Murashko Artem B20-02

Theoretical part.

3.1 Minimum spanning tree

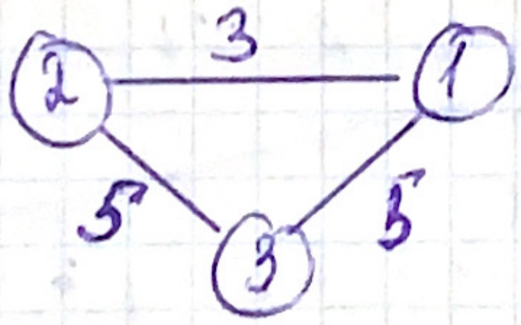
1. G_n, $n > 2$

2. edge: $\begin{cases} |i-j| \le 2 \\ \lfloor (i-1)/4 \rfloor = \lfloor (j-1)/4 \rfloor \\ \begin{cases} |i-j| \ge 4 \\ i+j \ne 4 \end{cases} \end{cases}$

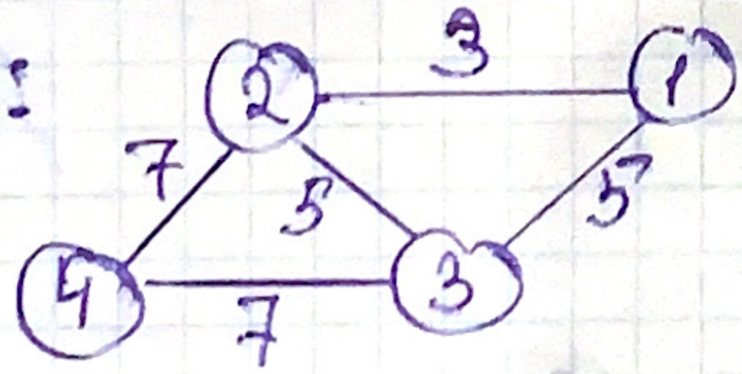
3. $W(v_i, v_j) = i + j + (|i-j| - 1)^2$

(1) G_2: 

Cost of MST = 3

G_3: 

Cost of MST = 3+5

G_4: 

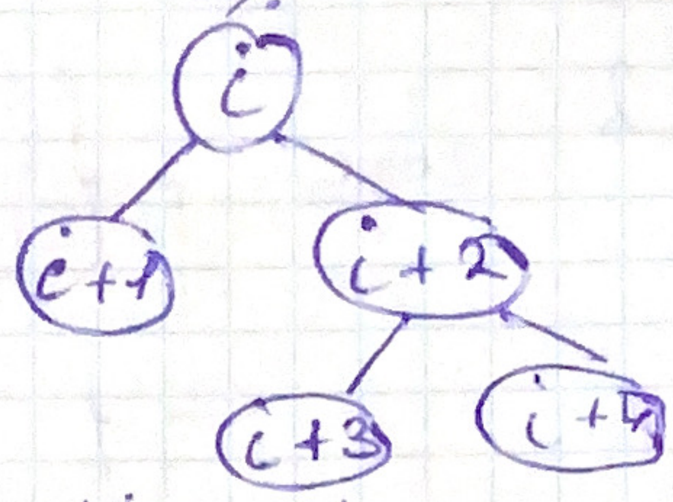
Cost of MST = 3+5+7

Let's build adjacency matrix. The cells will contain the type of the edge: 1 if $|-i-j| \le 2$ & $\lfloor (i-1)/4 \rfloor = \lfloor (j-1)/4 \rfloor$
2 if $|-i-j| \le 4$ & $i+j \ne 4$

	1	2	3	4	5	6	7	8	9	10	11	12
1		1	1/2									
2	1		1	1		2						
3	1/2	1		1	2							
4		1	1					2				
5			1/2			1	1/2					
6		2			1		1	1		2		
7					1/2	1		1	2			
8				2		1	1					2
9							2			1	1/2	
10						2			1		1	1
11									1/2	1		1
12								2		1	1	

Let's find a pattern.

