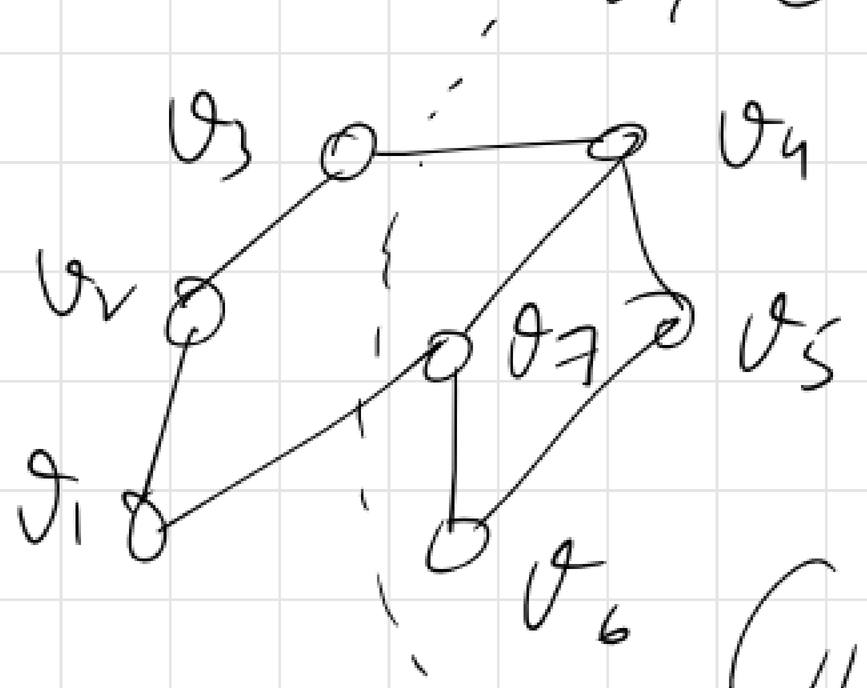
"Prims Algorithm"

Cut: A cut in G is a partition of V into V_1 and V_2 (two parts)

