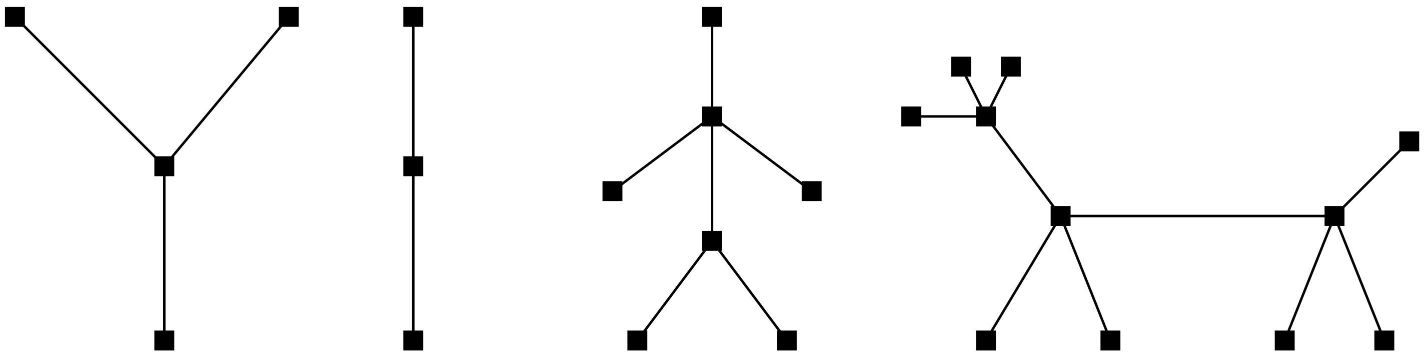
Trees

A *forest* is a graph without cycles.

A connected forest is a *tree*.



A *leaf* is a vertex of degree 1.

Proposition. Any tree is a bipartite graph.

Proof. All cycles (of which there are none) of a tree have even length. A graph is bipartite iff all its cycles have even length (shown in the first lecture). $\square$

Proposition. Let G be a graph with n vertices, m edges and k connected components. Then $n - k \leq m$.

Proof. Induction over m.

If $m = 0$, then $k = n$ because each vertex is a separate connected component. The inequality holds.

Let $m > 0$. Remove an edge from G, giving us a graph with $m - 1$ edges. There are two possibilities:

- The number of connected components remained the same. Induction hypothesis gives us $n - k \leq m - 1$. Then also $n - k \leq m$.
- The number of connected components increased by one. The induction hypothesis gives $n-(k+1) \leq m-1$. Then also $n - k \leq m$. $\square$

Theorem. Let $T = (V, E)$ be a graph with n vertices. Any two of the claims below imply the third one.

(i). T is connected.

(ii). T has no cycles.

(iii). T has $(n-1)$ edges.

This theorem gives us two alternative definitions for trees.

Proof.

(i) & (ii) $\Rightarrow$ (iii). Induction over n.

If T has one vertex then all edges of T are loops. I.e. each edge of T is a cycle. By (ii), T has no cycles, hence no edges.

Let T have n vertices.

T has no cycles $\Longrightarrow$ T contains a vertex v of degree 0 or 1.

Theorem. If all vertices of a graph have degree ≥ 2, then that graph contains a cycle.

T is connected $\Longrightarrow$ degree of v is not 0.

The subgraph $T' \leq T$ induced by $V \backslash \{v\}$ is connected and without cycles, hence it has $n-2$ edges.

T has one more edge than T'.

(ii) & (iii) $\Rightarrow$ (i). Assume that T is not connected.

Let $T_1, \ldots, T_k$ be the connected components of graph T. All of them are connected and without cycles, hence the number of edges in any T_i is one less than the number of vertices in T_i. (by (i) & (ii) $\Rightarrow$ (iii))

Hence T has $n - k$ edges. By (iii), T has $n - 1$ edges, i.e. $k = 1$, i.e. T is connected.

(i) & (iii) $\Rightarrow$ (ii). Assume T has a cycle. Remove an edge from it. The result is a connected graph with n vertices and $n - 2$ edges. Contradiction with the previous proposition. $\square$

Theorem. A graph T is a tree iff it is connected and all its edges are bridges.

Proof. $\Rightarrow$: T has n vertices and $n-1$ edges for some n. Consider an edge. After removing it, a graph with n vertices and $n-2$ edges remains. This graph is not connected. Thus that edge was a bridge.

$\Leftarrow$: T is connected. If it had any cycles, then after removing an edge in a cycle the graph is still connected. I.e. these edges are not bridges. Hence T is without cycles. $\square$

Teoreem. Let T be a graph with n vertices. The following claims are equivalent.

1. T is a tree.
2. Between any two vertices of T there is exactly one path.
3. T has no cycles, but adding an edge between any two vertices creates a cycle.

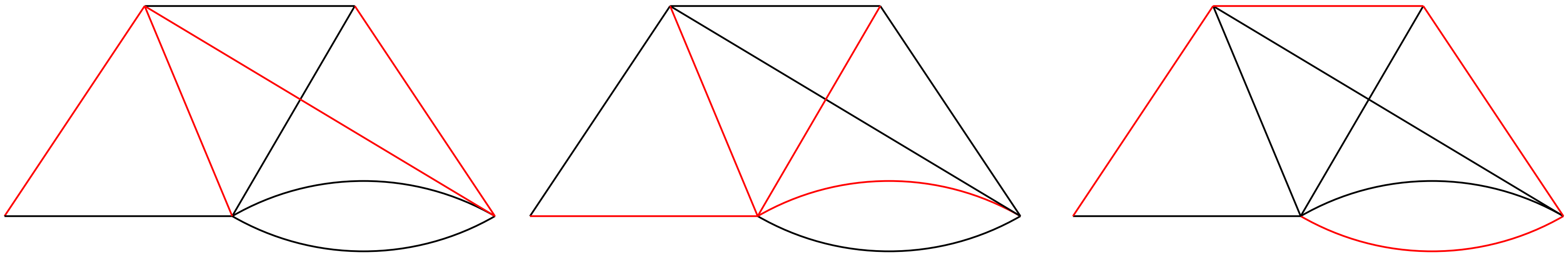
Proof. $1 \Rightarrow 2$. Between any two vertices there is at least one path – otherwise T would not be connected. If there were two different paths between two vertices, we would get a cycle and T would not be a tree.

$2 \Rightarrow 3$. T has no cycles, since otherwise we would get two different paths bewteen any two vertices on the cycle. Adding a new edge e between the vertices u and v, we obtain a cycle $u \rightsquigarrow v \overset{e}{\text{——}} u$.

$3 \Rightarrow 1$. Suppose T is not connected. When adding an edge between the vertices in different connected components we get no cycles, a contradiction with the assumption. $\square$

Spanning tree (aluspuu) of the connected graph $G = (V, E)$ is such a subgraph T, that T is a tree and $V(T) = V$.

A *spanning forest (alusmets)* of a non-connected graph is the union of spanning trees of its connected components.



Let $G = (V, E)$ be a graph with n vertices and let us have a *weight* $w(e)$ defined for each of its edges $e \in E$.

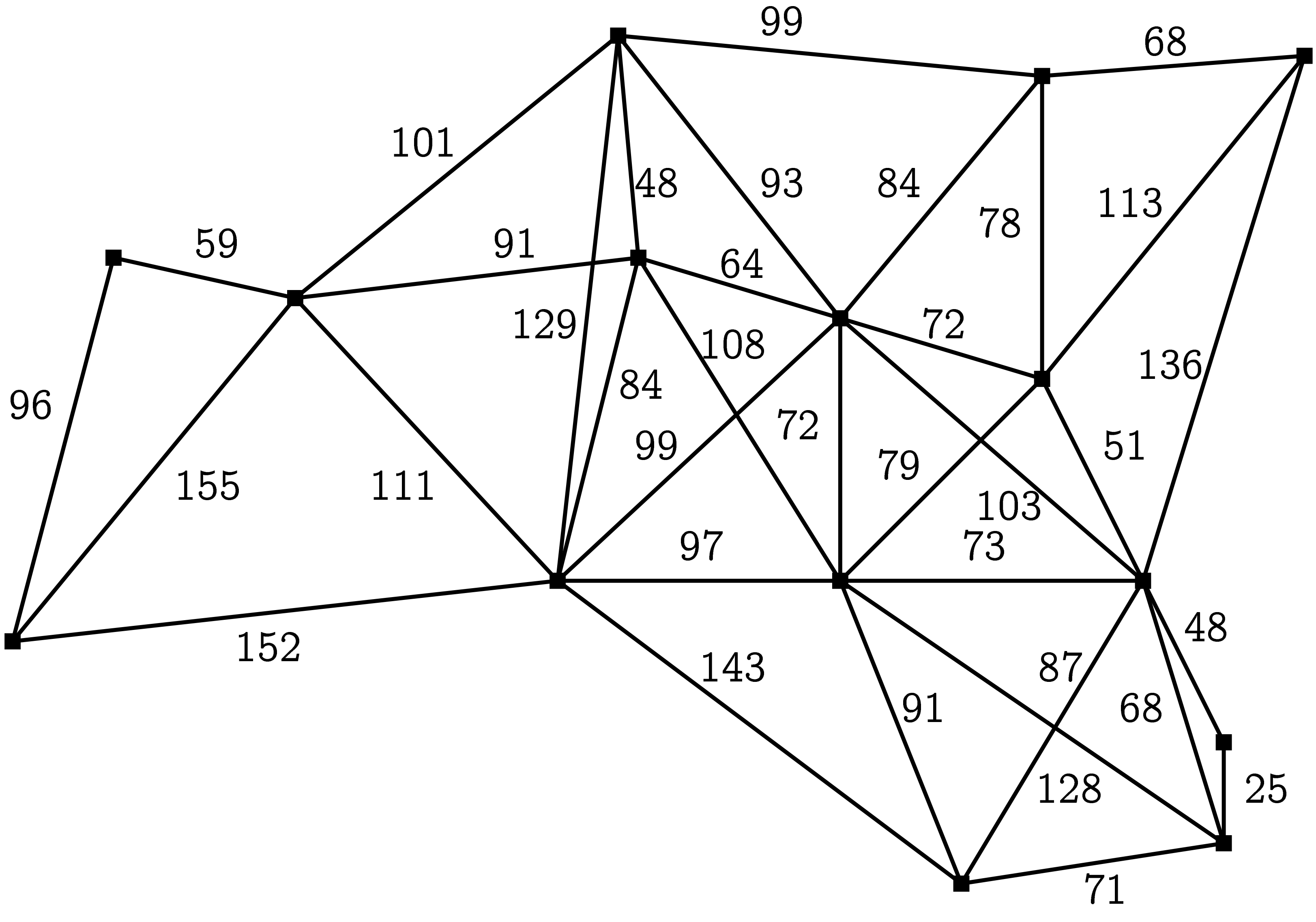
If $G' = (V', E')$ is a subgraph of G, then define $w(G') = \sum_{e \in E'} w(e)$.

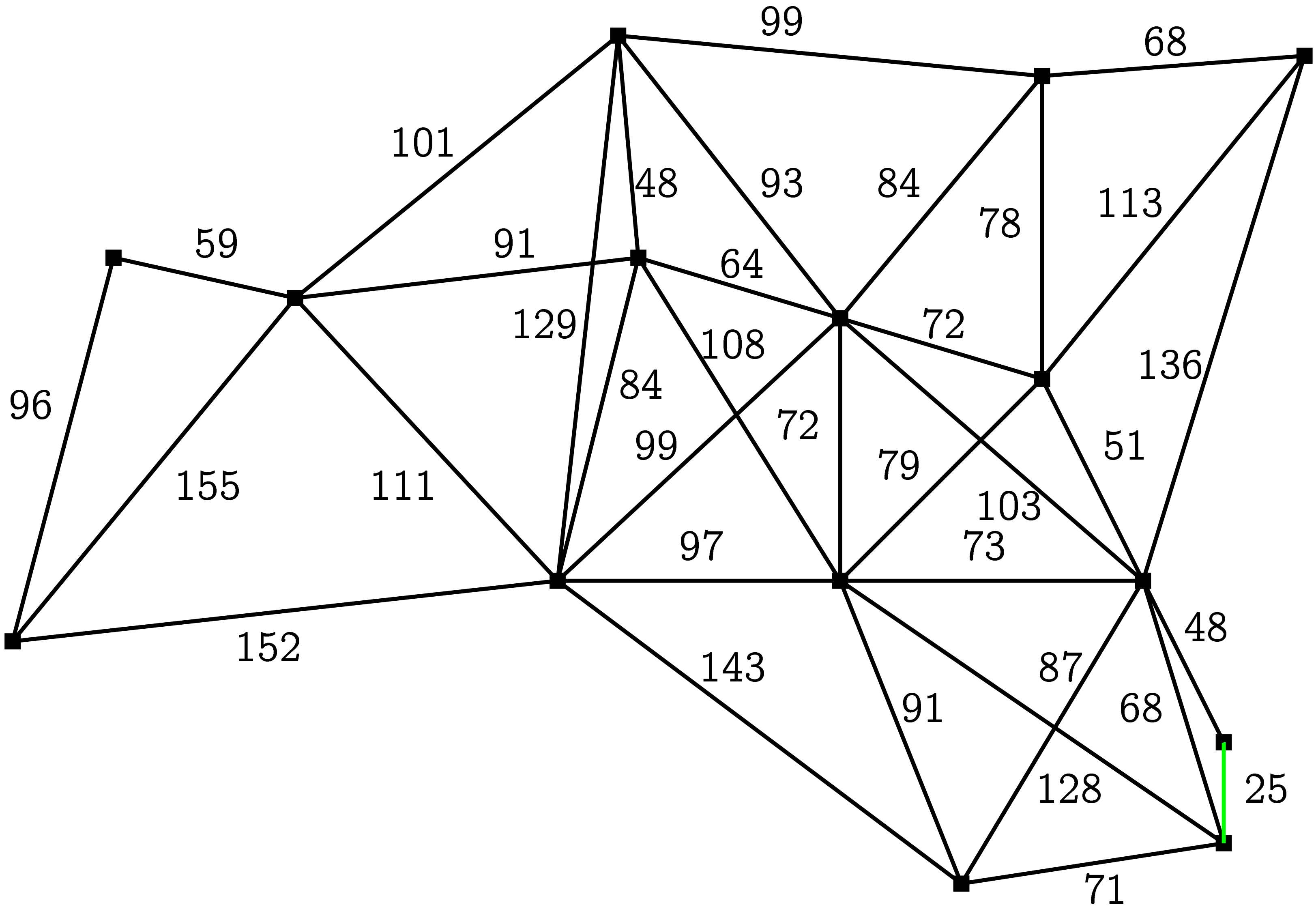
Algorithm (for finding the minimal weight spanning tree of G).

