Backlog Coverage Session

Hello Everyone
Very Special Good Evening
to all of you 😊😊😊
We will start session
from 9:06 PM

Agenda

1. Z- Algorithm → KMP
2. DSU — Disjoint Set Union
3. KrushKal Algo

Expectation: MST
Prims Algo

Z- Algorithm:

Requirment:

* no. of occurrence of pattern in text.
* indexes of pattern which is available in text.

for eg: pattern → a b a

text →

a	b	a	b	a	d	a	b	a	b	a	e
0	1	2	3	4	5	6	7	8	9	10	11

no. of occurrence of pattern → 4

Starting indices → [0, 2, 6, 8]

Brute force:

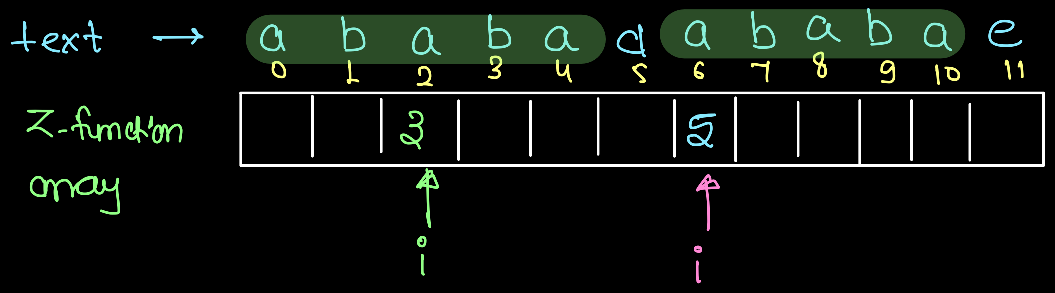
Using sliding window we can do that.

→ for more reference of brute fr
visit → Pattern matching session.

When we solved KMP, we take help of LPS → length of longest prefix suffix

(NOTE: excluding complete string)

similarly for Z-Algo, we have to use Z-function



text →	a	b	a	b	a	d	a	b	a	b	a	e
	0	1	2	3	4	5	6	7	8	9	10	11
Z-function array			3				5					

length of longest substring which is starting from index 'i' and it is also proper prefix.

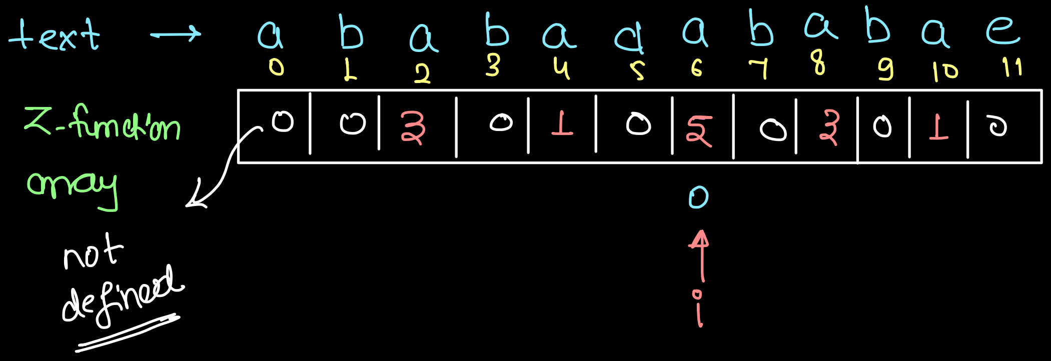
proper prefix

str= "abac"

a
ab
aba

proper prefix

~~abac~~



text →	a	b	a	b	a	d	a	b	a	b	a	e
	0	1	2	3	4	5	6	7	8	9	10	11
Z-function array	0	0	3	0	1	0	5	0	3	0	1	0

pseudocode:

```
int[] zfunction(String s) {
    int[] z = new int[n];  // value of z[i], initially 0.
    for(int i=1; i<n; i++) {
        while( s[z[i]] == s[i+z[i]] ) {
            z[i]++;
        }
    }
    return z;
}
```

