CS202
HW 4 Report

Halil Arda Özongun
22202709
Section 1

13 December 2024

Question 1:

The graph given in this question is a complete graph, therefore it is an dense graph. For finding the minimum spanning tree, it would be better if we use Prim's Algorithm rather than Kruskal Algorithm.

We have a graph like this: (matrix[i][j] represents the weight of the edge between vertex i and j)

	1	2	3	4	5	6	7	8	9	10
1	0	5	10	5	6	7	8	9	10	11
2	5	0	5	6	7	8	9	10	11	12
3	10	5	0	7	8	9	10	11	12	13
4	5	6	7	0	9	10	11	12	13	14
5	6	7	8	9	0	11	12	13	14	15
6	7	8	9	10	11	0	13	14	15	16
7	8	9	10	11	12	13	0	15	16	19
8	9	10	11	12	13	14	15	0	17	18
9	10	11	12	13	14	15	16	17	0	19
10	11	12	13	14	15	16	17	18	19	0

Prims Algorithm:

We have to choose a random vertex as the starting vertex. I will select vertex 1 here.

After, in each iteration we need to look for edges and take the minimum one which is a edge starts from vertex's in tree, and goes to vertex's not in tree. We can have a min-heap, and insert all the edges' of a vertex when a vertex added. If i==j we musn't add to min-heap since it is unnecessary to deal. And when we pop a vertex from min-heap we shouldn't insert it if it is already in tree. We can have a bool array so that we can keep track of which vertex is in tree. I am just showing the min-heap (partially) not others since they are easy to follow from photos.

1-

Minimum Spanning Tree:

Total weight = 0

Vertex List after adding this node (weight, from, to):

5, 1, 2

5, 1, 4

6, 1, 5

7, 1, 6

8, 1, 7

9, 1, 8

10, 1, 3

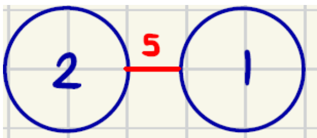
10, 1, 9

11, 1, 10

Take the minimum:

2-

Minimum Spanning Tree:



Total weight = 5

Vertex List after adding this node (weight, from, to):

5, 1, 4

5, 2, 3

6, 1, 5

6, 2, 4

7, 1, 6

7, 2, 5

8, 1, 7

8, 2, 6

9, 1, 8

9, 2, 7

10, 1, 3

10, 1, 9

10, 2, 8

11, 1, 10

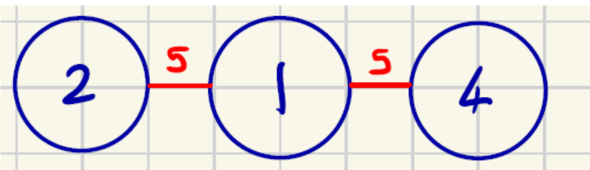
11, 2, 9

12, 2, 10

Take the minimum:

3-

Minimum Spanning Tree:



Total weight = 10

Vertex List after adding this node (weight, from, to):

5, 2, 3

6, 1, 5

6, 2, 4

7, 1, 6

7, 2, 5

7, 4, 3

8, 1, 7

8, 2, 6

9, 1, 8

9, 2, 7

9, 4, 5

10, 1, 3

10, 1, 9

10, 2, 8

10, 4, 6

11, 1, 10

11, 2, 9

11, 4, 7

12, 2, 10

12, 4, 8

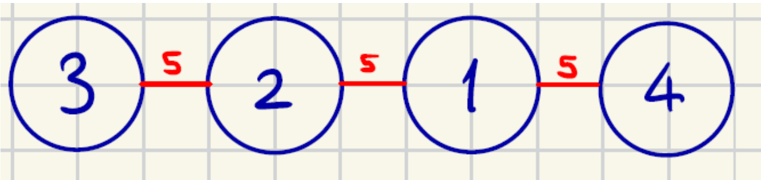
13, 4, 9

14, 4, 10

Take the minimum:

4-

Minimum Spanning Tree:



Total weight = 15

Vertex List after adding this node (weight, from, to):

(stopped adding after here since it is obvious and takes too much space to write whole list. Also as you will see they will be useless after a while since these nodes will be inserted.)