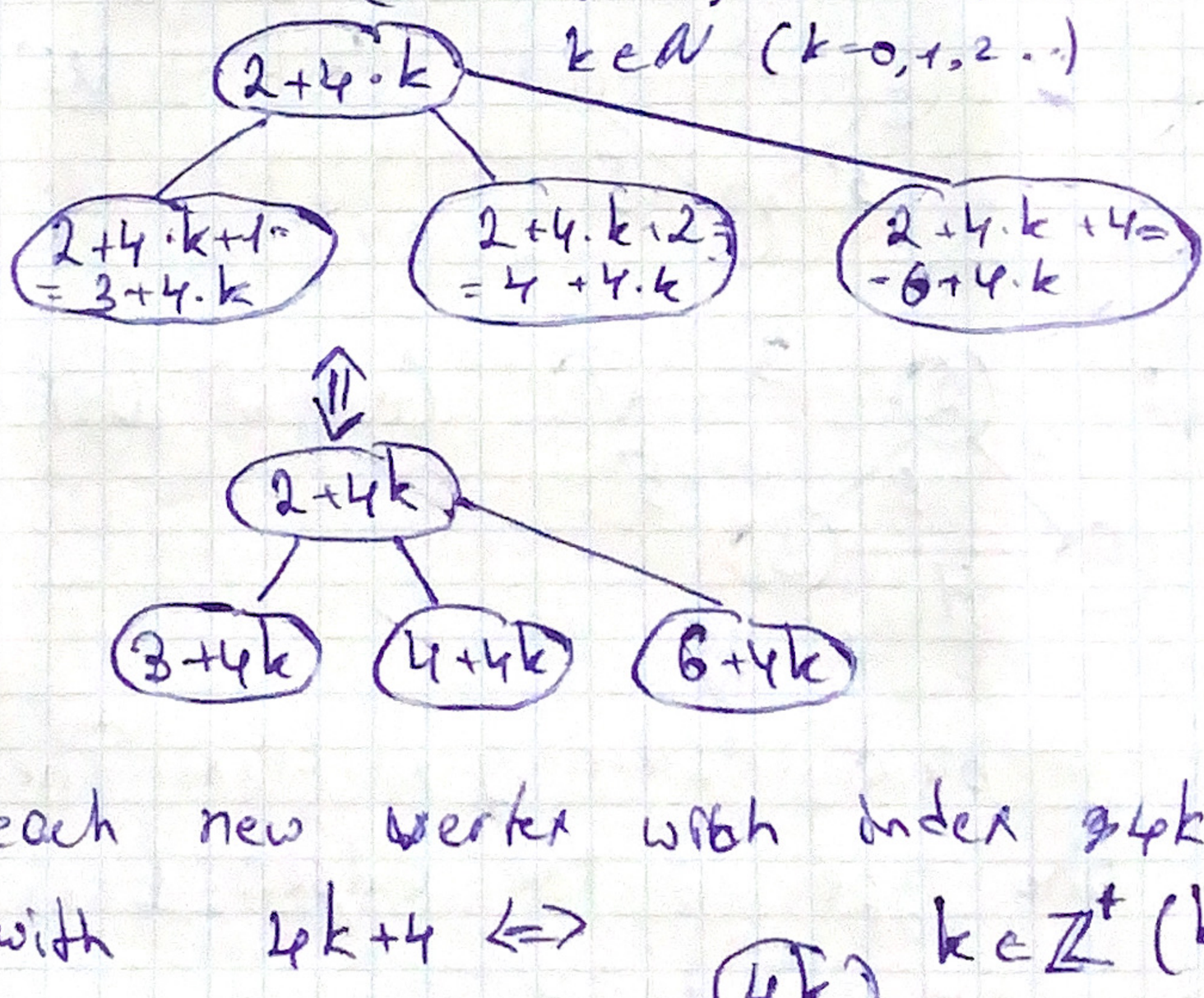
. Note that each new added odd vertex will be connected to the next (new) two vertices $\Leftrightarrow$ i / 2



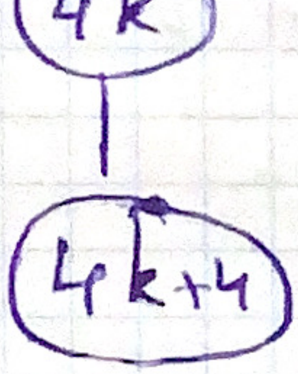
! / don't take into account vertices with less than the considered vertex.

