3.3.1

1. The total cost of MST = $n^2 - 1$
   a. Let's check all possible weights of edges to vertex i (i > j). Then we have a few rules of creating edges between i and j (W = weight of edge):
      i. If: $|i-j| = 1 \quad and \quad \left\lfloor \frac{i-1}{4} \right\rfloor = \left\lfloor \frac{j-1}{4} \right\rfloor \quad => \quad \boldsymbol{i\%4 \neq 1}$
      Then: $W = i + j + (1-1)^2 = i + j$
      But: i > j. Therefore: i = j + 1
      We have: $W = i + j + (1-1)^2 = i + j = 2j + 1 = \boldsymbol{2i - 1}$
      ii. If: $|i-j| = 2 \quad and \quad \left\lfloor \frac{i-1}{4} \right\rfloor = \left\lfloor \frac{j-1}{4} \right\rfloor \quad => \quad \boldsymbol{i\%4 \neq 1;\ 2}$
      Then: $W = i + j + (2-1)^2 = i + j + 1$
      But: i > j. Therefore: i = j + 2
      We have: $W = i + j + (2-1)^2 = i + j + 1 = 2j + 3 = \boldsymbol{2i - 1}$
      iii. If: $|i-j| = 4 \quad and \quad (i+j)\%4 \equiv 0 \quad => \quad \boldsymbol{i\%4 = 2\ and\ j\%4 = 2}$
      $\boldsymbol{i\%4 = 0\ and\ j\%4 = 0}$
      Then: $W = i + j + (4-1)^2 = i + j + 9$
      But: i > j. Therefore: i = j + 4
      We have: $W = i + j + (4-1)^2 = i + j + 9 = 2j + 13 = \boldsymbol{2i + 5}$
      iv. If: $|i-j| = 2 \ and \quad (i+j)\%4 \equiv 0 \quad => \quad \boldsymbol{i\%4 = 3\ and\ j\%4 = 1}$
      $\boldsymbol{i\%4 = 1\ and\ j\%4 = 3}$
      Then: $W = i + j + (2-1)^2 = i + j + 1$
      But: i > j. Therefore: i = j + 2
      We have: $W = i + j + (2-1)^2 = i + j + 1 = 2j + 3 = \boldsymbol{2i - 1}$
   b. To construct MST from given graph we need for each vertex take edge with minimal weight. Therefore:
      For each vertex with index i, where $\boldsymbol{i\%4 \in \{2, 3, 0\}}$, we can use first rule and connect it with vertex with index = i – 1. We will have edge (i-1, i) with W = 2i-1
      For each vertex with index i, where $\boldsymbol{i\%4 \in \{1\}}$, we can use fourth rule and connect it with vertex with index = i – 2. We will have edge (i-2, i) with W = 2i-1
   c. Example of edges: (1, 2) = 3, (2, 3) = 5, (3, 4) = 7, (3, 5) = 9 …
   d. Now we can calculate total cost of MST for n > 2. MST weight for n:
      $$\sum_{i=2}^{n} (2i - 1) = 3 + 5 + \ldots + (2n - 1)$$
      This sum equals to sum of arithmetic series: $S_k = k(\frac{a_1 + a_k}{2})$, where k = count of elements in arithmetic series. In our series k = n – 1 (because we start with i = 2)
      $$S_k = k\left(\frac{a_1 + a_k}{2}\right) = (n-1)\left(\frac{3 + (2n-1)}{2}\right) = (n-1)\left(\frac{2n-2}{2}\right) =$$
      $$(n-1)(n+1) = \boldsymbol{n^2 - 1}$$

2. Let's verify it using Prim's algorithm with priority queue.
    a. We will start our path with vertex with index 1. (I will write index and distance to this vertex in format: (index, weight))
    On first step we have only one vertex in our MST. And we update for adjacent vertices distance in PQ. (All edges, that we check will be written in PQ table in format: (distance/weight, [from ,to]))
    Vertices: (1, 0), (2, INF), (3, INF), (4, INF), (5, INF), (6, INF), (7, INF), (8, INF), (9, INF), (10, INF), (11, INF), (12, INF)

