Strongest Friendship Group - 2022 December, Gold

Monday, August 7, 2023 1:47 PM

Farmer John has N cows ($2 \le N \le 10 \le 10^5$), conveniently labeled 1...N. There are M ($1 \le M \le 2 \cdot 10^5$) pairs of friends among these cows.
A group of cows is called a "friendship group" if every cow in the group is reachable from every other cow in the group via a chain of friendships that lies solely within the group (friendships connecting to cows outside the group have no impact). The "strength" of a friendship group is the minimum number of friends of any cow in the group within the group times the number of cows in the group (again, note that friendships connecting to cows outside the group do not count for this definition).
Please find the maximum strength over all friendship groups.

SAMPLE INPUT:

8 10
1 2
1 3
1 4
2 3
2 4
3 4
1 5
2 6
3 7
4 8

SAMPLE OUTPUT:

12

The maximum strength can be observed to be with the group of cows numbered 1,2,3,41,2,3,4. The minimum number of friends of any cow in this group within the group is 33, so the answer is $4 \cdot 3 = 12 4 \cdot 3 = 12$.

From <http://www.usaco.org/index.php?page=viewproblem2&cpid=1259>

Clearly we need to use a DSU to keep track of connected components. A few questions:

1. How are we going to build the DSU(s)?
2. Where do we start? What paradigm do we follow?
3. What criteria/observation do all the optimal solutions have?

Approaches that won't work

1. Sorting nodes by the number of outgoing edges
2. Just making a DSU
3. Excluding nodes one at a time
4. Including nodes one at a time
5. Arbitrarily picking a starting point, or really picking any starting point to build one DSU from
6. Adding edges in some order

Criteria for the answer
Answer is the smallest # of outgoing edges * number of nodes
As the number of nodes increases, the impact that the smallest # of outgoing edges has on the answer increases. Unfortunately we can't just sort by largest number of outgoing edges. How are we supposed to even pick a start?

If we don't pick a start, everywhere is. Suppose we have trees rooted at every node now. How do we get the optimal addition?
For group size of 2 only 2 can be the answer.
For size 3 the maximum is 6, with a minimum of 3. In all case adding one more node will be beneficial.
For size 4, we can have 4, 8, or 12.
For size 5, we can have 5, 10, 15, 20.
For size n, we can have n, 2n, 3n..., (n-1)n.

Attempting to make pairs of trees would just put us back in the impossible task of trying to pair the correct ones without knowing what's ahead. What if we worked backwards? Is a greedy sol possible? Likely not. Maybe?

If we work backwards, every edge removed we know is not in the final cc. But how do we find these edges? And how do we know when to stop?

Possible solution: keep removing the smallest arry node until the answer begins to decrease?

If we rebuild DSU every time, it will be too slow. Ideally we re-reverse when making DSU, but we only know the order when we go backwards and need DSU then. Ok idea: we first sort the edges in descending order, then add each component to the DSU, recording the largest CC as we go. Then to simulate removal, we simply go backwards on the list.

Another maybe solution:

We can try and calculate the highest group size per minimum-arry size. We have 2000 total such values, so we just build 2000 DSUs. This might actually work, I will go with this one. To get the size quickly, we can return the new size of the larger tree every time we union.