GRAPH MINIMUM SPANNING TREE

- SPANNING TREE
- EXAMPLE OF A SPANNING TREE
- MINIMUM SPANNING TREE
- PRIM’S ALGORITHM
- KRUSKAL’S ALGORITHM
- PRACTICE: ISLANDS CONNECTION PROBLEM

Author: prof. Yevhenii Borodavka

SPANNING TREE

A spanning tree is a sub-graph of an undirected connected graph, which includes all the vertices of the graph with a minimum possible number of edges. If a vertex is missed, then it is not a spanning tree.

The edges may or may not have weights assigned to them.

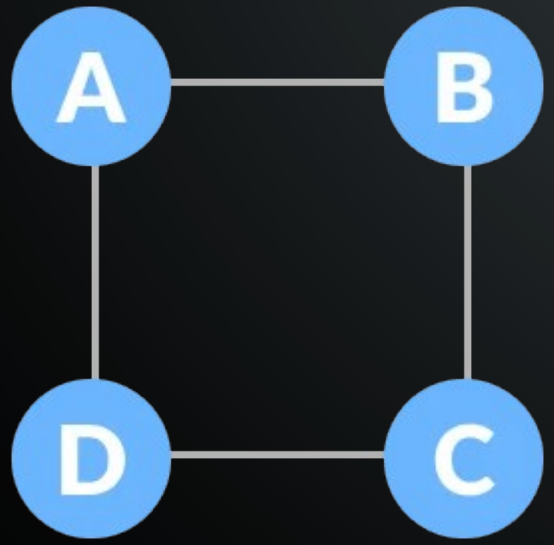
The total number of spanning trees with **n** vertices that can be created from a complete graph is equal to $\mathbf{n}^{(n-2)}$.

If we have **n**=4, the maximum number of possible spanning trees is equal to $4^{4-2}=16$. Thus, 16 spanning trees can be formed from a complete graph with 4 vertices.

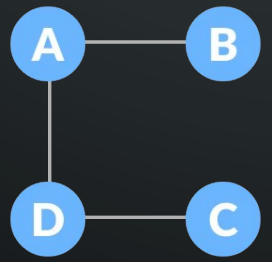
EXAMPLE OF A SPANNING TREE

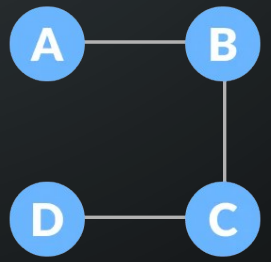
Let's understand the spanning tree with examples below.

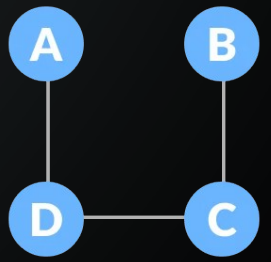
Let the original graph be:

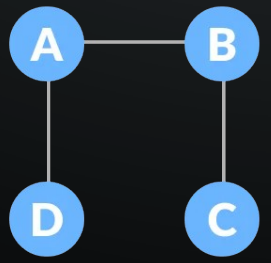


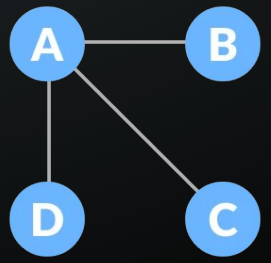
Some of the possible spanning trees that can be created from the above graph are:

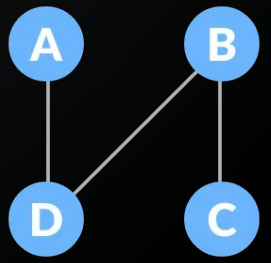








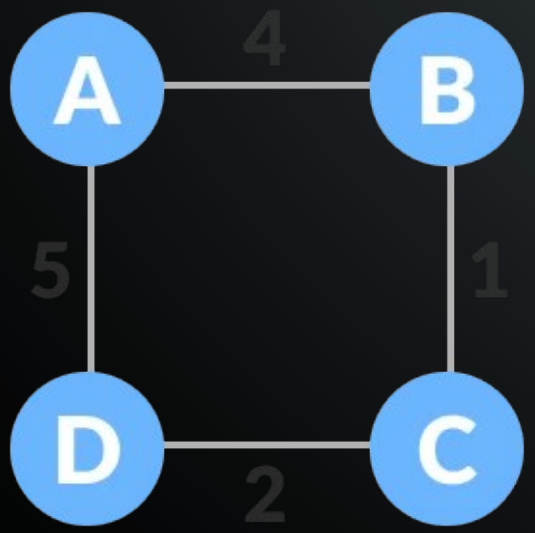




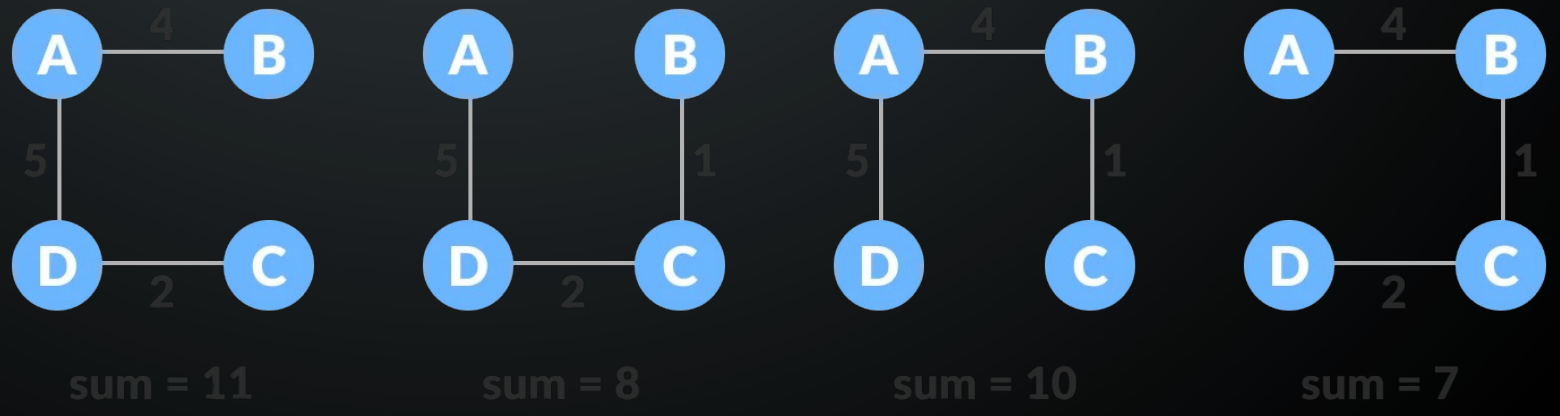
MINIMUM SPANNING TREE

A minimum spanning tree is a spanning tree in which the sum of the weight of the edges is as minimum as possible.