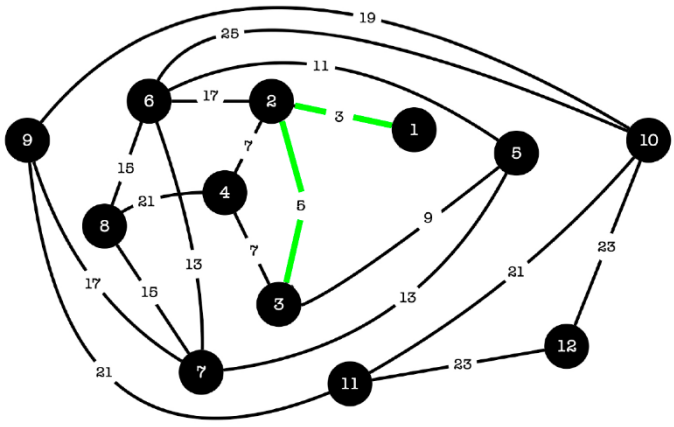
PQ
(3, [2, 1])
(5, [3, 1])
(INF, [4, NIL])
(INF, [5, NIL])
…

    b. Second step:
    We pop minimal edge from PQ. And add this edge and vertex to our MST. After that we again update PQ(but if distance equal to distance in PQ we also update vertex to).
    Vertices: (1, 0), (2, 3), (3, INF), (4, INF), (5, INF), (6, INF), (7, INF), (8, INF), (9, INF), (10, INF), (11, INF), (12, INF)

PQ
(5, [3, 2])
(7, [4, 2])
(17, [6, 2])
(INF, [5, NIL])
…

    c. Third step:
    We repeat all we did on step b.
    Vertices: (1, 0), (2, 3), (3, 5), (4, INF), (5, INF), (6, INF), (7, INF), (8, INF), (9, INF), (10, INF), (11, INF), (12, INF)

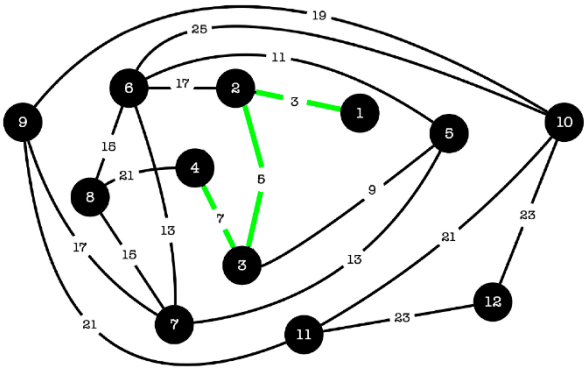
PQ
(7, [4, 3])
(9, [5, 3])
(17, [6, 2])
(INF, [7, NIL])
…



Green edges – Our MST
Also I there Exist edge (8, 12) = 29 – I haven't it on picture, but I didn't forget about it in PQ

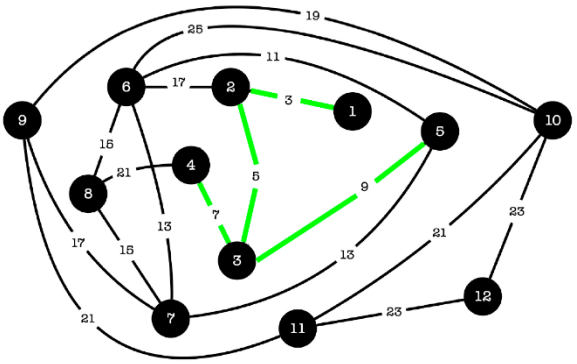
d. Fourth step:
We repeat all we did on step b.
Vertices: (1, 0), (2, 3), (3, 5), (4, 7), (5, INF), (6, INF), (7, INF), (8, INF), (9, INF), (10, INF), (11, INF), (12, INF)

