Problem Link :-

https://www.hackerearth.com/problem/algorithm/infinity-stones/

Solution:-

Decide a cutoff for the votes where cutoff means that everyone would be having votes less than cutoff and the winner would be having votes more than cutoff.

So lets say the cutoff is X. Then everyone should have less votes than cutoff so we would try to iterate over all the persons having votes greater than X,and would take there votes which are lowest in cost and would increase the votes of the winner.Then if the winner is still not having votes greater than X then we would try to make them X+1 and would take the minimum of all the votes which are left.

Implementation:-

```
#include <bits/stdc++.h>
using namespace std;
#define ll long long int

int main()
{
   ll n, m, i, j, k;
   cin >> n >> m;
   // n no. of students
   // m stones

   vector<vector<ll>> stones(n + 1);
   for (i = 1; i <= m; i++)
   {
      cin >> j >> k; // possession // cost
      stones[j].push_back(k);
   }

   // sorting the cost of stones each student has
   for (i = 1; i <= n; i++)
      sort(stones[i].begin(), stones[i].end());

   ll ans = LLONG_MAX;
   for (i = 0; i < m; i++)
   {
      ll cutoff = i;            // everyone is having less(or equal) stones than cutoff [except the first student]
      ll curr = stones[1].size(); // stones that student 1 has currently
      ll cost = 0;              // initial cost to have  curr number of stones

      vector<ll> left_stones;
      for (j = 2; j <= n; j++)
      {
         if (stones[j].size() > cutoff)
         {
            for (k = 0; k < stones[j].size() - cutoff; k++)
```

```
                cost += stones[j][k], curr++;
            for (; k < stones[j].size(); k++)
                left_stones.push_back(stones[j][k]);
        }
        else
        {
            for (k = 0; k < stones[j].size(); k++)
                left_stones.push_back(stones[j][k]);
        }
    }

    // If 1st is not having less stones than cutoff
    // than pick minimum cost stones from the left stones
    // so as to make his stones greater than cutoff
    j = 0;
    sort(left_stones.begin(), left_stones.end());
    while (curr <= cutoff && j < left_stones.size())
        cost += left_stones[j], j++, curr++;

    ans = min(ans, cost);
  }
  cout << ans << endl;
}
```