The initial graph is:

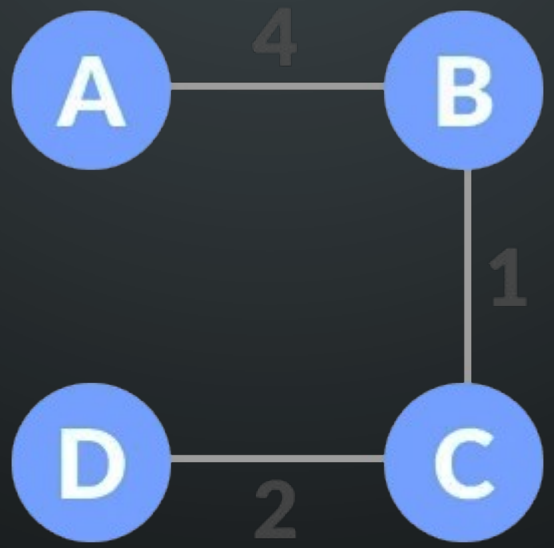


The possible spanning trees from the above graph are:



MINIMUM SPANNING TREE

The minimum spanning tree from the above spanning trees is:



sum = 7

PRIM’S ALGORITHM

Prim's algorithm is a minimum spanning tree algorithm that takes a graph as input and finds the subset of the edges of that graph which:

- form a tree that includes every vertex
- has the minimum sum of weights among all the trees that can be formed from the graph

HOW PRIM'S ALGORITHM WORKS

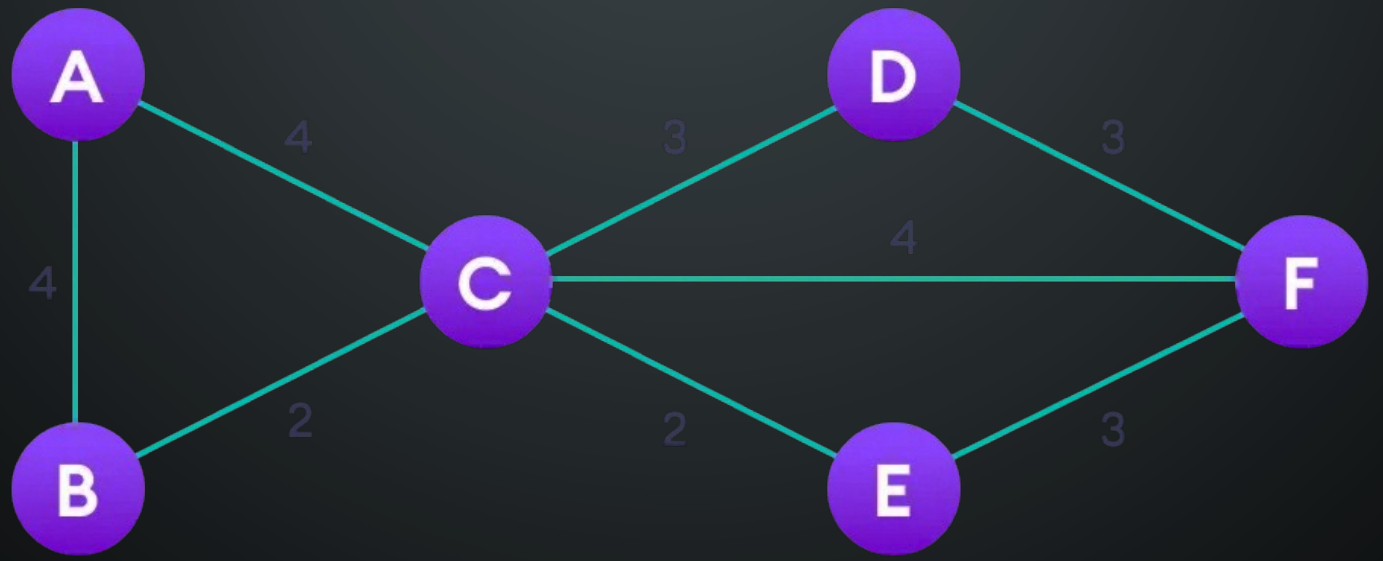
It falls under a class of algorithms called greedy algorithms that find the local optimum in the hopes of finding a global optimum.

We start from one vertex and keep adding edges with the lowest weight until we reach our goal.

The steps for implementing Prim's algorithm are as follows:

1. Initialize the minimum spanning tree with a vertex chosen at random.
2. Find all the edges that connect the tree to new vertices, find the minimum and add it to the tree.
3. Keep repeating step 2 until we get a minimum spanning