worst case : str - a a a a a a

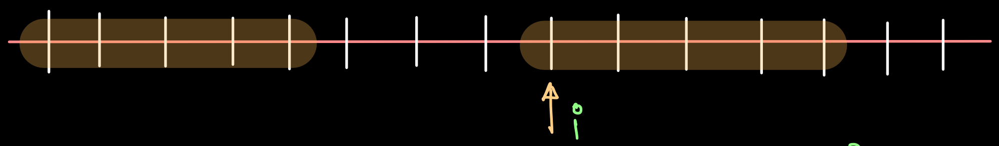
for (n-1) (n-2) (n-3) - - - 1

total itr = $1+2+3+\cdots+(n-1)$

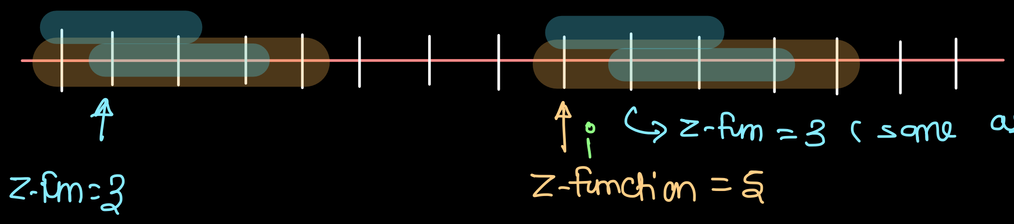
$= \frac{n(n-1)}{2}$

T.C: $O(n^2)$

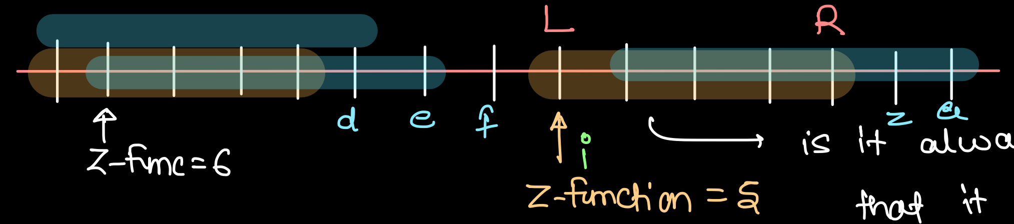
How to optimise Z-function: (Assumption, vertical lins are char)



Z-function = 5 → first 5 chars are equal to substring of length 5 which is starting form index i



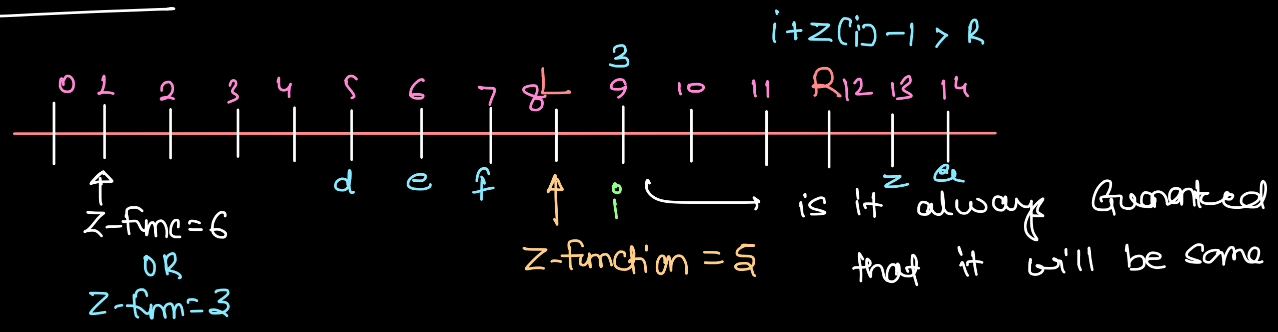
↪ z-fun = 3 (same as left side)



→ is it always Guaranteed that it will be same

NOTE: for this particular point we are not sure about z-funct 4 chars since it is beyond the range of R

