Problem-Sheet 3
My Solution

Michael Single

08-917-445

msingle@students.unibe.ch

Task 1:

1.1: Answers to the questions:

A cable network connecting a certain number of houses by an antenna – directly or indirectly – is modeled as an undirected connected graph. Each vertex represents either a house the antenna (only one vertex represents the antenna, later labeled by one – see figure 1.1 below). An edge between two vertices represents a cable connection between two houses or between a house and the antenna. The graph is undirected since the metric distance measure is symmetric (i.e. The distance from a vertex a to a vertex b is equal to the distance from vertex b to vertex a – assuming these vertices are connected with each other).

We are interested in the minimal total length connecting all houses to the antenna. This corresponds to the minimal spanning tree in this graph. Connect all vertices by a minimal amount of edges (just that many that they are connected but not more) such that the length is minimal.

One possibility would be to enumerate all possible subgraphs which are a spanning tree. From these we whould have to take the one which has the lowest cost (here length).

Unforutnaltely, the number of subgraphs which are a spanning tree is enormously large. For example if we are considering the complete graph K_n with n vertices, then there are n^(n-2) subgraphs in total that are a spanning tree. However, when dealing with a graph which is a subgraph of the complete graph (usually the case) the number of such spanning tree subgraphs is smaller – but still extremely large.

A recursive formula for computing the exact number of such subgraphs in a Graph of n vertices can be found in the paper „Counting Spanning Trees“ by Bang Ye Wu and Kun-Mao Chao.

Algorithms which give us the minimal spanning tree from a given graph are either **Prim's Algorithm** or **Kruskal's Algorithm.**

1.2: Solve the Problem:

Vertex with label 1 correspinds to the antenna, vertices with label 2 to 11 correspond to the houses. An edge between two vertices (i.e. either between two houses or between a house and the antenna) represents a cable connection between these two. Distances between two vertices (i.e. The edge weight) are indicated by a number on drawn on top of an edge.

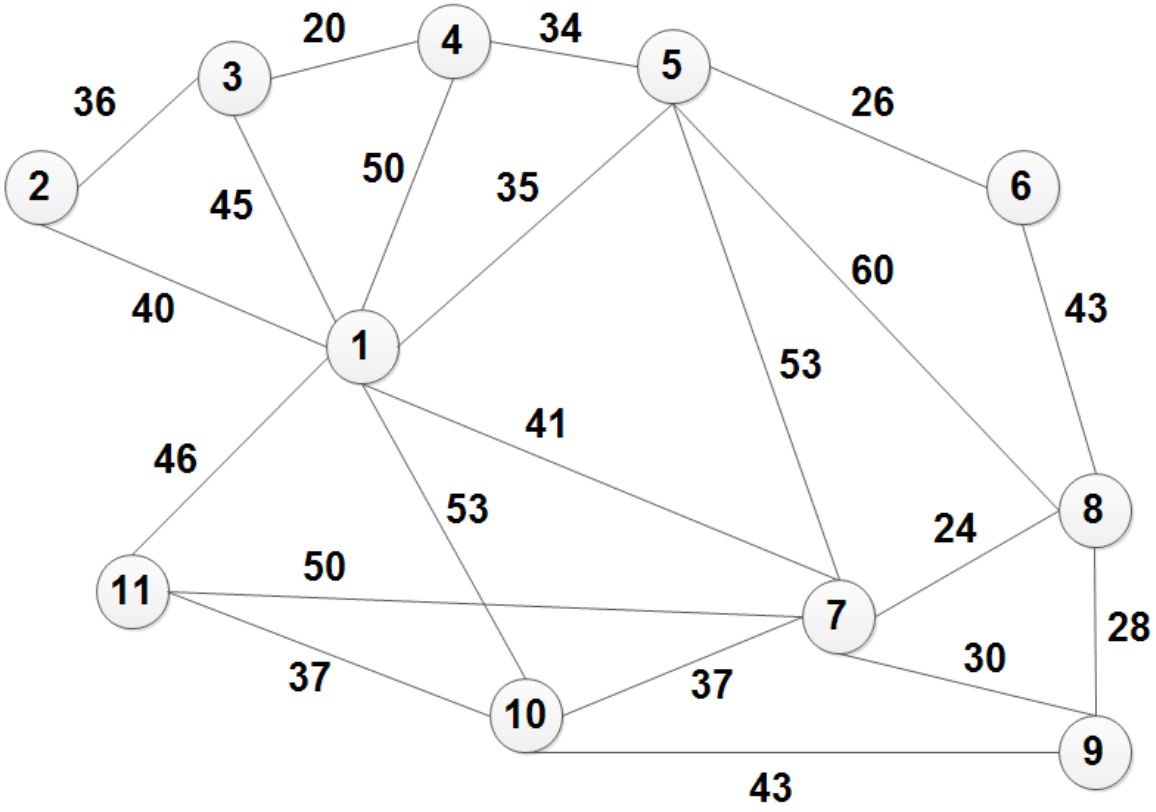

Figure 1.1: Cable network between houses and antenna modeled as a undirected Grap

Strategy: Greedely take edge between two vertices with mininum weight.

