Laboratory work # 6

Student: *Zhanghao Chen*
Student ID: 21321205
Timus Name: *BennyChan*

Mail: 1824362950@qq.com

Problem # 1628 *White Streaks*

Screenshot from Timus:

10285692	19:55:45 18 May 2023	BennyChan	1628. White Streaks	G++ 9.2 x64	Accepted	0.203	4 024 KB

Explanation of algorithm:

1. Create two vectors of size m+1 and n+1, and then adds n+1 and m+1 to each element of $p[i]$ and $q[i]$ by a traversal loop to ensure that any element in the vector has a comparison object.

2. Add the input values x and y to the p and q vectors through another loop. Then I use two nested loops to iterate over each element in the p and q vector, using the "temp" variable to keep track of previous elements. If the difference between the current element and the "temp" variable is greater than 2, the "sum" variable is incremented by 1. If the difference is greater than 1, I use another loop to traverse the elements of the $p[t\_q]$ vector, where t_q is a specific value, to find the element corresponding to the current element. If the condition is met, the "sum" variable is incremented by 1.

3. Finally, I output the value of "sum".

Computational complexity of algorithm:

$$O(N \times \log_2 N)$$

Source code:

```
#include <iostream>
#include <vector>
#include <algorithm>

```

```cpp
using namespace std;

int main() {
    int m, n, k;
    cin >> m >> n >> k;
    int sum = 0;
    vector<vector<int>> p(m+1);
    vector<vector<int>> q(n+1);

    for (int i = 1; i <= m; i++) {
        p[i].push_back(n+1);
    }
    for (int i = 1; i <= n; i++) {
        q[i].push_back(m+1);
    }
    for (int i = 0; i < k; i++) {
        int x, y;
        cin >> x >> y;
        p[x].push_back(y);
        q[y].push_back(x);
    }

    for (int i = 1; i <= m; i++) {
        int temp = 0;
        sort(p[i].begin(), p[i].end());
        for (int j = 0; j < p[i].size(); j++) {
            if (p[i][j]-temp > 2) sum++;
            temp = p[i][j];
        }
    }
    for (int i = 1; i <= n; i++) {
        int temp = 0;
        sort(q[i].begin(), q[i].end());
        for (int j = 0; j < q[i].size(); j++) {
            if (q[i][j]-temp > 2) sum++;
            else if (q[i][j]-temp > 1) {
                int t_q = q[i][j]-1;
                temp = 0;
                for (int l = 0; l < p[t_q].size(); l++) {
                    if (p[t_q][l] > i) {
                        if (p[t_q][l]-temp <= 2) sum++;
                        break;
                    }
                    temp = p[t_q][l];
                }
            }
            temp = q[i][j];
        }
    }
    cout << sum << endl;
    return 0;
}
```

Problem # 1650 *Billionaires*

Screenshot from Timus:

ID	Date	Author	Problem	Language	Judgement result	Test #	Execution time	Memory used
10285971	07:27:41 19 May 2023	BennyChan	1650. Billionaires	G++ 9.2 x64	Accepted		0.875	18 392 KB

Explanation of algorithm:

1. Initialize some data structures, including pairs and maps.

2. Create a vector of pairs, load in the values of m and q, and enter another loop that keeps reading in day, name, and city, adding them to the vector. Then, we sort the vector and then enter a loop where for each m, we check the elements of the vector whose day is equal to the current m and update the map and set based on name and city.

3. Check the size of the set, and if the size is greater than 1, we remove the largest element and compare it with the next largest element. If the largest element has more money than the second largest element, the counter for that city is incremented by 1.

4. Reinsert the largest element into set and output the names of all cities whose counter is not 0.

Computational complexity of algorithm:

$$O((q+m)\times \log_2 N)$$

- q is the number of queries and m is the number of days.

Source code:

```cpp
#include <iostream>
#include <map>
#include <set>
#include <utility>
#include <vector>

using ll = long long;

int main() {
    int num;
    std::cin >> num;

    std::set<std::pair<ll, std::string>> cities;
    std::map<std::string, ll> moneys;
    std::map<std::string, ll> bankirs;
    std::map<std::string, std::string> bankirsCity;

    for (int i = 0; i < num; i++) {
        std::string name, city;
        ll money;
        std::cin >> name >> city >> money;
        bankirs[name] = money;
        bankirsCity[name] = city;
        moneys[city] += money;
    }

    for (const auto&[city, money] : moneys) {
```

```
        cities.insert(std::make_pair(money, city));
    }

    std::vector<std::pair<int, std::pair<std::string, std::string>>> v;
    int m, q;
    std::cin >> m >> q;

    for (int i = 0; i < q; i++) {
        int day;
        std::string name, city;
        std::cin >> day >> name >> city;
        v.push_back(std::make_pair(day, std::make_pair(name, city)));
    }

    std::sort(v.begin(), v.end());

    std::map<std::string, int> ans;

    int curr = 0;
    for (int i = 0; i <= m - 1; i++) {
        while (curr < v.size() && v[curr].first == i) {
            std::string bankir, oldCity, newCity;
            ll sum;
            bankir = v[curr].second.first;
            oldCity = bankirsCity[bankir];
            newCity = v[curr].second.second;
            sum = bankirs[bankir];

            if (cities.count(std::make_pair(moneys[oldCity], oldCity))) {
                cities.erase(std::make_pair(moneys[oldCity], oldCity));
            }
            if (cities.count(std::make_pair(moneys[newCity], newCity))) {
                cities.erase(std::make_pair(moneys[newCity], newCity));
            }
            bankirsCity[bankir] = newCity;
            moneys[oldCity] -= sum;
            moneys[newCity] += sum;
            cities.insert(std::make_pair(moneys[oldCity], oldCity));
            cities.insert(std::make_pair(moneys[newCity], newCity));

            curr++;
        }

        if (cities.size() > 1) {
            auto a = *cities.rbegin();
            cities.erase(a);
            auto b = *cities.rbegin();
            if (a.first > b.first) {
                ans[a.second]++;
            }
            cities.insert(a);
        } else {
            ans[(*cities.rbegin()).second]++;
        }
    }

    for (const auto&[city, count] : ans) {
        if (count == 0) {
            continue;
        }
        std::cout << city << " " << count << std::endl;
    }
}
```