$$V_1 \cap V_2 = \emptyset$$
$$V_1 \cup V_2 = V$$

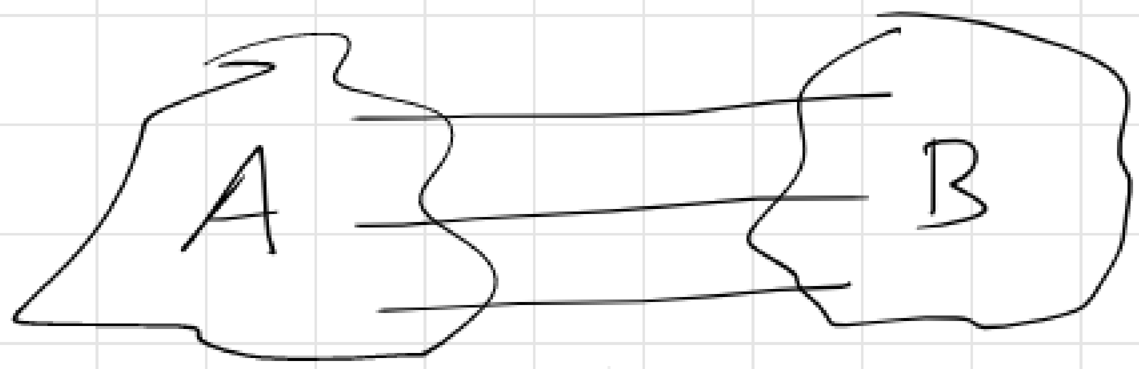


$V_1 = v_1\ v_2\ v_3$

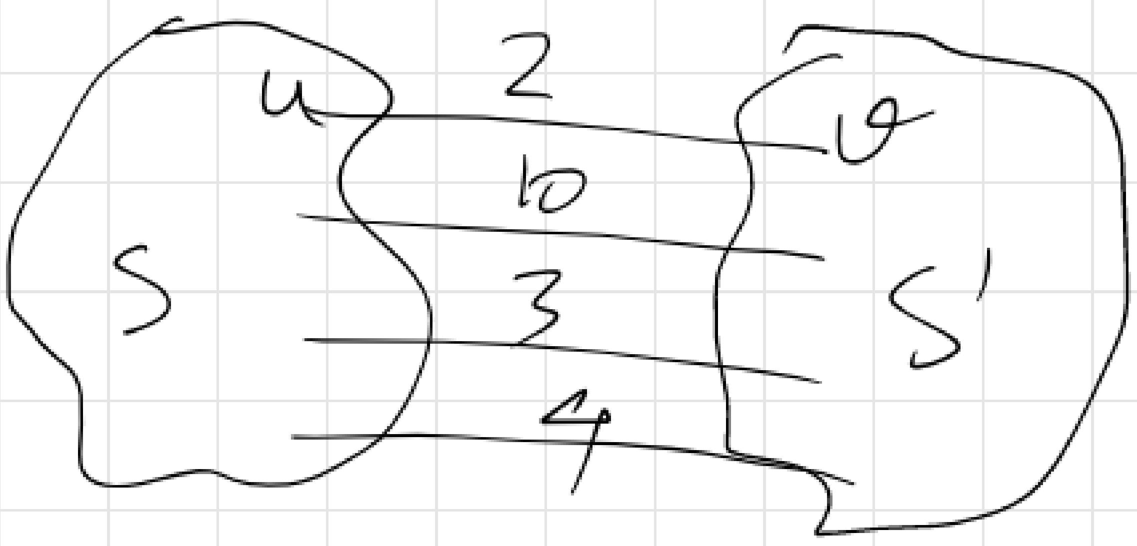
$V_2 = v_4\ v_5\ v_6\ v_7$

[# possible cuts $= 2^{n-1} - 1$]

Edges in the cut



Assume cut of G S and S'



Claim: For any S, S' cut, the minimum edge of the cut belongs to the MST

Imp: Note that there can be multiple cut edges in MST.

Proof of the claim

let MST T does not contain the min cut edge e.