Select the edges $e_1, \ldots, e_{n-1}$ so that

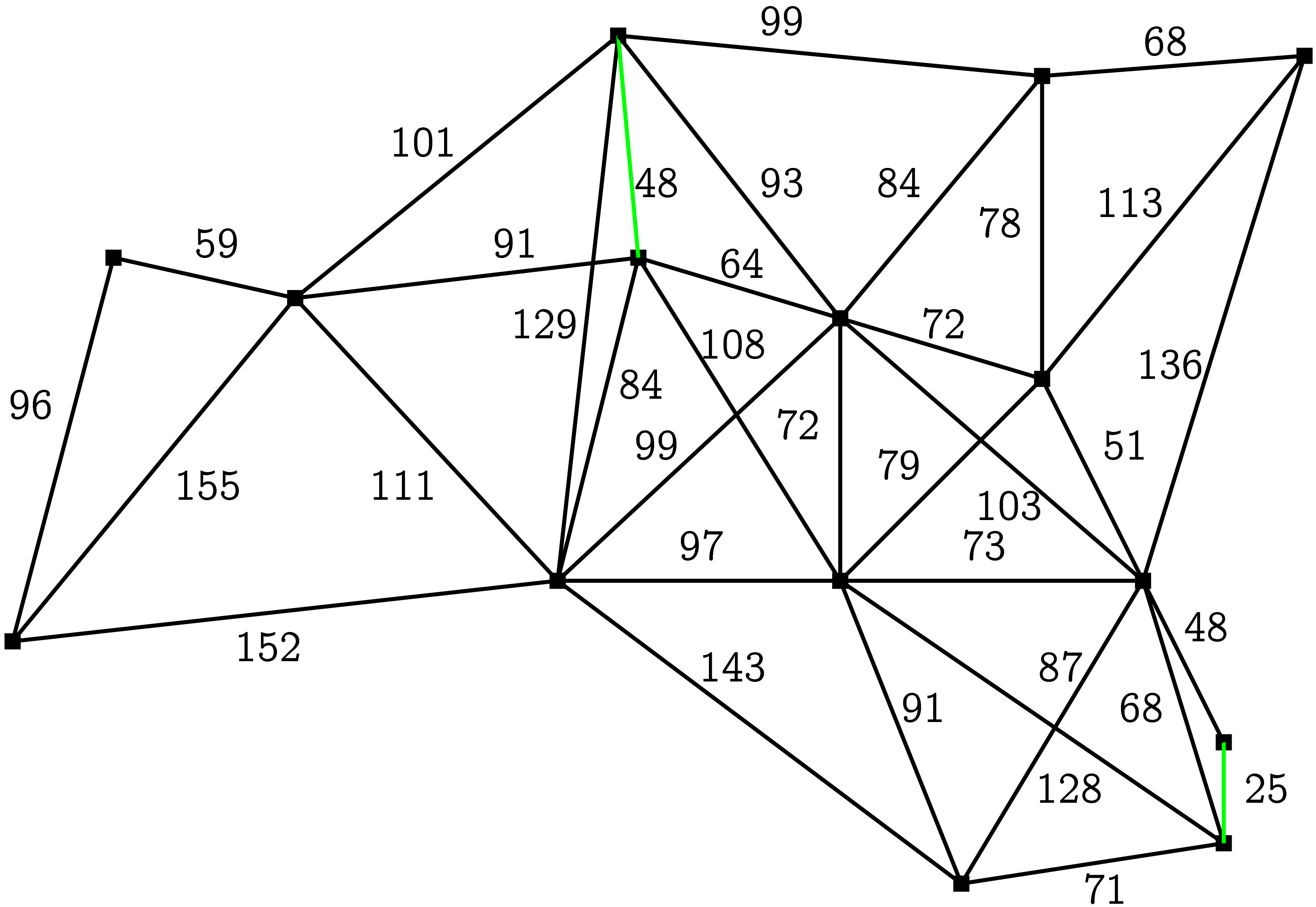
- e_i differs from the edges $e_1, \ldots, e_{i-1}$;
- e_i does not form a cycle together with $e_1, \ldots, e_{i-1}$;
- e_i has the minimal weight among the edges satisfying the two conditions above.

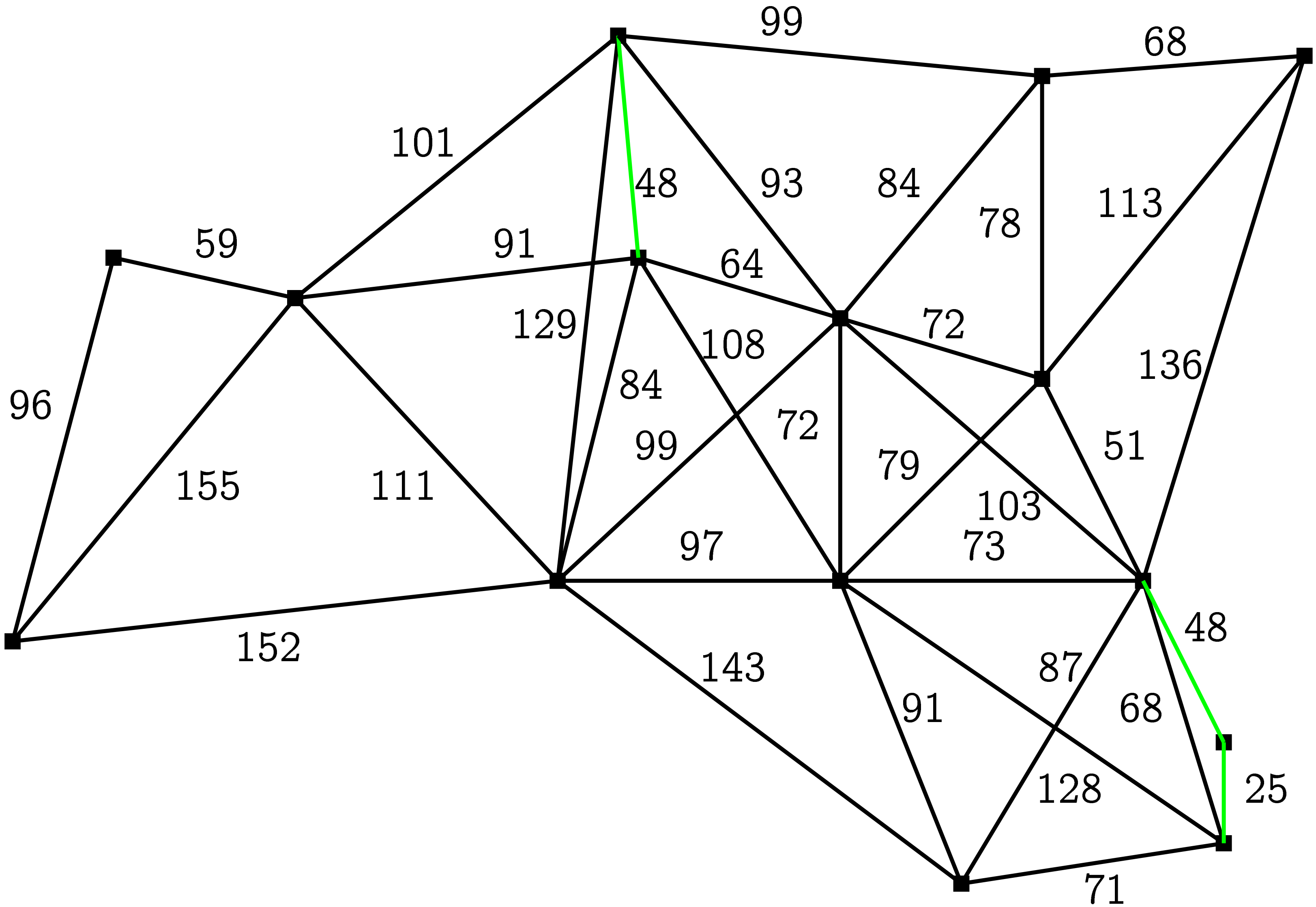
Output $T = (V, \{e_1, \ldots, e_{n-1}\})$.

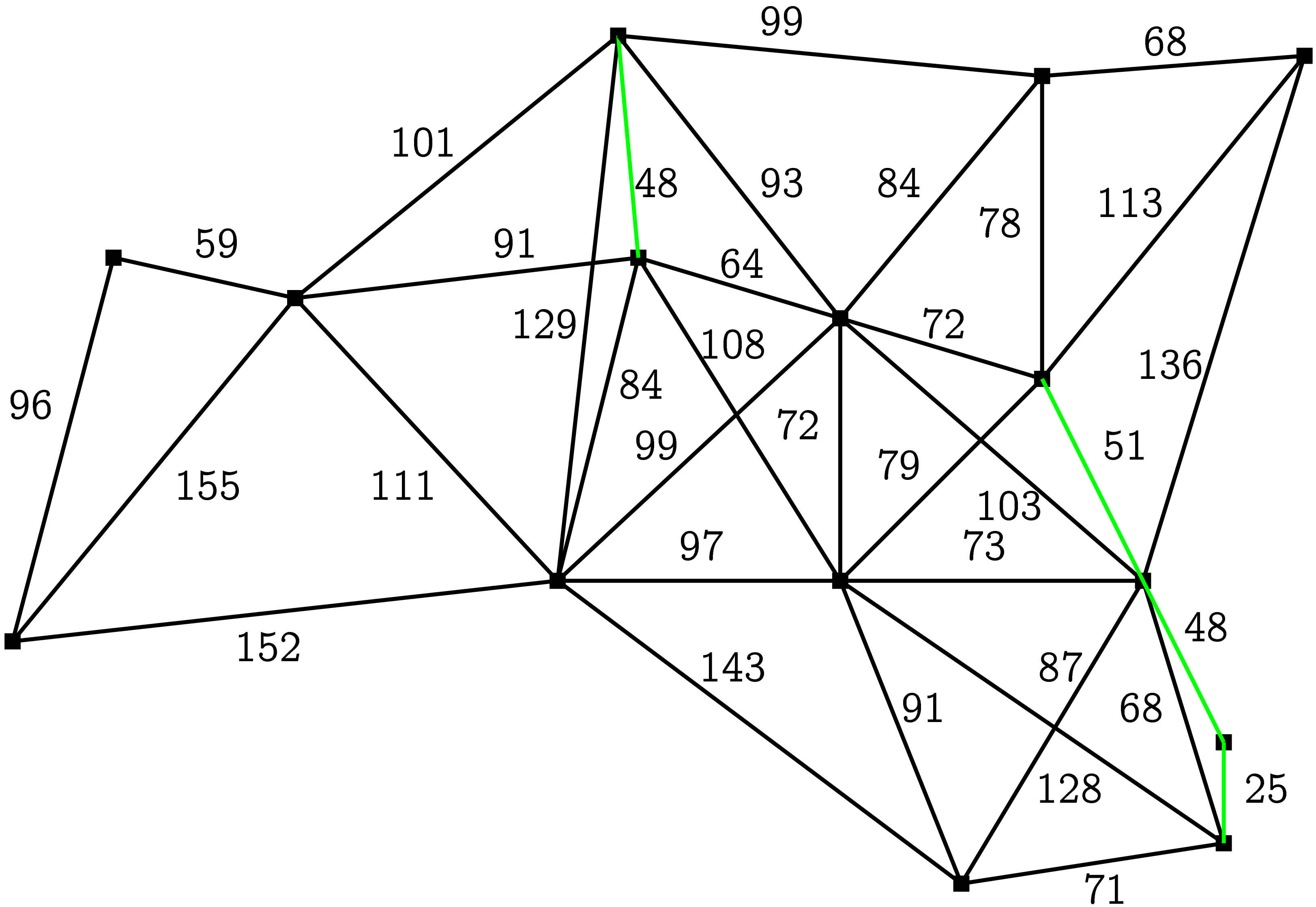

99
68
101
48
93
84
78
113
59
91
64
129
108
72
136
96
84
72
99
79
51
155
111
103
97
73
152
143
87
48
91
68
128
25
71



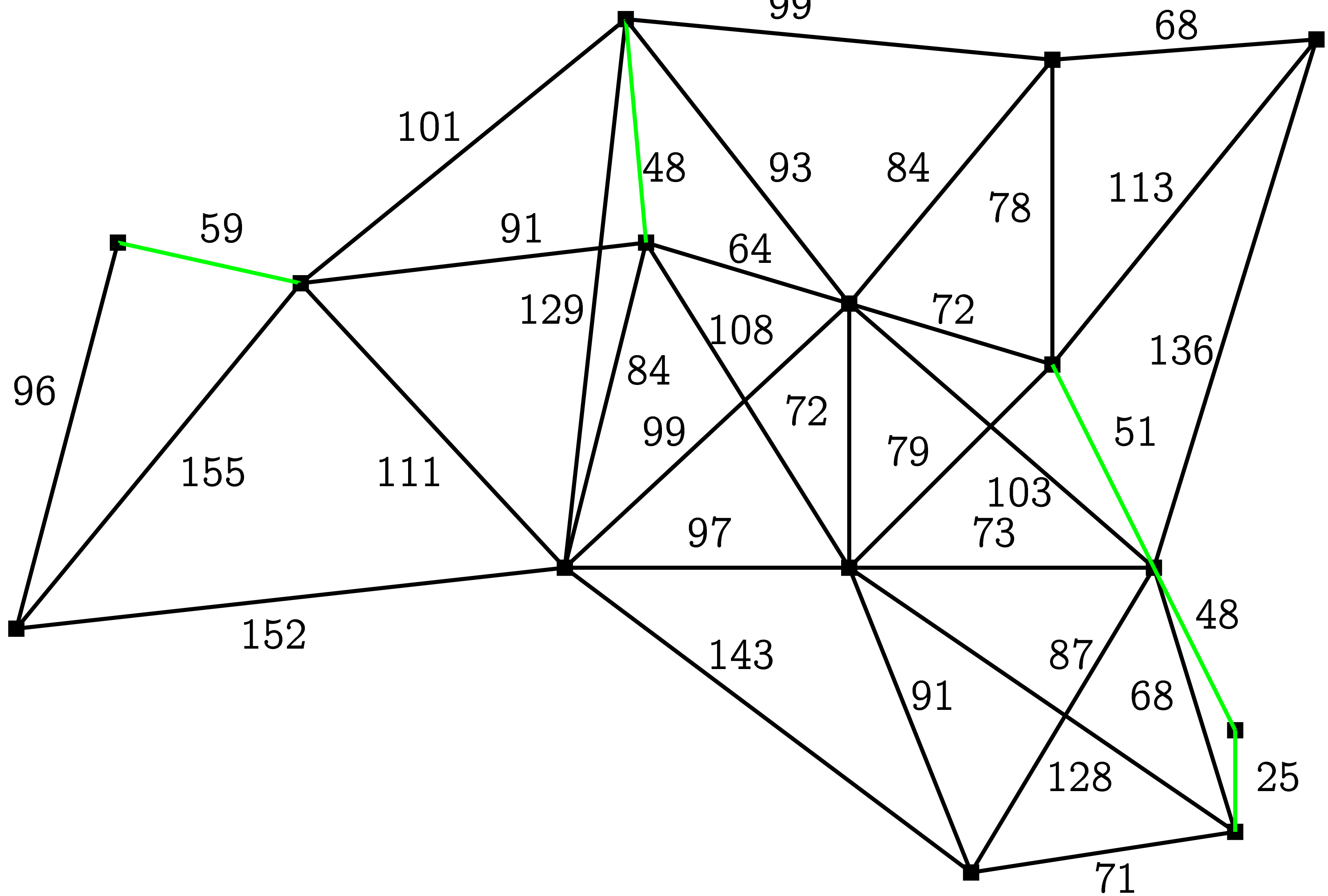
99
68
101
48
93
84
78
113
59
91
64
129
72
108
136
96
84
72
99
79
51
155
111
103
97
73
152
48
143
87
91
68
128
25
71


99
68
101
48
93
84
78
113
59
91
64
129
108
72
136
96
84
72
99
79
51
155
111
103
97
73
48
152
143
87
91
68
128
25
71

