Alicecream

Alice has an array A of length N and a list of bad pairs (u_i, v_i). Her father has asked her to calculate to the maximum value of (count(x) + count(y)) * (x + y) over all pairs of x and y where,

1. x != y
2. x and y both exist in A
3. count(x) denotes the frequency of x in A
4. (x, y) is not a bad pair

Otherwise, he won't let her have ice cream for dinner. Not that if (u, v) is a bad pair, then so is (v, u). Alice desperately wants to have ice cream for dinner. However, she is facing difficulty solving this problem. So, she has asked for your help. Help Alice solve this problem.

Input

- ➔ The first line contains N ($2 <= N <= 2 \times 10^5$) and M ($0 <= M <= 2 \times 10^5$) - number of integers in the array and the number of bad pairs.
- ➔ The second line contains A_1, A_2, …, A_N ($1 <= A_i <= 10^9$) - elements of the array.
- ➔ The ith of the next M lines contain two integers u and v ($1 <= u < v <= 10^9$), which represents a bad pair. It is guaranteed that no pair occurs twice in the input. It is also guaranteed that count(u) > 0 and count(v) > 0.
- ➔ It is guaranteed that there always exists a pair (a, b) which is not bad and both a and b exist in A.

Output

Print a single integer, the answer to the problem.

Sample Input
6 1
6 3 6 7 3 3
3 6

Sample Output
40

C ++

```cpp
#include <bits/stdc++.h>
using namespace std;

#define ll long long

int main()
{
    ll N, M;
    cin >> N >> M;
    vector<ll> A(N);
    for (ll i = 0; i < N; i++) {
        cin >> A[i];
    }
    set<pair<ll, ll>> bad;
    for (ll i = 0; i < M; i++) {
        ll x, y;
        cin >> x >> y;
        bad.insert({x, y});
    }

    set<ll> ST;
    map<ll, ll> frequency;
    map<ll, bool> visited;
    map<ll, set<ll, greater<ll>>> mp;
    for (auto x : A) {
        frequency[x]++;
    }
    for (ll i = 0; i < N; i++) {
        if (!visited[A[i]]) {
            mp[frequency[A[i]]].insert(A[i]);
            visited[A[i]] = true;
        }
    }
    for (auto x : mp) {
        ST.insert(x.first);
    }
```

```cpp
    ll ans = INT_MIN;
    for (auto fx : ST) {
        for (auto x : mp[fx]) {
            for (auto fy : ST) {
                if (fy > fx) {
                    break;
                }

                bool flag = false;
                for (auto y : mp[fy]) {
                    if (x == y or bad.count({x, y}) or bad.count({y, x})) {
                        continue;
                    }
                    ans = max(ans, (fx + fy) * (x + y));
                    flag = true;
                    break;
                }
                if (flag) {
                    break;
                }
            }
        }
    }
    cout << ans << endl;
}
```