* Every odd vertex except 1st connects with 4 others vertices

. Note that each vertex will be connected with $(2+4\cdot k)+1$, $(2+4\cdot k)+2$ and $(2+4\cdot k)+4$ $\Leftrightarrow$ with index $2+4\cdot k$, $k=0,1,2\ldots$



. Note that each new vertex with index $4k$, $k=1,2\ldots$ will be connected with $4k+4$ $\Leftrightarrow$ $k\in\mathbb{Z}^+$ $(k=1,2,3,\ldots)$

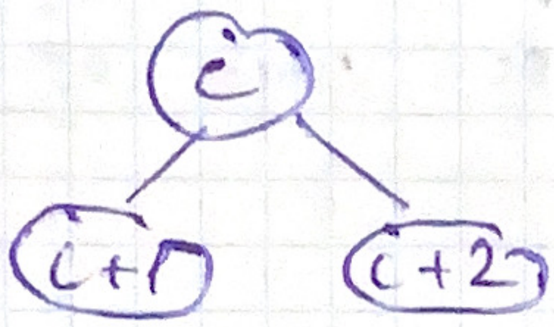


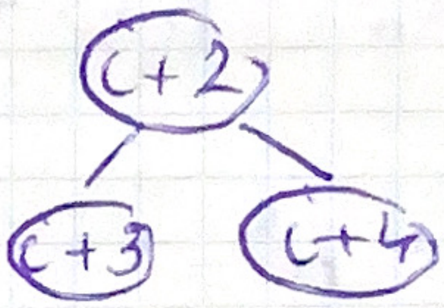
Since weight of edge equal to $i+j+(|i-j|-1)^2$,
weight takes a minimum when $|i-j|$ is minimum
$\Leftrightarrow |i-j| = 1 \;//\; i \neq j,\ i, j \in \mathbb{Z}^+$

$$\underbrace{i}_{>0} + \underbrace{j}_{>0} + \underbrace{(|i-j|-1)^2}_{\geq 0}$$

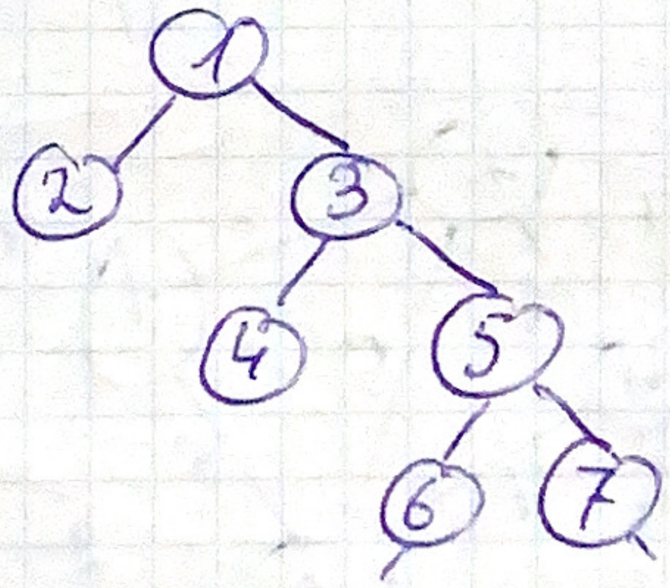
Let's take a look at the first template:

$i \not\equiv 2 \quad (i = 1, 3, 5 \ldots)$



since $i \not\equiv 2 \Rightarrow i+2 \not\equiv 2 \Rightarrow$  *

This means, that using only this templabe we can build minimum spanning tree.

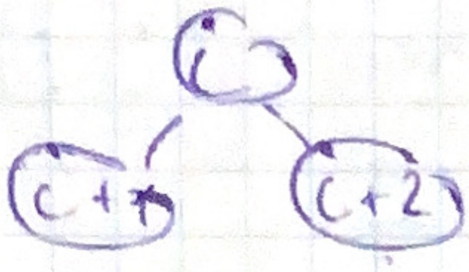


*Now let's proof that weight of edge between i and $i+2$ less than weight of edges between $i - (i+1) - (i+2)$

$$i + (i+2) + (|i-(i+2)|-1)^2 \overset{?}{<} i + (i+1) + (|i-(i+1)|-1)^2 +$$
$$+ (i+1) + (i+2) + (|(i+1)-(i+2)|-1)^2$$

$$2i + 2 + 1^2 \overset{?}{<} (2i+1) + (2i+3)$$
$$2i + 3 < 4i + 4, \text{ q.e.d}$$

Let's derive a formula of cost of MST:



$i + i+1 - (|i - i-1| - 1)^2 + = 2i + 1$

$i + i+2 - (|i - i-2| - 1)^2 = 2i + 2 + 1 = 2i + 3$

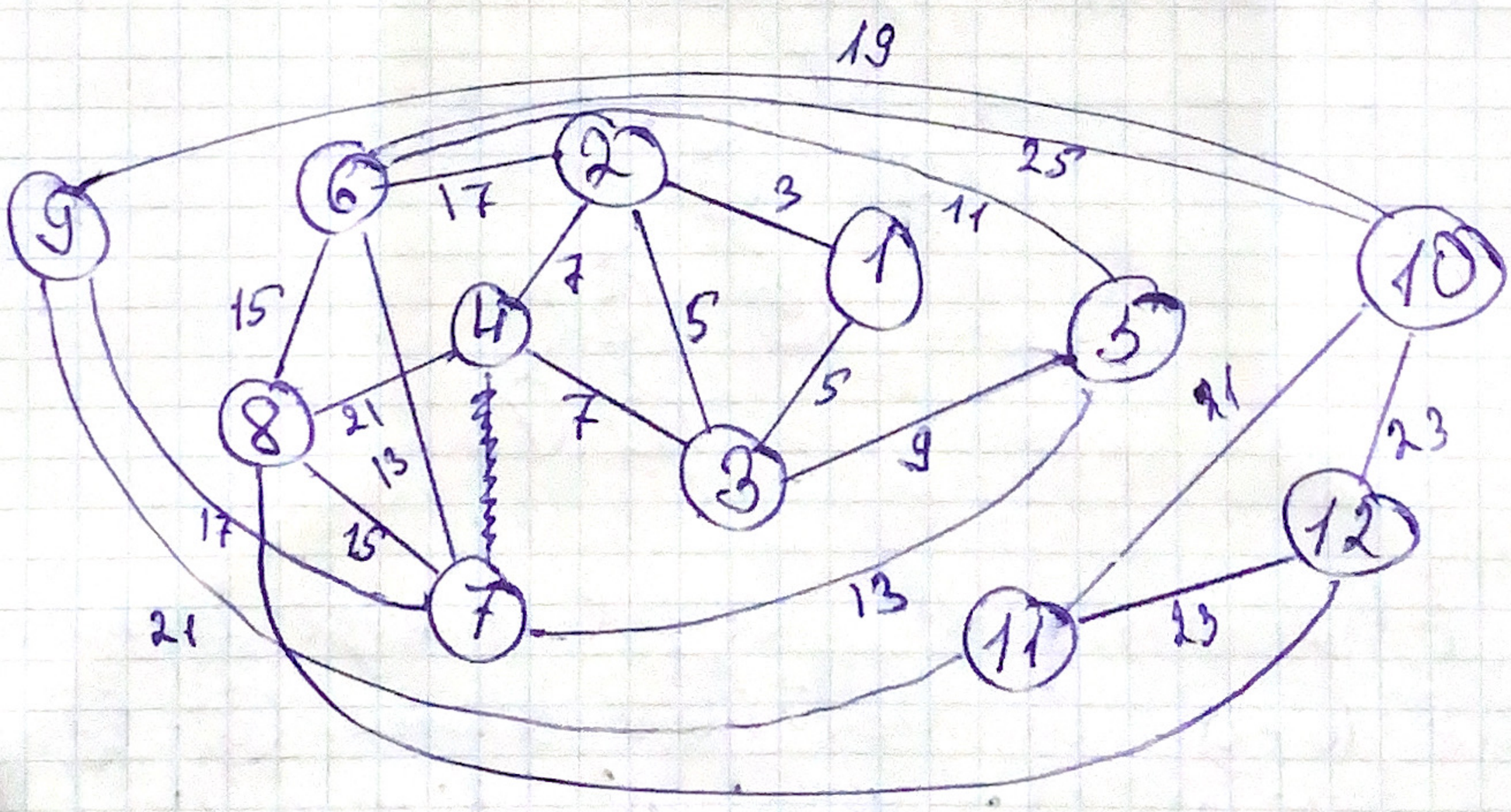
...

All in all, the cost will be sum of the odd numbers starting from 3.

$$C_n : 3 + 5 + 7 + \dots + (2n-1) \Leftrightarrow \sum_{i=1}^{n} 2i+1 = n^2 - 1$$

Answer: $n^2 - 1$

① Prim's algorithm



1 step

Let's start from vertex 1.
Initially, our priority queue (PQ) looks like this:

PQ.