6, 1, 5

6, 2, 4

7, 1, 6

7, 2, 5

7, 4, 3

8, 1, 7

8, 2, 6

9, 1, 8

9, 2, 7

9, 4, 5

10, 1, 3

10, 1, 9

10, 2, 8

10, 4, 6

11, 1, 10

11, 2, 9

11, 4, 7

12, 2, 10

12, 4, 8

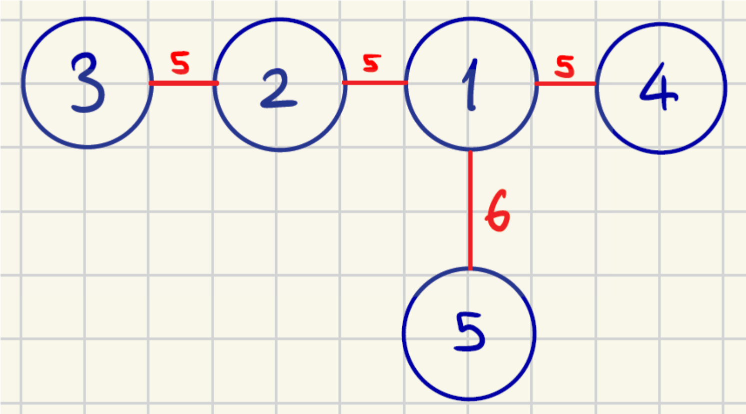
13, 4, 9

14, 4, 10

Take the minimum:

5-

Minimum Spanning Tree:



Total weight = 21

Vertex List after adding this node (weight, from, to):

6, 2, 4

7, 1, 6

7, 2, 5

7, 4, 3

8, 1, 7

8, 2, 6

9, 1, 8

9, 2, 7

9, 4, 5

10, 1, 3

10, 1, 9

10, 2, 8

10, 4, 6

11, 1, 10

11, 2, 9

11, 4, 7

12, 2, 10

12, 4, 8

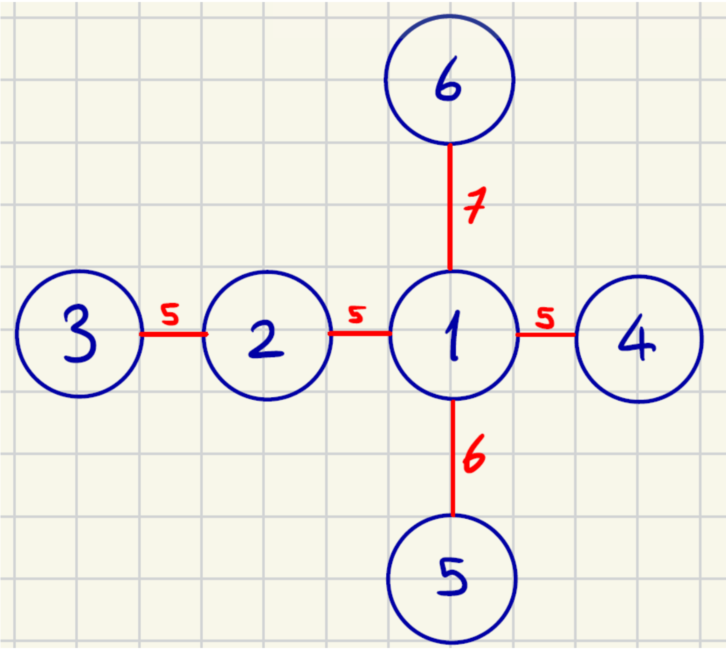
13, 4, 9

14, 4, 10

Take the minimum: 4 is in tree so 6, 2, 4 is useless. Take 7, 1, 6

6-

Minimum Spanning Tree:



Total weight = 28

Vertex List after adding this node (weight, from, to):

