1056. Mice and Rice (25)

时间限制

30 ms

内存限制

65536 kB

代码长度限制

16000 B

判题程序

Standard

作者

CHEN, Yue

*Mice and Rice* is the name of a programming contest in which each programmer must write a piece of code to control the movements of a mouse in a given map. The goal of each mouse is to eat as much rice as possible in order to become a FatMouse.

First the playing order is randomly decided for NP programmers. Then every NG programmers are grouped in a match. The fattest mouse in a group wins and enters the next turn. All the losers in this turn are ranked the same. Every NG winners are then grouped in the next match until a final winner is determined.

For the sake of simplicity, assume that the weight of each mouse is fixed once the programmer submits his/her code. Given the weights of all the mice and the initial playing order, you are supposed to output the ranks for the programmers.

**Input Specification:**

Each input file contains one test case. For each case, the first line contains 2 positive integers: NP and NG (<= 1000), the number of programmers and the maximum number of mice in a group, respectively. If there are less than NG mice at the end of the player's list, then all the mice left will be put into the last group. The second line contains NP distinct non-negative numbers Wi (i=0,...NP-1) where each Wi is the weight of the i-th mouse respectively. The third line gives the initial playing order which is a permutation of 0,...NP-1 (assume that the programmers are numbered from 0 to NP-1). All the numbers in a line are separated by a space.

**Output Specification:**

For each test case, print the final ranks in a line. The i-th number is the rank of the i-th programmer, and all the numbers must be separated by a space, with no extra space at the end of the line.

**Sample Input:**

11 3

25 18 0 46 37 3 19 22 57 56 10

6 0 8 7 10 5 9 1 4 2 3

**Sample Output:**

5 5 5 2 5 5 5 3 1 3 5

#include<iostream>

#include<string>

#include<algorithm>

#include<queue>

#include<vector>

#include<sstream>

#include<stack>

using namespace std;

struct ranking {

int weight;

int rank;

int finalrank;

};

int comp(ranking\*a, ranking \*b)

{

if (a->weight > b->weight)

return 1;

else

return 0;

}

int comp2(ranking\*a, ranking\*b)

{

return a->rank > b -> rank;

}

vector<ranking\*>all\_num;

vector<ranking\*>list\_a;

vector<ranking\*>list\_b;

int main()

{

int num, group;

cin >> num >> group;

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

{

ranking\* temp;

temp = new ranking;

cin >> temp->weight;

all\_num.push\_back(temp);

}

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

{

int temp;

cin >> temp;

list\_a.push\_back(all\_num[temp]);

}

int flag = 0,cnt=0;

while (list\_a.size()!=1)

{

while (list\_a.size())

{

if (list\_a.size() > group)

{

sort(list\_a.begin(), list\_a.begin() + group, comp);

list\_b.push\_back(list\_a.front());

list\_a.erase(list\_a.begin());

for (int i = 1; i < group; i++)

{

list\_a.front()->rank = cnt;

list\_a.erase(list\_a.begin());

}

}

else

{

sort(list\_a.begin(), list\_a.end(), comp);

list\_b.push\_back(list\_a.front());

list\_a.erase(list\_a.begin());

for (int i = 0; i < list\_a.size(); i++)

{

list\_a.front()->rank = cnt;

list\_a.erase(list\_a.begin());

}

}

}

list\_a = list\_b;

list\_b = vector<ranking\*>();

cnt ++;

}

list\_a.front()->rank = cnt;

vector<ranking\*>rank(all\_num);

sort(rank.begin(), rank.end(), comp2);

rank[0]->finalrank = 1;

for (int i = 1; i < num; i++)

{

if (rank[i]->rank == rank[i - 1]->rank)

rank[i]->finalrank = rank[i - 1]->finalrank;

else

rank[i]->finalrank = i + 1;

}

cout << all\_num[0]->finalrank;

for (int i = 1; i < all\_num.size(); i++)

{

cout<<" " << all\_num[i]->finalrank;

}

}