Isaac is tired of his daily trip to his office, using the same shortest route everyday. Although this saves his time, he must see the same scenery again and again. He cannot stand such a boring commutation any more.

One day, he decided to improve the situation. He would change his route everyday at least slightly. His new scheme is as follows. On the first day, he uses the shortest route. On the second day, he uses the second shortest route, namely the shortest except one used on the first day. In general, on the k-th day, the k-th shortest route is chosen. Visiting the same place twice on a route should be avoided, of course.

You are invited to help Isaac, by writing a program which finds his route on the k-th day. The problem is easily modeled using terms in the graph theory. Your program should find the k-th shortest path in the given directed graph.

Input

The input consists of multiple datasets, each in the following format.

$$
\begin{array}{lllll}
n & m & k & a & b \\
x_1 & y_1 & d_1 & & \\
x_2 & y_2 & d_2 & & \\
 & \vdots & & & \\
x_m & y_m & d_m & &
\end{array}
$$

Every input item in a dataset is a non-negative integer. Two or more input items in a line are separated by a space.

n is the number of nodes in the graph. You can assume the inequality $2 \leq n \leq 50$. m is the number of (directed) edges. a is the start node, and b is the goal node. They are between 1 and n, inclusive. You are required to find the k-th shortest path from a to b. You can assume $1 \leq k \leq 200$ and $a \neq b$.

The i-th edge is from the node x_i to y_i with the length d_i ($1 \leq i \leq m$). Both x_i and y_i are between 1 and n, inclusive. d_i is between 1 and 10000, inclusive. You can directly go from x_i to y_i, but not from y_i to x_i unless an edge from y_i to x_i is explicitly given. The edge connecting the same pair of nodes is unique, if any, that is, if $i \neq j$, it is never the case that x_i equals x_j and y_i equals y_j. Edges are not connecting a node to itself, that is, x_i never equals y_i. Thus the inequality $0 \leq m \leq n(n-1)$ holds.

Note that the given graph may be quite unrealistic as a road network. Both the cases $m = 0$ and $m = n(n-1)$ are included in the judges' data.

The last dataset is followed by a line containing five zeros (separated by a space).

Output

For each dataset in the input, one line should be output as specified below. An output line should not contain extra characters such as spaces.

If the number of distinct paths from a to b is less than k, the string `None` should be printed. Note that the first letter of `None` is in uppercase, while the other letters are in lowercase.

If the number of distinct paths from a to b is k or more, the node numbers visited in the k-th shortest path should be printed in the visited order, separated by a hyphen (minus sign). Note that a must be the first, and b must be the last in the printed line.

In this problem the term *shorter* (thus *shortest* also) has a special meaning. A path P is defined to be shorter than Q, if and only if one of the following conditions holds.

1. The length of P is less than the length of Q. The length of a path is defined to be the sum of lengths of edges on the path.

2. The length of P is equal to the length of Q, and P's sequence of node numbers comes earlier than Q's in the dictionary order. Let's specify the latter condition more precisely. Denote P's sequence of node numbers by $p_1, p_2, \ldots, p_s$, and Q's by $q_1, q_2, \ldots, q_t$. $p_1 = q_1 = a$ and $p_s = q_t = b$ should be observed. The sequence P comes earlier than Q in the dictionary order, if for some r ($1 \leq r \leq s$ and $r \leq t$), $p_1 = q_1, \ldots, p_{r-1} = q_{r-1}$, and $p_r < q_r$ (p_r is numerically smaller than q_r).

A path visiting the same node twice or more is not allowed.

Note: In the case of the first dataset, there are 16 paths from the node 1 to 5. They are ordered as follows (The number in parentheses is the length of the path).

1	(3)	1-2-3-5	9	(5)	1-2-3-4-5
2	(3)	1-2-5	10	(5)	1-2-4-3-5
3	(3)	1-3-5	11	(5)	1-2-4-5
4	(3)	1-4-3-5	12	(5)	1-3-4-5
5	(3)	1-4-5	13	(6)	1-3-2-5
6	(3)	1-5	14	(6)	1-3-4-2-5
7	(4)	1-4-2-3-5	15	(6)	1-4-3-2-5
8	(4)	1-4-2-5	16	(8)	1-3-2-4-5

Sample Input

```
5 20 10 1 5
1 2 1
1 3 2
1 4 1
1 5 3
2 1 1
2 3 1
2 4 2
2 5 2
3 1 1
3 2 2
3 4 1
3 5 1
4 1 1
4 2 1
4 3 1
4 5 2
5 1 1
5 2 1
5 3 1
5 4 1
4 6 1 1 4
2 4 2
1 3 2
1 2 1
1 4 3
2 3 1
3 4 1
3 3 5 1 3
1 2 1
2 3 1
1 3 1
0 0 0 0 0
```

Sample Output

```
1-2-4-3-5
1-2-3-4
None
```