But we are sure within the Range of z-function will be 4 (R-i+1) pseudo code: (optimise)



```
int[] zfunction(String s){
    int[] z= new int[n]; // value of z[i], initially 0.
    int l=0, r=0;
    for(int i=1; i<n; i++){
        if( i <= R)
        |   z[i]= Min(z[i-l) , R-i+1)
        }
        while( s[z[i]] == s[i+z[i]] ){
        |   z[i]++;
        }
        if( i+z[i] -1 > R) {
        |   L= i;
        |   R= i+z[i]-1;
        }
    }
    return z;
}
```

overall itr by for-loop => n

overall " " while - loop => n

total itr = 2n

=> T.C: O(n)

S.C: O(n)

							L				R	
	a	b	a	b	a	d	a	b	a	b	a	e
	0	1	2	3	4	5	6	7	8	9	10	11
Z -Fnction	0	0	3	0	1	0	5	0	3	0	1	0

↑
i

```
int[] zfunction(String S){
    ✓int[] z= new int[n];
    ✓int l=0, r=0;
    for(int i=1; i<n; i++){
        ✓if( i<= R)
        |    z[i]= Min( z[i-l) , R-i+1)
        }
        ✓while( s[z[i]] == s[ i+z[i]] ){
        |    z[i]++;
        }
        if( i+z[i] -1 >R) {
        ✓|    L= i;
        |    R= i+z[i]-1;
        }
    }
    return z;
}
```

L = ~~0~~ ~~2~~ 6

R = ~~0~~ ~~4~~ 10

i = ~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~ ~~6~~ ~~7~~ ~~8~~ ~~9~~ ~~10~~ ~~11~~ 12

Pattern matching using Z- Algo:

pattern → a b a

text →	a	b	a	b	a	d	a	b	a	b	a	e
	0	1	2	3	4	5	6	7	8	9	10	11

pat + # + text ⇒	a	b	a	#	a	b	a	b	a	d	a	b	a	b	a	e
z_function →	0	0	1	0	3	0	3	0	1	0	3	0	3	0	1	0

substring with length 3
is same as Proper prefix

pseudocode:

```
int countOfPatternInText( String pat, String text) {
    String str= pat + "#" + text;
    int[] z= zfunction( str); → (n)
    for(int ele : z) { → (n)
    |    if(ele == pat.length){
    |    |    count++;
    |    }
    }
    return count;
}
```

T.C : O(n+m)

S.C : O(n+m)

n → pat.length

m → text.length

10:21 - 10:35
Break

DSU - Disjoint Set Union

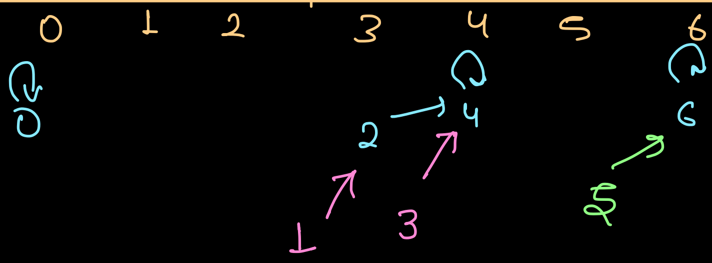
→ union → Merging their leader
→ find → leader

DSU follows transitivity.

→ a similar to b
b " " c
⇒ a " " c

for mengemt of leaders:

parent:	0	2	4	4	4	6	6
	0	1	2	3	4	5	6



Edges

u	v
1	2 ✓
3	4 ✓
1	4 ✓
5	6 ✓

2-4

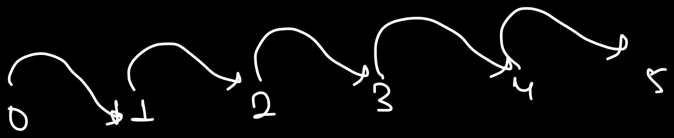
u = 1 → find leader = 2
v = 4 → find leader = 4

u = 5 → leader(5) = 5
v = 6 → leader(6) = 6

int[] parent; → parent array globally available)

```
int find(int x) {
    if( par[x] == x) {
        return x;
    }
    int temp= find(par[x]);
    return temp;
}

void union(int x, int y) {
    int px= find(x);
    int py = find(y);
    if( px != py){
        // merge them
        par[px] = py;
    }
}
```

worst situation:



travel for parent of
0 to 4 → 1000 time'

