 0 then
            if label[w] > c[v, w] then
                label[w] = c[v, w]
                H.decresePriority[w, label[w]]
    S[w] = 1
    H.delete[v]
}
```

- S[ ] is an array, H is the heap
- S[***v***] = 1 if ***v*** ∈ S
  = 0 if ***v*** ∈ S'
- label[] array is keeping track of the minimum label of each vertex in ***S'***
- How many times the loops are going to be executed?
  - While loop?
    - ***V – 1*** times
  - For loop?
    - $\sum_v$ ***deg(v) log V*** = ***E log V*** times
- Each *H.findMin()* takes how much time?
  - ***O(1)*** time

Next Lecture

Single Source Shortest Path Algorithms

Thank you for your attention...

Any question?

Contact:
Department of Information Technology, NITK Surathkal, India
6th Floor, Room: 13
Phone: +91-9477678768
E-mail: shrutilipi@nitk.edu.in, shrutilipi.bhattacharjee@tum.de