PQ
(9, [5, 3])
(17, [6, 2])
(21, [8, 4])
(INF, [7, NIL])
…



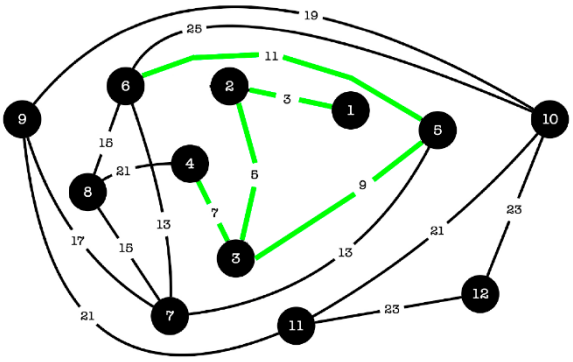
e. Fifth step:
Vertices: (1, 0), (2, 3), (3, 5), (4, 7), (5, 9), (6, INF), (7, INF), (8, INF), (9, INF), (10, INF), (11, INF), (12, INF)

PQ
(11, [6, 5])
(13, [7, 5])
(21, [8, 4])
(INF, [9, NIL])
…



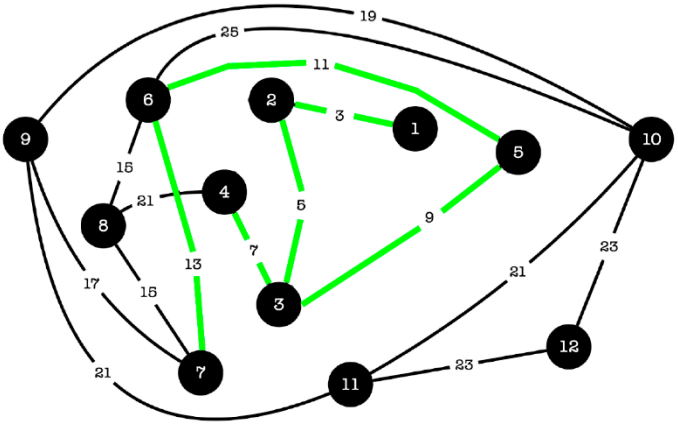
f. Sixth step:
Vertices: (1, 0), (2, 3), (3, 5), (4, 7), (5, 9), (6, 11), (7, INF), (8, INF), (9, INF), (10, INF), (11, INF), (12, INF)

PQ
(13, [7, 6])
(15, [8, 6])
(25, [10, 6])
(INF, [9, NIL])
…



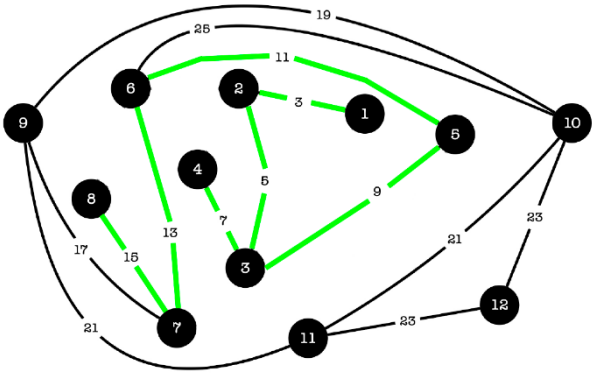
g. Seventh step:
**Six edges added*
On this step we take edge (7, 6) with weight = 13 because this edge placed on the top of PQ. After adding this edge and vertex with index 7, we check all edges from this vertex, and update our PQ. From vertex 7 we can go to vertices: 8 and 9. (Vertices 5 and 6 in MST, therefore we don't check edges to them (Not in PQ)). And we update PQ for this vertices (check PQ table).
Vertices: (1, 0), (2, 3), (3, 5), (4, 7), (5, 9), (6, 11), (7, 13), (8, INF), (9, INF), (10, INF), (11, INF), (12, INF)