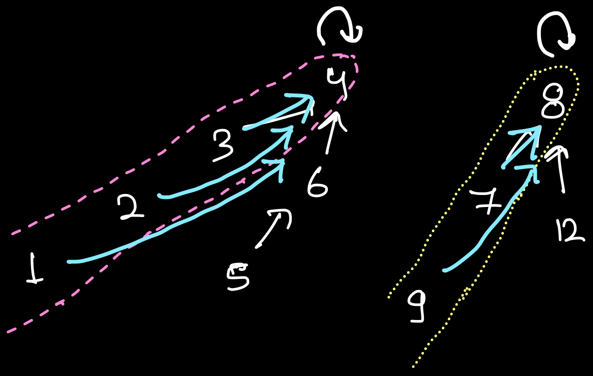
→ Repeat n - iterations
every time.

* concept for optimation of DSU.

① Path compression

② union by Rank

situation:

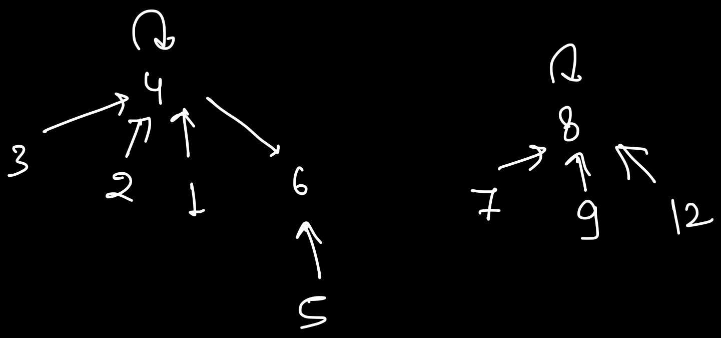


parent(1) → parents are set.

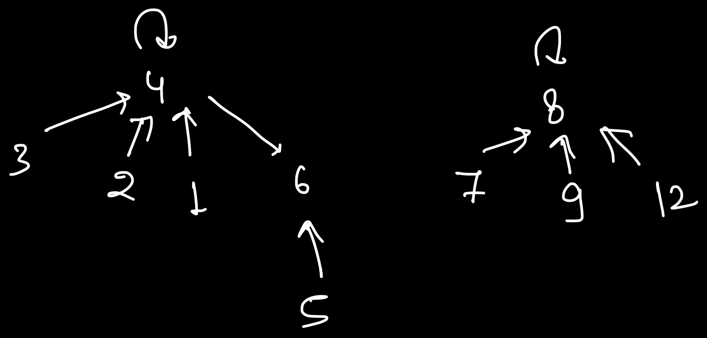
parent(9)

path comparison.

find function



union by Rank :



leader

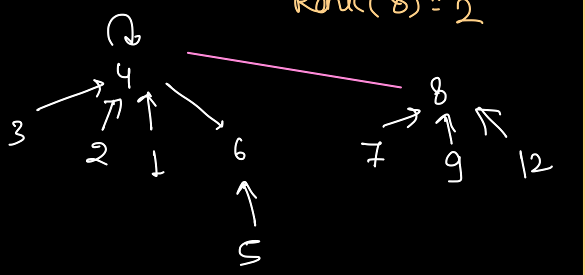
u → 6 → 4]
v → 12 → 8]