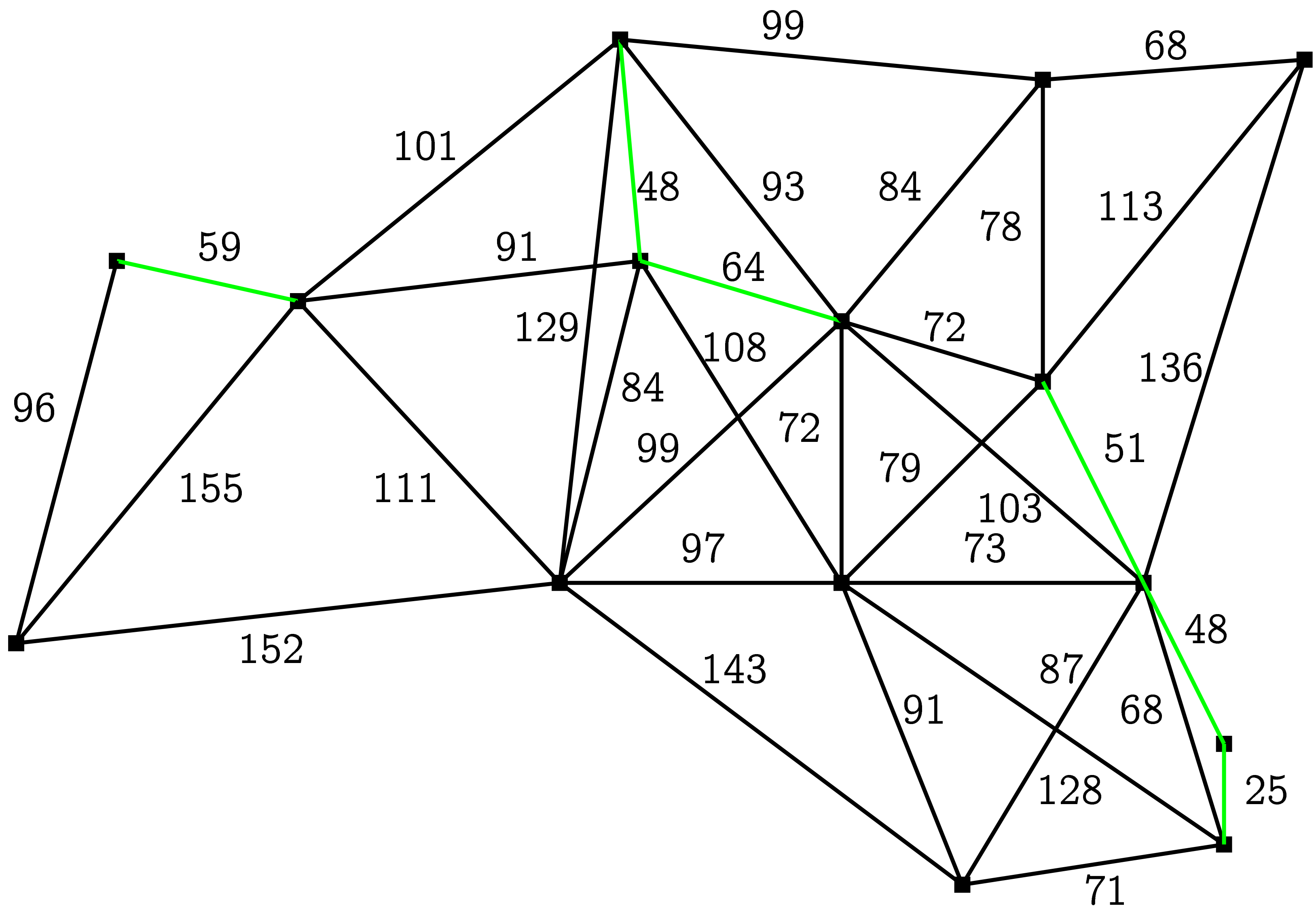

99
68
101
48
93
84
78
113
59
91
64
129
108
72
96
84
136
99
72
51
79
155
111
103
97
73
152
48
143
87
91
68
128
25
71

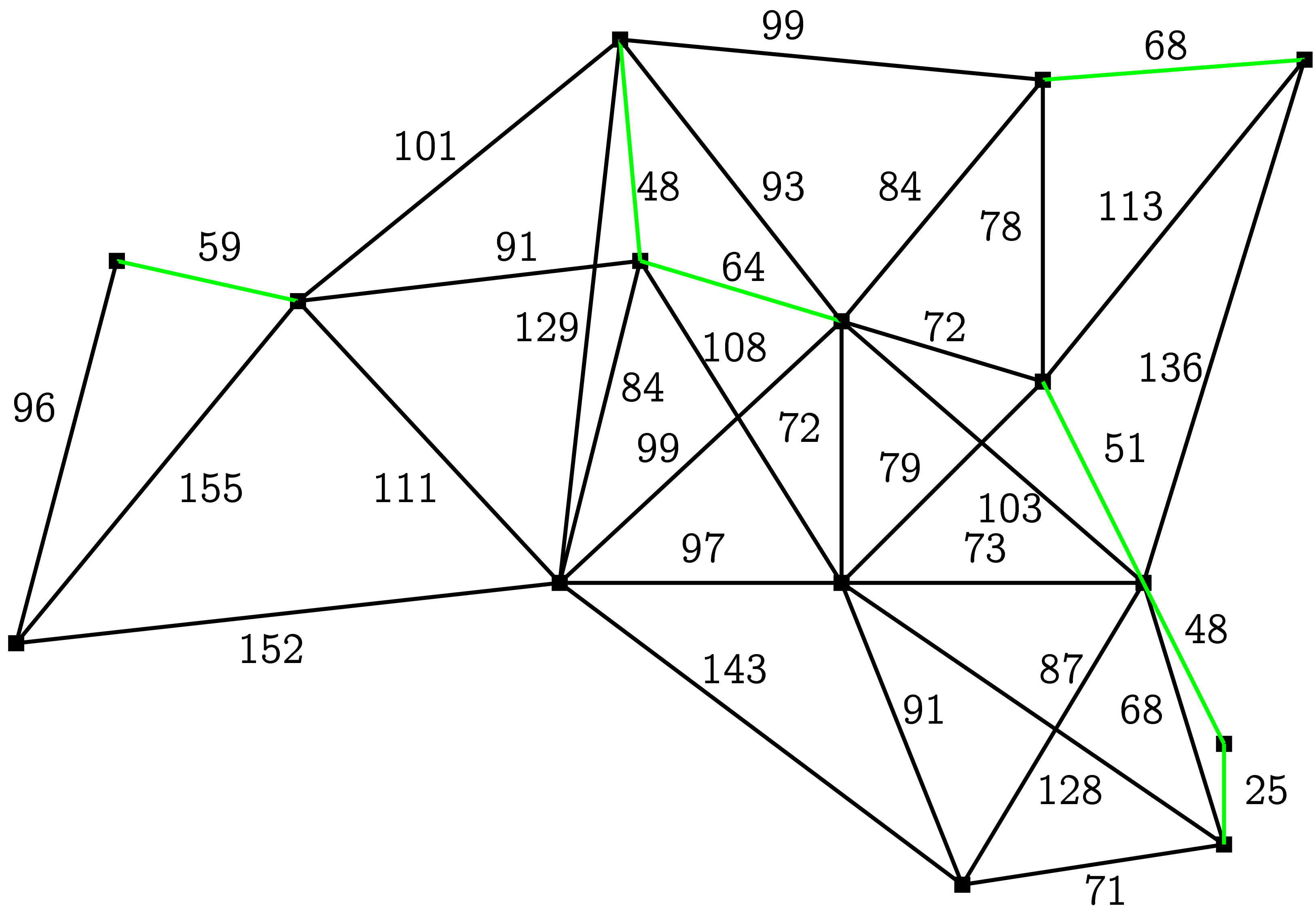


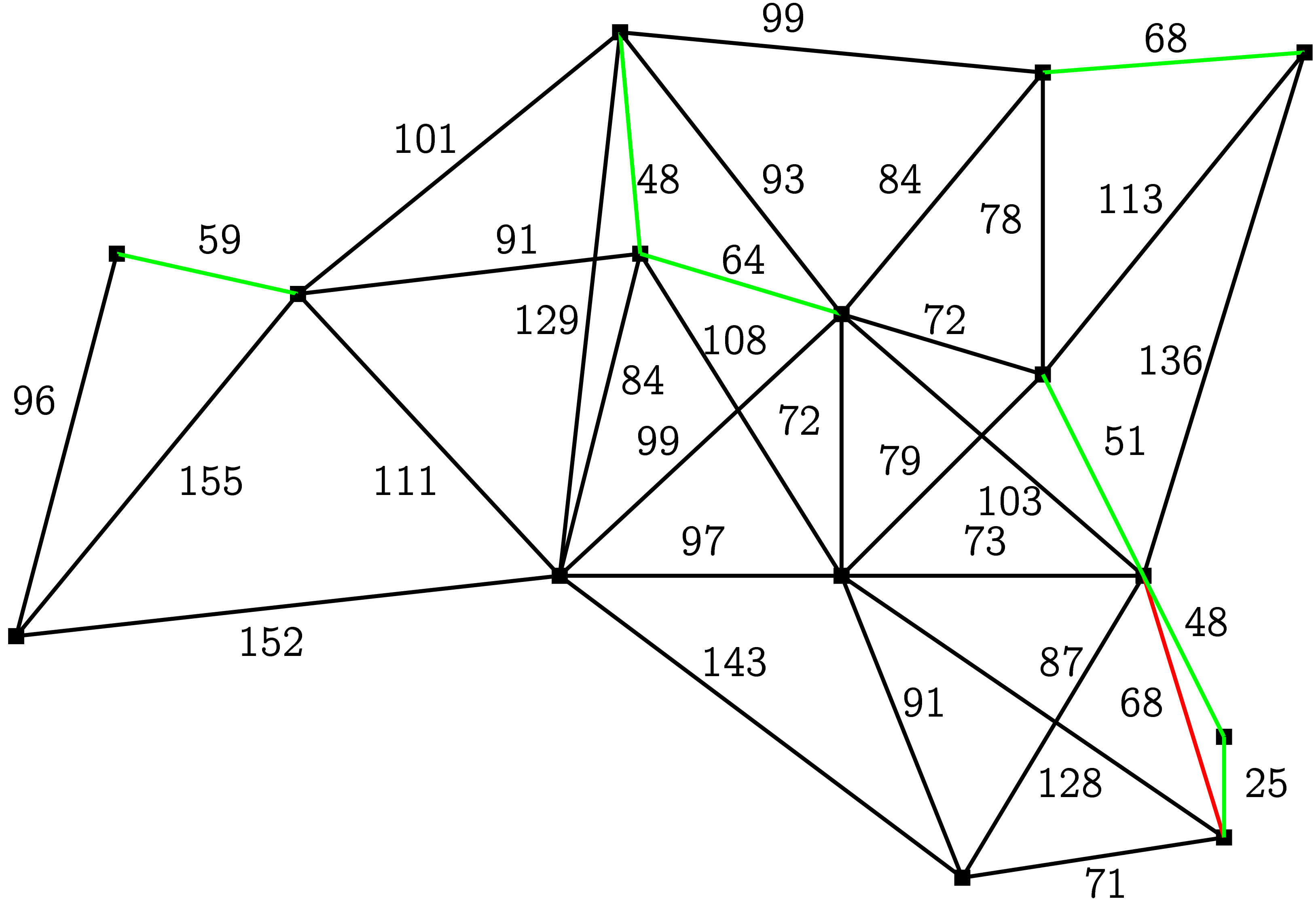
99
68
101
48
93
84
78
113
59
91
64
129
72
108
136
96
84
99
72
79
51
155
111
103
97
73
152
143
48
87
91
68
128
25
71

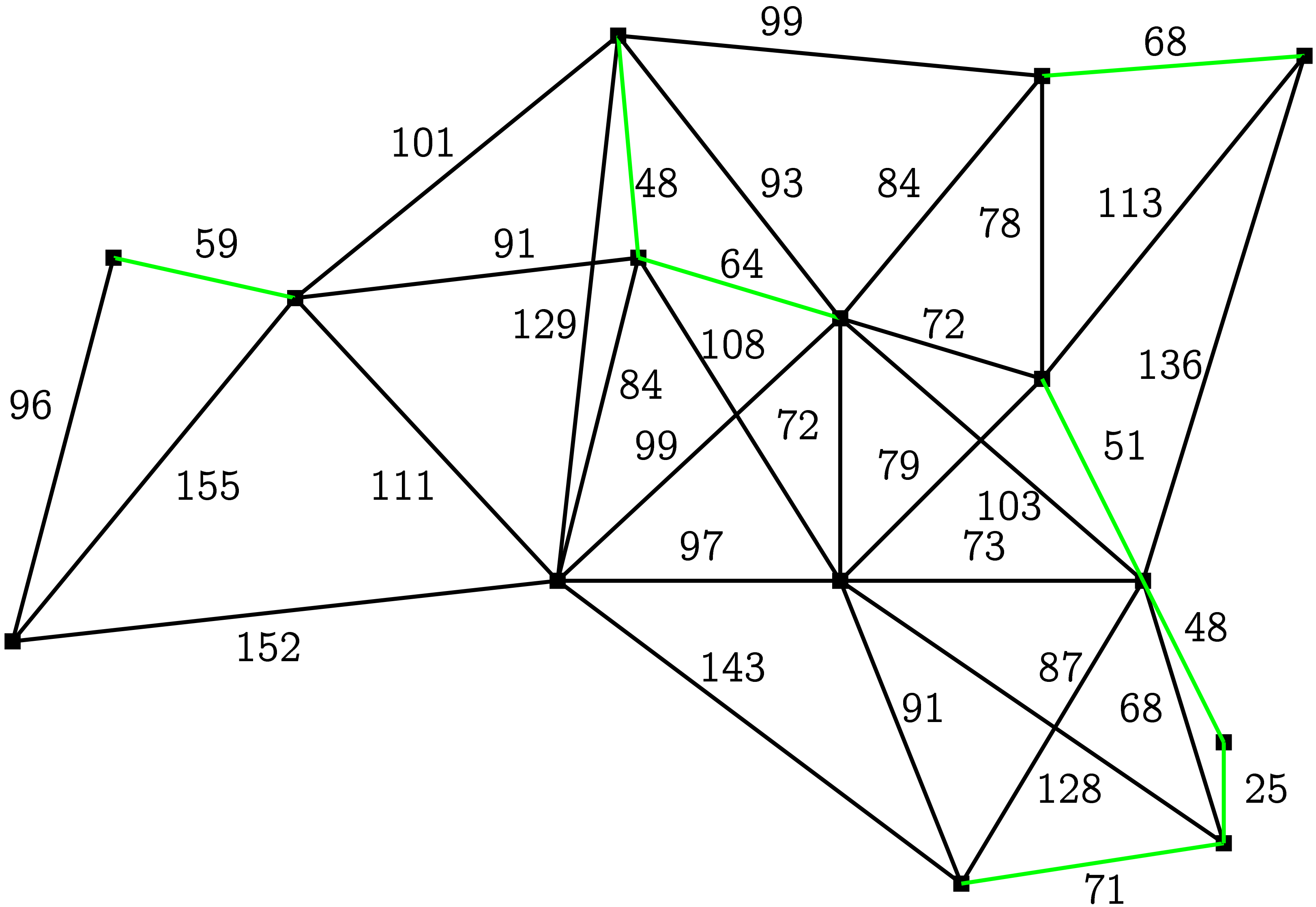


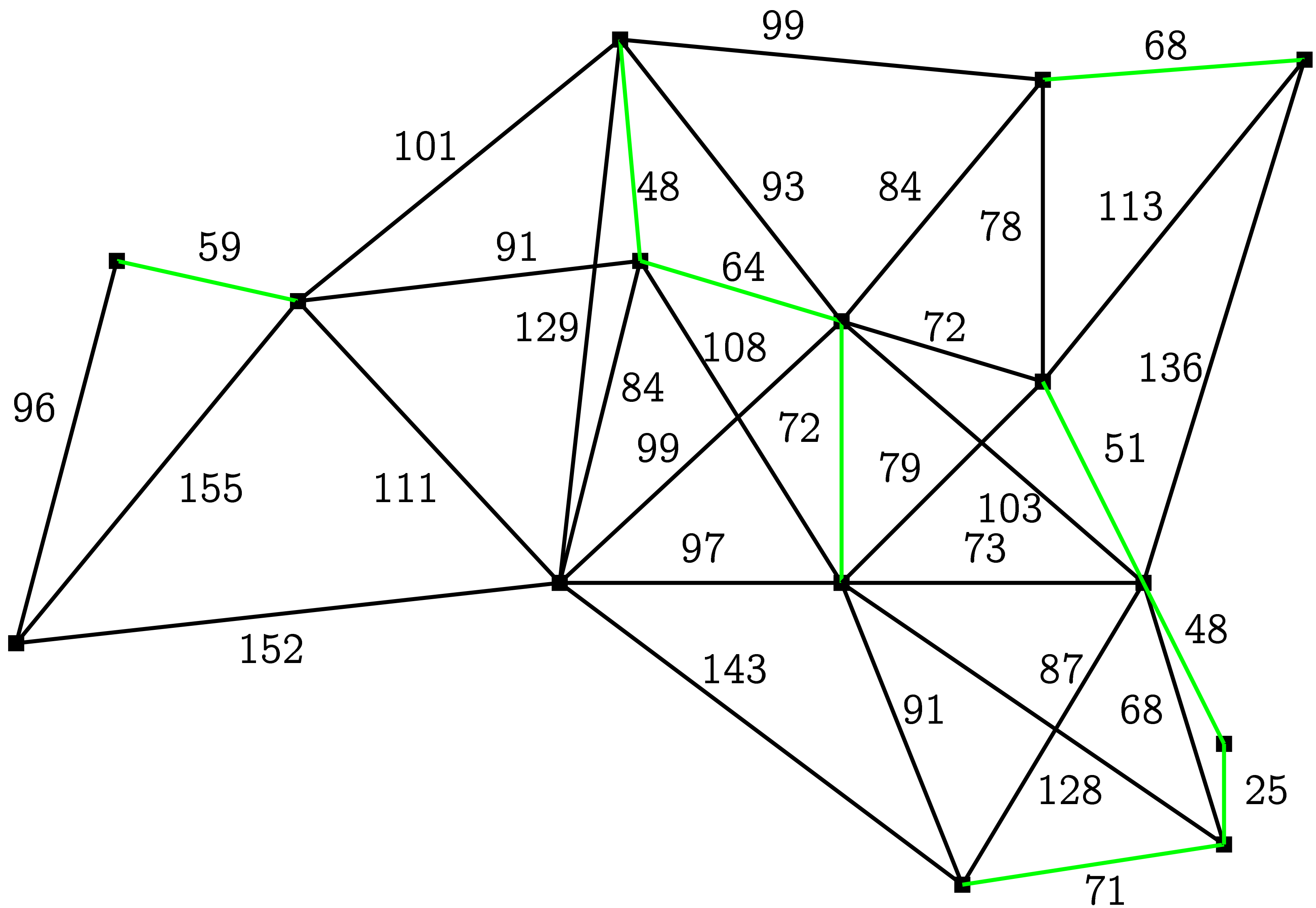
99
68
101
48
93
84
78
113
59
91
64
129
108
72
136
96
84
72
99
79
51
155
111
103
97
73
152
143
48
87
91
68
128
25
71

