Usually, results of competitions are based on the scores of participants. However, we are planning a change for the next year of IPSC. During the registration each team will be able to enter a single positive integer – their preferred place in the ranklist. We would take all these preferences into account, and at the end of the competition we will simply announce a ranklist that would please all of you.

But waitHow would that ranklist look like if it won’t be possible to satisfy all the requests?

Suppose that we already have a ranklist. For each team, compute the distance between their preferred place and their place in the ranklist. The sum of these distances will be called the *badness* of this ranklist.

Given team names and their preferred placements find one ranklist with the minimal possible badness.

**Input**

The first line of the input file contains an integer T,T≤20T,T≤20, specifying the number of test cases. Each test case is preceded by a blank line.

Each test case looks as follows: The first line contains an integer N(N≤100000)N(N≤100000)– the number of teams participating in the competition. Each of the next NN lines contains a team name (a string of letters and numbers of length at most 20) and its preferred place (an integer between 1 and NN, inclusive). No two team names will be equal.

**Output**

For each of the test cases output a single line with a single integer – the badness of the best ranklist for the given teams.

|  |  |
| --- | --- |
| **Sample Input 1** | **Sample Output 1** |
| 2  7  noobz 1  llamas 2  Winn3rz 2  5thwheel 1  NotoricCoders 5  StrangeCase 7  WhoKnows 7  3  ThreeHeadedMonkey 1  MoscowSUx13 1  NeedForSuccess 1 |  |

这个算是瞎贪心的错误吧……

一开始想想应该挺简单

然后就没有推理原理直接贪心了一发

错了。

后面就开始自己瞎凑数据……

嗯果不其然的也错了

当时想的是：

推理第一步：把出现过的名次都分配一个到最后名次上面去

这样造成的不满度是0.

第二步：一个空的位置，最合理的排布应该是向左取到一个已经出现且需求不止一个的名次，向右取到一个已经出现且需求不止一个的名次，然后取比较近的那个，赋值给这个空位。

但是这个推理没有考虑到，有可能会出现溢出的情况

即，如果一个数字的需求足够大到覆盖两个数字之间一半的空间，那这个想法就会出现错误，因为每个队伍都应该被分到相应的值上，放在后面的话很明显就存在了更优的情况。

所以贪心还是贪心，但是这个贪心是：

第一步：把需求过的名次都给一次

第二步：从第一名开始，把多余的数字挨个排到空位上，这个不存在更优解，因为任何两个非固定数位的对换只能导致相同或者更大的结果。

#include <iostream>

#include<sstream>

#include<cstdio>

#include<iomanip>

#include<string>

#include<vector>

#include<stack>

#include<queue>

#include<algorithm>

#include<map>

#include<cmath>

#include<climits>

#include<cstring>

#define hash 10007

#define MAX 900005

#define ll long long

using namespace std;

int main()

{

int T;

cin >> T;

while (T--)

{

vector<int>R;

vector<int>FS;

int num;

cin >> num;

ll total = 0;

R.resize(num,0);

FS.resize(num, 0);

for (int i = 0; i < num; i++)

{

int t;

scanf("%\*s %d", &t);

R[t-1]++;

}

for (int i = 0; i < R.size(); i++)

{

if (R[i])

{

R[i]--;

FS[i] = 1;

}

}

int now=0;

while (FS[now])

{

now++;

}

for (int i = 0; i < R.size(); i++)

{

while (R[i])

{

total += abs(i - now);

FS[now++] = 1;

while (now<num&&FS[now])

now++;

R[i]--;

}

}

cout << total << endl;

}

}