PQ
(15, [8, 7])
(17, [9, 7])
(25, [10, 6])
(INF, [11, NIL])
…



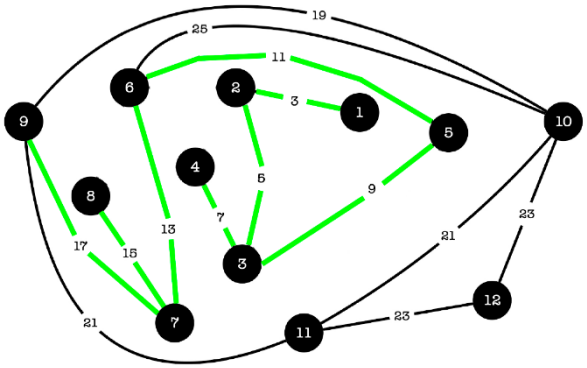
h. Eighth step:
Vertices: (1, 0), (2, 3), (3, 5), (4, 7), (5, 9), (6, 11), (7, 13), (8, 15), (9, INF), (10, INF), (11, INF), (12, INF)

PQ
(17, [9, 7])
(25, [10, 6])
(29, [12, 8])
(INF, [11, NIL])



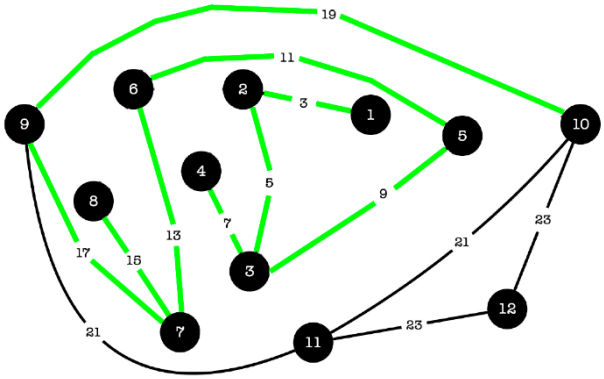
i. Ninth step:
Vertices: (1, 0), (2, 3), (3, 5), (4, 7), (5, 9), (6, 11), (7, 13), (8, 15), (9, 17), (10, INF), (11, INF), (12, INF)

PQ
(19, [10, 9])
(21, [11, 9])
(29, [12, 8])



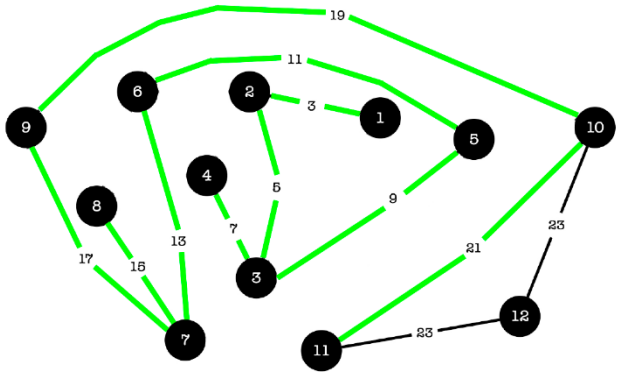
j. Tenth step:
Vertices: (1, 0), (2, 3), (3, 5), (4, 7), (5, 9), (6, 11), (7, 13), (8, 15), (9, 17), (10, 19), (11, INF), (12, INF)

PQ
(21, [11, 10])
(23, [12, 10])