EXAMPLE OF PRIM'S ALGORITHM

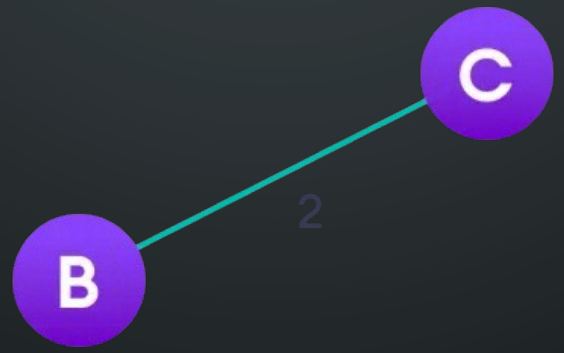


Step: 1

EXAMPLE OF PRIM'S ALGORITHM

Step: 2

EXAMPLE OF PRIM'S ALGORITHM

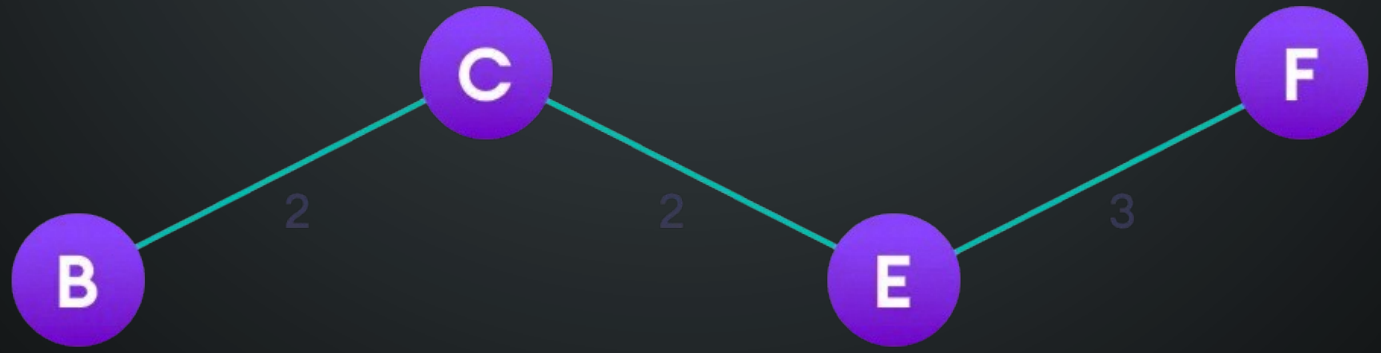


Step: 3

EXAMPLE OF PRIM'S ALGORITHM

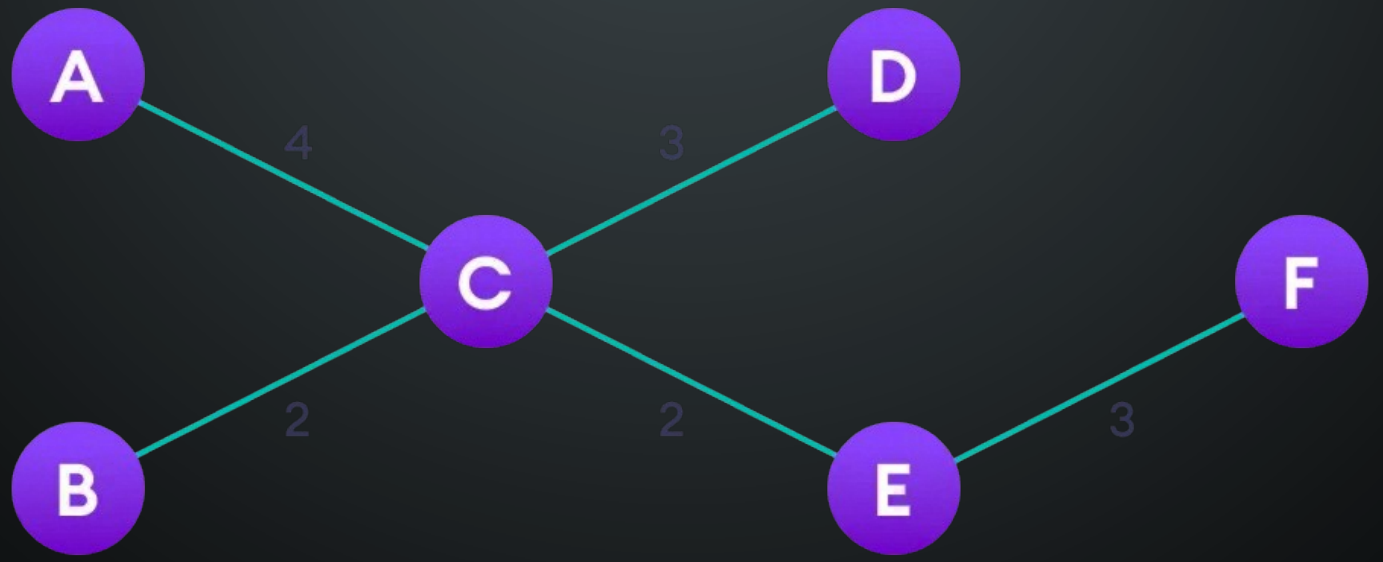
Step: 4

EXAMPLE OF PRIM'S ALGORITHM



Step: 5

EXAMPLE OF PRIM'S ALGORITHM



Step: 6

PRIM'S ALGORITHM PSEUDOCODE

```
T = ∅;
U = { 1 };
while (U ≠ V)
    let (u, v) be the lowest cost edge such that u ∈ U and v ∈ V - U;
    T = T ∪ {(u, v)}
    U = U ∪ {v}
```

The time complexity of Prim's algorithm is **O**(**E** log **V**).

KRUSKAL'S ALGORITHM

Kruskal's algorithm is a minimum spanning tree algorithm that takes a graph as input and finds the subset of the edges of that graph which:

- form a tree that includes every vertex
- has the minimum sum of weights among all the trees that can be formed from the graph

HOW KRUSKAL'S ALGORITHM WORKS

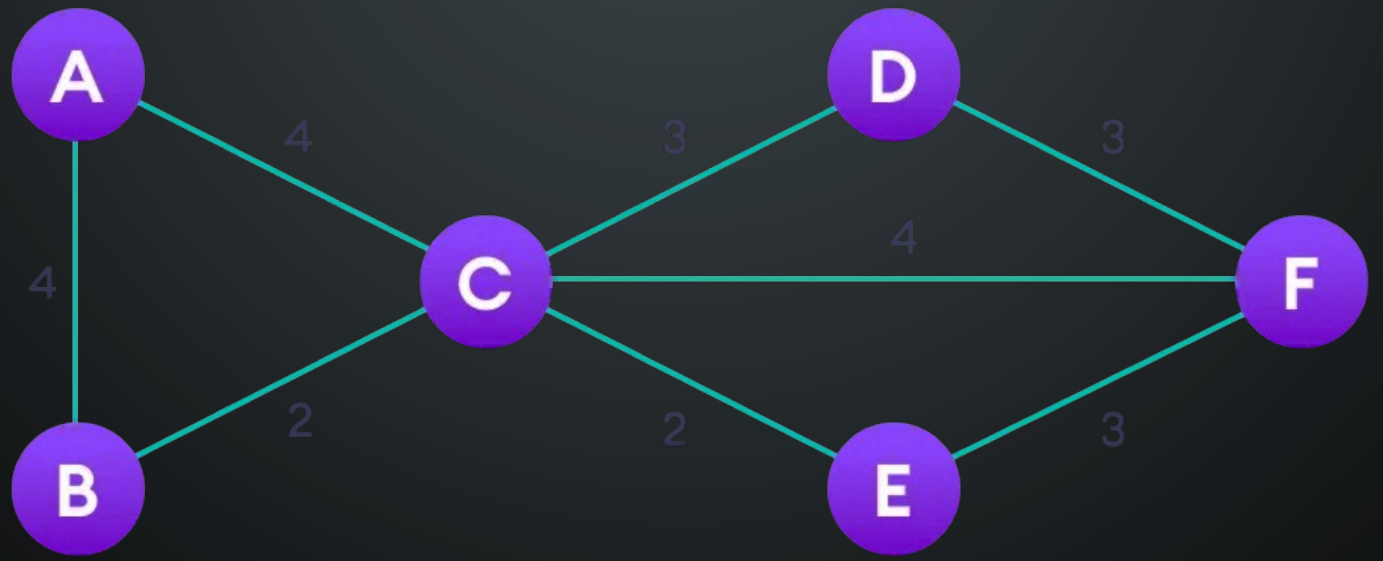
It falls under a class of algorithms called greedy algorithms that find the local optimum in the hopes of finding a global optimum.