0, (1, None)
∞, (2, None)
∞, (3, None)
...

Remove min from PQ:

M.S. Tree

A ①

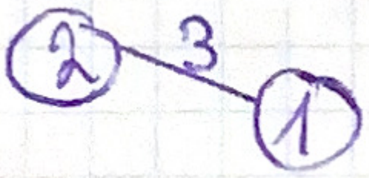
→

PQ

3, (2, 12)
5, (3, 13)
∞, (4, None)
...

Remove min from PQ:

M.S. Tree



→

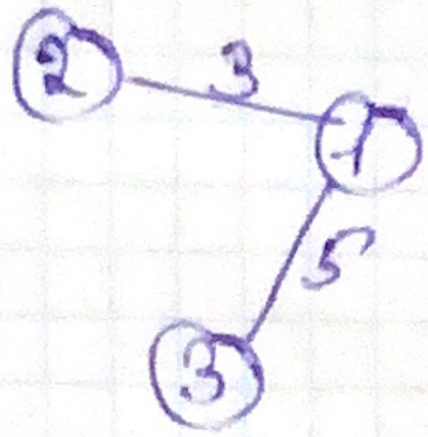
PQ

5, (3, 13)
7, (4, 24)
17, (6, 26)
∞, (5, None)
...

2 step

Remove min from PQ:

M.S.T



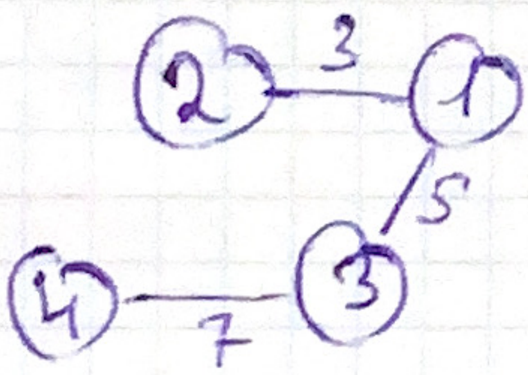
PQ

7, (4, 34)
9, (5, 35)
17, (6, 26)
∞, (7, None)
...

3 step

Remove min from PQ.

M.S.T.



PQ

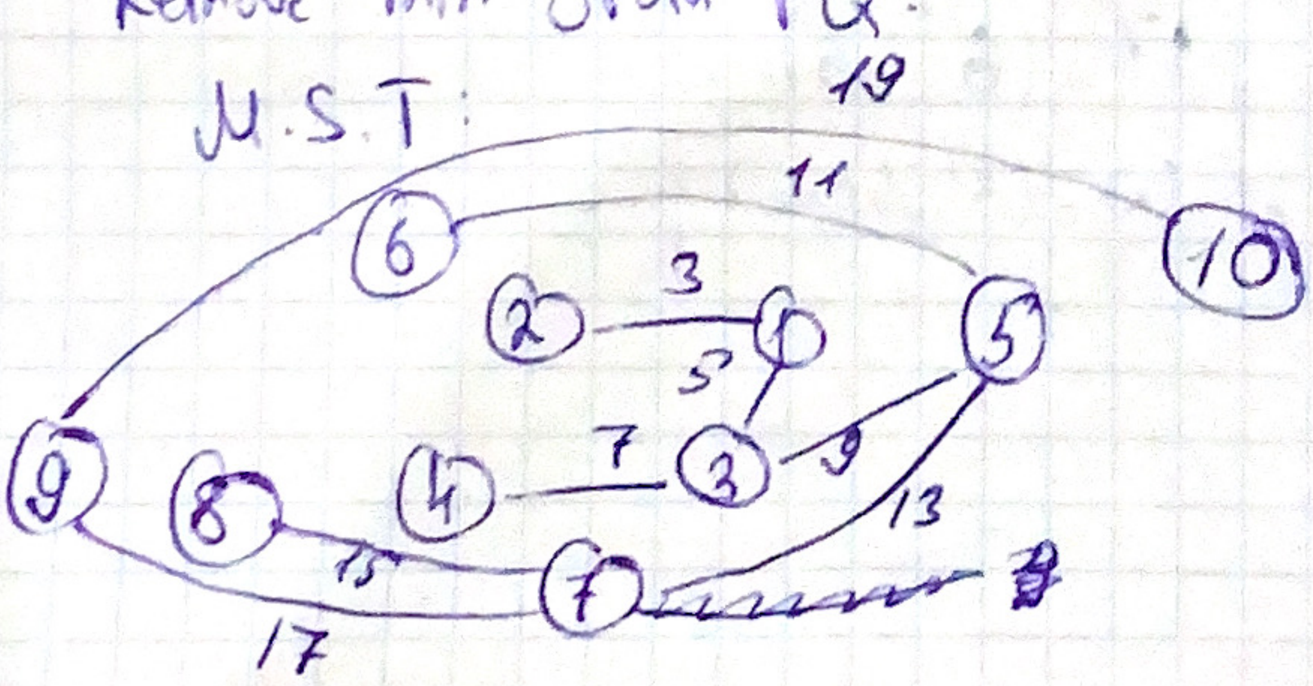
9, (5, 35)
~~11, (7, 33)~~ 17, (6, 26)
21, (8, 48)
∞, (7, None)
...