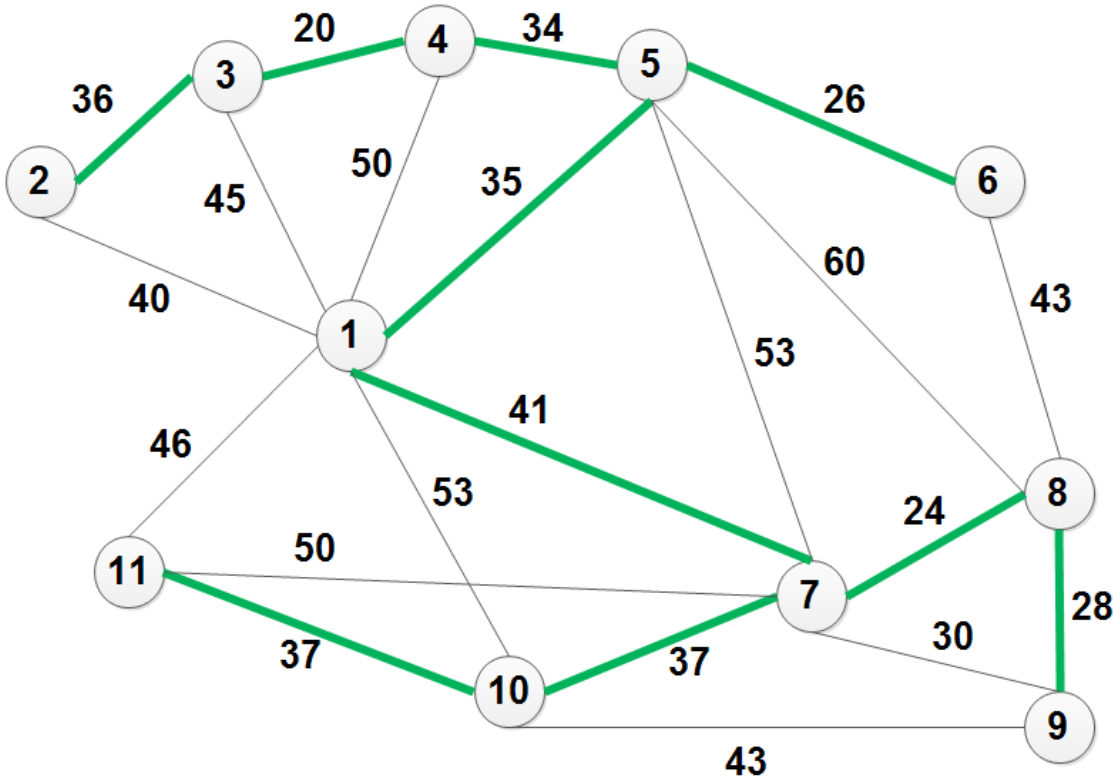

Figure 1.2: Minimal Spanning Tree colored in green inside the given initial graph from figure 1.1.

Minimal Distance

= 36+ 20 + 34 + 26 + 35 + 41 + 24 + 28 + 37 + 37

= 318

thus the minimal total length is equal to 318 meters.

Task 2:

2.1.1: Describe in words how the algorithm works:

Prim's Algorithms is a greedy algorithm which computes the Minimal Spanning Tree for a given undirected, conected Graph G = (V,E)

Note that each edge in such a Graph has a certain weight assigned.

Initially, this version of Prim's algorithm start by introducing a helper set called S.

T contains the solution to the MST (acutally it contains the iterative solution /

a temporar solution which's state is depending on the number of successfully performed most outer

iterations).

The set S consists of elements that are sets containing the vertices of V.

S basically depicts the current solution at a certain iteration (of the most other loop - while loop) of Prim's Algorithm.

More precicely after each most outer iteration, S is closer to the final set of edges of the MST (for G).

Therefore we notice that this algorithm iteratively builds up the MST of G by adding edges to the current solution (greedy approach).

Initially, we Start with S which is then a forest, containing all the vertices of G having no conntections.

Hence, S initially the biggest possible forest which implies that |S| = |V|.

For each element in S (i.e. the compononets of the current solution graph) has two quantities associtated with:

+ minC which is a minimal cost currently found for this component.

+ minE the edge which has the minC as its weight.

Again, please notice that each element in S has two separate of such quantities assigned to (however they do not have to be destinct).

In every Iteation over the elements of S we set their minC value equal to infinity.

Next we iterate over each edge in our given initial graph G.

For the current edge in iteration we consider its vertices v and w

(remember an edge is made by a connection between two vertices).

We then perform a query which tells us from which component in S the vertex v (and w respectively) came from.

We label these compoents by K_v (K_w repsecitvely).

If the vertices v and w came from distinct components (i.e. the vertices v and w belong to separate components which currently are not connected),

then we perform an update to the quantities of the components K_v, K_w (remember these quantities are minC and minE).

the update wokrs as the following: if the current minimal cost of a component is bigger than the weight of the current edge in iteration,

then update the weight minC of the considered component by the weight of the edge.

Furthermore, set its minimal edge equal to the current edge in iteration.

Roughly speaking the algorithm does the following:

For each component in S we look for the edge with the lowest cost in order to connect two unconnected components.

Note that instead of directly iterating over the components, we iterate over the set of edges and perform certain queries in order to find the related components from S.

For any pair of unconnected components in S for which there is at least one connecting edge

(between the those components) we find their minimal connecting edge. (the most inner if statements).

After iterating over all edges, we collect all minimal edges from the components in S and merge them with the solution T of the current MST.

Then we determine a new Set of S which is equal to the set of unconnected components after having perfromed the merging with T.

We repeat the described procedure until |S| is equal to 1.

2.1.2: Apply it to the graph of Exercise

Applying Prime's Algorithm to the graph from figure 1.1.

Ater Initialization:

S = {{1}, {2}, ..., {11}}; T ={}

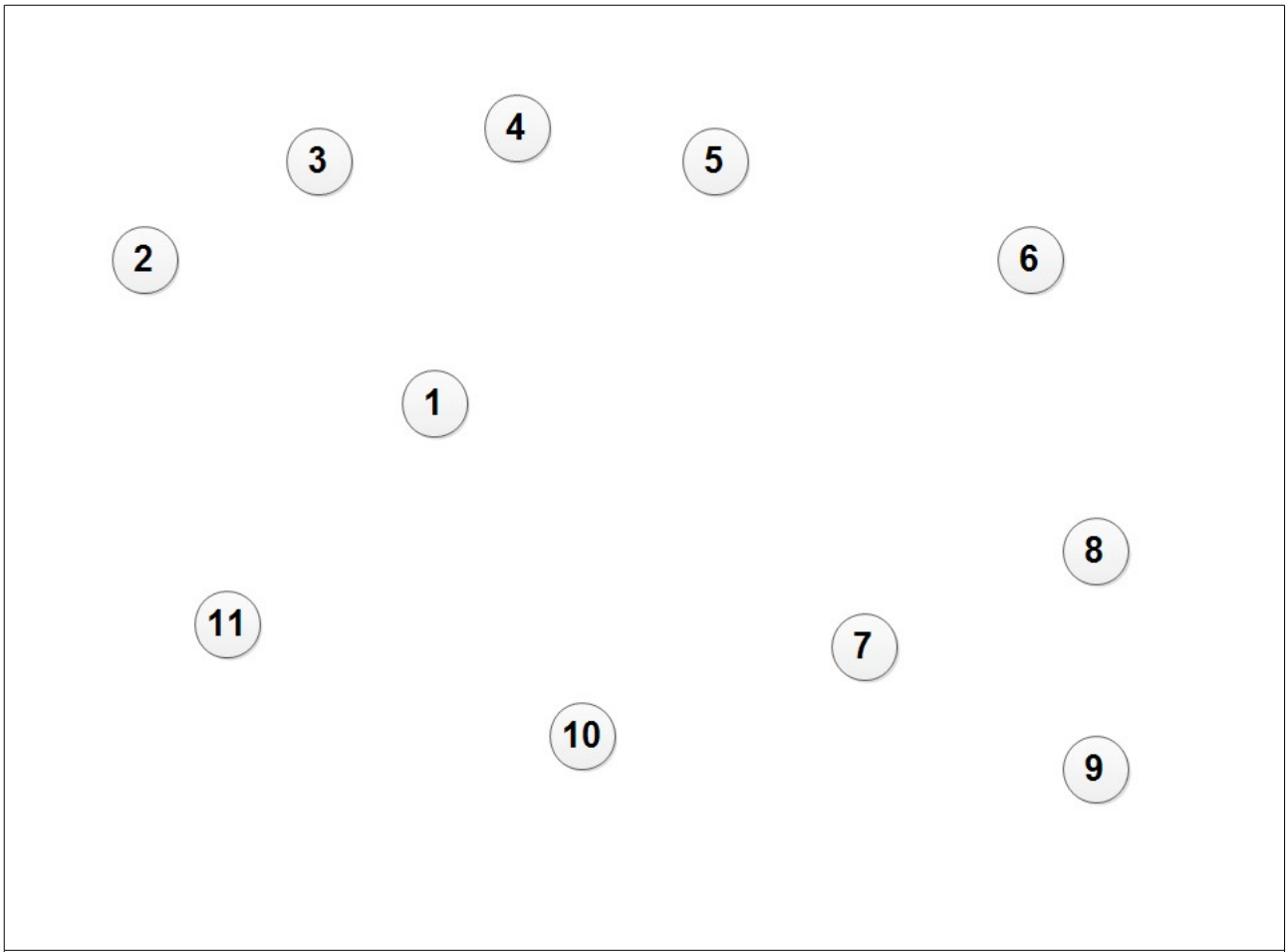

Figure 2.1: Initial value of S – the biggest possible forest .

After 1. While-Iteration:

New determined set S: {{(2,3),(2,4)}, {(1,5), (5,6)}, {(7,8), (7,10), (8,9), (10,11)}}

therefore S = {{(2,3),(2,4)}, {(1,5), (5,6)}, {(7,8), (7,10), (8,9), (10,11)}}

= {K1, K2, K3}

T = S

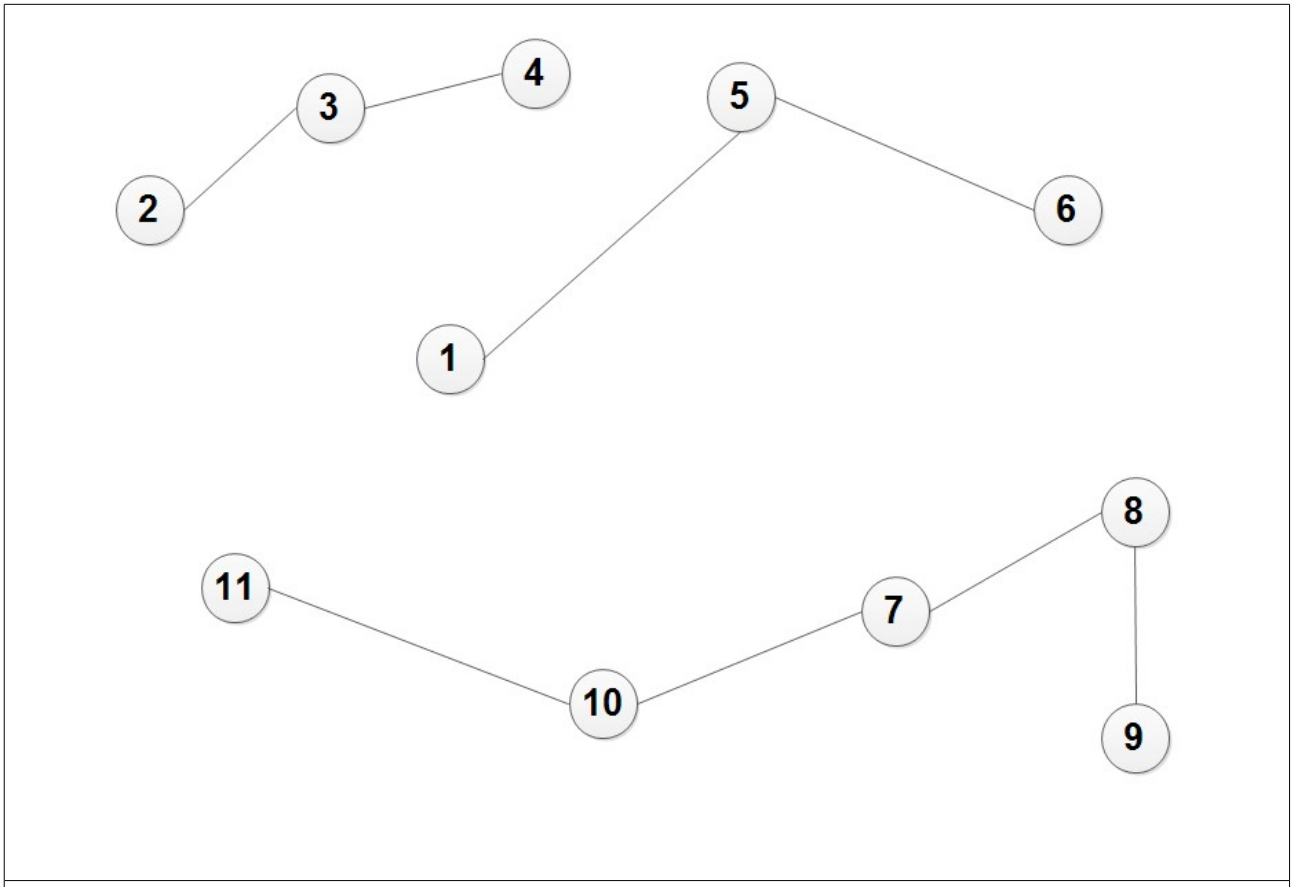

Figure 2.2: After 1. Iteration

minCK1	minCK2	minCK3	minCK4	minCK5	minCK6	minCK7	minCK8	minCK9	minCK10	minCK11
35	36	36	20	26	26	24	24	28	37	37
minEK1	minEK2	minEK3	minEK4	minEK5	minEK6	minEK7	minEK8	minEK9	minEK10	minEK11
(1,5)	(2,3)	(3,4)	(3,4)	(5,6)	(5,6)	(7,8)	(7,8)	(8,9)	(7,10)	(10,11)

After 2. While-Iteration:

minCK1	minCK2	minCK3
34	34	41
minEK1	minEK2	minEK3
(4,5)	(4,5)	(5,7)

New determined set S: {{(1,5),(2,3),(2,4),(4,5),(5,6)}, {(7,8), (7,10), (8,9), (10,11)}}