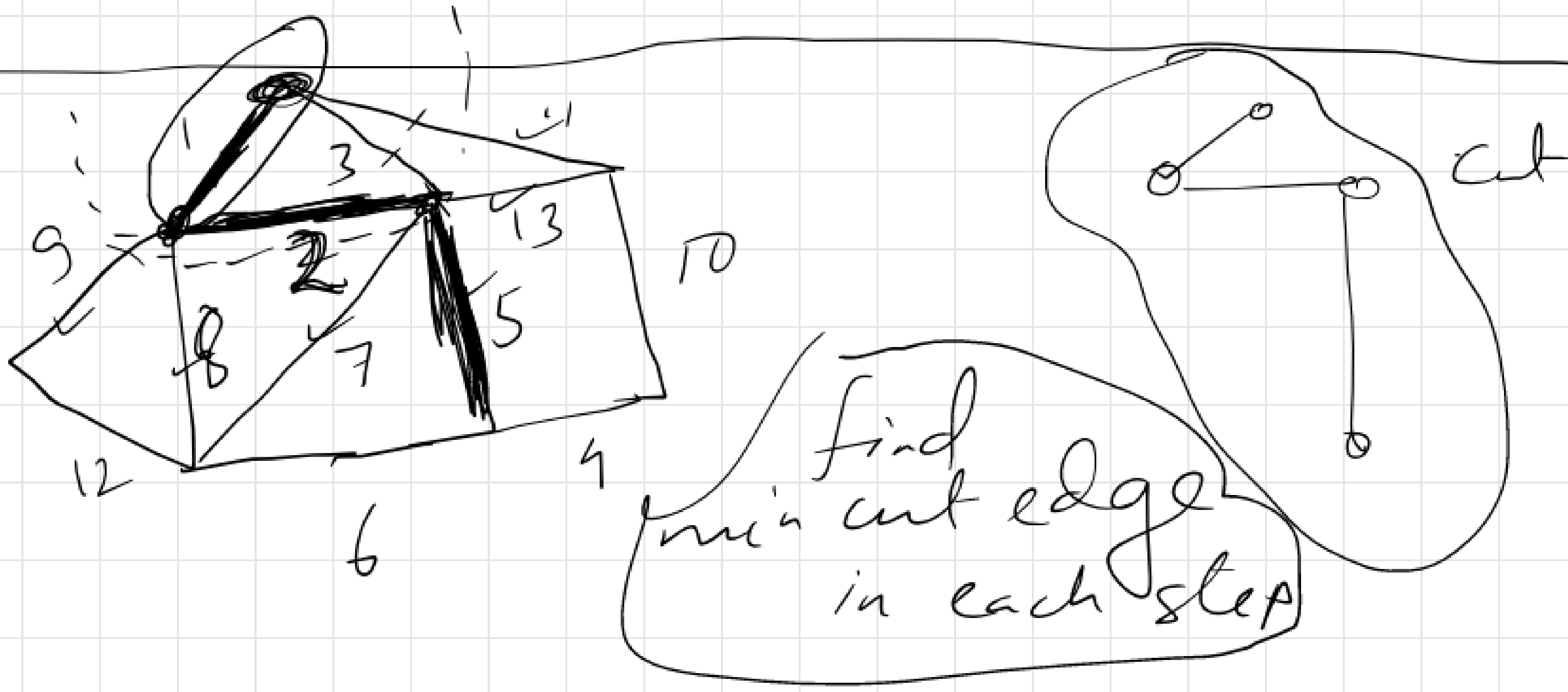
Add e to T $\rightarrow$ form a cycle C

$\Downarrow$

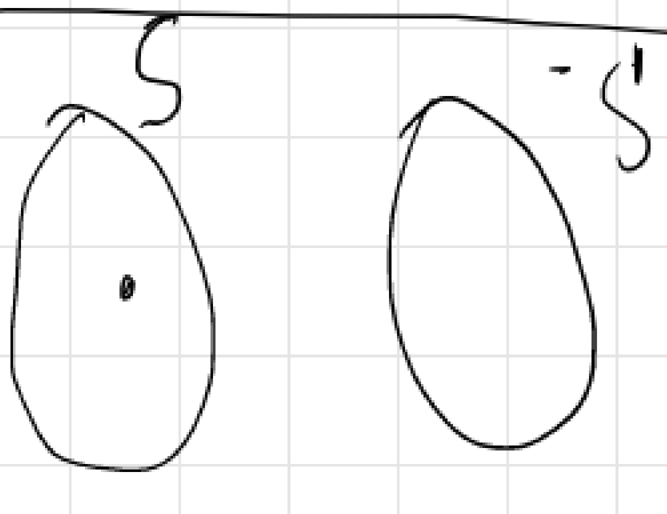
C contains at least one cut edge other than e.

add e, remove that edge

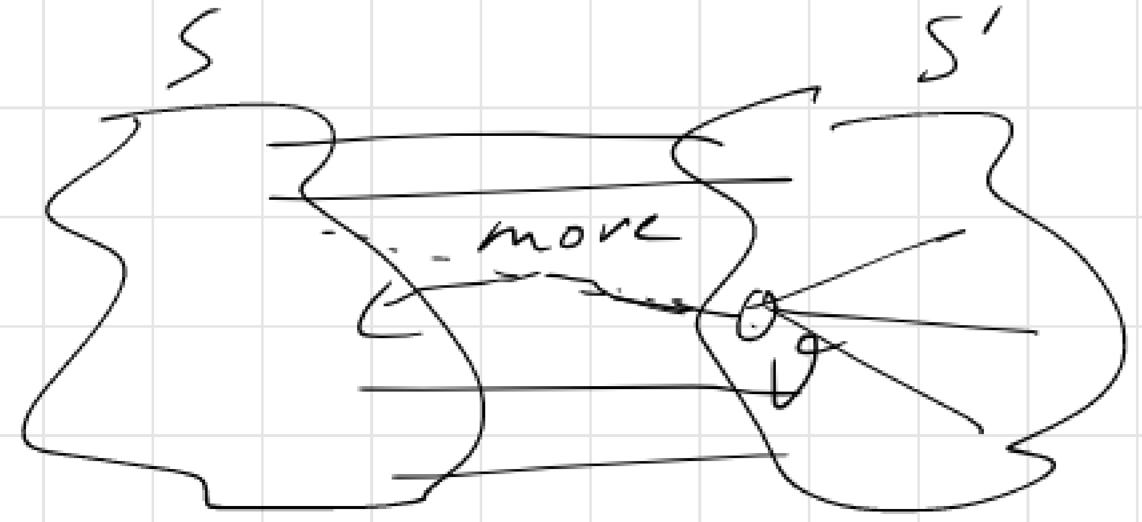
new MST $T' < T$ $\Rightarrow \Leftarrow$