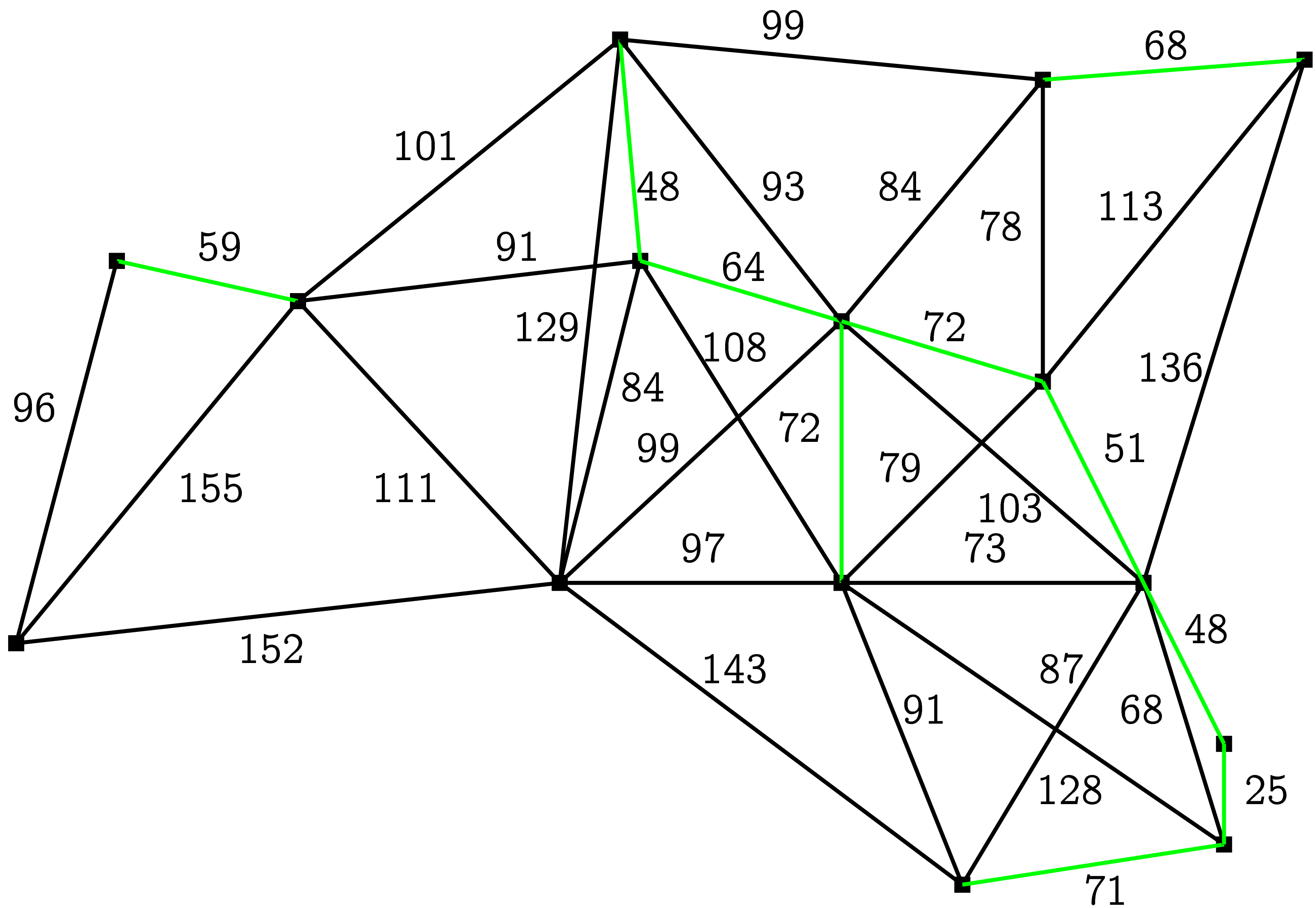

99
68
101
48
93
84
78
113
59
91
64
72
129
108
96
84
136
99
72
79
51
155
111
103
97
73
152
143
48
87
91
68
128
25
71

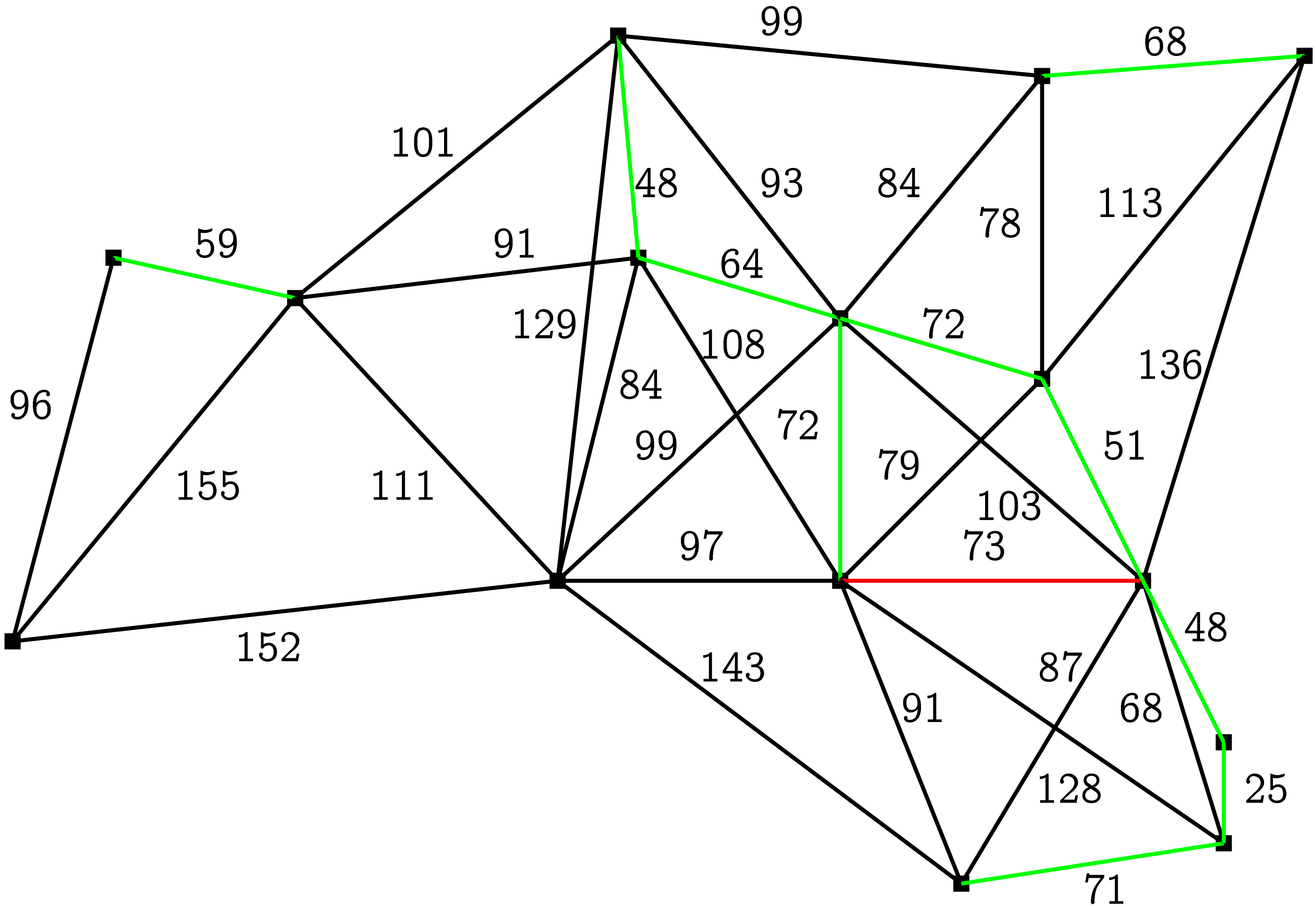

99
68
101
48
93
84
78
113
59
91
64
129
108
72
96
84
136
99
72
79
51
155
111
103
97
73
48
152
143
87
91
68
128
25
71

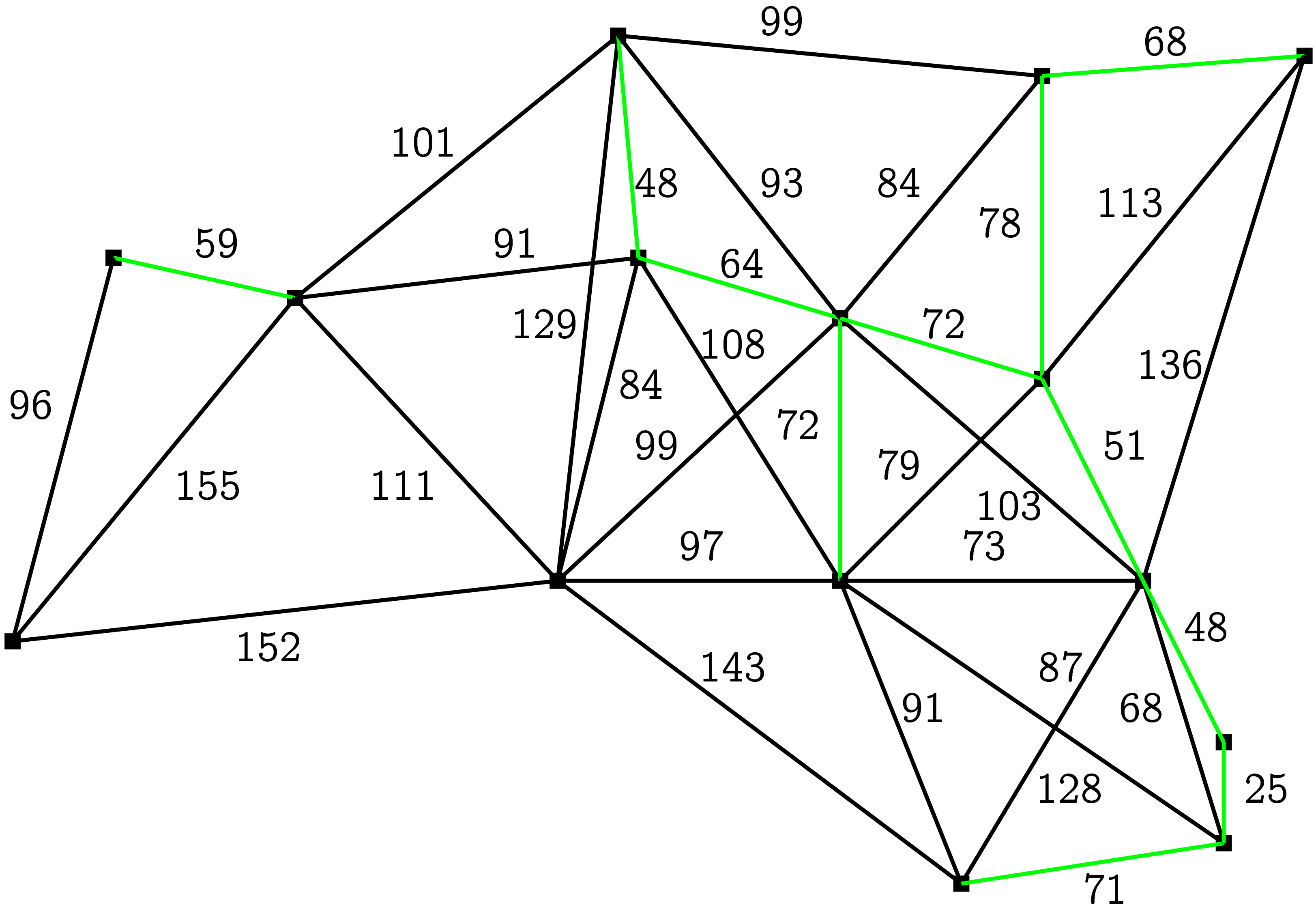

99
68
101
48
93
84
78
113
59
91
64
129
72
108
136
96
84
72
99
79
51
155
111
103
97
73
152
48
143
87
68
91
128
25
71

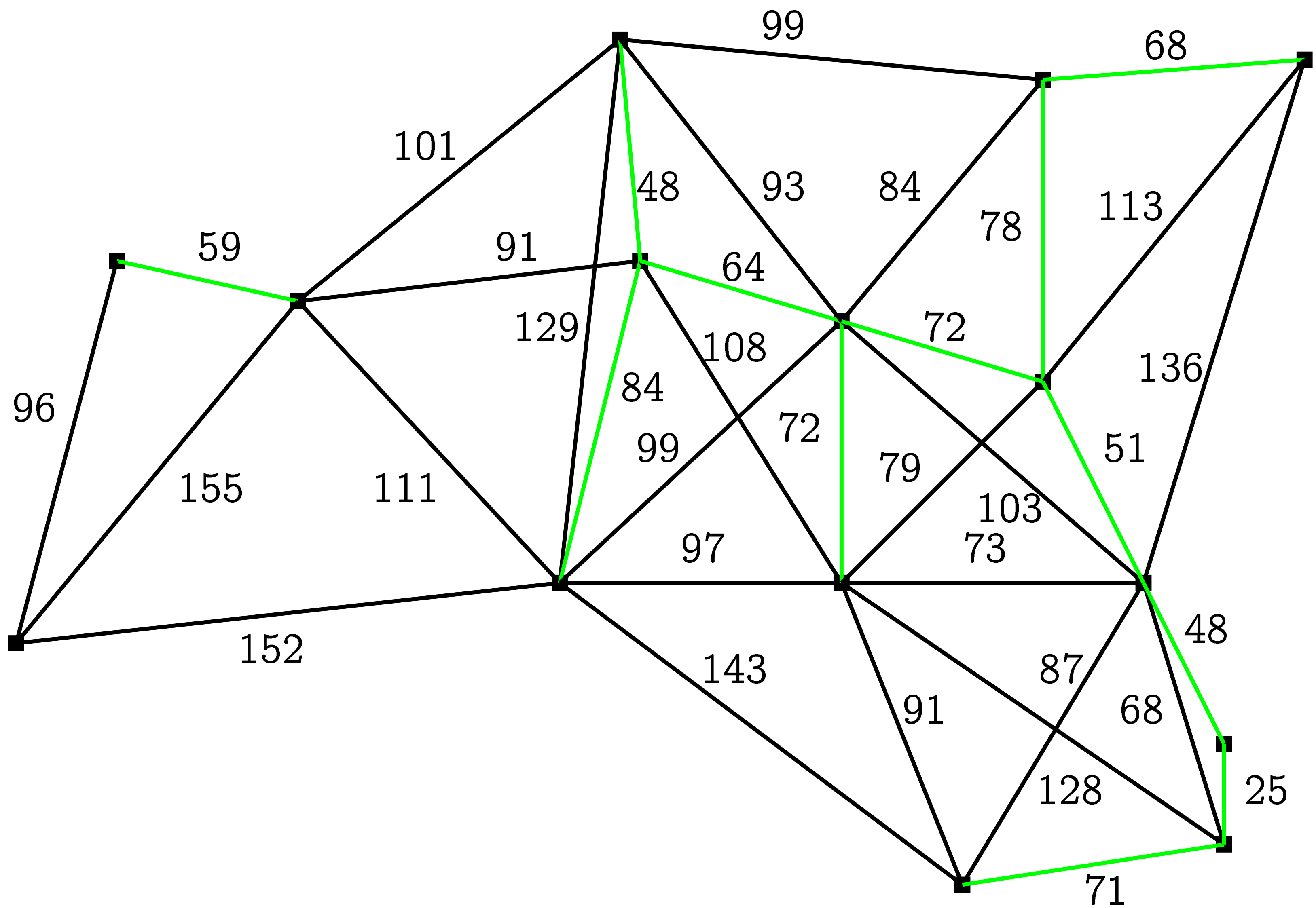

99
68
101
48
93
84
78
113
59
91
64
129
108
72
96
84
136
72
99
79
51
155
111
103
97
73
48
152
143
87
91
68
128
25
71