...

#8 step

Remove min from PQ:

M.S.T:



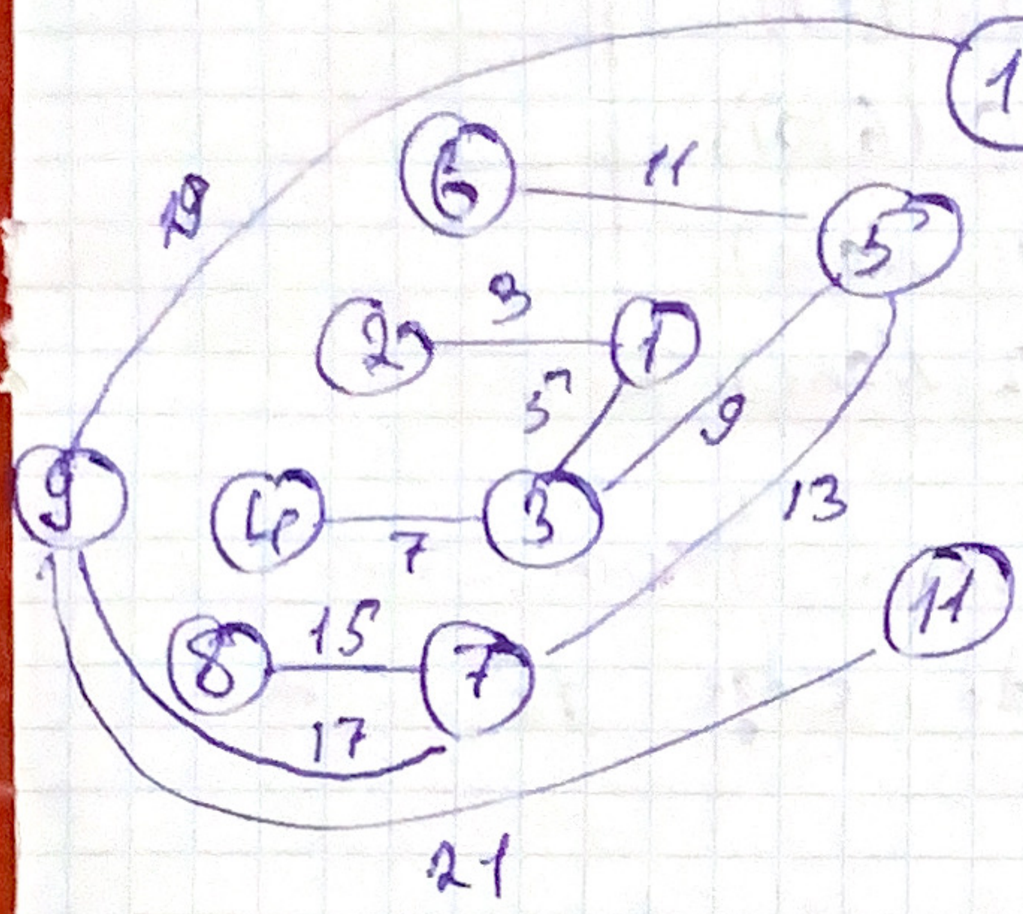
PQ

21, (11, (9, 11))
23, (12, (10, 12))