We start from the edges with the lowest weight and keep adding edges until we reach our goal.

The steps for implementing Kruskal's algorithm are as follows:

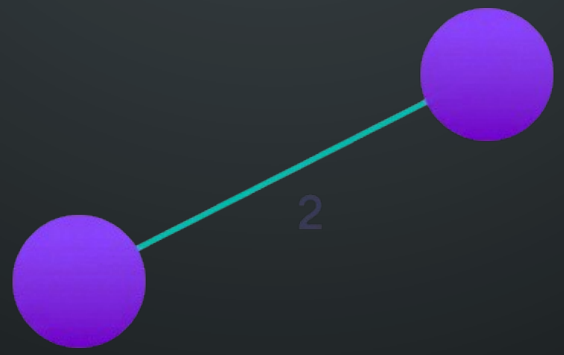
1. Sort all the edges from low weight to high.
2. Take the edge with the lowest weight and add it to the spanning tree. If adding the edge created a cycle, then reject this edge.
3. Keep adding edges until we reach all vertices.

EXAMPLE OF KRUSKAL'S ALGORITHM

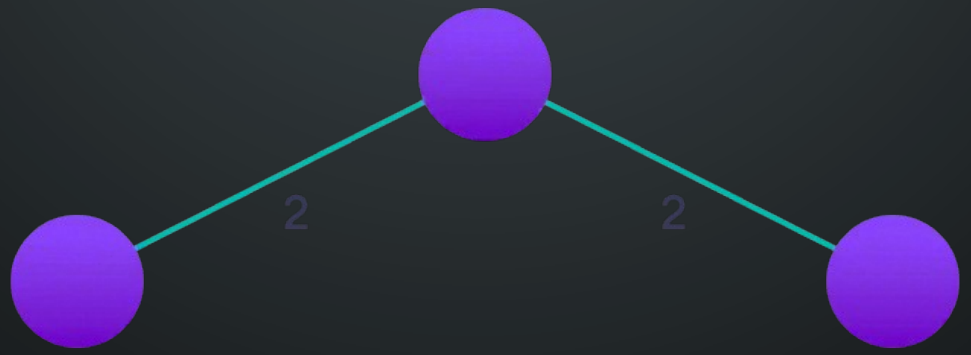


Step: 1

EXAMPLE OF KRUSKAL'S ALGORITHM

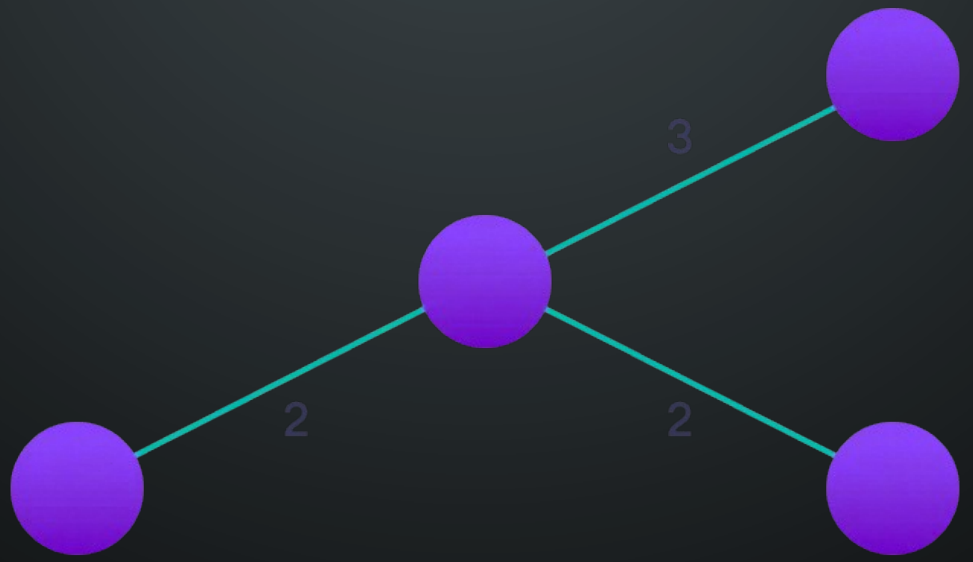


EXAMPLE OF KRUSKAL'S ALGORITHM



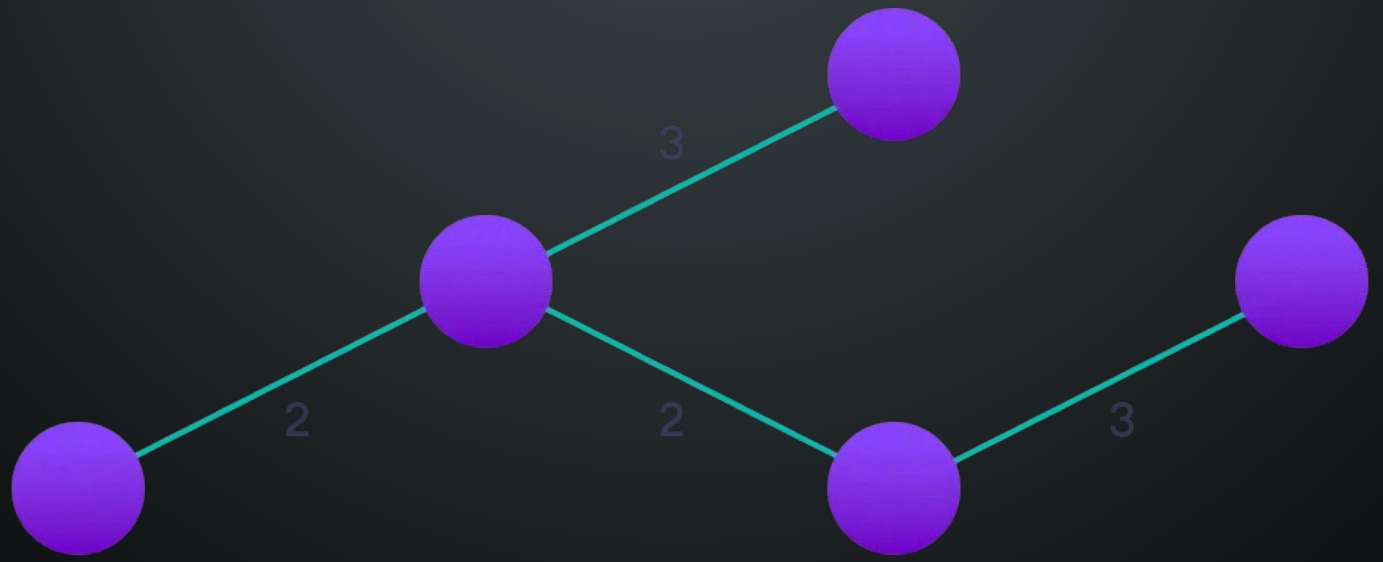
Step: 3

EXAMPLE OF KRUSKAL'S ALGORITHM



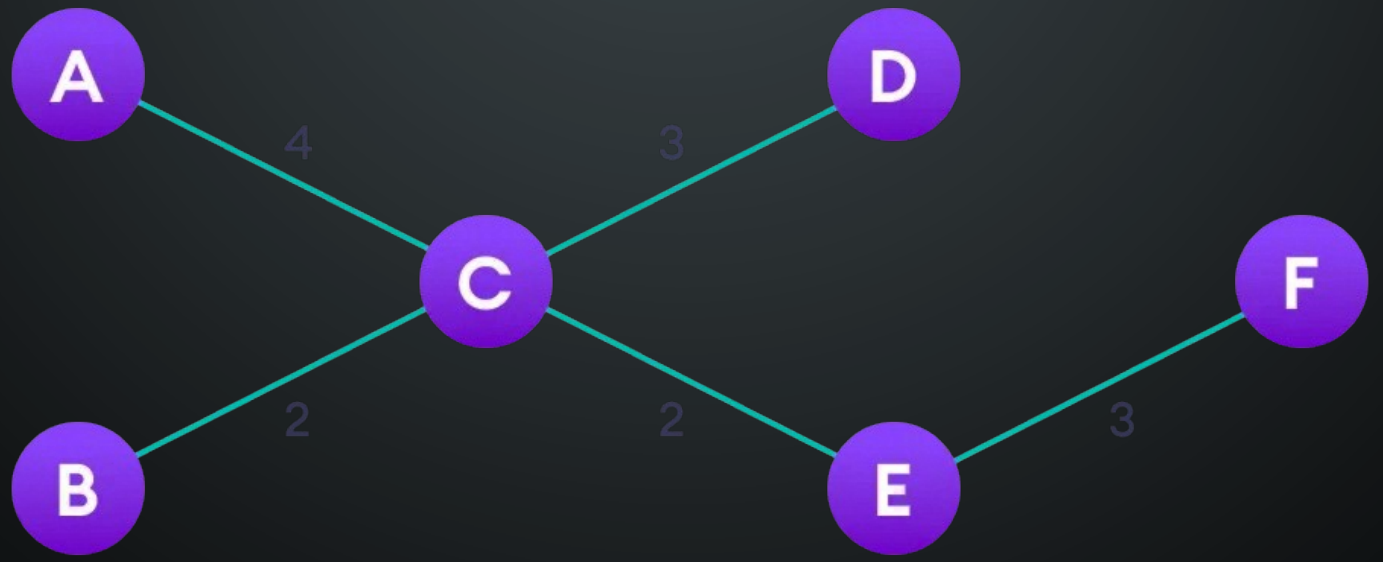
Step: 4

EXAMPLE OF KRUSKAL'S ALGORITHM



Step: 5

EXAMPLE OF KRUSKAL'S ALGORITHM



Step: 6

KRUSKAL'S ALGORITHM COMPLEXITY

Any minimum spanning tree algorithm revolves around checking if adding an edge creates a loop or not.

The most common way to find this out is an algorithm called Union Find. The Union-Find algorithm divides the vertices into clusters and allows us to check if two vertices belong to the same cluster or not and hence decide whether adding an edge creates a cycle.