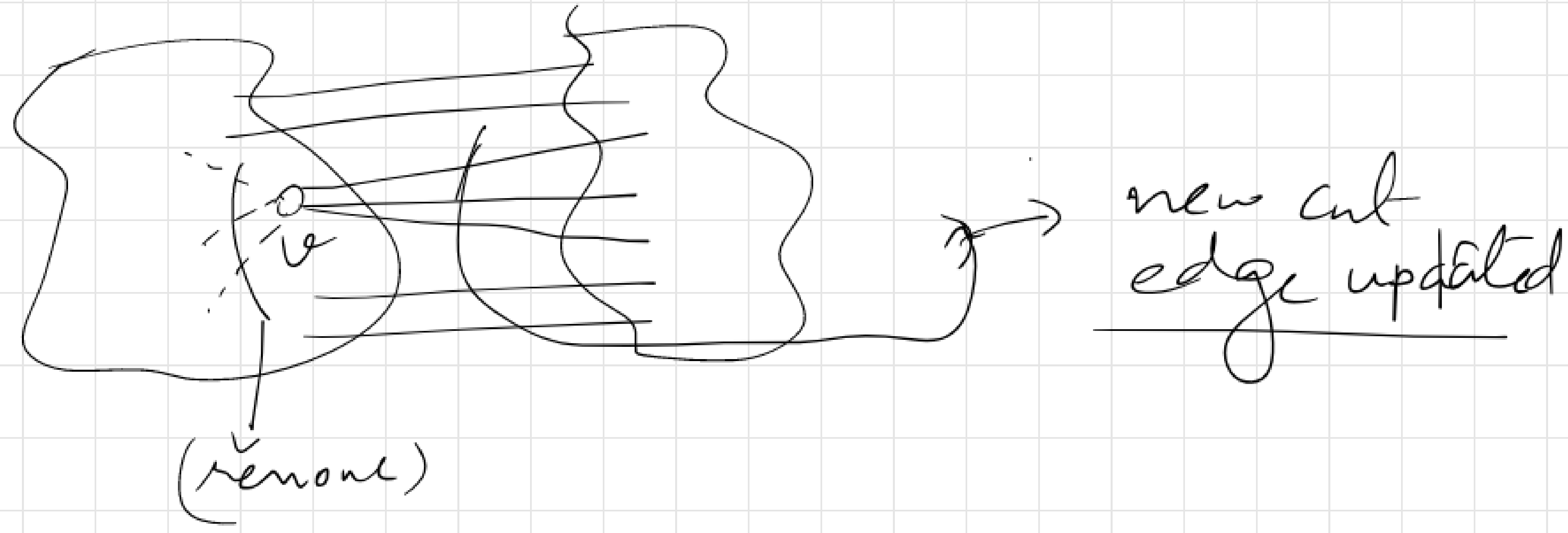


find min cut edge in each step



At each step, find min cut edge, then add one vertex to S, update new cut edge





Heap : containing only cut-edge

After update : $\xrightarrow{\log m}$ add some cut edge in the heap

$\xrightarrow{\log m}$ delete some previous cut edge from the heap

Heap opⁿ

$$\left[\left\{ \sum_v \deg(v) \cdot \log m \right\} = m \log m \right]$$

Find min : n times : $O(n)$

Pseudo code

S is an array $S[v] = 1$ if $v \in S$
$= 0$, else

$S[\text{root vertex}] = 1$;