10 step

Remove mm from PQ

MST



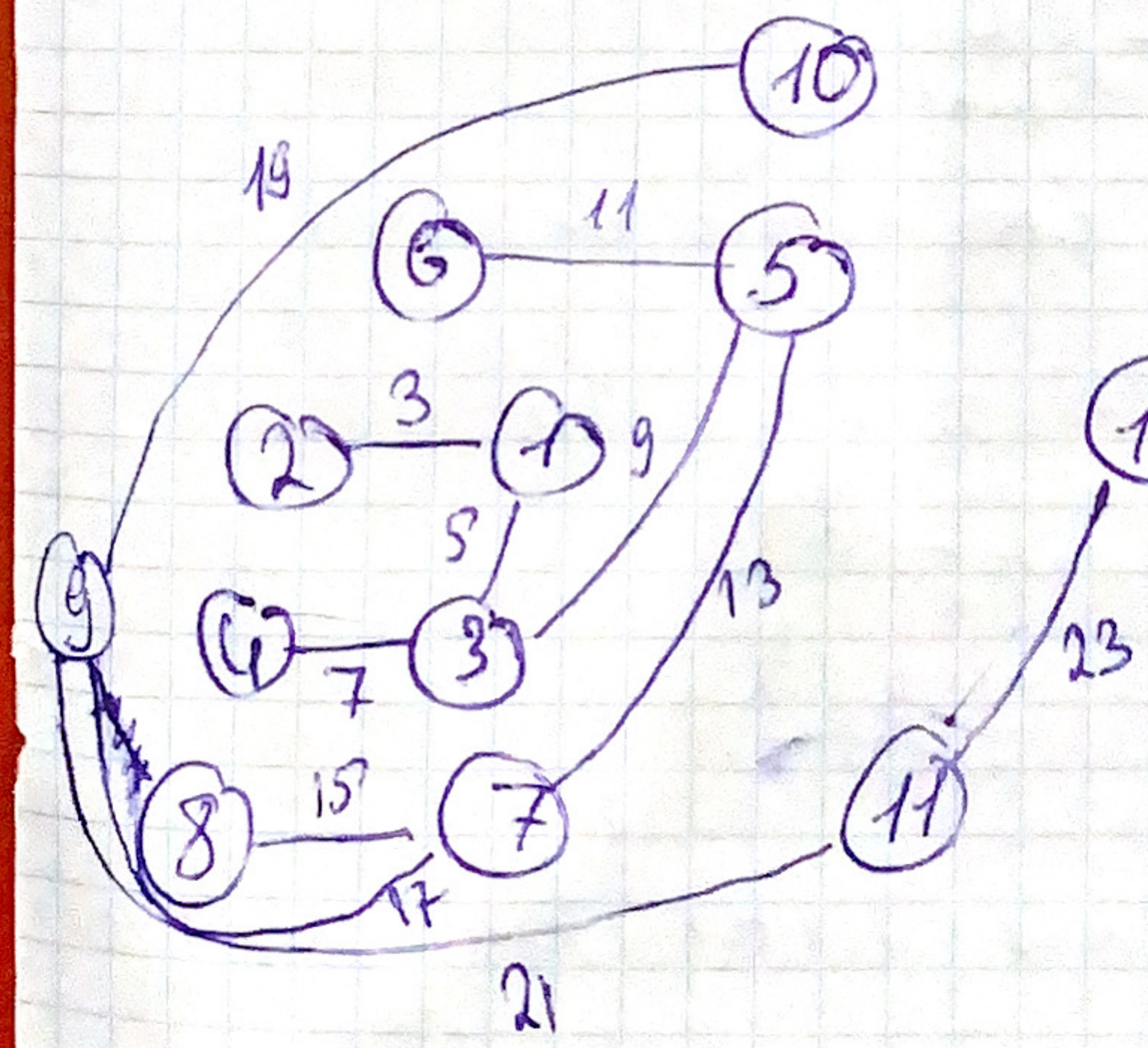
PQ

23, (12, (11, 12))