The time complexity of Kruskal's Algorithm is: **O**(**E** log **E**).

PRACTICE: ISLANDS CONNECTION PROBLEM

Problem. You have **N** islands ($1 <= N <= 100$) that you need to connect with tunnels. In addition, you need to keep environmental pollution as low as possible. The pollution is calculated by formula: $P = L^2 E$, where **L** – tunnel length, **E** – pollution coefficient.

Task. Create a program using C/C++/Python to solve this problem.

Input. Four strings: the first one with single number **N**; the second one with **N** numbers – the **X** coordinates ($0 <= X <= 10^6$) of the islands divided by spaces; the third one with **N** numbers – the **Y** coordinates ($0 <= Y <= 10^6$) of the islands divided by spaces; the last one with single number **E** ($0 <= E <= 1$) – pollution coefficient.

Output. The accumulated environmental pollution is rounded to the closest integer.

PRACTICE: ISLANDS CONNECTION PROBLEM

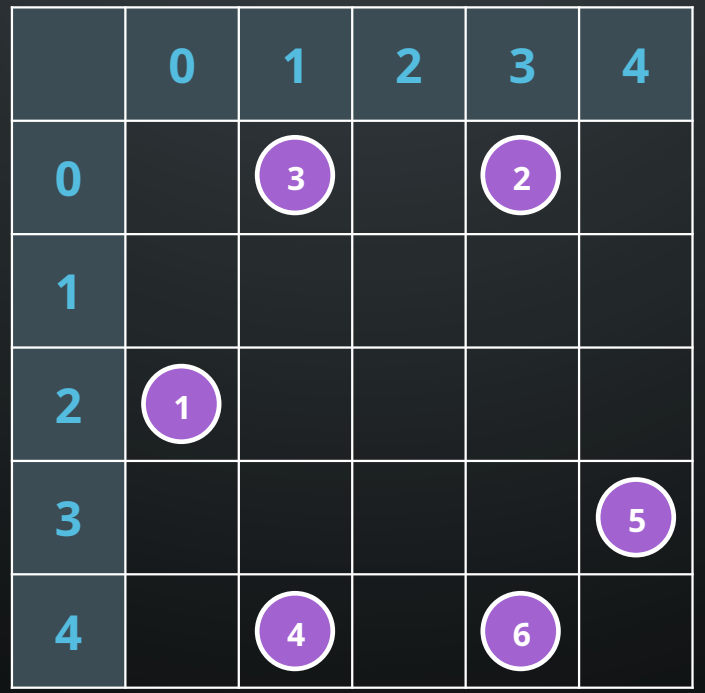
Example. There are **N**=6 islands and pollution coefficient **E**=0.005