$S[v] = 0 \ \forall v \in V \setminus \text{root}$; $T \leftarrow \emptyset$

$\forall$ e incident to root vertex do H. insert(e)

while !H. empty do

f = H. findmin()

let v be the endpoint of f s.t. $S[v] = 0$

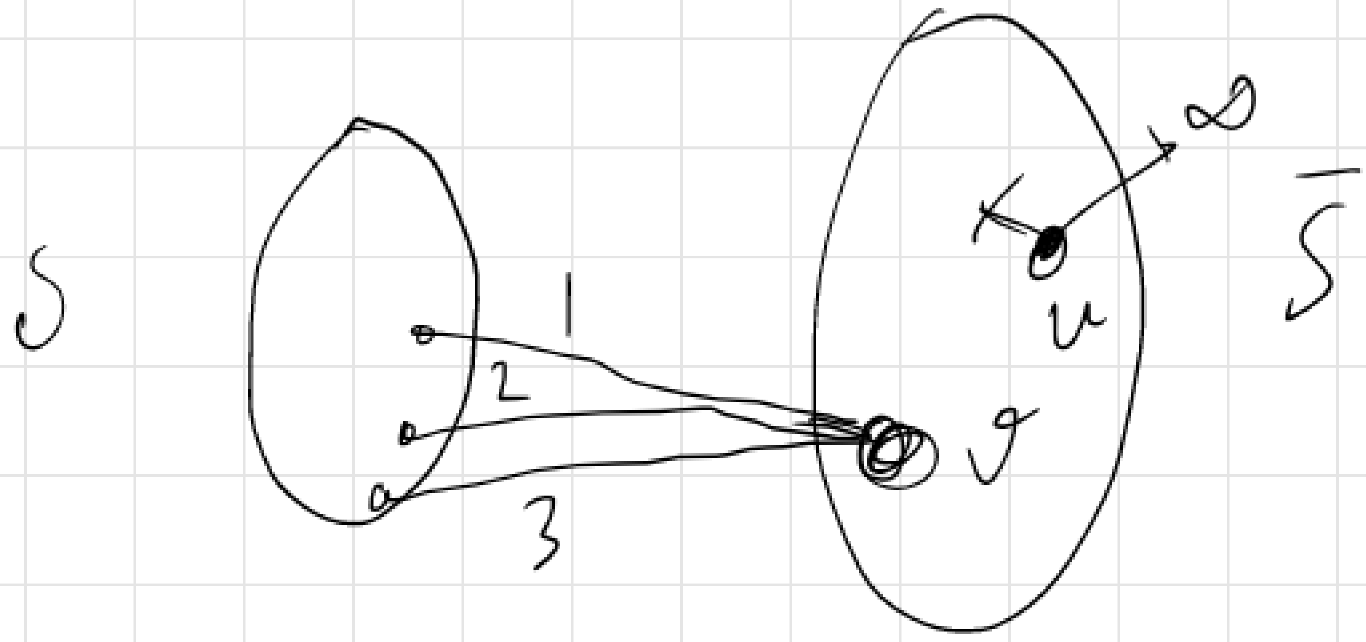
in while

for all $e = (v, w)$ adjacent to v do

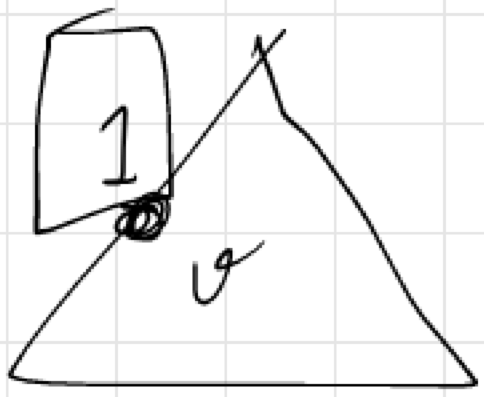
  if $s[w] == 0$ then H.insert(e)

  else H.delete(e)