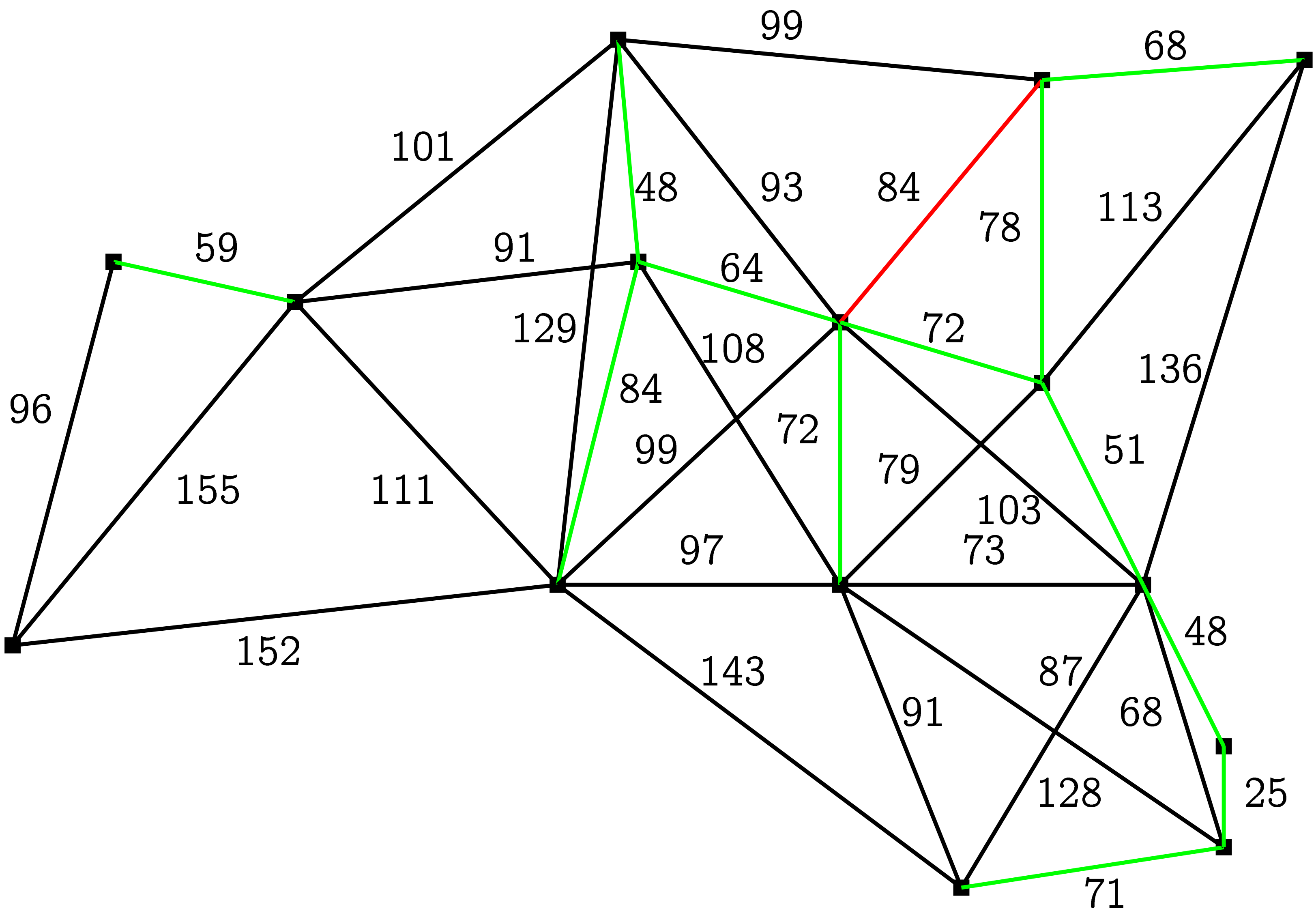

99
68
101
48
93
84
78
113
59
91
64
129
72
108
136
96
84
72
99
79
51
155
111
103
97
73
152
143
87
48
91
68
128
25
71

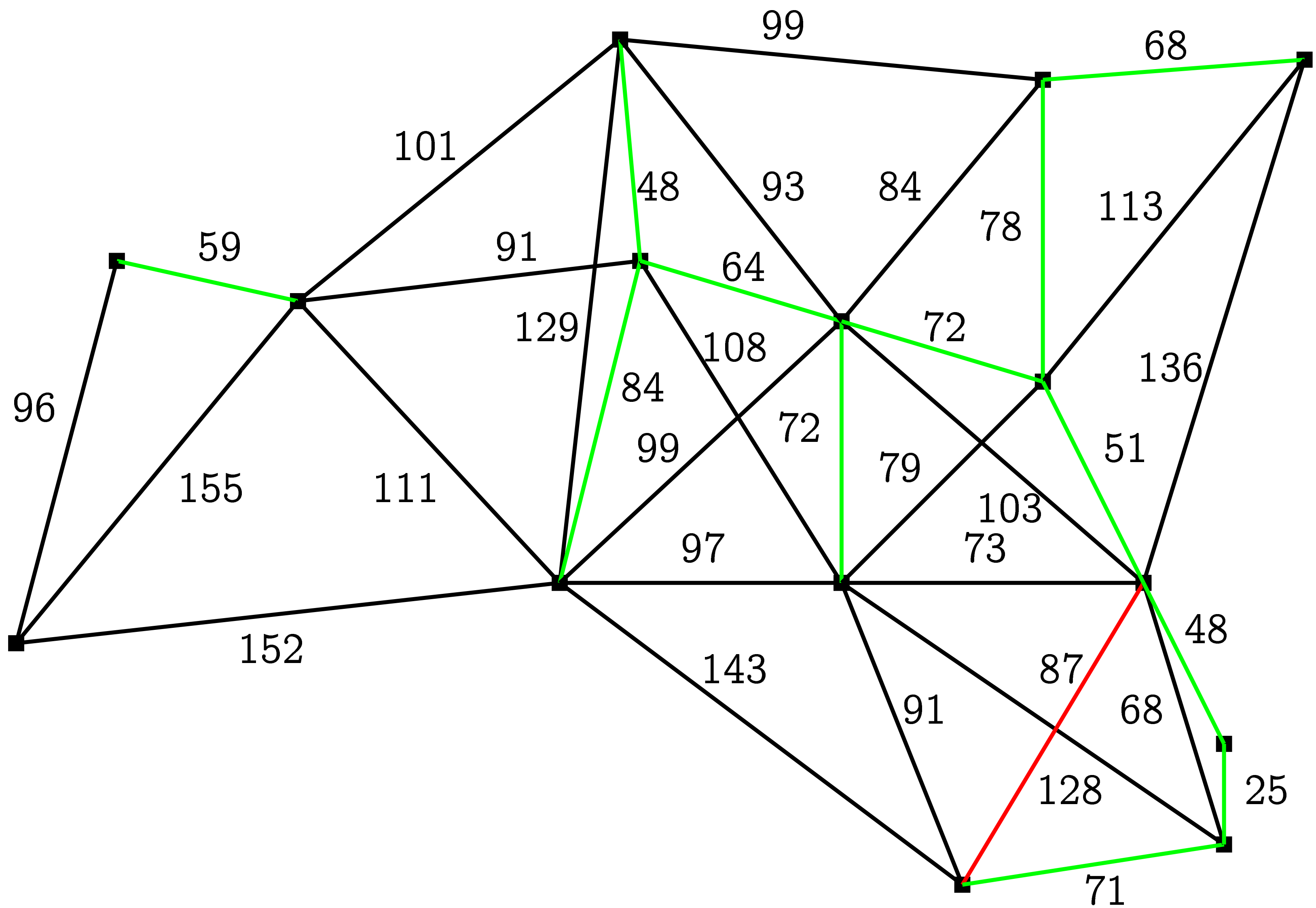

99
68
101
48
93
84
78
113
59
91
64
129
108
72
136
84
96
99
72
79
51
155
111
103
97
73
152
48
143
87
91
68
128
25
71

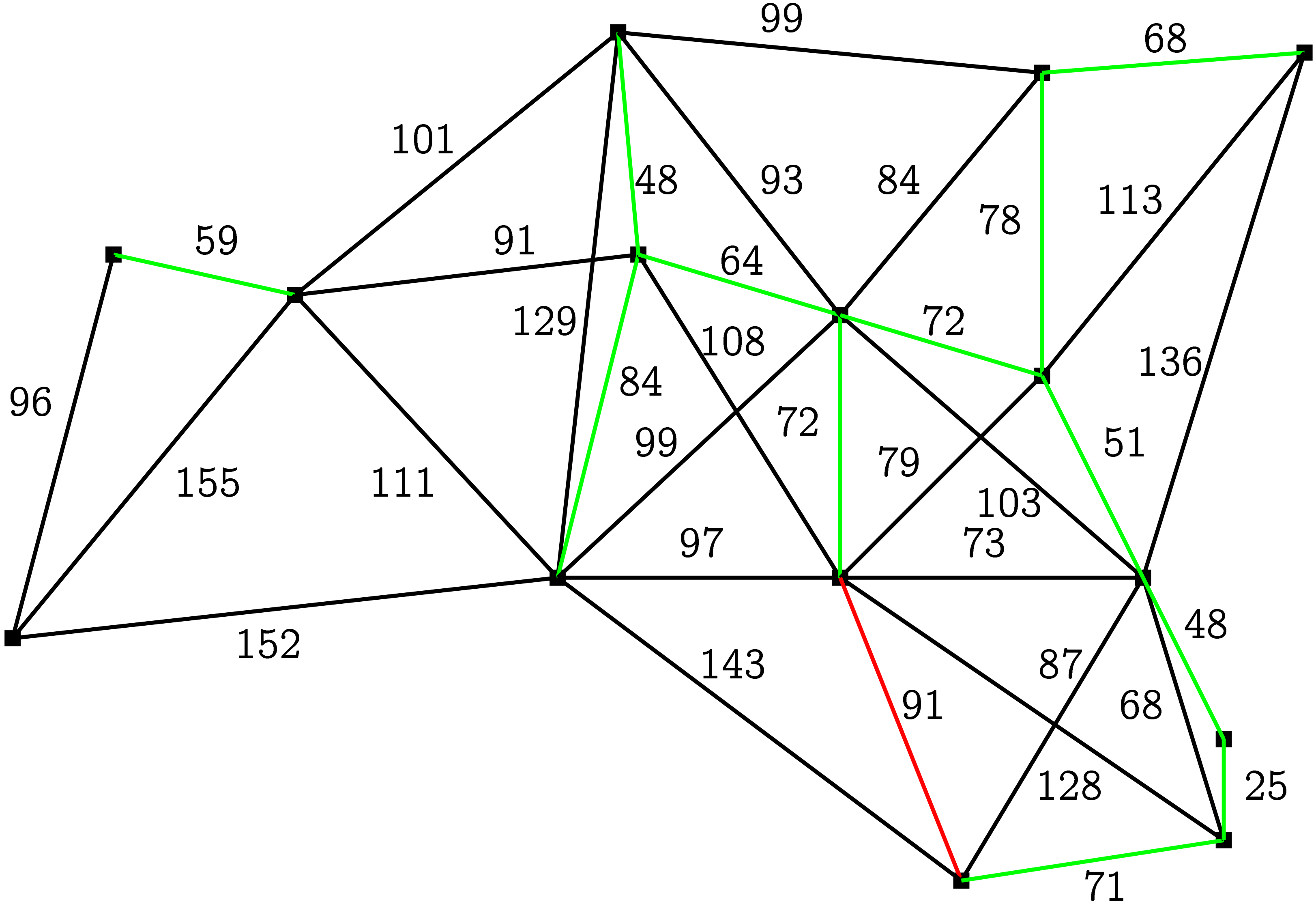

99
68
101
48
93
84
78
113
59
91
64
129
72
108
84
136
96
72
99
79
51
155
111
103
97
73
152
143
87
48
91
68
128
25
71

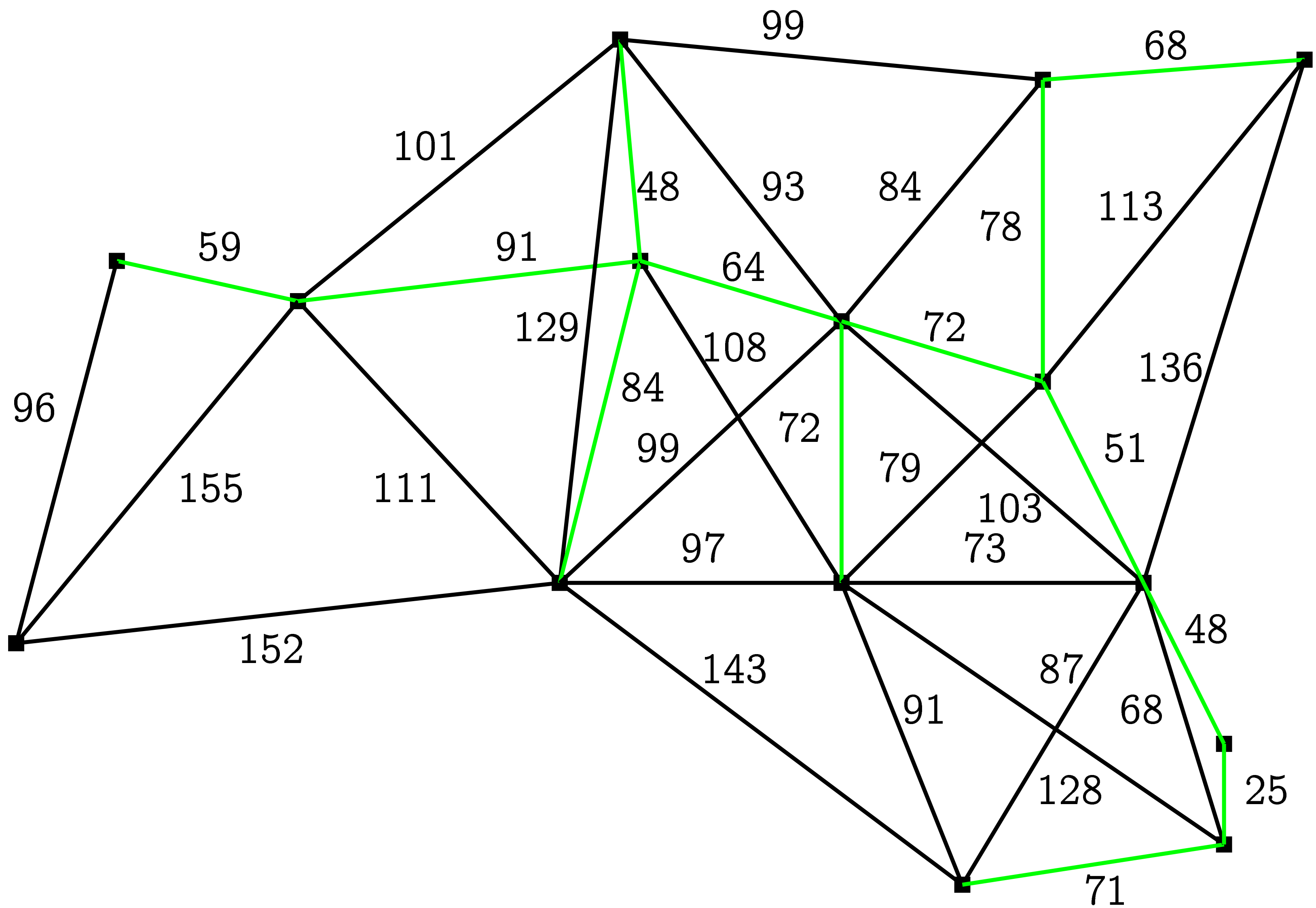

99
68
101
48
93
84
78
113
59
91
64
129
72
108
136
96
84
72
99
79
51
155
111
103
97
73
48
152
143
87
91
68
128
25
71