6

0 3 1 1 4 3

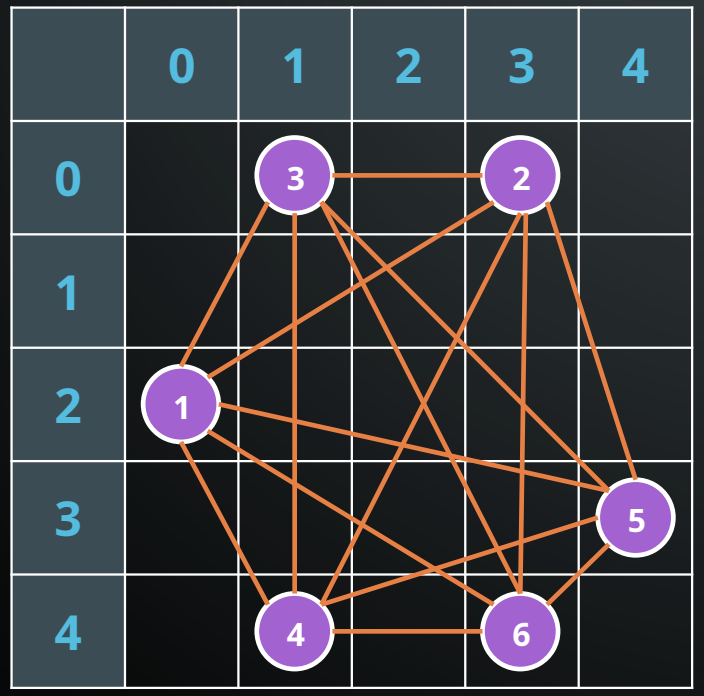
2 0 0 4 3 4

0.005



PRACTICE: ISLANDS CONNECTION PROBLEM

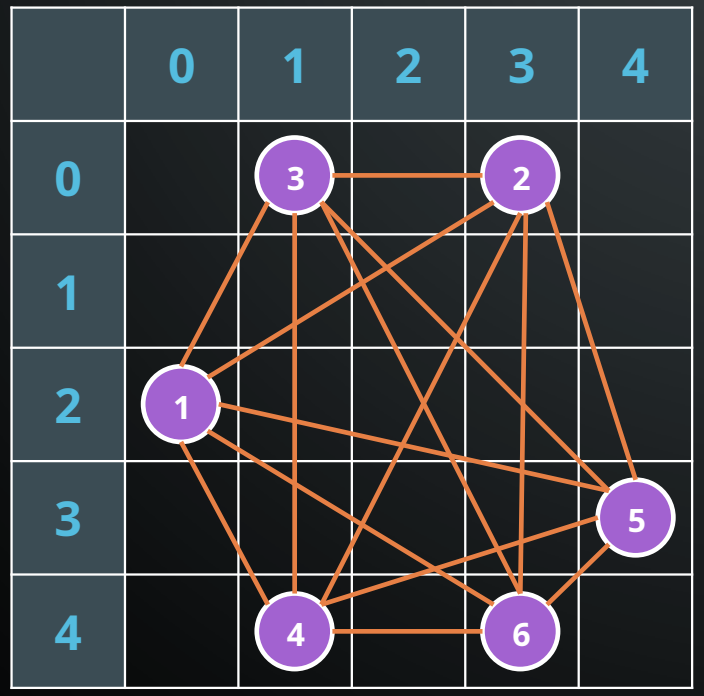
Step 1. Calculate squares length for all edges.



N	EDGE	LENGTH
1	1 - 2	$(3-0)^2 + (2-0)^2 = 13$
2	1 - 3	$(1-0)^2 + (2-0)^2 = 5$
3	1 - 4	$(1-0)^2 + (4-2)^2 = 5$
4	1 - 5	$(4-0)^2 + (3-2)^2 = 17$
5	1 - 6	$(3-0)^2 + (4-2)^2 = 13$
6	2 - 3	$(3-1)^2 + (0-0)^2 = 4$
7	2 - 4	$(3-1)^2 + (4-0)^2 = 20$
8	2 - 5	$(4-3)^2 + (3-0)^2 = 10$
9	2 - 6	$(0-0)^2 + (4-0)^2 = 16$
10	3 - 4	$(1-1)^2 + (4-0)^2 = 16$
11	3 - 5	$(4-1)^2 + (3-1)^2 = 13$
12	3 - 6	$(3-1)^2 + (4-0)^2 = 20$
13	4 - 5	$(4-1)^2 + (4-3)^2 = 10$
14	4 - 6	$(3-1)^2 + (4-4)^2 = 4$

PRACTICE: ISLANDS CONNECTION PROBLEM

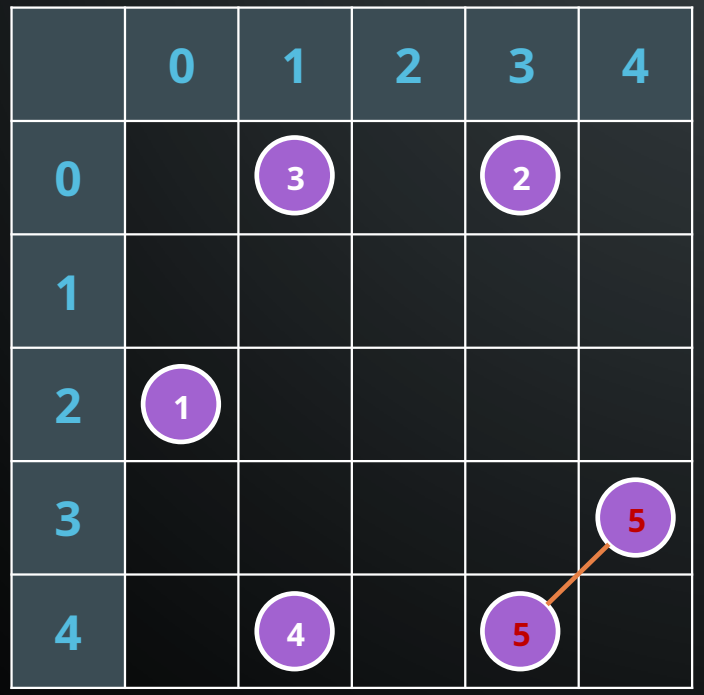
Step 2. Sort lengths in ascending order.