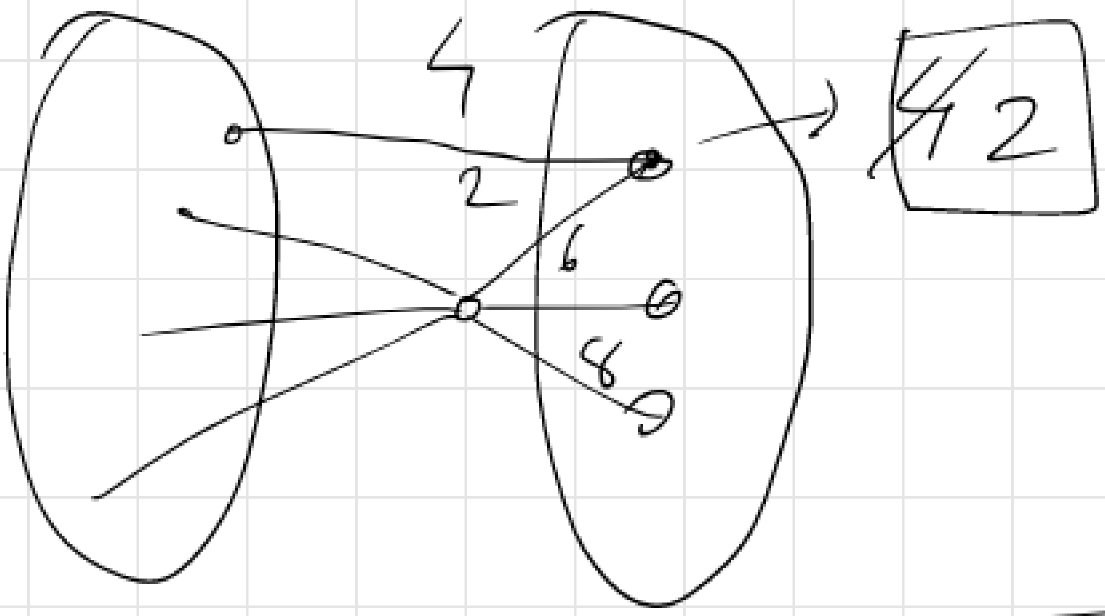
$s[v] = 1$



create Heap with vertices



heap has one element for each vertex in $\bar{S}$



decrease key

Pseudo code

$\forall v, S[v] = 0$, H.insert(v, ∞) , label[w]

, H.decpri$(r, 0)$

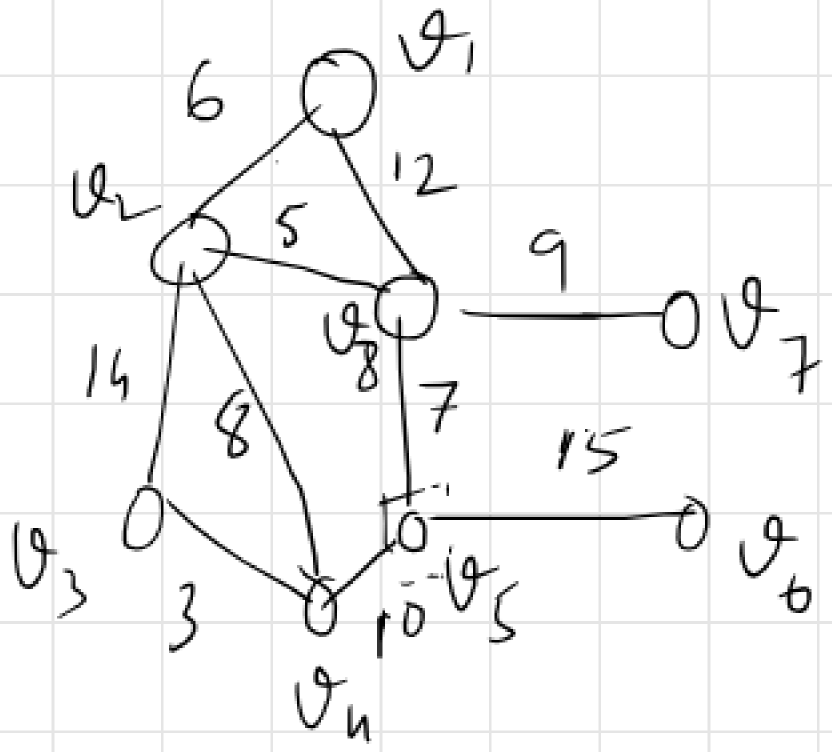
```
While !H.empty {
    u = H.findmin()
    for all v adjacent to u do
        if S[v] == 0 then
            if label[v] > W(u, v)
                label[v] = W(u, v))
                H.decrepri(v, label(v))

    S[v] = 1
}
```