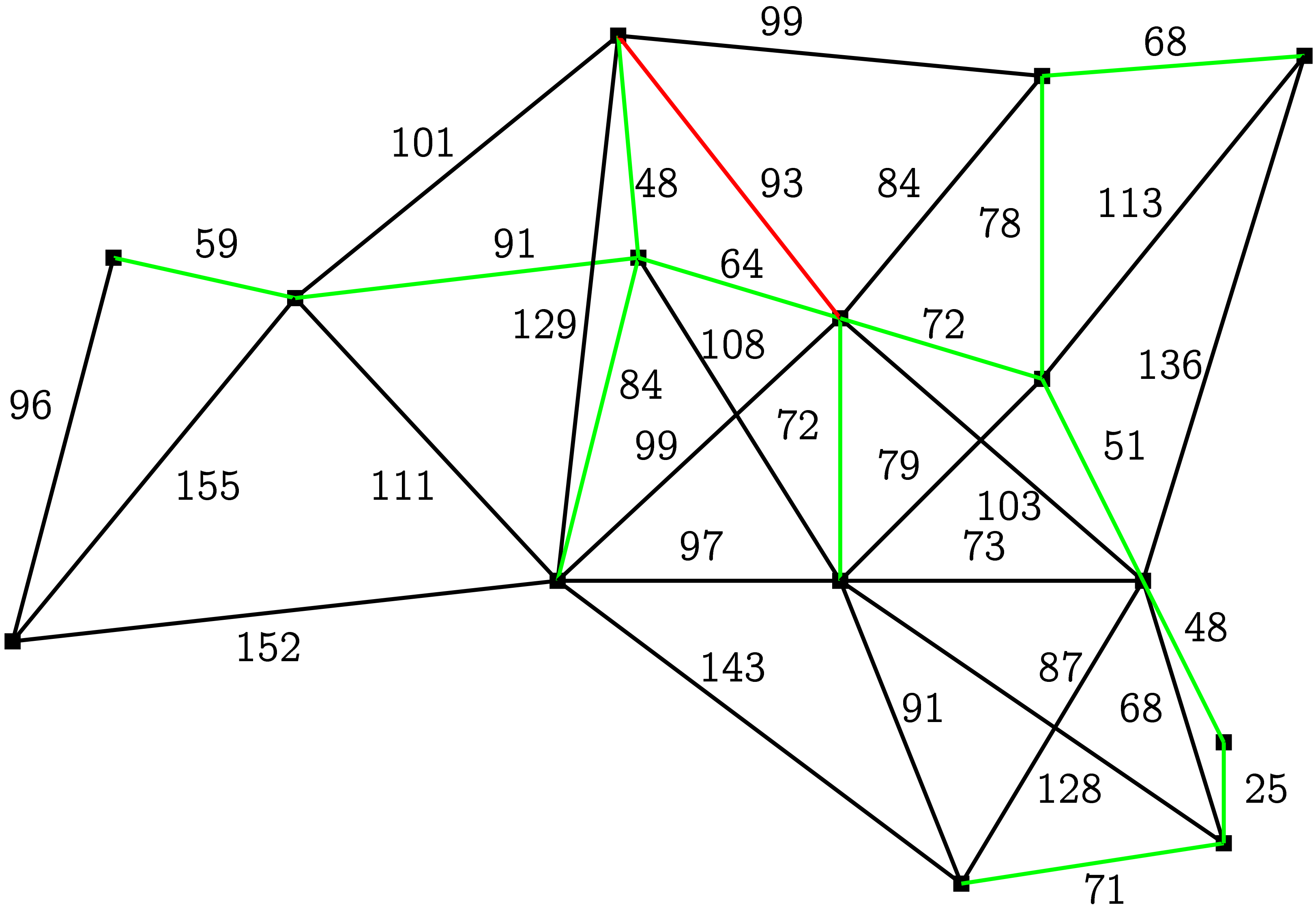

99
68
101
48
93
84
78
113
59
91
64
72
129
108
136
96
84
72
99
79
51
155
111
103
97
73
48
152
143
87
91
68
128
25
71

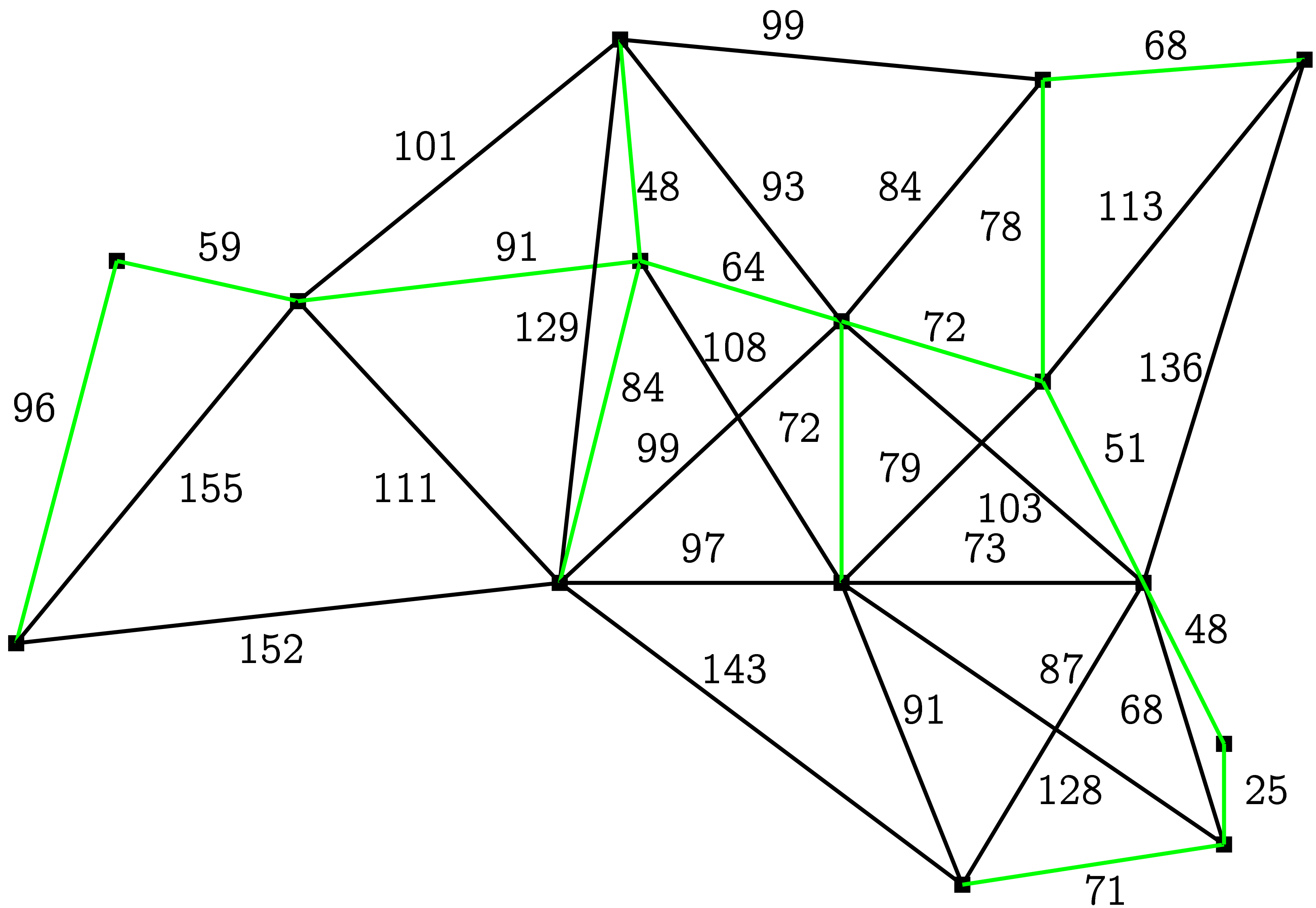

99
68
101
48
93
84
78
113
59
91
64
129
108
72
96
84
136
72
99
79
51
155
111
103
97
73
152
143
87
48
91
68
128
25
71


99
68
101
48
93
84
78
113
59
91
64
72
129
108
136
96
84
72
99
79
51
155
111
103
97
73
152
143
48
87
91
68
128
25
71


99
68
101
48
93
84
78
113
59
91
64
129
72
108
136
96
84
72
99
79
51
155
111
103
97
73
152
143
48
87
91
68
128
25
71


99
68
101
48
93
84
78
113
59
91
64
72
129
108
136
96
84
72
99
79
51
155
111
103
97
73
152
143
87
48
91
68
128
25
71


99
68
101
48
93
84
78
113
59
91
64
72
129
108
136
96
84
72
99
79
51
155
111
103
97
73
152
48
143
87
68
91
128
25
71


99
68
101
48
93
84
78
113
59
91
64
129
108
72
136
96
84
99
72
79
51
155
111
103
97
73
152
143
87
48
91
68
128
25
71

Theorem. The presented algorithm is correct.

Proof. T is a (spanning) tree — it has no cycles, but does have n vertices and $n-1$ edges.

Assume that $w(T)$ is not minimal possible. Let T' be some minimal spanning tree of G. Let T' be such that is has the maximal possible number of edges in common with T.

Let $k \in \{1, \ldots, n-1\}$ be the least number such that $e_k \notin E(T')$.

Let $S = T' \cup \{e_k\}$. The graph S has a cycle C.

Since T and T' have no cycles, we must have $e_k \in C$ and there exists an edge $e \in E(T') \backslash E(T)$ such that $e \in C$.

The graph $T'' = S \backslash \{e\}$ is connected and has $n-1$ edges, i.e. it is a spanning tree.

Edge e

- is different from $e_1, \dots, e_{k-1}$,
- does not form a cycle together with $e_1, \dots, e_{k-1}$ (since $e_1, \dots, e_{k-1} \in E(T')$).

The edge e_k has minimal weight among the edges such that

- are different from $e_1, \dots, e_{k-1}$,
- do not form a cycle together with $e_1, \dots, e_{k-1}$.

Thus $w(e_k) \leq w(e)$.

We obtain $w(T'') = w(T') - w(e) + w(e_k) \leq w(T')$, i.e. T'' is a minimal weight spanning tree.

The tree T'' has more edges in common with T than T' does. A contradiction with the choice of T'. $\square$

Proposition. Let $G = (V, E)$ be connected and $v \in V$. The next three claims are equivalent.

(i) v is a cut-vertex.

(ii) there exist $u, w \in V \setminus \{v\}$, such that any path $u \rightsquigarrow w$ passes v.

(iii) The set $V \setminus \{v\}$ can be partitioned to U and W, such that for any $u \in U$ and $w \in W$, any path $u \rightsquigarrow w$ passes the vertex v.

Proof. (i) $\Rightarrow$ (iii). Graph $G\backslash v$ is not connected. Let one of its connected components be U and the rest of the components be W.

If $u \in U$ and $w \in W$ then the graph $G\backslash v$ has no paths from u to w. Hence any path $u \rightsquigarrow w$ in G passes v.

(iii) $\Rightarrow$ (ii). Take u from U and w from W.

(ii) $\Rightarrow$ (i). If v is located on all paths $u \rightsquigarrow w$, then $G\backslash v$ contains no paths from u to w, i.e. $G\backslash v$ is not connected i.e. v is a cut-vertex. $\square$

A connected graph is a *block*, if it has no cut-vertices.

Theorem. Let $G = (V, E)$ be a connected simple graph with at least 3 vertices. The next claims are equivalent.

(i) G is a block. [i.e. without cut-vertices]

(ii) Any two vertices are located on some cycle.

(iii) Any vertex and any edge are located on some cycle.

(iv) Any two edges are located on some cycle.

(v) For any two vertices and one edge, there is a path connecting those vertices and passing through that edge.

(vi) For any three vertices, there exists a path connecting the first two of them and passing the third one.

(vii) For any three vertices, there exists a path connecting the first two of them and not passing the third one.

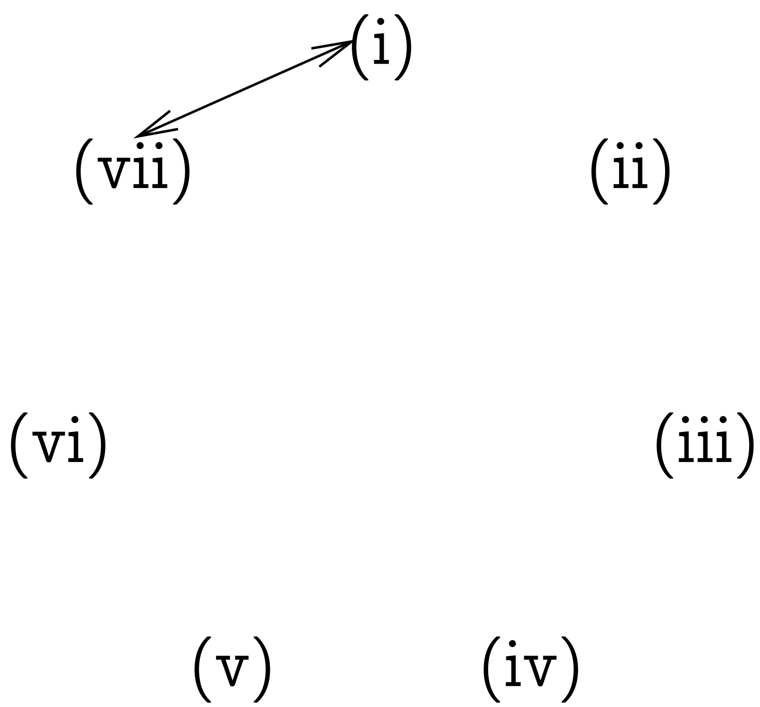
Proof.

(i) $\Rightarrow$ (vii)

Let $u, v, w \in V$. As v is no cut-vertex, the claim (ii) of the previous proposition is not true, i.e. for any u, w there is a path $u \rightsquigarrow w$ that does not pass v.

(vii) $\Rightarrow$ (i)

Let $v \in V$. We show that it is not a cut-vertex. For any $u, w \in V$ there exists a path $u \rightsquigarrow w$ not passing v, thus (ii) of the previous proposition is false.


(i)
(ii)
(iii)
(iv)
(v)
(vi)
(vii)

(i) $\Rightarrow$ (ii)

Let $u, v \in V$ and let $U \subseteq V \backslash \{u\}$ be the set of all vertices that are located on some cycle together with u.

