Kruskal's Algoorithm

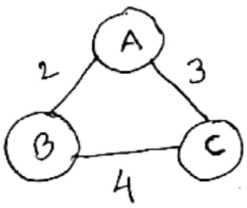
Minimum Spanning Tree

Kruskals():

```
Sort edges in increasing order of length $\{e_1, e_2, e_3, \dots, e_m\}$

T = { }

for i = 1 to m
    if $e_i$ does not add a cycle:
        add $e_i$ to T

return T.
```

* But how can we determine that adding e_i to T won't add a cycle?

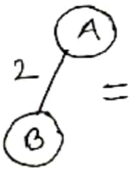
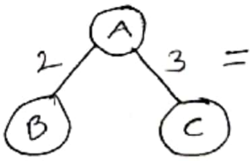
Ans:



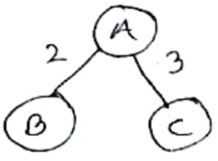
$T = \{\ \}$

$E = \text{edges} = \{AB, AC, BC\}$

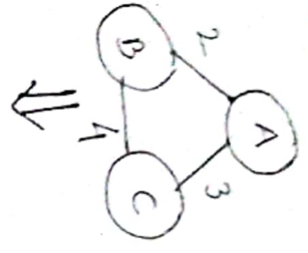
$\text{cost} = \{2, 3, 4\}$

Selected Edge	Accept/Reject	Total cost	Tree
e_1 = "AB"	Accept	2	 = T
e_2 = "AC"	Accept	2 + 3	 = T
e_3 = "BC"	Reject (Addition will form cycle)	×	×

Minimum Spanning Tree:



Kruskal's algorithm Using Union and Find



⇓

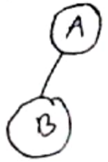
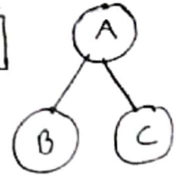
	A	B	C
Father	A	B	C

$E = \{AB, AC, BC\}$

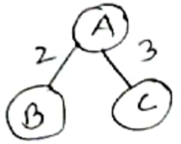
$\text{Weight} = \{2, 3, 4\}$

~~Selected Edge uv Father(u) ≠ Father(v) T cost~~

~~AB u = A V = B father(A) ≠ father(B) ⇒ A~~

Selected Edge	uv	Father(u) ≠ Father(v)	Father Array	T	cost
e = AB	u = A V = B	Father('A') ≠ Father(B) ⇒ 'A' ≠ 'B' True	A B C A A C		2
e = AC	u = A V = C	Father('A') ≠ Father(C) ⇒ A ≠ C True	A B C A A A		2+3
e = BC	u = B V = C	Father(B) ≠ Father(C) ⇒ A ≠ A False	×	×	×

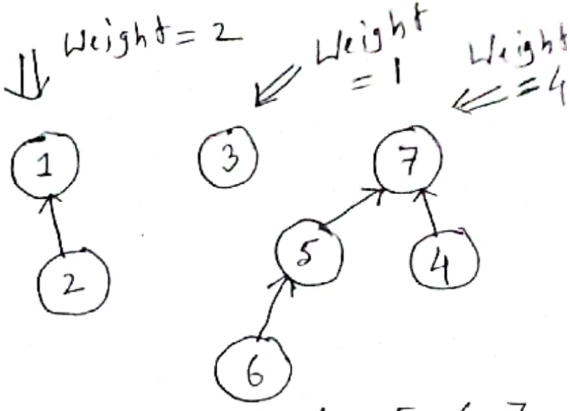
Minimum spanning Tree =



" " cost = 5

7 Algorithm ~~by~~ For Union by Rank

```
Union( i, j : index) {
    Wi = Weight[i];
    Wj = Weight[j];
    if (Wi < Wj) then
        up[i] = j;
        Weight[j] = Wi + Wj;
        Weight[i] = NULL;

    else  up[j] = i;
          Weight[i] = Wi + Wj
          Weight[j] = NULL;

}
```



	1	2	3	4	5	6	7
up	0	1	0	7	7	5	0
Weight	2		1				4

i = 1, j = 7

Union(1, 7):

W_i = Weight[1] = 2

W_j = Weight[7] = 4

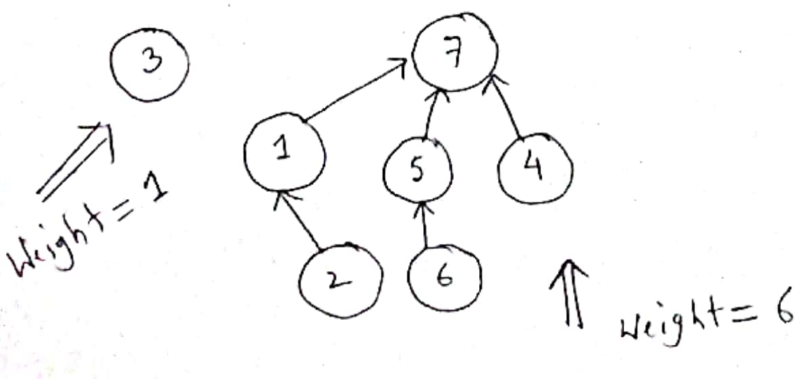
if $(\underbrace{2 < 4}_{\text{True}})$

up[1] = 7

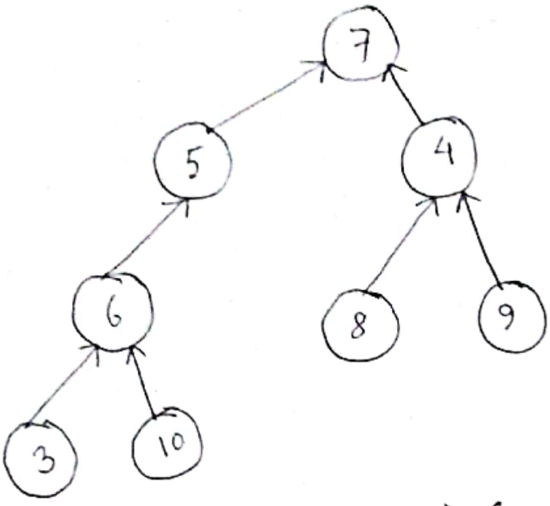
Weight[7] = $W_i + W_j$ = 2 + 4 = 6

Weight[1] = NULL

	1	2	3	4	5	6	7
Up	7	1	0	7	7	5	0
Weight			1				6



Path Compression



	7	5	4	3	10	6	8	9
Up	0	7	7	6	6	5	4	4

```
PC-Find(i: index){
    r = i
    while $up[r] \neq 0$ do
        r = up[r]
    if $i \neq r$ then
        k = up[i];
        while $k \neq r$ do
            up[i] = r
            i = k
            k = up[k]
    return (r)
}
```

st

pc-Find($\overset{i}{3}$):

i=3:

r = 3

r = 3   up[3] = 6 True

r = 6   up[6] = 5 True

r = 5   up[5] = 7 True

r = 7   up[7] = 0 False

if $\overset{i}{3} \neq \overset{r}{7}$ then

k = up[3] = 6

6 ≠ 7 T | up[3] = 7
i = 6
k = up[6]
= 5

5 ≠ 7 T | up[6] = 7
i = 5
k = up[5]
= 7

7 ≠ 7 False

	7	5	4	3	10	6	8	9
up	0	7	7	~~6~~ 7	6	~~5~~ 7	4	4

After Path Compression: