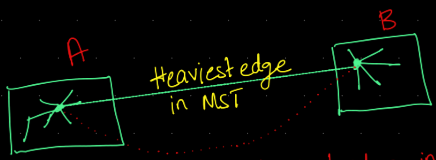
MST → Tree with all vertices of graph but with min total edge cost

Cycle property — Heaviest edge in a cycle will never be part of MST

$\boxed{\text{assumption distinct edges}}$



Since Heaviest edge in cycle there is a path from area A to B which is having lesser than or equal to cost to the heaviest edge.

Cut property → Smallest edge crossing any cut must be part of any mst
↳ given n vertices, we can cut in $2^{n-1}-1$ #ways (ie 2^n binary partitions)

$\boxed{\text{assumption:- distinct edges}}$

eg: $G(V,E)$; $V = \{a,b,c\}$

cuts possible = $\{a\};\{b,c\}$
$\{b\};\{a,c\}$
$\{c\};\{a,b\}$

→ Intuition → cuts not labelled so selecting a set of vertex ≡ not selecting that set of vertex
↳ empty set not allowed

☆ Theorm:- A edge is not in any MST, if and only if it is not the unique heaviest edge in any cycle.

Edge not in MST ⟷ the edge is heaviest edge in some cycle

Edge in MST ⟷ the edge is not heaviest edge in any cycle in graph

} distinct edges weight assumption

Kruskal's algorithm — Take smallest edge first — $\boxed{\text{greedy}}$
— Unless it forms a cycle

```
Kruscal(G):
    E' = sort edges in increasing order   → O(E log E)
    MST T = {}
    for each edge ei in E':               → O[E]
        if not cycle(ei ∪ T):             → O(α)
            T = T.union(ei)
```

Kruscal's time complexity
$= O(E\log E + E\alpha)$

$\alpha = V+E$ using DFS
$\Rightarrow$ Kruscal's $= O(E\log E + E(V+E))$ too much

Use union find data structure works incrementally only $\alpha = O(1)$

$\boxed{\Rightarrow \text{Kruscal's complexity} = O(E\log E + E)}$

a) Kruscal's can handle negative edge

b) If we already have sorted order
$\Rightarrow$ Time complexity $= O(E)$

c) If we assume appropriately about range of weights $\Rightarrow$ Radix sort possible $\Rightarrow O(E)$

d) Adding weights do not change MST but only the MST cost.