

- 2nd LGE Code Jam 2019 - Problem D

2nd LGE Code Jam 2019 - Problem D (English)

2nd LGE Code Jam 2019 - Problem E

2nd LGE Code Jam 2019 - Problem E (English)

2nd LGE Code Jam 2019 - Problem F

2nd LGE Code Jam 2019 - Problem F (English)

- 코드잼 준비하기
- 2019년 활동
- 2018년 활동
- 2016년 활동
- 2015년 활동
- 2014년 활동
- 2013년 활동
- 2012년 활동
- 코딩전문가와 함께하는 코딩 도장
- 코딩 전문가가 참여하는 커뮤니티 모임
- 코딩전문가의 동영상 강의
- SW 개발 관련 Q&A
- 정기 모임
- History

Space tools

Pages / ... / 2019 Online Code Jam 2 - Editorial

EditSave for laterWatchShare

2nd LGE Code Jam 2019 - Problem E (English)

Created by 박선현 sunhyun.park, last modified by 배성호 baver.bae on 2019/07/12

Subtask 1)

This can be considered as a maximum matching problem in bipartite graphs.

Since $N, M \leq 100$, we have $V \leq 200$, $E \leq 10000$, So well-known $O(VE)$ algorithms like Edmonds-Karp, Ford-Fulkerson is enough.

Subtask 2)

Bipartite matching doesn't work here, since $N, M \leq 100000$ gives $O(VE) = 10^{15}$ and $O(\sqrt{V}E) > O(10^{12})$.

Then how about considering greedy solution? Consider the following approach:

- Store information of passengers with balanced binary search tree with **(distance, number of people)** pairs:
You can use C++ std::map, Java TreeMap
- Sort cabs first in **ascending order by its endpoint**. For tiebreakers, sort in **descending order by its starting point** (only when endpoints equals)
- Make decisions from the leftmost (which having the **minimum endpoint**) cab.
To do so, first search a passenger **nearest from its start point** who can **actually take this cab**.
You **should use binary search** to reduce time complexity. (C++ - std::map::lower_bound, Java - TreeMap.ceilingEntry)
When the passenger-cab pair is matched, each values of entries is decremented. Don't forget erase the entry when the number of people with a typical distance becomes zero.
- If we iterate over all cabs and apply 3., the number of maximum matching is the answer!!

Were we just lucky? Or is this really right?

Actually this is the optimal solution.

(Of course you can get the similar correct solution by sorting everything in the reversed order. The important thing is you should use sorting + greedy idea)

If we sorted **passengers distances' in ascending order**, we have to make decisions from **the cab which has the least capability for taking passengers** to get the optimal solution.

For this, at first, sort it in ascending order by its endpoint, and for tiebreakers, sort in descending order by its start point, which is the only way to "restrict" possible options of selecting passengers.

To check if this cab can actually take the passenger, pick the passenger nearest from the start point of cab = leftmost passenger this cab can take.

Therefore, if some people belongs to multiple ranges of cab, **we are always giving the priority to the cab which has the least capability**, so our solution is globally optimal.

Complexity : Building TreeMap $O(N \log N)$ + Sorting Cabs $O(M \log M)$ + Finding passengers $O(M \log N)$ = $O(M \log M + (M + N) \log N)$

JavaC++

C++ Solution: $O(N \log N)$

```
1 #include <iostream>
2 #include <vector>
3 #include <numeric>
4 #include <algorithm>
5 #include <set>
6 using namespace std;
7 int match(int n, int m, vector<int> &x, vector<int> &y, vector<int> &z) {
8     multiset<int> ms(x.begin(), x.end());
9     vector<int> idx(m);
10    iota(idx.begin(), idx.end(), 0);
11    sort(idx.begin(), idx.end(), [&](int i, int j) {
12        if (z[i] != z[j])
13            return z[i] < z[j];
14        return y[i] < y[j];
15    });
16    int ans = 0;
17    for (auto i : idx) {
18        auto it = ms.lower_bound(y[i]);
19        if (it == ms.end() || *it > z[i])
20            continue;
21        ++ans;
22        ms.erase(it);
23    }
24    return ans;
25 }
26 int main(void) {
27     cout.sync_with_stdio(false);
28     cin.tie(nullptr);
29     int nTests;
30     cin >> nTests;
31     while (nTests--) {
32         int n, m;
33         cin >> n >> m;
34         vector<int> x(n);
35         for (int i = 0; i < n; ++i)
36             cin >> x[i];
37         vector<int> y(m), z(m);
38         for (int i = 0; i < m; ++i)
39             cin >> y[i] >> z[i];
40         cout << match(n, m, x, y, z) << '\n';
41     }
42     return 0;
43 }
```

Like 배성호 baver.bae likes this

No labels



Write a comment...