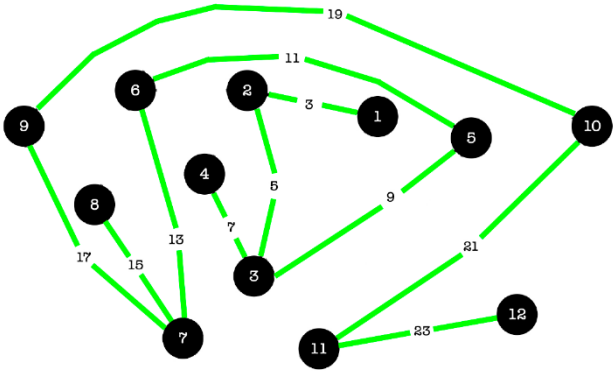
k. Eleventh step:
Vertices: (1, 0), (2, 3), (3, 5), (4, 7), (5, 9), (6, 11), (7, 13), (8, 15), (9, 17), (10, 19), (11, 21), (12, INF)

PQ
(23, [12, 11])

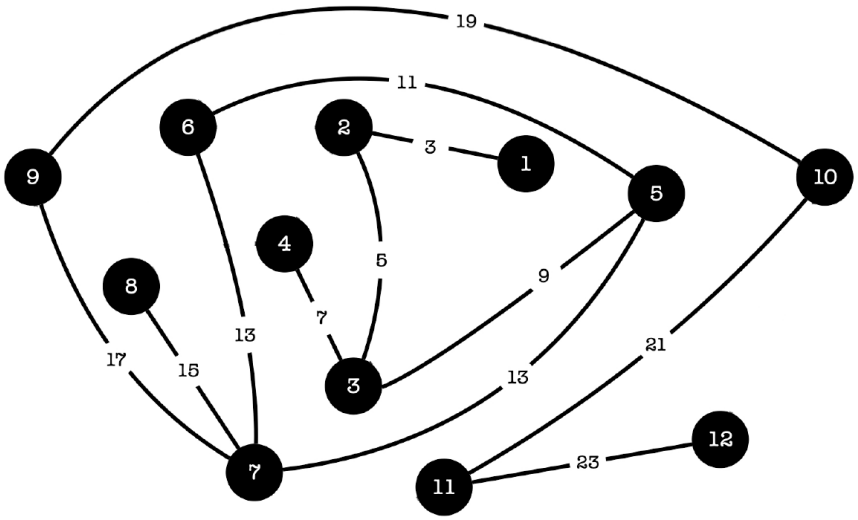


l. Twelfth step:
Vertices: (1, 0), (2, 3), (3, 5), (4, 7), (5, 9), (6, 11), (7, 13), (8, 15), (9, 17), (10, 19), (11, 21), (12, 23)

PQ



Final MST (on each step we repeated actions from step b):



Total cost of this MST = $3 + 5 + 7 + 9 + 11 + 13 + 15 + 17 + 19 + 21 + 23 =$
$= 143 = 12^2 - 1 = n^2 - 1 - QED$

3.3.2

1. Let's store our graph as Adjacency Matrix = G using vectors. If we haven't edge between two vertices i and j, G[i][j] = INF.
   N = number of Vertices
2. Updating states:
   a. If we add vertex:

```
addVertex(v)
  G.push_back(vector)          //O(1)
  For i = 0, i < N, i++        //O(N+1)
    G[i].push_back(INF)        //O(N)
    G[N+1].push_back(INF)      //O(N)
  G[N+1][N+1] = 0              //O(1)
  N++                          //O(1)
```

   Assume that push_back and take element by index in vector works with O(1)
   Therefore, our time complexity = O(N)

   b. If we add edge:

```
addEdge(e)
  G[e.from][e.to] = e.weight   //O(1)
  For i = 0, i < N, i++        //O(N+1)
    For j = 0, j < N, j++      //O(N^2 + N)
      If G[i][j] > G[i][e.from]+G[e.from][e.to]+G[e.to][j]
        G[i][j] = G[i][e.from]+G[e.from][e.to]+G[e.to][j]
```

   Time complexity = O(N^2)
3. Time complexity of *addVertex* = O(N)
   *If we assume that complexity of push_back = O(1)
   **If we assume that complexity of push_back = O(N), then complexity of *addVertex* = O(N^2)

   Time complexity of *addEdge* = O(N^2).
   *Because our inner 'if' will not work more than (N^2 + N) times