7, 2, 5

7, 4, 3

8, 1, 7

8, 2, 6

9, 1, 8

9, 2, 7

9, 4, 5

10, 1, 3

10, 1, 9

10, 2, 8

10, 4, 6

11, 1, 10

11, 2, 9

11, 4, 7

12, 2, 10

12, 4, 8

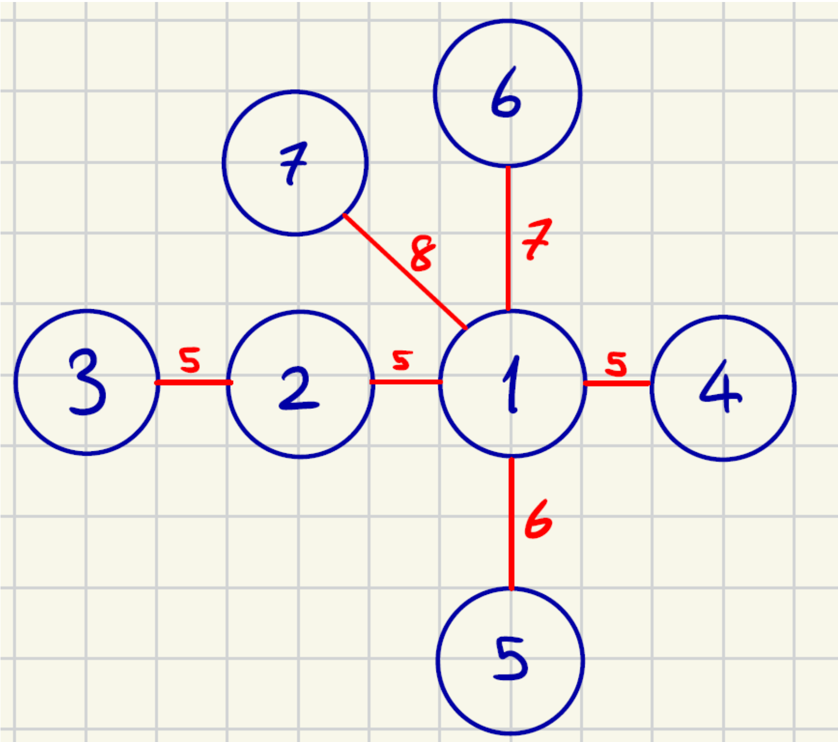
13, 4, 9

14, 4, 10

Take the minimum: 5 and 4 is in tree, pass this edges. Take 8, 1, 7

7-

Minimum Spanning Tree:



Total weight = 36

Vertex List after adding this node (weight, from, to):

8, 2, 6

9, 1, 8

9, 2, 7

9, 4, 5

10, 1, 3

10, 1, 9

10, 2, 8

10, 4, 6

11, 1, 10

11, 2, 9

11, 4, 7

12, 2, 10

12, 4, 8

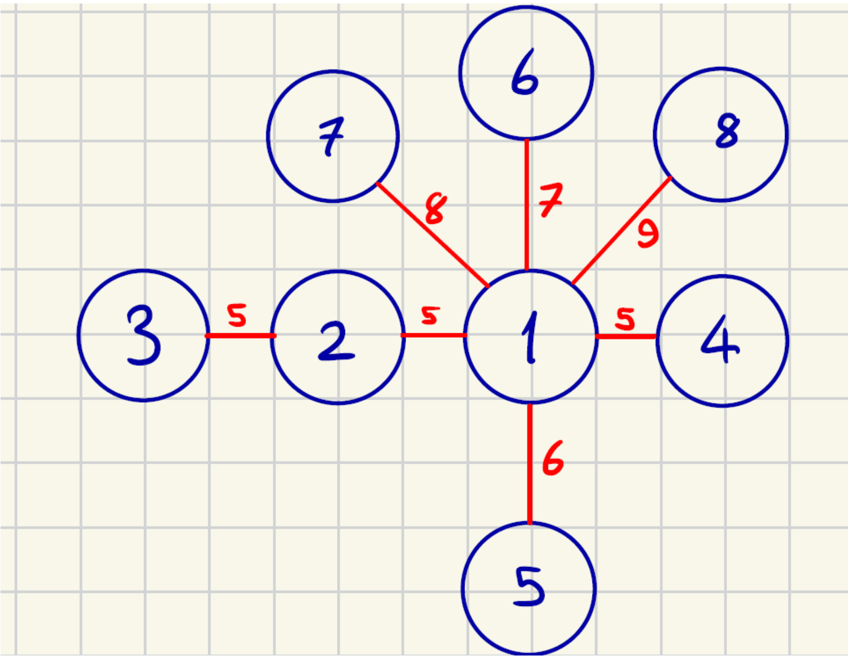
13, 4, 9

14, 4, 10

Take the minimum: 5 and 6 and 3 in tree, pass this edges. Take 9, 1, 8.

