2. Maximum Spanning Tree

Given a weighted, undirected graph of N vertices and E edges satisfies the followings:

1) A path between any arbitrary vertices always exists. (G is a connected graph) Therefore, there will always be a spanning tree.
2. G is a normal graph so that there is no self loop and two paths between any pairs of vertices.

Write an algorithm that find the sum of weights of the maximum spanning tree of G.

[Constraints]
- $5 \le N \le 20{,}000$
- $5 \le E \le 40{,}000$

Note : You may get penalised if the run-time of your algorithm is significantly longer than others.

[File]
Write your implementations in “Solution2.java” file with “Solution2” class.
When you run the code with the following command, the output file “output2.txt” should be generated.

```
> javac Solution2.java −encoding UTF8 && java Solution2
```

[Input]
An input file “input2.txt” will be given, which as 10 test cases.
Each case consists of 2 lines; first has the number of vertices of G N, the number of edges of G E, and in the second line, E pieces of information about the edges as source, vertex, weight with space in each.

For example, “3 9 17 4 3 5 ...” means that there is an edge between vertex 3 and vertex 9 and its weight is 17, and there is another edge between vertex 4 and vertex 3 and its weight is 5. So there will be 3*E integers split by space.

[Output]

For each case, you should print the case number as #x where x is the index of the case. Then print the sum of the weight of the maximum spanning tree of G followed by space.

You must produce your results as “output2.txt”.

[Example]

Input (input2.txt)

```
5 10                                                                         ← 1st Case
4 1 3 1 5 1 2 1 1 5 3 5 2 3 6 3 4 6 4 2 6 4 5 2 3 1 5 2 5 1
7 12                                                                         ← 2nd Case
1 2 7 6 4 1 7 2 1 1 3 1 4 2 10 4 3 3 4 1 5 2 5 4 7 1 9 7 4 7 6 3 7 5 4 6
…
```

Output (output2.txt)

```
#1 22
#2 42
…
```