therefore S = {{(1,5),(2,3),(2,4),(4,5),(5,6)}, {(7,8), (7,10), (8,9), (10,11)}}

= {K1, K2}

T = S

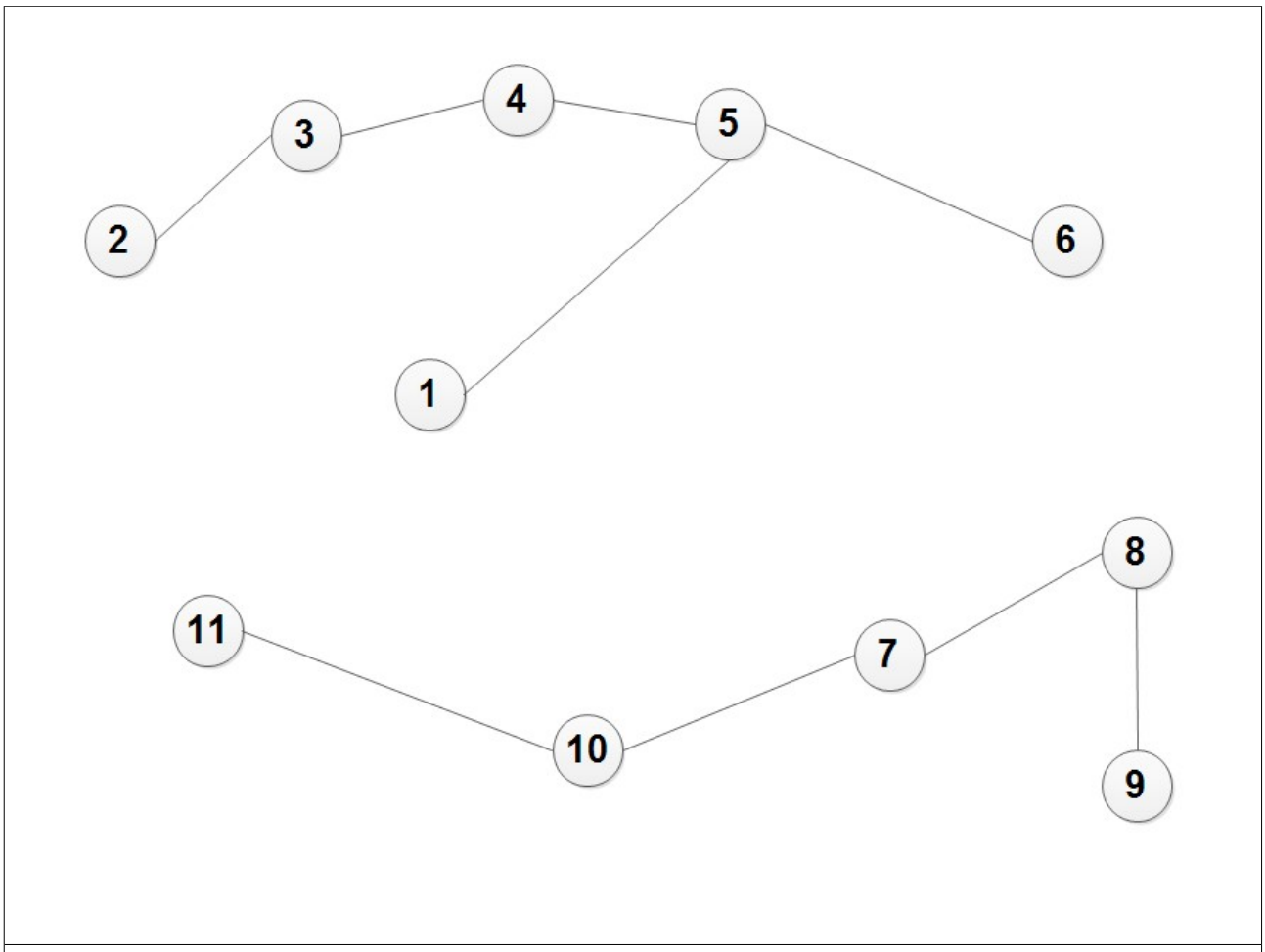

Figure 2.3: After 2. Iteration

After 3. While-Iteration:

minCK1	minCK2
41	41
minEK1	minEK2
(1,7)	(1,7)

New determined set S: {{(1,5),(2,3),(2,4),(4,5),(5,6),(1,7),(7,8), (7,10), (8,9), (10,11)}}

therefore S = {{(1,5),(2,3),(2,4),(4,5),(5,6),(1,7),(7,8), (7,10), (8,9), (10,11)}}

