Codefoces Round 810, Div.2

https://codeforces.com/contest/1711

***Problem 1 (1711A) Perfect Permutation.***

<https://codeforces.com/contest/1711/problem/A>

***Code:***

#include <iostream>

#include <vector>

using namespace std;

void generatePermutation(int n) {

cout << n << " ";

for (int i = 1; i < n; ++i) {

cout << i << " ";

}

cout << endl;

}

int main() {

int t;

cin >> t;

for (int j = 0; j < t; ++j) {

int n;

cin >> n;

generatePermutation(n);

}

return 0;

}

***Idea:***

Since k+1 does not divide k, a permutation with weight equal to 1 is: [n,1,2,⋯,n−1].

***Algorithm complexity:***

O(n)

***Problem 2. (1711B) Party.***

https://codeforces.com/contest/1711/problem/B?locale=en

***Code:***

#include <iostream>

#include <climits>

using namespace std;

const int MAXN = 100010;

int x[MAXN], y[MAXN], a[MAXN], degree[MAXN];

void calc() {

int n, m;

cin >> n >> m;

for (int i = 1; i <= n; i++) {

degree[i] = 0;

cin >> a[i];

}

for (int i = 1; i <= m; i++) {

cin >> x[i] >> y[i];

degree[x[i]]++;

degree[y[i]]++;

}

int ans = INT\_MAX;

if (m % 2 == 0)

ans = 0;

for (int i = 1; i <= n; i++) {

if (degree[i] % 2 == 1)

ans = min(ans, a[i]);

}

for (int i = 1; i <= m; i++) {

if (degree[x[i]] % 2 == 0 && degree[y[i]] % 2 == 0)

ans = min(ans, a[x[i]] + a[y[i]]);

}

cout << ans << endl;

}

int main() {

int n = 1;

cin >> n;

for (int j = 1; j <= n; j++)

calc();

return 0;

}

***Idea:***

Let's consider the case where m is odd only, since if m is even the answer is 0.

Assume that you delete x vertices with even degrees and you vertices with odd degrees.

If y≥1, then only deleting one vertex with an odd degree would lead to a not worse answer, so you do not need to consider it except for (x,y)=(0,1).

If y=0 , then the parity of the edges at the end is determined only by the number of edges whose both endpoints are deleted. In particular, there must be at least two adjacent vertices deleted with even degrees. So you do not need to consider it except for (x,y)=(2,0) and they are neighbours.

Thus, an optimal solution either has (x,y)=(0,1) or (x,y)=(2,0) and the two vertices are adjacent.

One can iterate over all possible solutions with such a structure and take the optimal one.

***Algorithm complexity:***

O(n+m)