11 step

Remove mm (last element) from PQ

MST



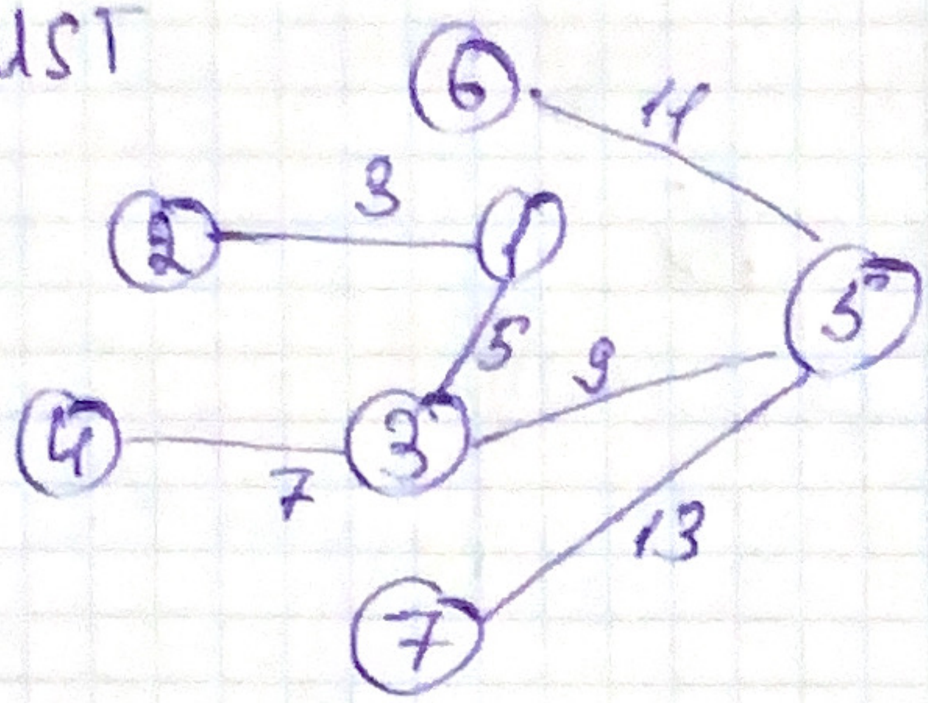
PQ

empty

Total cost: 143

After 6 step.

MST



Our PQ:

| 15, (8, (7,8))
| 17, (9, (7,9))
| 25, (10, (6,10))
| ∞, (11, None)
| ∞, (12, None)

On 7 step we will extract nom from our priority queue, i.e. we should add edge 78 with weight 15

3.2 Shortest path

① We need two 2d arrays in order ~~to~~ to store minimum distances and to store id of ancestors. // d[][], p[][]

d[][] — matrix of distances

p[][] — matrix of ancestors (to restore the path)

Also we need two arrays in order to store vertices and edges // V[], E[]

V[] — array of vertices

E[] — array of edges

② If we want to add new vertex, we just need to resize our matrix d[][] and p[][], because the added vertex will be in another connected component

```
addVertex(val)
    temp_d[][] = d[][]
    temp_p[][] = p[][]
    resize(d, d.size()+1) // (increas size of matrix from
                                n×n to (n+1)×(n+1)
    resize(p, p.size()+1)
    for i in 1..to d.size()
        for j in 1 to d.size()
            d[0][j] = INF
            p[i][j] = 0
            if (i = j)
                d[i][j] = 0
for i in 1 to temp_d.size()
    for j in 1 to temp_d.size()
        d[i][j] = temp_d[i][j]
        p[i][j] = temp_p[i][j]
v.add(val)

addEdge(from, to)
    e.add(from, to)
    if (from, to).weight < d[from][to]
        d[from][to] = (from, to).weight
        k = from
        for i in 1 to v.size()
            for j in 1 to v.size()
                if d[0][j] > d[0][k] + d[k][j]
                    d[0][j] = d[i][k] + d[k][j]
```

```
                    p[i][j] = k
k = to
for i in 1 to v.size()
        for j in 1 to v.size()
                if d[i][j] > d[i][k] + d[k][j]
                    d[i][j] = d[i][k] + d[k][j]
                    p[i][j] = k
```

③ addVertex(val)

$T(V) = O(V^2) + O(V^2) + O(V^2) = O(V^2)$

- $O(V^2)$ ↑ initialization of matrix (first loops)
- $O(V^2)$ ↑ resizing of matrix
- $O(V^2)$ ↑ copy temp matrices to d and p.

addEdge(from, to)

$T(V) = O(V^2) + O(V^2) = O(V^2)$

- $O(V^2)$ ↑ updating all paths with vertex from
- $O(V^2)$ ↑ updating all paths with vertex to