N	EDGE	LENGTH
1	5 - 6	$(4-3)^2 + (4-3)^2 = 2$
2	2 - 3	$(3-1)^2 + (0-0)^2 = 4$
3	4 - 6	$(3-1)^2 + (4-4)^2 = 4$
4	1 - 3	$(1-0)^2 + (2-0)^2 = 5$
5	1 - 4	$(1-0)^2 + (4-2)^2 = 5$
6	2 - 5	$(4-3)^2 + (3-0)^2 = 10$
7	4 - 5	$(4-1)^2 + (4-3)^2 = 10$
8	1 - 2	$(3-0)^2 + (2-0)^2 = 13$
9	1 - 6	$(3-0)^2 + (4-2)^2 = 13$
10	3 - 5	$(4-1)^2 + (3-1)^2 = 13$
11	2 - 6	$(0-0)^2 + (4-0)^2 = 16$
12	3 - 4	$(1-1)^2 + (4-0)^2 = 16$
13	1 - 5	$(4-0)^2 + (3-2)^2 = 17$
14	2 - 4	$(3-1)^2 + (4-0)^2 = 20$

PRACTICE: ISLANDS CONNECTION PROBLEM

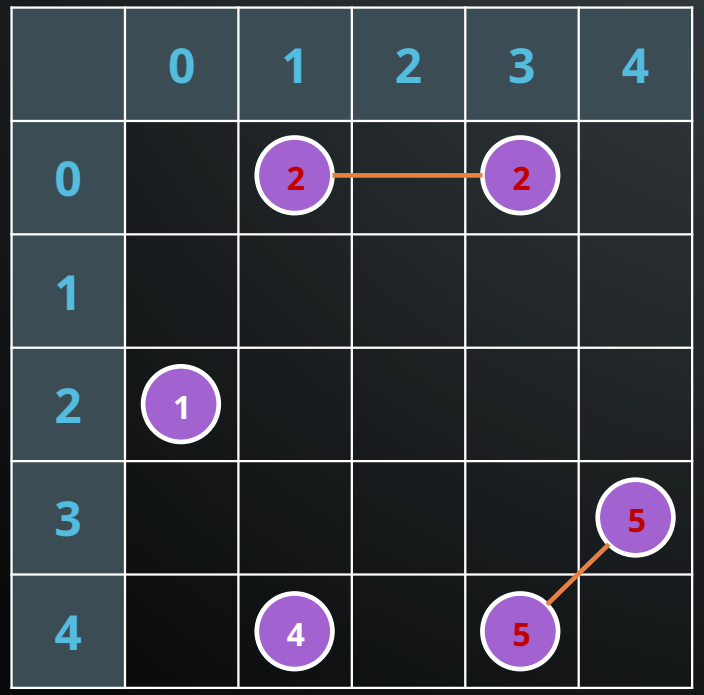
Step 3. Select the first edge from the list and add it to MST.



N	EDGE	LENGTH
1	5 - 6	$(4-3)^2 + (4-3)^2 = 2$
2	2 - 3	$(3-1)^2 + (0-0)^2 = 4$
3	4 - 6	$(3-1)^2 + (4-4)^2 = 4$
4	1 - 3	$(1-0)^2 + (2-0)^2 = 5$
5	1 - 4	$(1-0)^2 + (4-2)^2 = 5$
6	2 - 5	$(4-3)^2 + (3-0)^2 = 10$
7	4 - 5	$(4-1)^2 + (4-3)^2 = 10$
8	1 - 2	$(3-0)^2 + (2-0)^2 = 13$
9	1 - 6	$(3-0)^2 + (4-2)^2 = 13$
10	3 - 5	$(4-1)^2 + (3-1)^2 = 13$
11	2 - 6	$(0-0)^2 + (4-0)^2 = 16$
12	3 - 4	$(1-1)^2 + (4-0)^2 = 16$
13	1 - 5	$(4-0)^2 + (3-2)^2 = 17$
14	2 - 4	$(3-1)^2 + (4-0)^2 = 20$

PRACTICE: ISLANDS CONNECTION PROBLEM

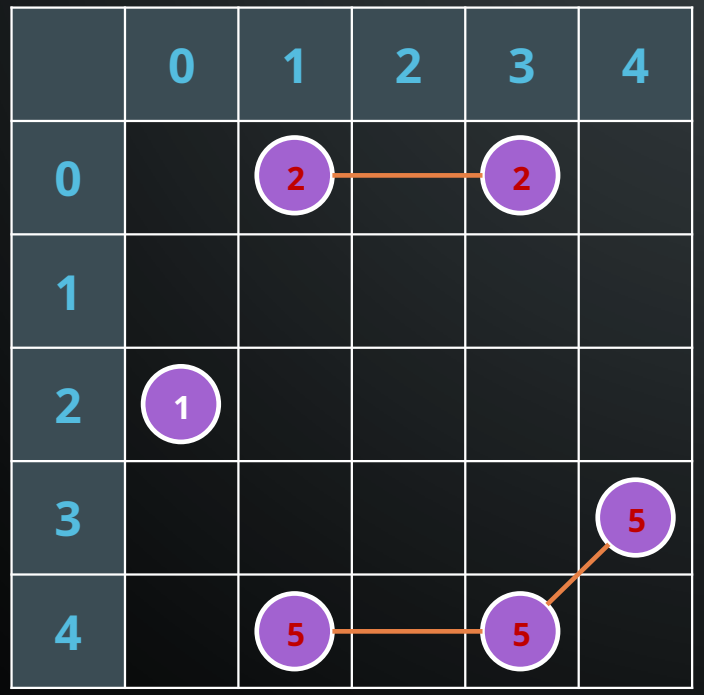
Step 4. Select the next edge from the list and add it to MST.



N	EDGE	LENGTH
1	5 - 6	$(4-3)^2 + (4-3)^2 = 2$
2	2 - 3	$(3-1)^2 + (0-0)^2 = 4$
3	4 - 6	$(3-1)^2 + (4-4)^2 = 4$
4	1 - 3	$(1-0)^2 + (2-0)^2 = 5$
5	1 - 4	$(1-0)^2 + (4-2)^2 = 5$
6	2 - 5	$(4-3)^2 + (3-0)^2 = 10$
7	4 - 5	$(4-1)^2 + (4-3)^2 = 10$
8	1 - 2	$(3-0)^2 + (2-0)^2 = 13$
9	1 - 6	$(3-0)^2 + (4-2)^2 = 13$
10	3 - 5	$(4-1)^2 + (3-1)^2 = 13$
11	2 - 6	$(0-0)^2 + (4-0)^2 = 16$
12	3 - 4	$(1-1)^2 + (4-0)^2 = 16$
13	1 - 5	$(4-0)^2 + (3-2)^2 = 17$
14	2 - 4	$(3-1)^2 + (4-0)^2 = 20$