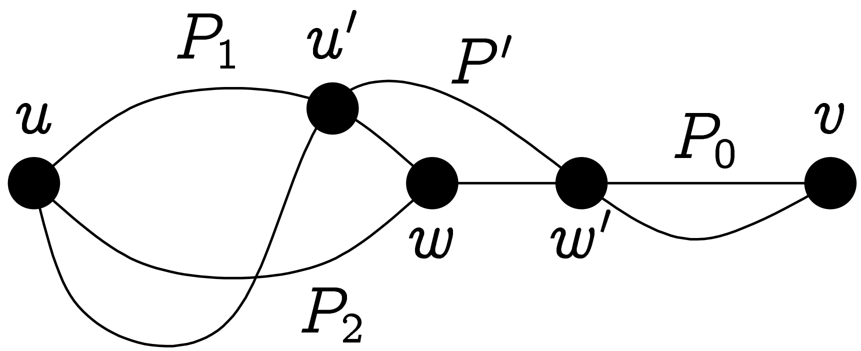
Assume the opposite: $v \notin U$.

U is not empty — it contains at least all neighbours u' of u. Indeed, $G - (u, u')$ is connected, as G has no bridges. Hence there is a path $u \rightsquigarrow u'$ that does not use the edge (u, u'). This path and this edge together form a cycle.

Let $w \in U$ have the minimal possible distance from v. Let

- P_0 be the shortest path $w \rightsquigarrow v$;
- P_1 ja P_2 paths $u \rightsquigarrow w$ that do not intersect.

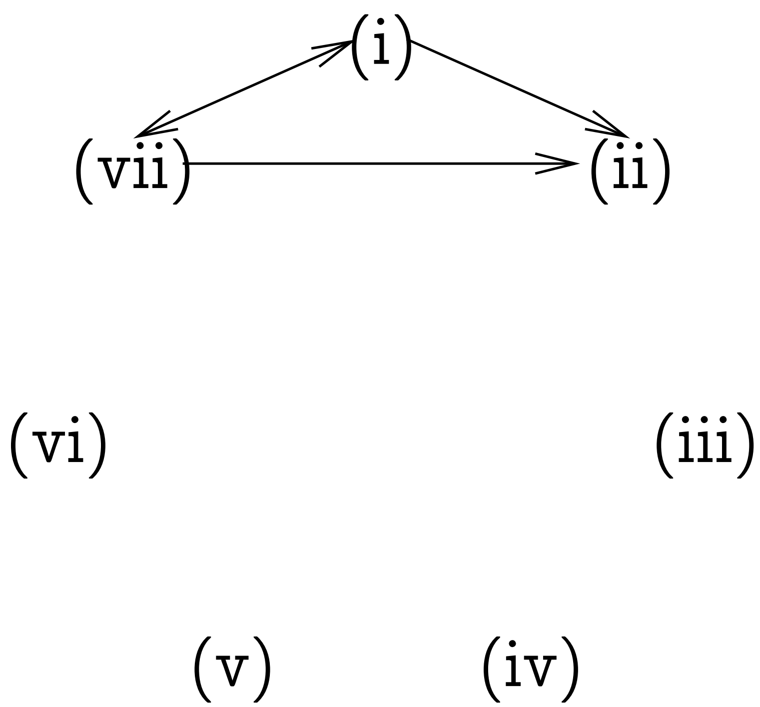
By choice of w, paths P_1 and P_2 do not intersect P_0.



Also define

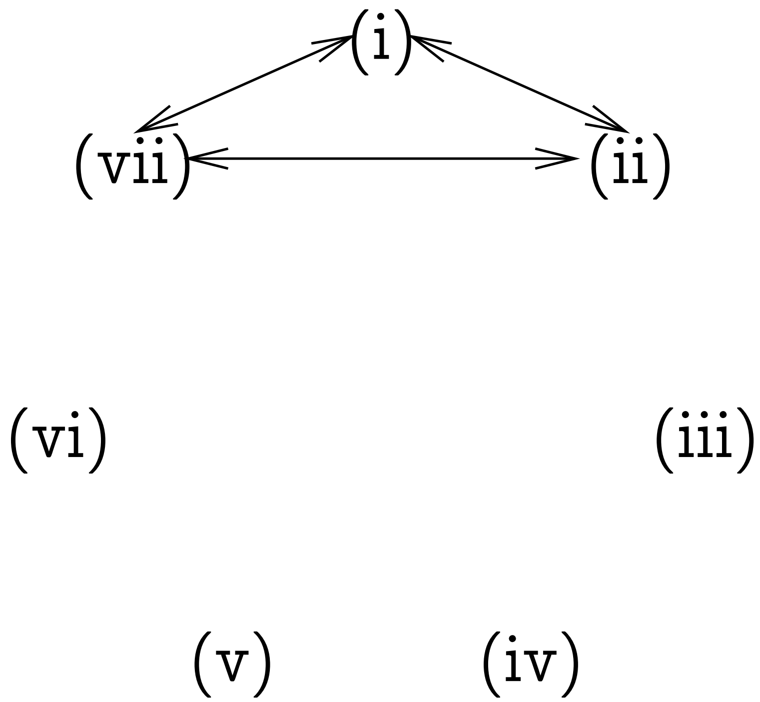
- P' — some path $u \rightsquigarrow v$ that does not pass w (exists by (vii));
- w' — the first (from u) vertex on P' that is also on P_0;
- u' — the last (from u) vertex on P' before w', that is on either P_1 or P_2. Assume w.l.o.g. that it is on P_1.

$u \overset{P_2}{\rightsquigarrow} w \overset{P_0}{\rightsquigarrow} w' \overset{P'}{\rightsquigarrow} u' \overset{P_1}{\rightsquigarrow} u$ is a cycle, hence $w' \in U$ and $d(w', v) < d(w, v)$. Contradiction with the choice of w.


(i)
(ii)
(iii)
(iv)
(v)
(vi)
(vii)

(ii) $\Rightarrow$ (vii)

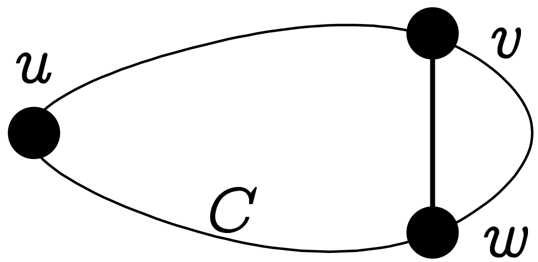
Let $u, v, w \in V$. There exists a cycle containing u and w. Hence there are two non-intersecting paths $u \rightsquigarrow w$. At least one of them does not contain v.


(i)
(vii)
(ii)
(vi)
(iii)
(v)
(iv)

(ii) $\Rightarrow$ (iii)

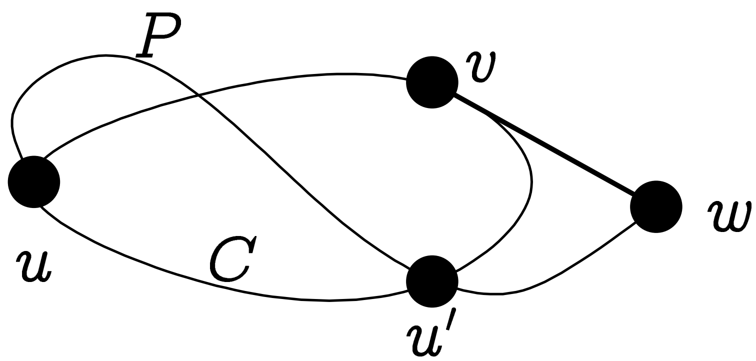
Let $u \in V$ and $(v, w) \in E$. Let C be a cycle passing through u and v.

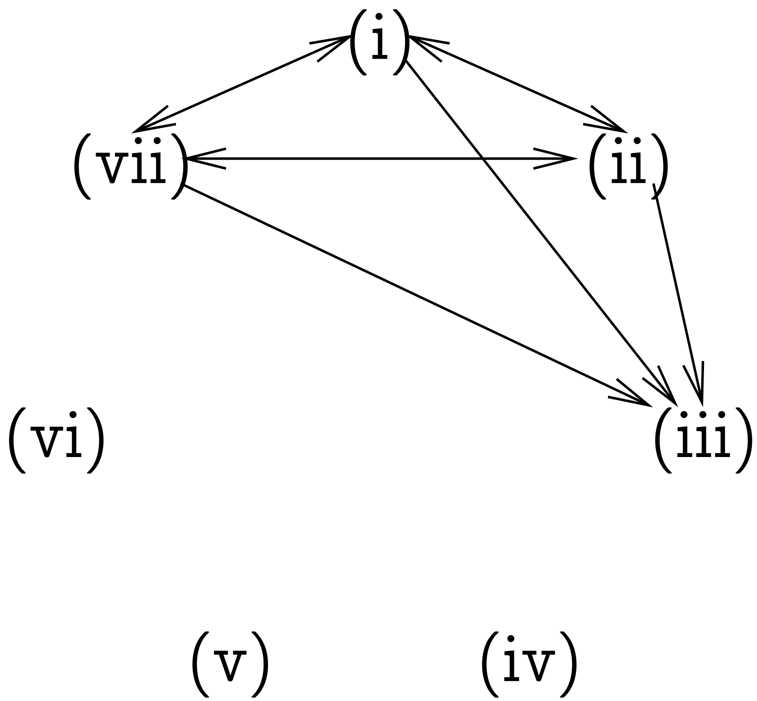
If w is on C then replace the subpath of C between v and w with the edge (v, w).



If w is not on C, then let P be a path $u \rightsquigarrow w$ that does not contain v (exists by (vii)). Let u' be the last (from u) vertex on that path, that is also on C.

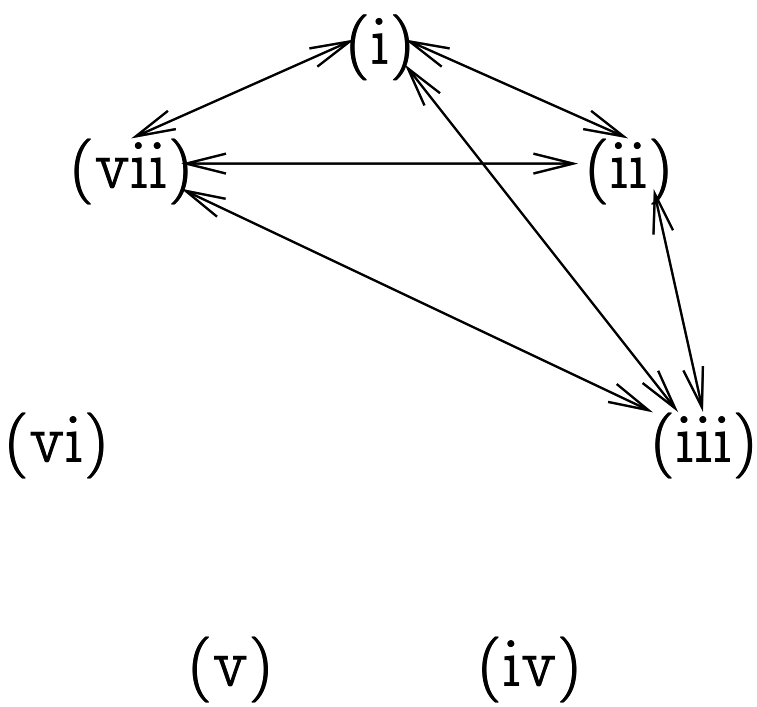
In C replace the subpath between u' and v by $u' \overset{P}{\rightsquigarrow} w$ — v.




(i)
(vii)
(ii)
(vi)
(iii)
(v)
(iv)

(iii) $\Rightarrow$ (ii)

Let $u, v \in V$. Let w be a vertex adjacent to v (there exists one, because G is connected). A cycle passing through u and the edge (v, w) passes through u and v.

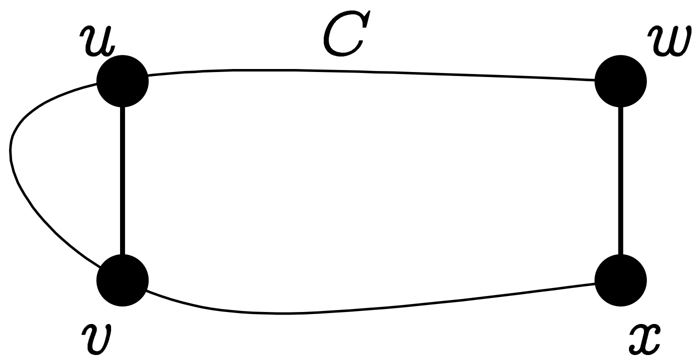

(i)
(vii)
(ii)
(vi)
(iii)
(v)
(iv)

(iii) $\Rightarrow$ (iv)

Let $(u, v), (w, x) \in E$. If these edges have a common vertex, e.g. $v = w$, then a cycle is given by these two edges and a path $u \rightsquigarrow x$ in the graph $G \backslash v$ (it is connected by (i)).

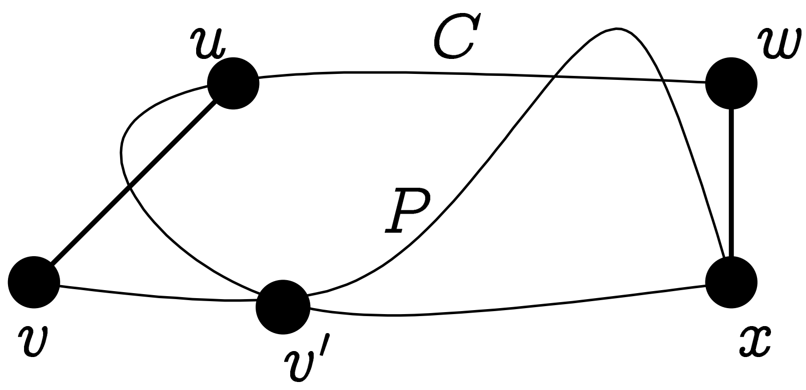
If u, v, w, x are all different then let C be a cycle passing through u and (w, x).

If v is on C then replace the subpath of C between u and v with the edge (u, v)

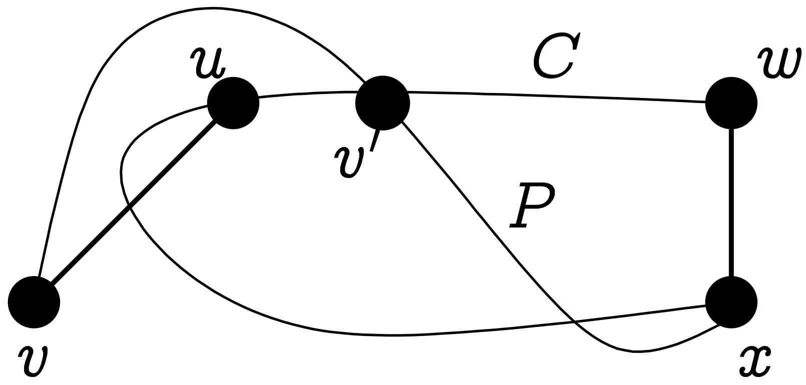


Otherwise let P be a path $x \rightsquigarrow v$ not passing u (it exists by (vii)). Let v' be the last (from x) vertex on P that is also on the cycle C.

If v' is on C between u and x, then the cycle we are looking for is $x \overset{C}{\rightsquigarrow} v' \overset{P}{\rightsquigarrow} v — u \overset{C}{\rightsquigarrow} w — x$.

