INDIAN INSTITUTE OF TECHNOLOGY ROPAR
Lab on Arborescence

You can implement this in C/C++ (C++ preferred though)

Write a program for computing min cost arborescence.
Ensure your code is compatible on linux machines.

Your Code would be also considered for indentation, description of your idea and functions, efficiency (provided its correct).

Input Format

First line: Number of Test Cases T and then follows their description
For each test case,
first row indicate N M s (single space separated)
where
N is number of vertices in directed graph where vertices are labelled 1 to N (not 0 indexing)
M is the number of edges and
s is the index of source vertex
And then M rows mentioning u v w(u,v) (single space separated, where u and v are vertex ids and w is weight, Safely assume all edge weights are non-negative, else print -1 for the output)

Output Format

T rows corresponding to T test cases
Output row for each test case has 2N+1+1 entries (all single space separated)
where first entry is the total sum of min cost arborescence
and then N entries corresponding to vertices V1, V2, V3..., VN providing the distance of i-th vertex from the source vertex s. (Indicate -1 if unreachable)
and then Symbol #
and then N entries (corresponding to vertices V1, V2, V3..., VN) providing the label of the parent node through which one reaches i-th vertex (basically second last node in path from source vertex to that i-th vertex. (Indicate 0 if no parent i.e. for source vertex and -1 if vertex i not reachable)

Considering the following sample input and output :

```
===========INPUT===============
2
6 9 1
1 2 10
1 3 2
1 4 10
2 3 1
3 4 4
4 5 2
2 6 8
5 2 2
4 6 4
8 13 1
1 2 5
1 5 11
5 2 6
6 5 10
6 2 2
```

```
2 3 3
2 7 13
7 6 7
3 7 9
3 4 12
4 8 1
8 3 4
7 8 8
```

```
===========OUTPUT ==========
14 0 10 2 6 8 10 # 0 5 1 3 4 4
47 0 5 8 20 34 24 17 21 # 0 1 2 3 6 7 3 4
```

It will help us easily examine correctness of solution for random input graphs that we may use in test cases. Also it is important in cases when we have different min-cost arborescences for a same graph (possible if there are more than one similar weight edges)

Q: Which edge do we select when there are multiple direct edges between two points ---- the first one or the one with smaller weights?
Ans: Assume input graph has no multiple direct edges between 2 vertices. In case such a situation arise during processing, choose one with smallest weight.
In input u to v if edge weight is w1 and then again there is mentioned an edge from u to v but with different edge weight value w2, then chose only one value i.e. min (w1,w2)
As such in input test cases, we would try not to give you any such case

Q: What are we supposed to do if a vertex is unreachable from the source ----as according to the definition of arborescence (given in slides) it is possible to reach every vertex by at least one way? or say, Q: Can you tell if it is okay to assume that the test cases you will provide have the following property : Every node is reachable from the source node.
Ans: Find min cost (partial) arborescence for the set of vertices that are reachable from given input source vertex. Mention -1 (unreachable) for other vertices.
Output format remains same i.e. having 2N+1+1 entries. But presence of -1 in output would indicate some vertices were unreachable.

Q: According to the definition of the min-cost arborescence (given in slides) no negative edge weight, is there any possibility of having an edge of negative weight in our test cases?
Ans: Safely assume no negative edge weights. We won't give you graph with negative edge weight. If any input test case does contain any negative edge weight, just output single entry i.e. -1 corresponding to that test case

Q: While selecting the edges lexicographically (in case of multiple possibilities of min-cost arborescence), can I consider the index of the super-node such that out of the vertices inside the super-node which lead to some min-cost arborescence the selected one has a minimum index?
Ans: When same min value but different edges to choose from, choose one whose destination vertex label is less. If that also same, choose one that has lesser value of source vertex label.
It is recommended (however it is not mandatory) to give the supernode as index value which is same as lowest index amongst the vertices/nodes forming the cycle that led to creation of that supernode.
OR one may give label n+1 to first supernode, n+2 for second supernode...It depends on your approach partly.
We would accept any answer provided its correct. We understand there may be more than one min cost arborescences possible for same input graph and with same min cost value.