Университет ИТМО

Факультет программной инженерии и компьютерной техники

**Лабораторная работа №1**

по «Алгоритмам и структурам данных»

Timus

Выполнил:

Студент группы P32092

Голиков А.С.

Преподаватели:

Косяков М.С.

Тараканов Д.С.

Санкт-Петербург

2023

Задача №1155 «Дуоны»

Пояснение к примененному алгоритму:

Заметим, что если дуоны находятся на четном расстоянии на нашем кубе (расстояние считаем по ребрам), то их никак нельзя спарить (провести серию созданий и аннигиляций, чтобы в итоге изменить количества в наших вершинах на 1 при неизменных остальных).

Тогда очевидно, вершины A, C, F, H можно онулять только вершинами B, D, E, G, и если изначально баланса среди этих вершин нет, то куб никак нельзя онулить целиком, итого слудеющее неравенство - критерий возможности онулить куб.

A + C + F + H != B + D + E + G

Если все хорошо и куб онуляем, давайте придумаем, как мы будем это делать. На самом деле, порядок не важен. Предположим, что в нашем порядке что-то пошло не так, и мы не можем аннигилировать очередную пару.

Но это значит, что баланс вершинок куба нарушен, а мы каждым действием равномерно его меняем, чего не могло произойти при корректном всем остальном - противоречие.

Асимптотика - мы делаем операций столько, чему равна сумма вводимых чисел, но так как они все ограниченны 100 сверху, количество операций не превышает константу - O(1).

Код решения:

#include <iostream>  
#include <vector>  
  
using namespace std;  
  
int main() {  
 vector<int> cube(8);  
  
 for (int i = 0; i < 8; ++i) {  
 cin >> cube[i];  
 }  
 if (cube[0] + cube[2] + cube[5] + cube[7] != cube[1] + cube[3] + cube[4] + cube[6]) {  
 cout << "IMPOSSIBLE";  
 return 0;  
 }  
 for (int i = 0; i < 8; ++i) {  
 int this\_i = i;  
 int next\_i = (i + 1) % 4 + 4 \* (i > 3);  
 int prev\_i = (i - 1 + 4) % 4 + 4 \* (i > 3);  
 int vert\_i = (i + 4) % 8;  
 int diag\_i = ((i + 2) % 4 + 4 \* (i > 3) + 4) % 8;  
 int next\_vert\_i = ((i + 1) % 4 + 4 \* (i > 3) + 4) % 8;  
  
 char this\_c = (char) ('A' + this\_i);  
 char next\_c = (char) ('A' + next\_i);  
 char prev\_c = (char) ('A' + prev\_i);  
 char vert\_c = (char) ('A' + vert\_i);  
 char diag\_c = (char) ('A' + diag\_i);  
 char next\_vert\_c = (char) ('A' + next\_vert\_i);  
  
  
 //kill\_with\_next  
 while (cube[this\_i] && cube[next\_i]) {  
 --cube[this\_i];  
 --cube[next\_i];  
 cout << this\_c << next\_c << "-\n";  
 }  
  
 //kill\_with\_prev  
 while (cube[this\_i] && cube[prev\_i]) {  
 --cube[this\_i];  
 --cube[prev\_i];  
 cout << this\_c << prev\_c << "-\n";  
 }  
  
 //kill\_with\_diag  
 while (cube[this\_i] && cube[diag\_i]) {  
 if (!cube[vert\_i] || !cube[next\_vert\_i]) {  
 cout << vert\_c << next\_vert\_c << "+\n";  
 ++cube[vert\_i];  
 ++cube[next\_vert\_i];  
 }  
 cout << this\_c << vert\_c << "-\n";  
 cout << next\_vert\_c << diag\_c << "-\n";  
 --cube[this\_i];  
 --cube[vert\_i];  
 --cube[next\_vert\_i];  
 --cube[diag\_i];  
 }  
  
 //kill\_with\_vert  
 while (cube[this\_i] && cube[vert\_i]) {  
 --cube[this\_i];  
 --cube[vert\_i];  
 cout << this\_c << vert\_c << "-\n";  
 }  
 }  
  
 return 0;  
}

Задача №2025 «Стенка на стенку»

Пояснение к примененному алгоритму:

Нам нужно максимизировать количество ребер в не более, чем k-дольном графе.  
Пусть у нас ровно k долей (если меньше - просто добавим доли с 0 вершин). Всего ребер может быть n(n-1)/2, в каждой доле мы теряем еще по k\_i(k\_i-1)/2 ребер  
Итого ребер у нас (n(n-1) - sum(k^2) (суммма кваратов мощностей всех долей) + sum(k) (= n))/2 =  
= (n^2 - sum(k^2))/2  
  
Для максимизации ребер нужно минимизировать сумму квадратов мощностей долей.  
Пусть выбрали такие оптимальные {k\_i}, что сумма квадратов минимальна  
Пусть у нас есть k\_i и k\_j такие, что k\_i - k\_j >= 2, но тогда (k\_i)^2 + (k\_j)^2 > (k\_i - 1)^2 + (k\_j + 1)^2 = (k\_i)^2 + (k\_j)^2 - 2(k\_i - k\_j) + 2  
Противоречие - значит все мощности отличаются не более чем на 1. А такое мы очень легко можем придумать - пусть n = qk + r, тогда имеем r долей размера (q + 1) и (k - r) долей размера q  
Пример есть - осталось посчитать ребра  
Итого их (n \* (n - 1) - r \* q \* (q + 1) - (k - r) \* q \* (q - 1)) / 2  
  
Асимптотика - O(T)

Код решения:

#include <iostream>  
  
using namespace std;  
  
int main() {  
 ios::sync\_with\_stdio(false);  
 int T, n, k;  
 cin >> T;  
 for (int i = 0; i < T; ++i) {  
 cin >> n >> k;  
 int r = n % k;  
 int q = n / k;  
 cout << (n \* (n - 1) - r \* q \* (q + 1) - (k - r) \* q \* (q - 1)) / 2 << "\n";  
 }  
 return 0;  
}