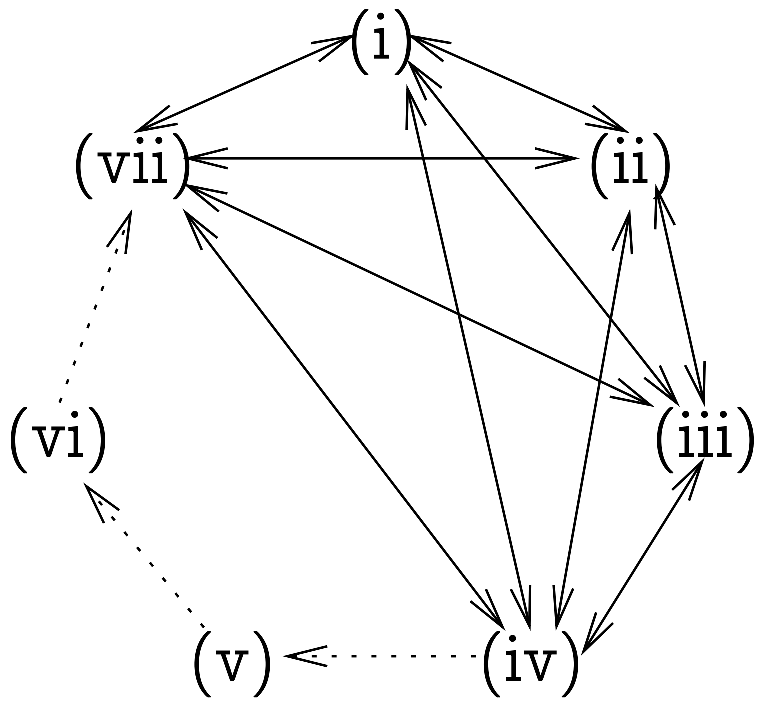


If v' is on C between u and w, then the cycle we are looking for is $x \overset{C}{\rightsquigarrow} u — v \overset{P}{\rightsquigarrow} v' \overset{C}{\rightsquigarrow} w — x$.



(iv) $\Rightarrow$ (iii)

Like (iii) $\Rightarrow$ (ii)



(i)
(vii)
(ii)
(vi)
(iii)
(v)
(iv)

(iv) $\Rightarrow$ (v)

Let $u, v \in V$ and $(w, x) \in E$. The graph is a block because of (iv)$\Rightarrow$(i). Define Olgu

$$G' = \begin{cases} G & \text{if } (u, v) \in E \\ G + (u, v) & \text{if } (u, v) \notin E \end{cases}.$$

By adding edges to a connected graph, we are not introducing any cut-vertices. Hence G' is a block and (iv) holds for it, too.

By (iv), there exists a cycle C in G' passing through the edges (u, v) and (w, x).

$C - (u, v)$ is the path connecting u and v and passing through the edge (w, x). All edges of that path are in G.

(v) $\Rightarrow$ (vi)

Let $u, v, w \in V$. Let x be adjacent to v. By (v), there exists a path $P : u \rightsquigarrow w$, containing the edge (v, x), hence also the vertex v.

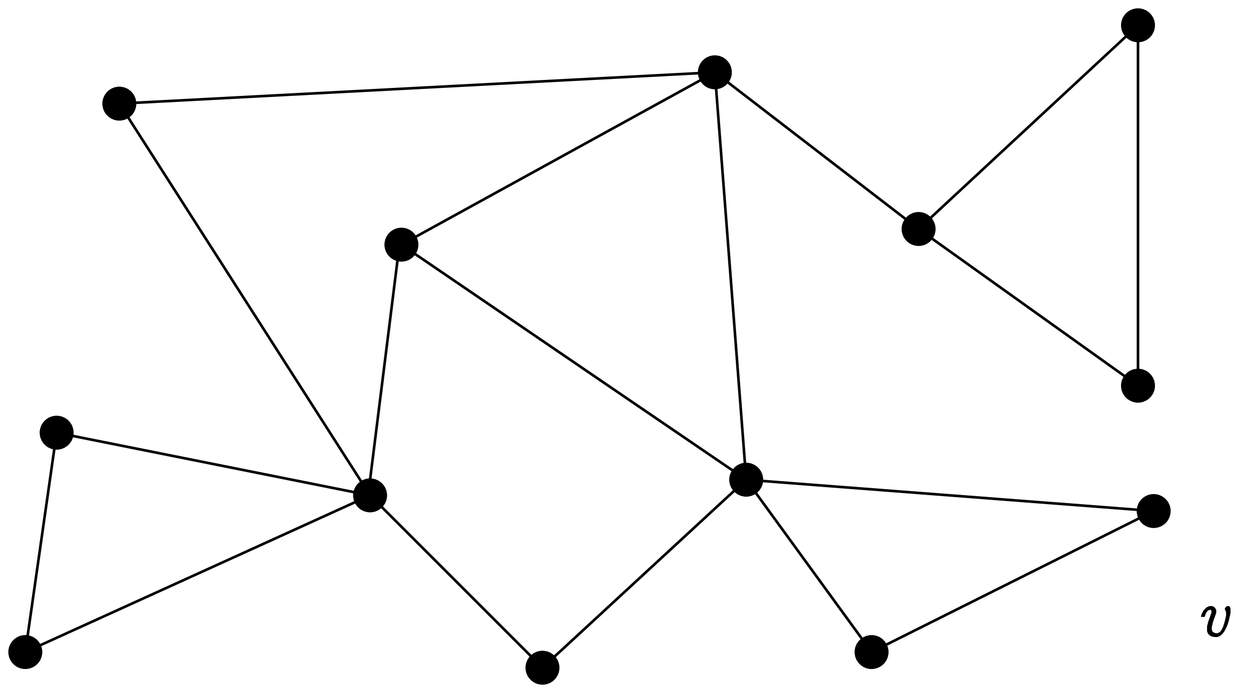
(vi) $\Rightarrow$ (vii)

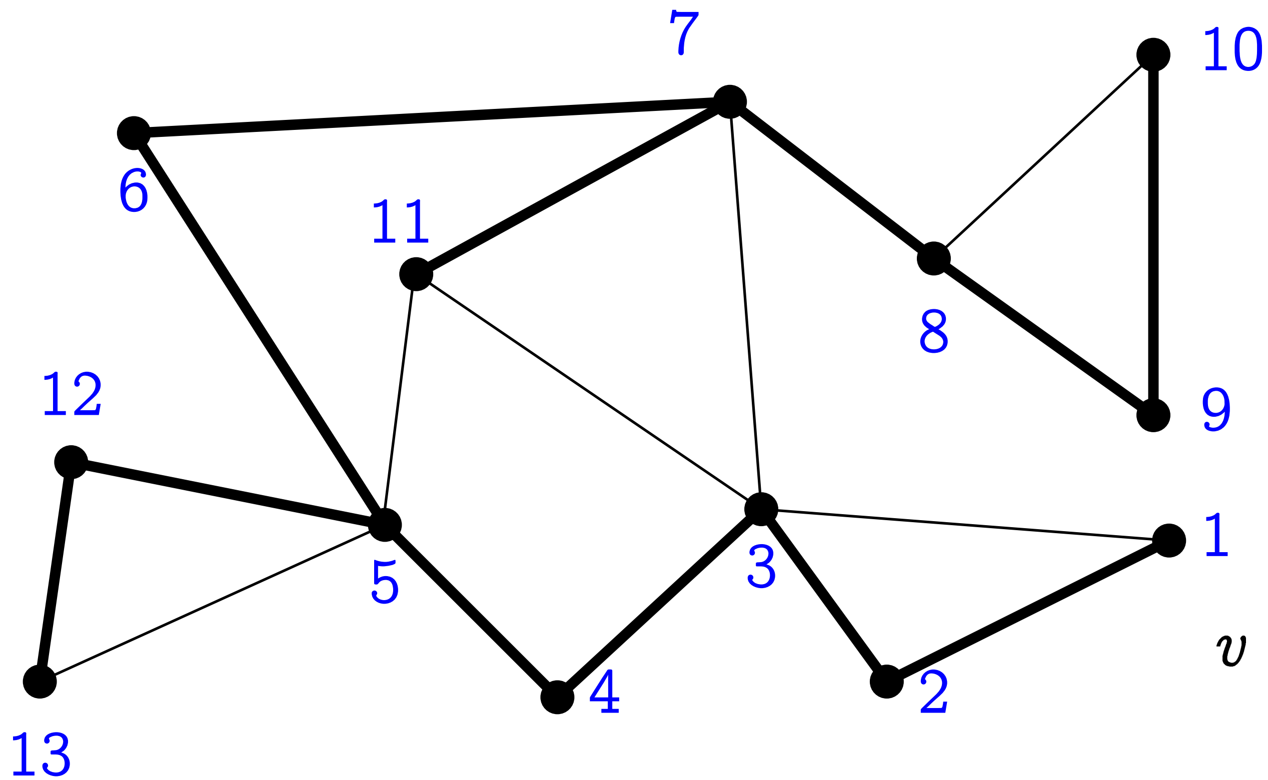
Let $u, v, w \in V$. By (vi) there exists a path $P : u \rightsquigarrow v$ passing through w. The subpath of P from u to w does not contain v. $\square$

Finding cut-vertices in a connected graph $G = (V, E)$:

First step.

- Pick $v \in V$.
- Do a depth-first search in G starting from v.
- Number the vertices of G in the order they are visited.
  - (pre-order)
  - Let $n(u)$ be the number of the vertex u.


v


7
10
6
11
8
12
9
1
5
3
v
4
2
13

Second step.

For each vertex $u \in U$ find the smallest number of a vertex that can be reached from u by following

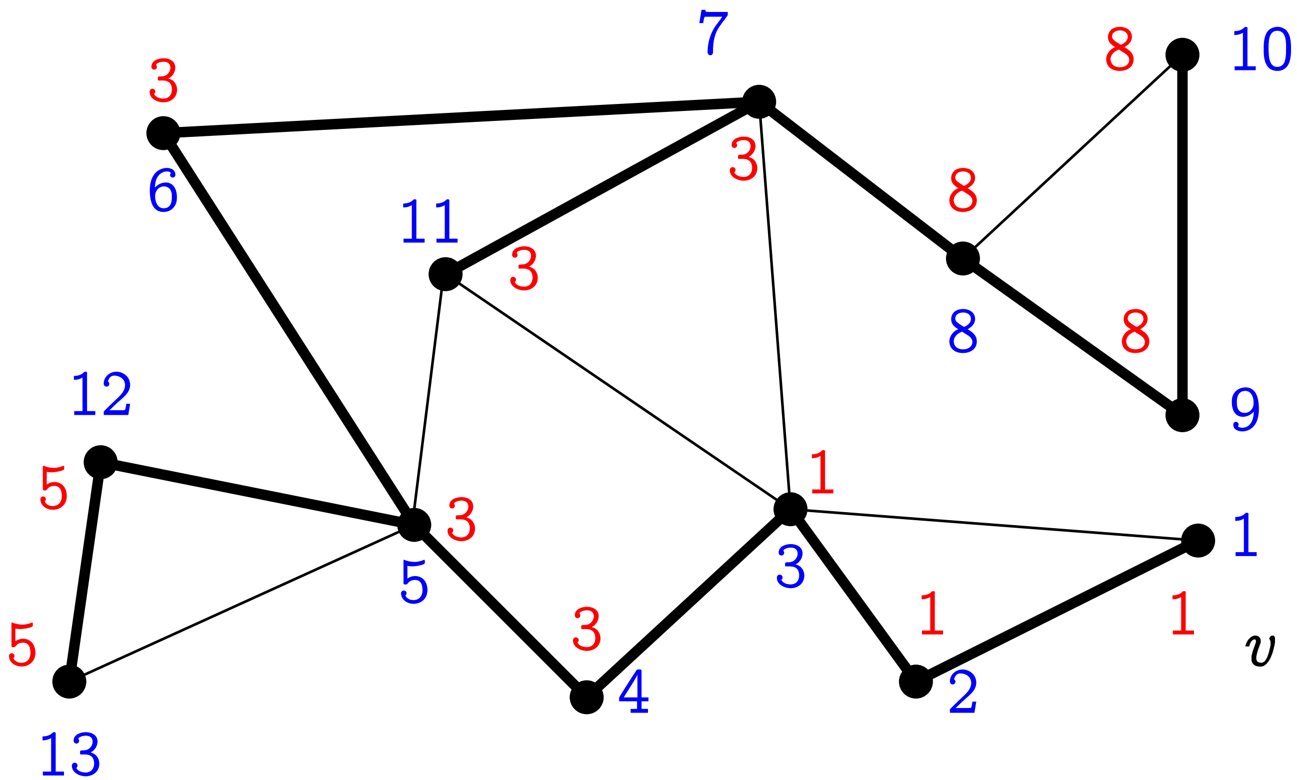
- Any number of tree edges, followed by
- At most one back-edge.

The edges have to be followed in the correct direction:

- tree-edges: from smaller- to greater-numbered vertex;
- back-edges: from greater- to smaller-numbered vertex.

To find those numbers $l(u)$:

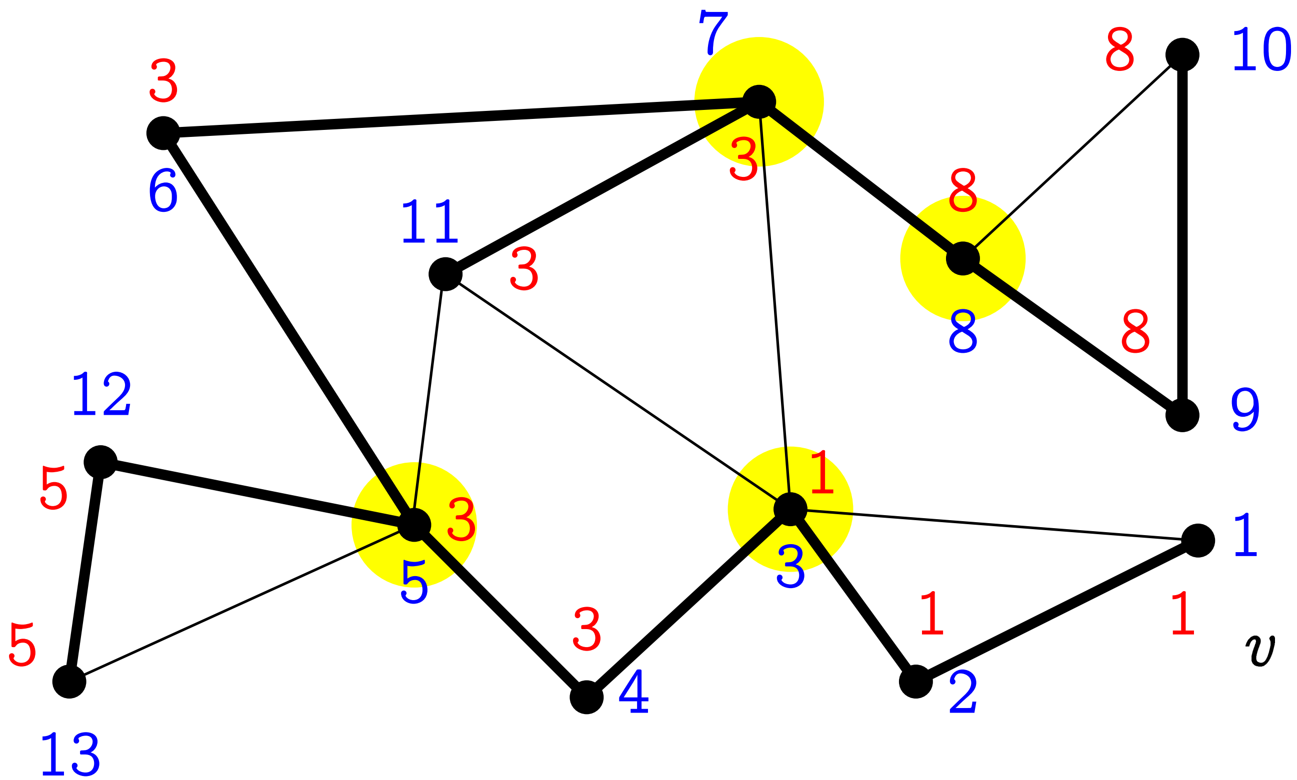
- Let u range over V, in the order of decreasing $n(u)$.
- Let $l(u)$ be the minimum of
  - $n(u)$;
  - $l(w)$ for any child w of u in the DFS-tree;
  - $n(w)$ for any w, such that (u, w) is a back-edge.



7
8
10
3
3
8
6
11
3
8
8
12
9
1
5
3
1
3
5
1
1
3
5
v
4
2
13

Third step.

- v is a cut-vertex if it has at least two children.
- Any other $u \in V$ is a cut vertex if it has a child w, such that $l(w) \geq n(u)$.



7
3
8
10
3
8
6
11
3
8
8
12
9
5
3
1
5
3
1
3
1
5
1
4
v
2
13