Σ deg

Pseudo Code:

```
Priority Queue Q ← V          ; π(v) = -1  ∀ v∈V
key(v) ← ∞   ∀ v ∈ V          ;
key(s) ← 0      for arbitrary vertex s ∈ V
While (Q ≠ ∅)
    do u ← ExtractMin(Q)
        for v ← adj(u)
            do if v ∈ Q and w(u,v) < key(v)
                then key(v) ← w(u,v)          } decrease key
                     π(v) ← u
[v, π(v)] → MST
```

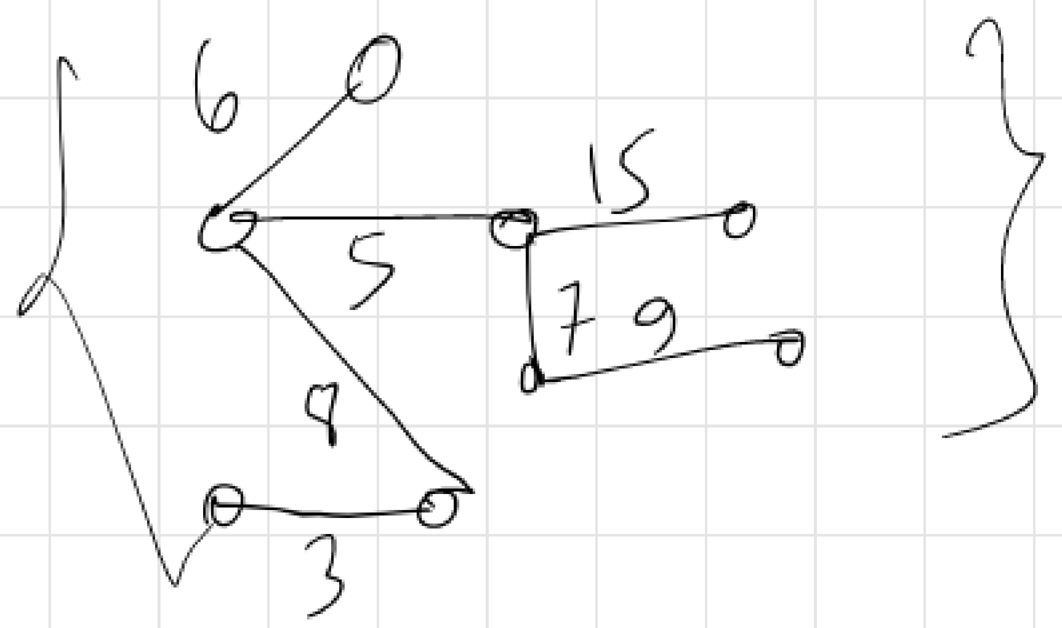
Example



v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	0	∞	∞	∞
∞	∞	∞	10	0	15	∞	7
12	5	∞	10	0	15	9	7
6	5	14	8	0	15	9	7

— continue —

6	5	3	8	1	15	9	7



[Time complexity]

$O(V) + O(E)$.

decrease key

1 → $O(V)$

2) $O(V)$

3) $O(1)$

4) while ($Q \neq \emptyset$)

$O(|E|)$ $O(V)$ times $O(\deg(v))$

▷ $|V| \cdot$ $\underbrace{\text{Time Extract Min}}$ + $|E| \cdot$ Time decrease key

Heap: $O(\log|V|)$ $O(\log|V|)$

$V\log V + E\log V \simeq O(E\log V)$