I

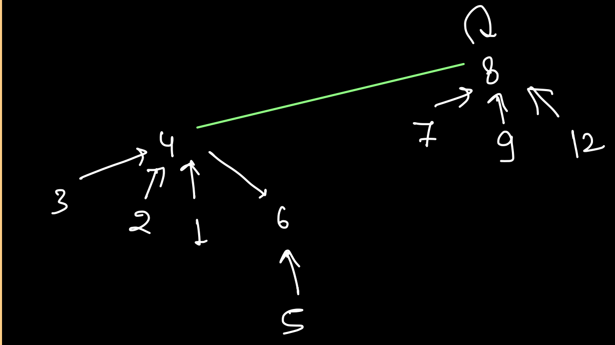
Rank is heigh

Rank(4) = 3

Rank(8) = 2



II



this is better possibility;

Note: union by Rank make leader which have highest Rank available.

↳ prepare one Rank array as well

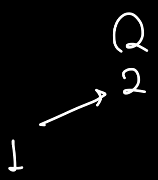
parent:	0	2	4	4	4	6	6
	0	1	2	3	4	5	6

Rank:	1	1	1	1	1	1	1
	0	1	2	3	4	5	6

initial Rank = 1

path Compression -

```
int[] parent;

int find(int x) {
    if( par[x] == x) {
        return x;
    }
    int temp = find(par[x]);
    par[x] = temp; // path Compression.
    return temp;
}
```



1 - 2 ✓

parent using find in O(1)

union is also in O(1)

union by Rank

```
void union(int x, int y) {
    int px = find(x);
    int py = find(y);
    if( px != py){
        if(rank[px] > rank[py]){
            par[py] = px;
        }
        else if( rank[px] < rank[py]){
            par[px] = py;
        }
        else {
            par[px] = py;
            rank[py]++;
        }
    }
}
```

Krushkal's Algorithm:

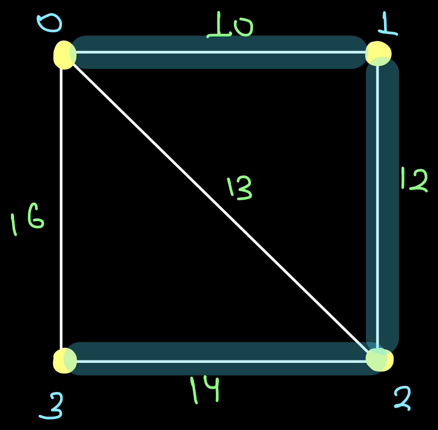
MST → min. Spanning Tree

make sure that (n-1) edges are there.