8-

Minimum Spanning Tree:



Total weight = 45

Vertex List after adding this node (weight, from, to):

9, 2, 7

9, 4, 5

10, 1, 3

10, 1, 9

10, 2, 8

10, 4, 6

11, 1, 10

11, 2, 9

11, 4, 7

12, 2, 10

12, 4, 8

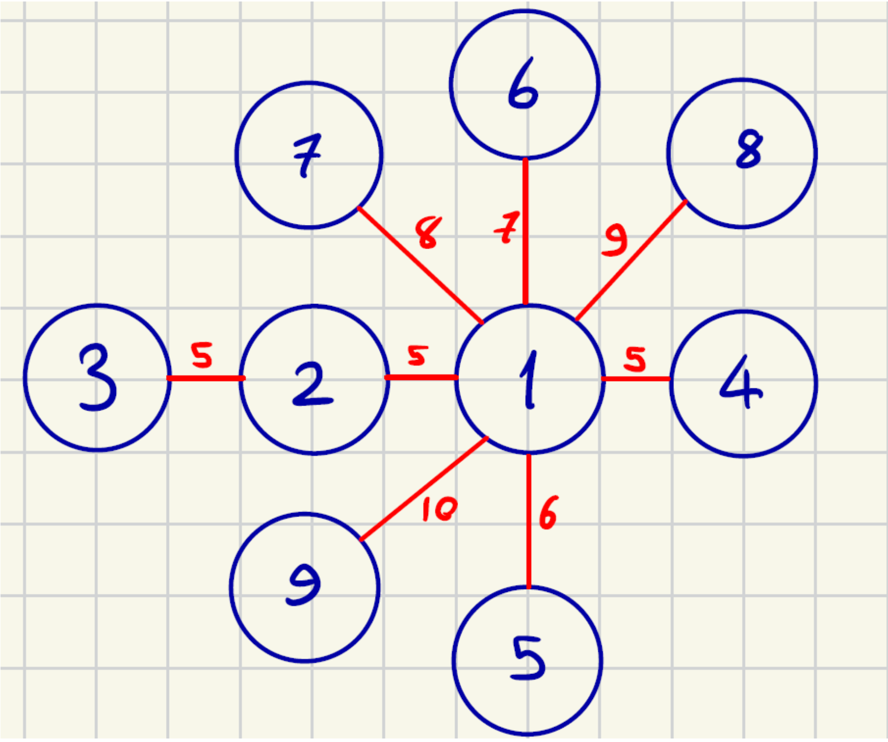
13, 4, 9

14, 4, 10

Take the minimum: 7 and 6 is in tree. Take 10, 1, 9.

9-

Minimum Spanning Tree:



Total weight = 55

Vertex List after adding this node (weight, from, to):