PRACTICE: ISLANDS CONNECTION PROBLEM

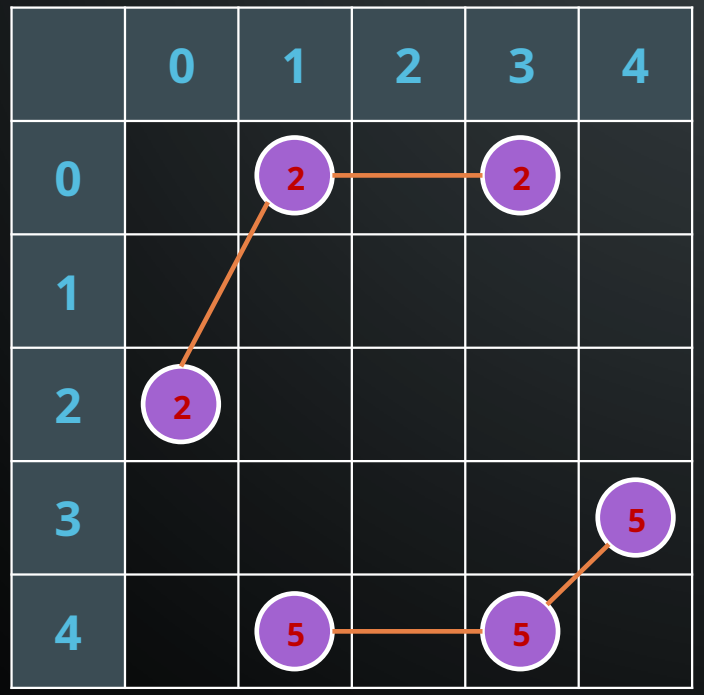
Step 5. Select the next edge from the list and add it to MST.



N	EDGE	LENGTH
1	5 - 6	$(4-3)^2 + (4-3)^2 = 2$
2	2 - 3	$(3-1)^2 + (0-0)^2 = 4$
3	4 - 6	$(3-1)^2 + (4-4)^2 = 4$
4	1 - 3	$(1-0)^2 + (2-0)^2 = 5$
5	1 - 4	$(1-0)^2 + (4-2)^2 = 5$
6	2 - 5	$(4-3)^2 + (3-0)^2 = 10$
7	4 - 5	$(4-1)^2 + (4-3)^2 = 10$
8	1 - 2	$(3-0)^2 + (2-0)^2 = 13$
9	1 - 6	$(3-0)^2 + (4-2)^2 = 13$
10	3 - 5	$(4-1)^2 + (3-1)^2 = 13$
11	2 - 6	$(0-0)^2 + (4-0)^2 = 16$
12	3 - 4	$(1-1)^2 + (4-0)^2 = 16$
13	1 - 5	$(4-0)^2 + (3-2)^2 = 17$
14	2 - 4	$(3-1)^2 + (4-0)^2 = 20$

PRACTICE: ISLANDS CONNECTION PROBLEM

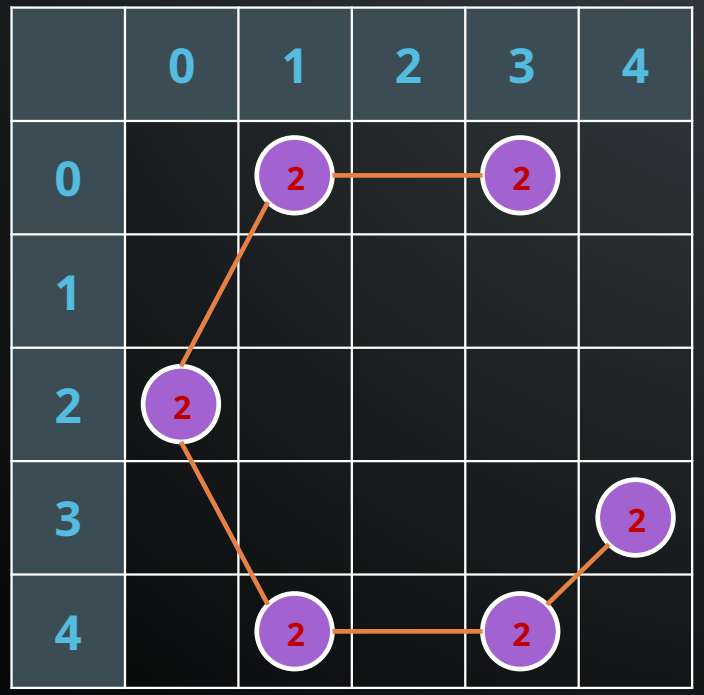
Step 6. Select the next edge from the list and add it to MST.



N	EDGE	LENGTH
1	5 - 6	$(4-3)^2 + (4-3)^2 = 2$
2	2 - 3	$(3-1)^2 + (0-0)^2 = 4$
3	4 - 6	$(3-1)^2 + (4-4)^2 = 4$
4	1 - 3	$(1-0)^2 + (2-0)^2 = 5$
5	1 - 4	$(1-0)^2 + (4-2)^2 = 5$
6	2 - 5	$(4-3)^2 + (3-0)^2 = 10$
7	4 - 5	$(4-1)^2 + (4-3)^2 = 10$
8	1 - 2	$(3-0)^2 + (2-0)^2 = 13$
9	1 - 6	$(3-0)^2 + (4-2)^2 = 13$
10	3 - 5	$(4-1)^2 + (3-1)^2 = 13$
11	2 - 6	$(0-0)^2 + (4-0)^2 = 16$
12	3 - 4	$(1-1)^2 + (4-0)^2 = 16$
13	1 - 5	$(4-0)^2 + (3-2)^2 = 17$
14	2 - 4	$(3-1)^2 + (4-0)^2 = 20$

PRACTICE: ISLANDS CONNECTION PROBLEM

Step 7. Select the next edge from the list and add it to MST.



N	EDGE	LENGTH
1	5 - 6	$(4-3)^2 + (4-3)^2 = 2$
2	2 - 3	$(3-1)^2 + (0-0)^2 = 4$
3	4 - 6	$(3-1)^2 + (4-4)^2 = 4$
4	1 - 3	$(1-0)^2 + (2-0)^2 = 5$
5	1 - 4	$(1-0)^2 + (4-2)^2 = 5$
6	2 - 5	$(4-3)^2 + (3-0)^2 = 10$
7	4 - 5	$(4-1)^2 + (4-3)^2 = 10$
8	1 - 2	$(3-0)^2 + (2-0)^2 = 13$
9	1 - 6	$(3-0)^2 + (4-2)^2 = 13$
10	3 - 5	$(4-1)^2 + (3-1)^2 = 13$
11	2 - 6	$(0-0)^2 + (4-0)^2 = 16$
12	3 - 4	$(1-1)^2 + (4-0)^2 = 16$
13	1 - 5	$(4-0)^2 + (3-2)^2 = 17$
14	2 - 4	$(3-1)^2 + (4-0)^2 = 20$

PRACTICE: ISLANDS CONNECTION PROBLEM

Code example:

```
#include <iostream>
#define MAX 1000

int N, A, X[MAX], Y[MAX];
double E;

int main()
{
    cin >> N;
    for(int i = 0; i < N; i++) cin >> X[i];
    for(int i = 0; i < N; i++) cin >> Y[i];
    cin >> E;
    /*your code here*/
    cout << A;

    return 0;
}
```

THANK YOU!