Tree

n-nodes

Edge ? → $\boxed{n-1}$



Connect all vertex with min possible cost

⇒ MST

0 – 1 @ 10

1 – 2 @ 12

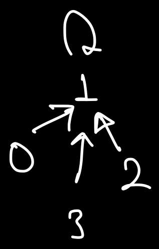
0 – 2 @ 13

0 – 3 @ 16

3 – 2 @ 14

} n-1 edges are relevant

Sort edgs on the basis of their weight in ↑ing order



$\boxed{\text{3 eds'}}$

0 – 1 @ 10 ✓

1 – 2 @ 12 ✓

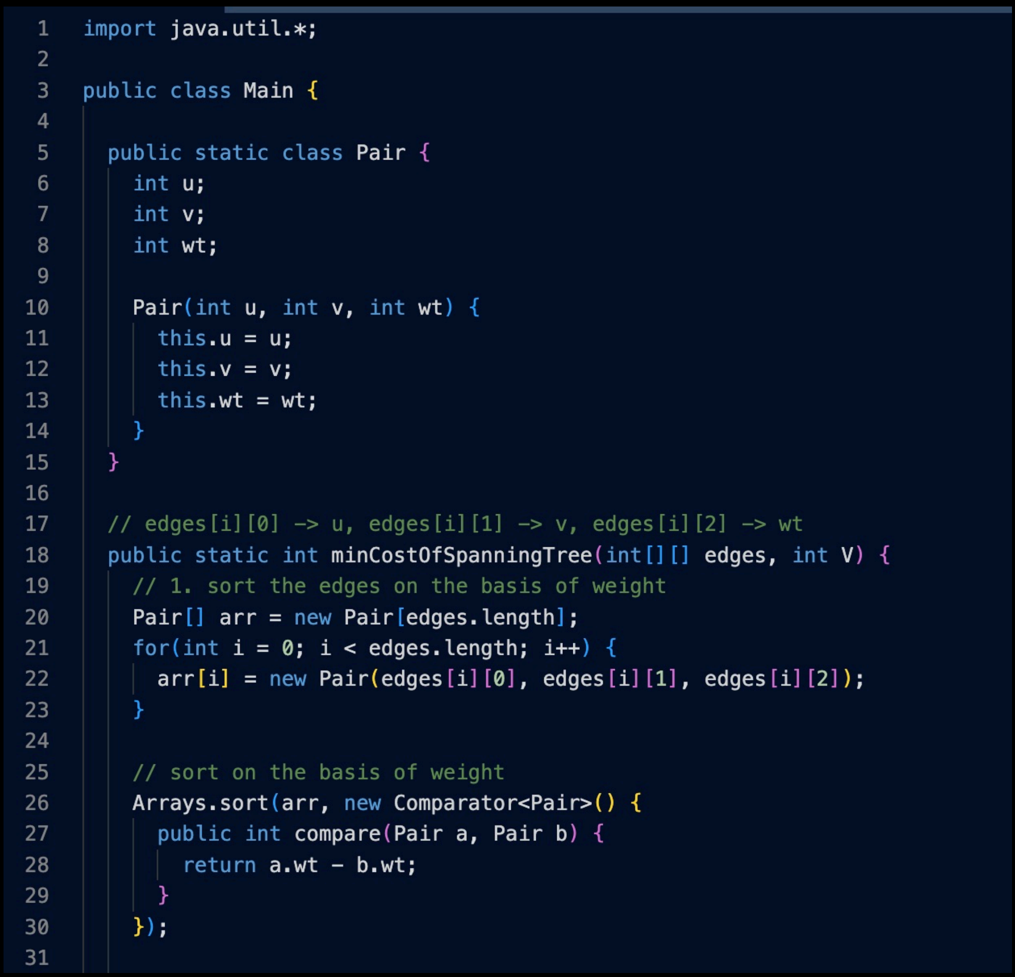
~~0 – 2 @ 13~~ ✓ → same parent they are alreay connec'