10, 2, 8

10, 4, 6

11, 1, 10

11, 2, 9

11, 4, 7

12, 2, 10

12, 4, 8

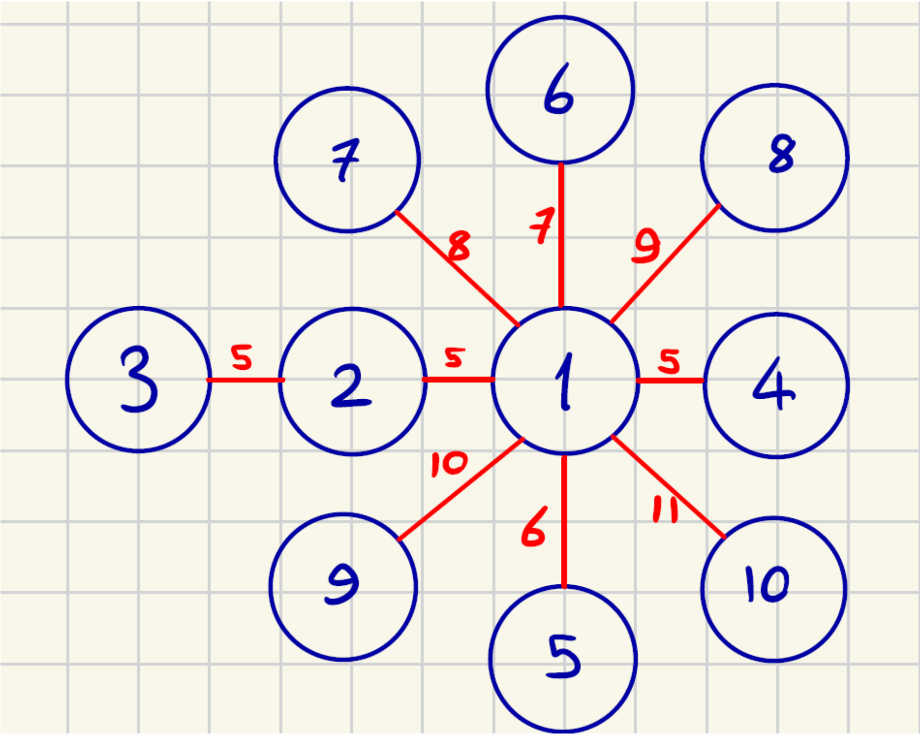
13, 4, 9

14, 4, 10

Take the minimum: 8 and 7 is in tree. Take 11, 1, 10.

10-

Minimum Spanning Tree:



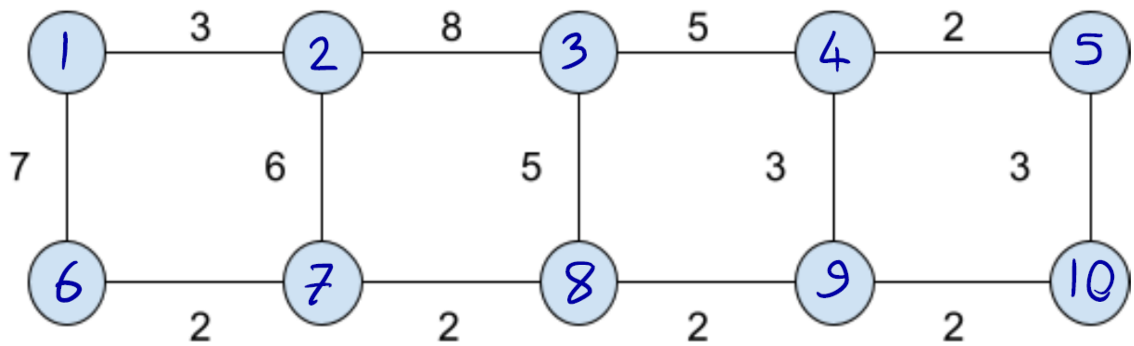
Total weight = 66

Tree is completed, no need to further go on in heap.

Question 2:

In this question we are asked for the minimum cable cost in the network. If we think of this Local Area Network as a graph, what is wanted from us is that the sum of the edges is minimum, when the network is complete. We can think of it as a minimum spanning tree question. We can use two algorithms, Prim's or Kruskal. Since this is a sparse graph, I prefer to use Kruskal.

To use Kruskal I will use an edge list, in a min heap, and here how i numbered the graph:



Edge list in min heap represented as {weight, u, v}:

2, 4, 5

2, 6, 7

2, 7, 8

2, 8, 9

2, 9, 10

3, 1, 2

3, 4, 9

3, 5, 10

5, 3, 8

5, 3, 4

6, 2, 7

7, 1, 6

8, 2, 3

Forest:

1, 2, 3, 4, 5, 6, 7, 8, 9, 10

Cost = 0

Iteration 1:

Pop min: 2, 4, 5

They are seperate, merge them:

Forest after: 1, 2, 3, [4 – 5], 6, 7, 8, 9, 10

Cost = 2

Heap after:

2, 6, 7

2, 7, 8

2, 8, 9

2, 9, 10

3, 1, 2

3, 4, 9

3, 5, 10

5, 3, 8

5, 3, 4

6, 2, 7

7, 1, 6

8, 2, 3

Iteration 2:

Pop min: 2, 6, 7

They are seperate, merge them:

Forest after: 1, 2, 3, [4 – 5], [6 – 7], 8, 9, 10

Cost = 4

Heap after:

2, 7, 8

2, 8, 9

2, 9, 10

3, 1, 2

3, 4, 9

3, 5, 10

5, 3, 8

5, 3, 4

6, 2, 7

7, 1, 6

8, 2, 3

Iteration 3:

Pop min: 2, 7, 8

They are seperate, merge them:

Forest after: 1, 2, 3, [4 – 5], [6 – 7 – 8], 9, 10

Cost = 6

Heap after:

2, 8, 9

2, 9, 10

3, 1, 2

3, 4, 9

3, 5, 10

5, 3, 8

5, 3, 4

6, 2, 7

7, 1, 6

8, 2, 3

Iteration 4:

Pop min: 2, 8, 9

They are seperate, merge them:

Forest after: 1, 2, 3, [4 – 5], [6 – 7 – 8 – 9], 10

Cost = 8

Heap after:

2, 9, 10

3, 1, 2

3, 4, 9

3, 5, 10

5, 3, 8

5, 3, 4

6, 2, 7

7, 1, 6

8, 2, 3

Iteration 5:

Pop min: 2, 9, 10

They are seperate, merge them:

Forest after: 1, 2, 3, [4 – 5], [6 – 7 – 8 – 9 -10]

Cost = 10

Heap after:

3, 1, 2

3, 4, 9

3, 5, 10

5, 3, 8

5, 3, 4

6, 2, 7

7, 1, 6

8, 2, 3

Iteration 6:

Pop min: 3, 1, 2

They are seperate, merge them:

Forest after: [1 -2] , 3, [4 – 5], [6 – 7 – 8 – 9 -10]

Cost = 13

Heap after:

3, 4, 9

3, 5, 10

5, 3, 8

5, 3, 4

6, 2, 7

7, 1, 6

8, 2, 3

Iteration 7:

Pop min: 3, 4, 9

They are seperate, merge them:

Forest after: [1 -2], 3, [4 – 5 - 6 – 7 – 8 – 9 -10]

Cost = 16

Heap after:

3, 5, 10

5, 3, 8

5, 3, 4

6, 2, 7

7, 1, 6

8, 2, 3

Iteration 8:

Pop min: 3, 5, 10

They are in same tree. Don’t add and move on.

Forest after: [1 -2] , 3, [4 – 5 - 6 – 7 – 8 – 9 -10]

Cost = 16

Heap after:

5, 3, 8

5, 3, 4

6, 2, 7

7, 1, 6

8, 2, 3

Iteration 9:

Pop min: 5, 3, 8

They are seperate, merge them:

Forest after: [1 -2] , [3 - 4 – 5 - 6 – 7 – 8 – 9 -10]

Cost = 21

Heap after:

5, 3, 4

6, 2, 7

7, 1, 6

8, 2, 3

Iteration 10:

Pop min: 5, 3, 4

They are in same tree. Don’t add and move on.

Forest after: [1 -2] , [3 - 4 – 5 - 6 – 7 – 8 – 9 -10]

Cost = 21

Heap after:

6, 2, 7

7, 1, 6

8, 2, 3

Iteration 11:

Pop min: 6, 2, 7

They are seperate, merge them:

Cost = 27

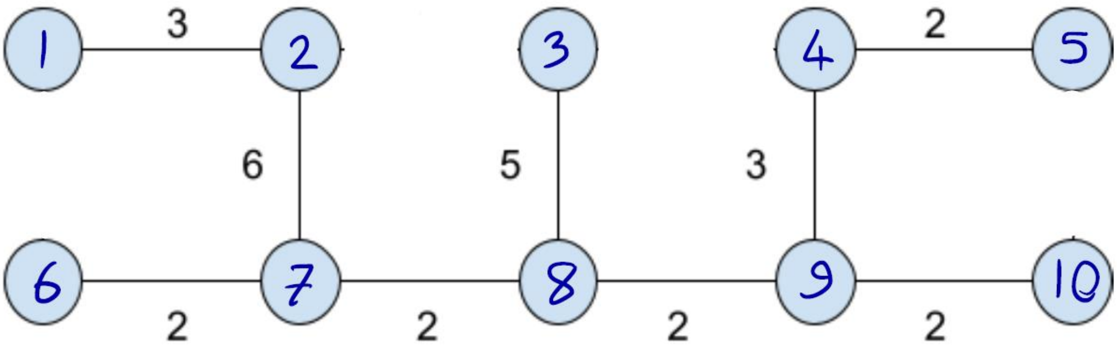
Forest after: [1 -2 - 3 - 4 – 5 - 6 – 7 – 8 – 9 -10]

Heap after:

7, 1, 6

8, 2, 3

Here in the final situation, the total cost of the wiring is 27. When we mark the lines we used here the final graph:



$$2 \cdot 5 + 3 \cdot 2 + 5 + 6 = 27$$