= {K1}

T = S

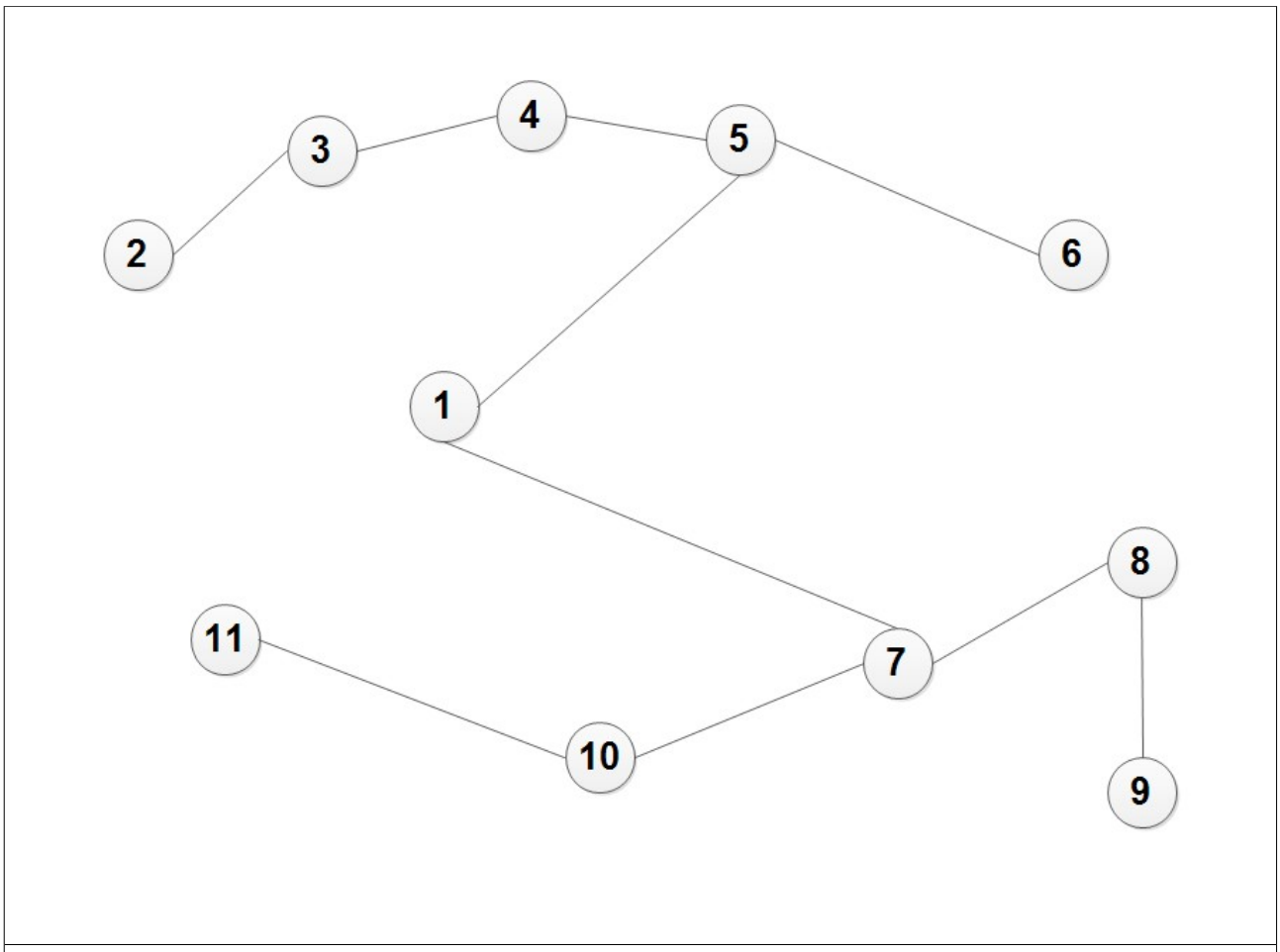

Figure 2.4: After 3. Iteration

Since now |S| is equalt to one, we will end the while loop and have found our Minimal Spanning Tree denoted by T.

As we can see in figure 2.4 we get the same results as we got in figure 1.2.

2.2. Why is the algorithm valid?

Given a Graph G which is connected we can see that this algorithm will evenetually terminate.

We merge all found edges which connect (at a minimal cost) with T and update the set accordingly. Connecting unconnected components by edges results in a set S that will contain fewer elements (when performing the statement at line 14 in the algorithm (determine set S of the connected component from forest (V,T)). Connecting unconnected components by edges from the graph works as long as there are edges to connect unconnected components among each other. We always will have such connectiong edges until we have no more unconnected components as long as we use an input graph which itself is conntected.

So this shows that the algorithm terminates. Furthermore, as aleardy sated we connect will connect unconnected components until we have no more any unconnected components in S (i.e. |S| is equal to 1, one connected component). This algorithm greedily takes edges to connect unconnected components (by taking the edges with the lowest weights). Since it alwas tries to connect unconnected components – but by the edge with a minimum weight – it will construct a spanning tree containing the edges with the minimum weight, i.e. A MST.

Task 3:

Firt of all, we have to compute the edge weights according to the given task:
in the following an edge represents a path
If a path is **Marking**: 1km costs 500.- (thin black line)
If a path is **Beautification**: 1km costs 1000.- (bold black line)
If a path is **Extension**: 1km costs 1500.- (bold gray line)

This will give us the following graph:

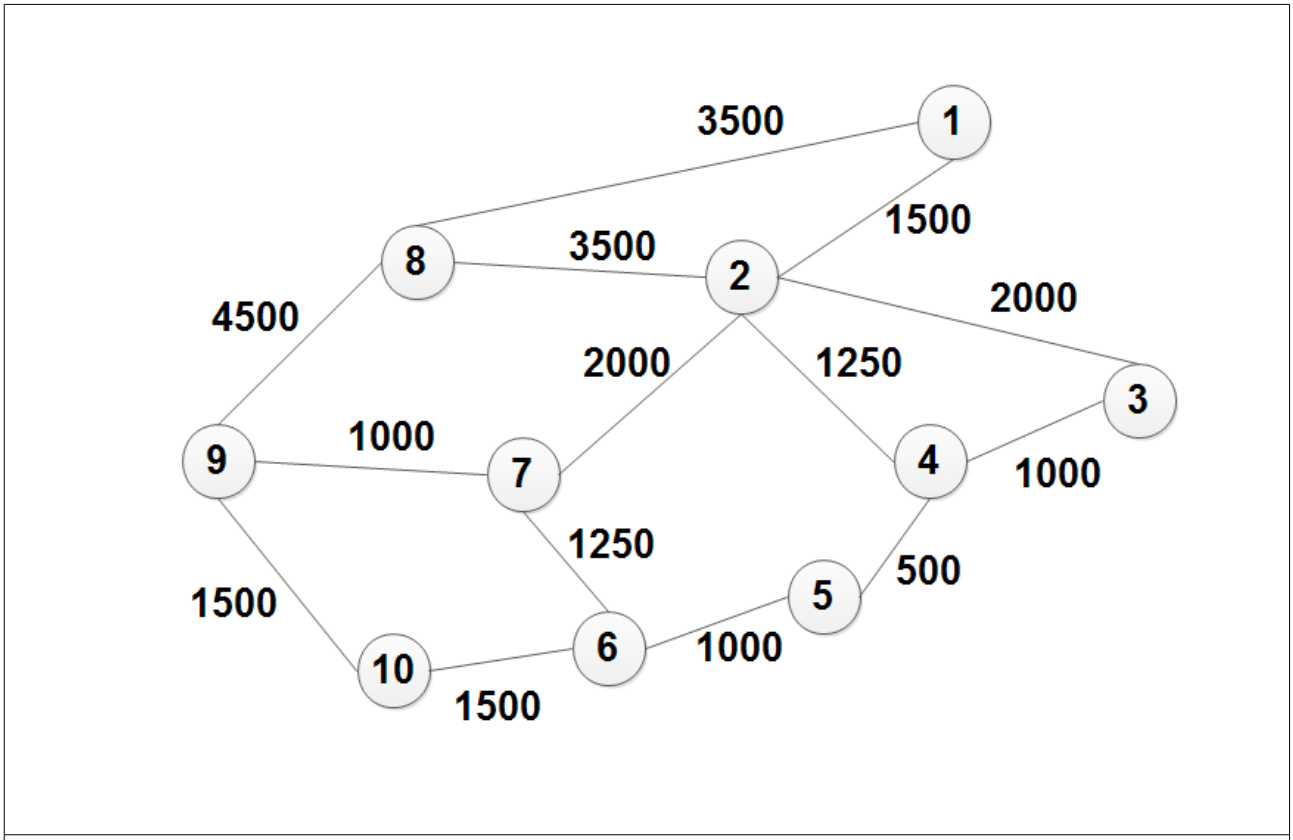

Figure 3.1: Reformulated graph based on given per km costs uwing the color encodings of the map

Note that we performed the following vertex relabeling:

1	2	3	4	5	6	7	8	9	10
Chalais	Vercorin	Pinsec	Crouja	Tracui	Crêt	Refuge	Moulins	La Lé	unnamed

Which gives us the following MST:

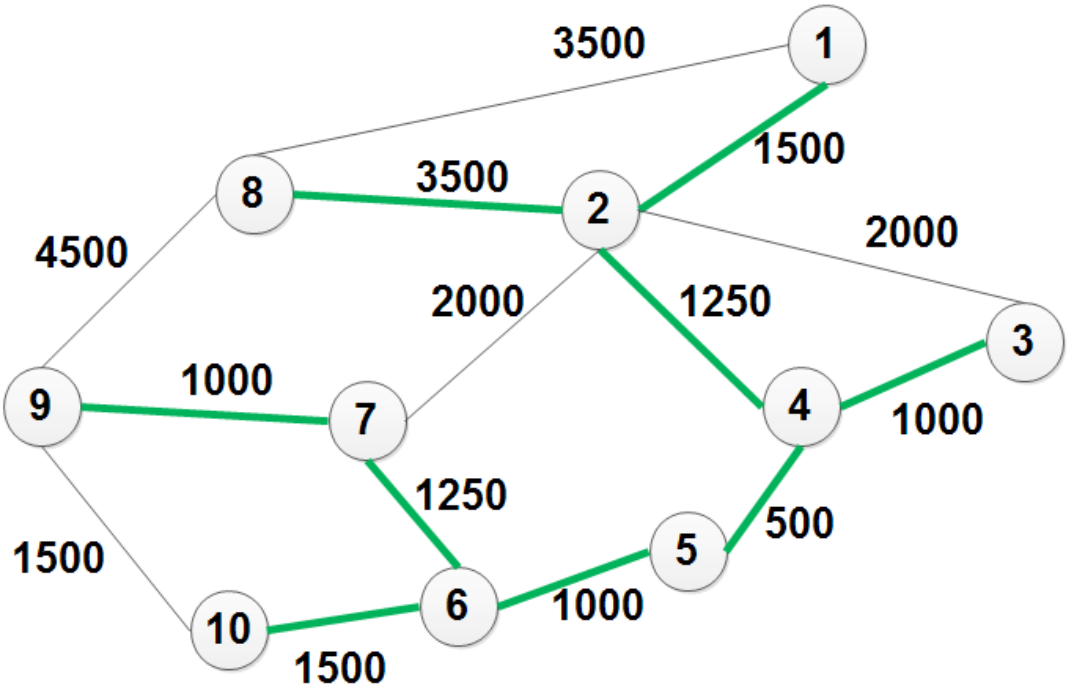

Figure 3.2: one particular MST for the graph from figure 3.1

Therefore one possible solution for figure 3.1 is the MST from figure 3.2.

Note that there are other possibe solutions: E.g. Instead of the edge (6,10) we could have taken the edge (9,10) for our spanning tree. Similarly, instead of taking the edge (2,8) we could have taken the edge from (1,8). This gives us 4 possible MST for this problem. (Assuming I have calculated my weights correct). Thus: Let us assume that I have computed the weight correclty :)