3 – 2 @ 14 ✓

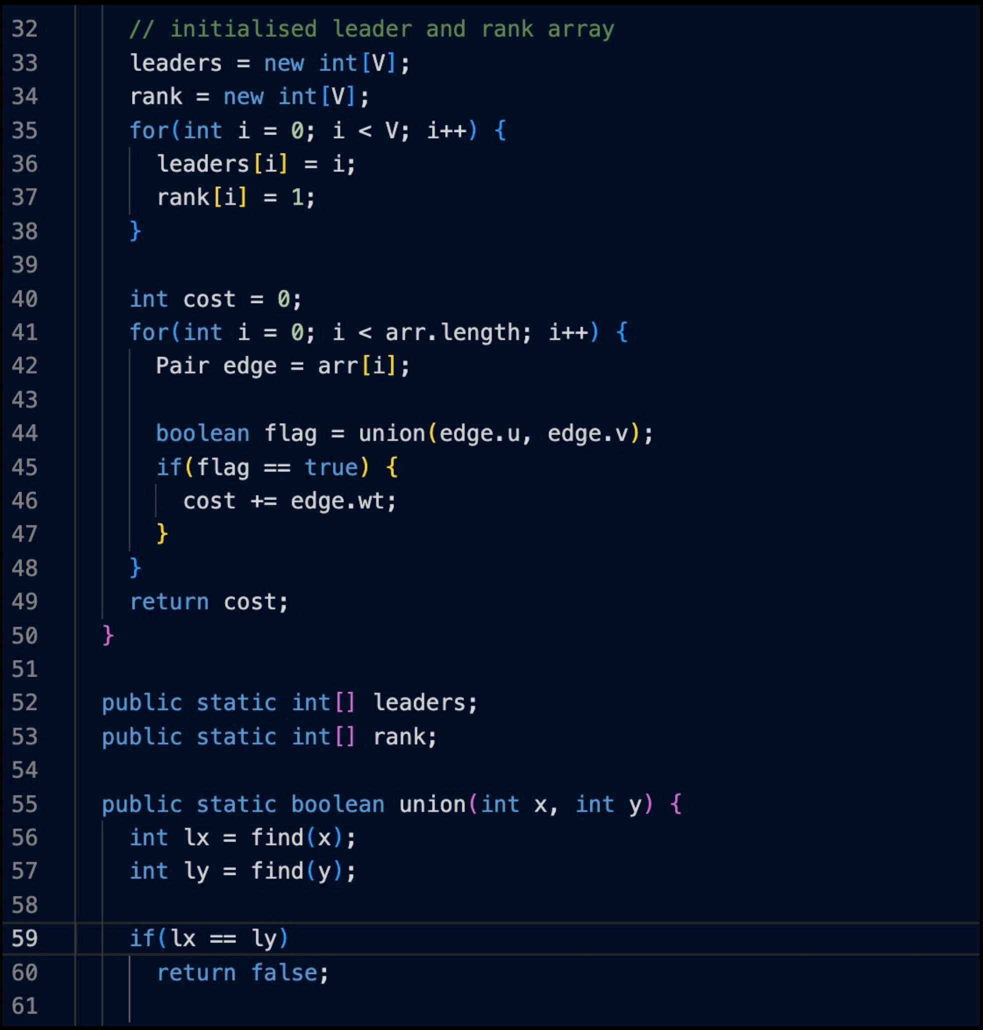
~~0 – 3 @ 16~~ ✓

Same parent they are alreay connec'

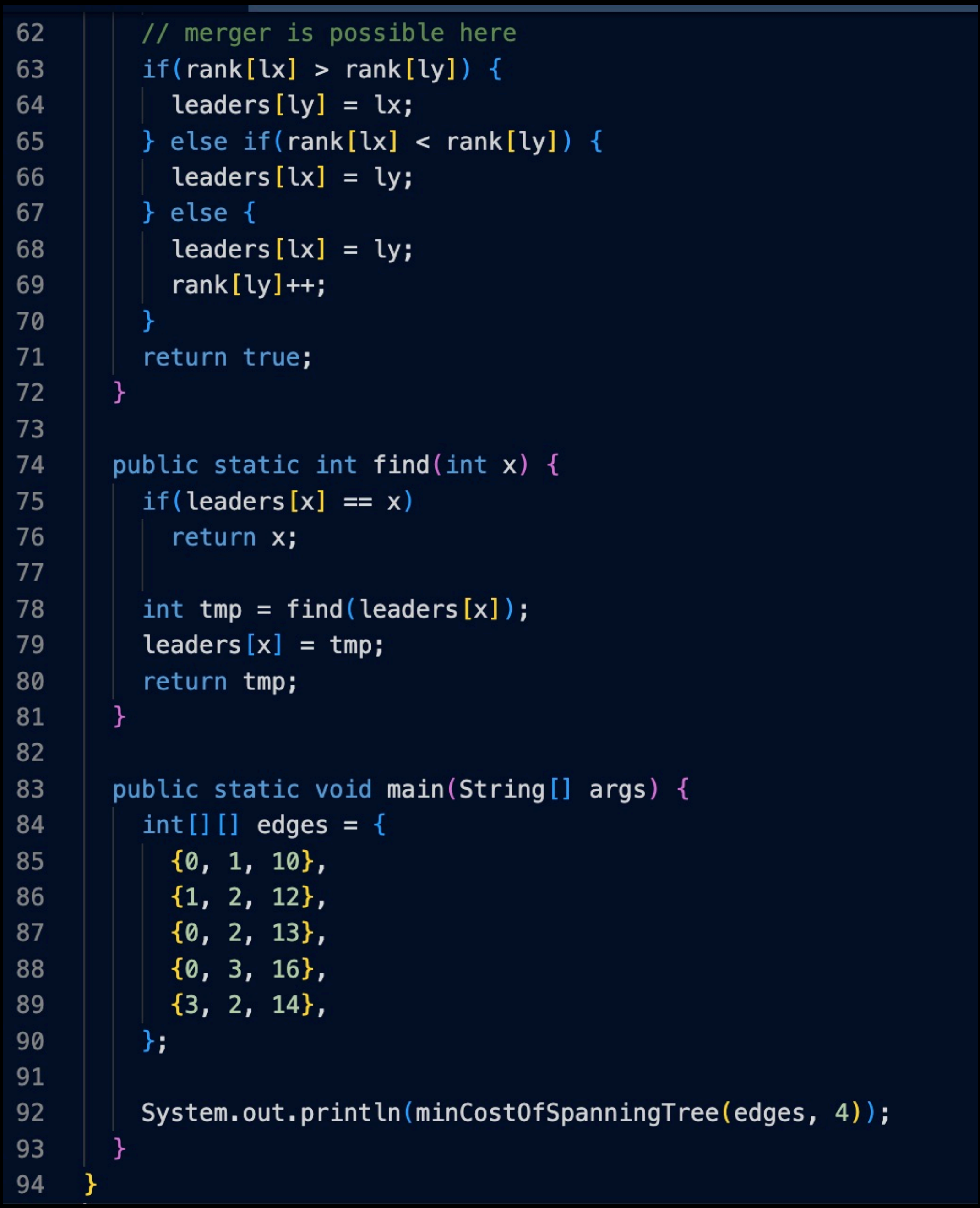
10 + 12 + 14 ⇒ $\boxed{36}$



```java
import java.util.*;

public class Main {

  public static class Pair {
    int u;
    int v;
    int wt;

    Pair(int u, int v, int wt) {
      this.u = u;
      this.v = v;
      this.wt = wt;
    }
  }

  // edges[i][0] -> u, edges[i][1] -> v, edges[i][2] -> wt
  public static int minCostOfSpanningTree(int[][] edges, int V) {
    // 1. sort the edges on the basis of weight
    Pair[] arr = new Pair[edges.length];
    for(int i = 0; i < edges.length; i++) {
      arr[i] = new Pair(edges[i][0], edges[i][1], edges[i][2]);
    }

    // sort on the basis of weight
    Arrays.sort(arr, new Comparator<Pair>() {
      public int compare(Pair a, Pair b) {
        return a.wt - b.wt;
      }
    });

```



```java
    // initialised leader and rank array
    leaders = new int[V];
    rank = new int[V];
    for(int i = 0; i < V; i++) {
      leaders[i] = i;
      rank[i] = 1;
    }

    int cost = 0;
    for(int i = 0; i < arr.length; i++) {
      Pair edge = arr[i];

      boolean flag = union(edge.u, edge.v);
      if(flag == true) {
        cost += edge.wt;
      }
    }
    return cost;
  }

  public static int[] leaders;
  public static int[] rank;

  public static boolean union(int x, int y) {
    int lx = find(x);
    int ly = find(y);

    if(lx == ly)
      return false;

```



```java
    // merger is possible here
    if(rank[lx] > rank[ly]) {
      leaders[ly] = lx;
    } else if(rank[lx] < rank[ly]) {
      leaders[lx] = ly;
    } else {
      leaders[lx] = ly;
      rank[ly]++;
    }
    return true;
  }

  public static int find(int x) {
    if(leaders[x] == x)
      return x;

    int tmp = find(leaders[x]);
    leaders[x] = tmp;
    return tmp;
  }

  public static void main(String[] args) {
    int[][] edges = {
      {0, 1, 10},
      {1, 2, 12},
      {0, 2, 13},
      {0, 3, 16},
      {3, 2, 14},
    };

    System.out.println(minCostOfSpanningTree(edges, 4));
  }
}
```