

**Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский государственный технический университет имени Н.Э. Баумана**

**(национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)**

ФАКУЛЬТЕТ *Робототехники и комплексной автоматизации*

КАФЕДРА *Системы автоматизированного проектирования (РК-6)*

# ОТЧЕТ О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ

Студент Журавлев Н.В.

Группа РК6-52Б

Тип задания Лабораторная работа №4

Студент

**Журавлев Н.В.**

*подпись, дата фамилия, и.о.*

Преподаватель

**Князева С.В.**

*подпись, дата фамилия, и.о.*

*Москва, 2021*

**Вариант 5**

**Задание**

Перечислить все разбиения заданного целого числа n > 0 на слагаемые, где максимальное из них равно заданной величине m. Для генерации разбиений следует применить алгоритм Эрлиха с мульти записью слагаемых каждого разбиения в порядке убывания своих величин слева направо и знаком + между ними.

**Теоретическая часть**

Не менее гибкий, но гораздо более совершенный в вычислительном отношении алгоритм Эрлиха порождает все разбиения любого заданного числа в словарном порядке, используя мультимножественную запись слагаемых, где учитывается их кратность. Это дает возможность записать разбиение любого числа n на m ≤ n мультислагаемых в следующем формате:

n = + … + + … +.

В этой записи ki обозначает кратность слагаемого pi, а все слагаемые перечислены в порядке убывания их величин:

> … > > … > .

Например, разбиение числа 7 на 2 мультислагаемых 2 и 1 с кратностями 2 и 3, соответственно, выглядит так:

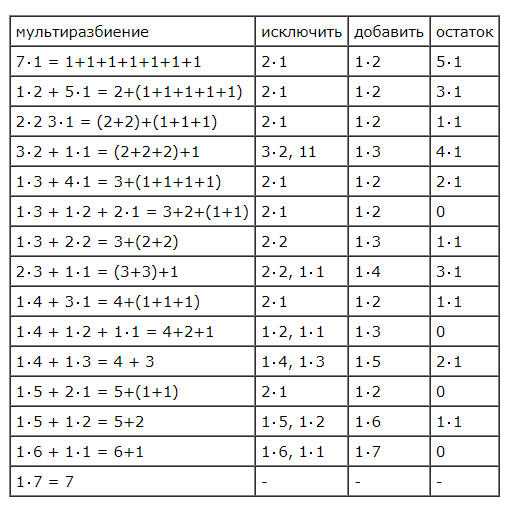
7 = 2∙2 + 3∙1 = (2 + 2) + (1 + 1 + 1).

Мультимножественная запись слагаемых позволяет построить алгоритм генерации разбиений целых чисел, который переходит от текущего разбиения к следующему в словарном порядке, рассматривая только самое правое мультислагаемое без анализа предыдущих частей разбиения как было в алгоритме Гинденбурга. Способ конструирования очередного разбиения зависит от кратности последнего мультислагаемого текущего разбиения. При этом требуется рассмотреть следующие два случая:

> 1 и = 1.

В первом случае нужно исключить из разбиения значение (2∙), чтобы получить возможность добавить (еще) одно мультислагаемое со значением (+1). Во втором случае нужно добавить еще одно мультислагаемое со значением (+1), исключив два последних мультислагаемых. В любом случае, то, что остается от уменьшаемых мультислагаемых должно быть превращено в соответствующее число единиц для последнего мультислагаемого следующего разбиения. Этот процесс должен начинать с разбиения, где все части равны единице, и завершаться разбиением из одного мультислагаемого. Выполнение шагов алгоритма Эрлиха иллюстрирует следующая таблица, где показана генерация разбиений числа 7:

**Алгоритм**



**Код программы**

#include <iostream>

#include <vector>

#define ERR\_INPUT\_N 1

#define ERR\_INPUT\_M 2

using namespace std;

void optimize(vector<pair<int, int>> &values){

int i = 0;

int check = 0;

for (auto x : values) {

int k = 0;

for (auto j : values) {

if (j.second == x.second && i != k) {

values[i].first += j.first;

values.erase(values.begin() + k);

check = 1;

break;

}

k++;

}

if (check == 1) {

break;

}

i++;

}

if (check == 0) {

return;

}

optimize(values);

}

void print(vector<pair<int, int>> &values) {

cout << endl;

unsigned long last\_element = values.size() - 1;

for (auto x : values) {

cout << x.first << "\*" << x.second;

if (x.first != values[last\_element].first || x.second != values[last\_element].second) {

cout << " + ";

}

}

cout << endl;

}

int calc\_sum(vector<pair<int, int>> &values) {

int sum = 0;

for (auto x : values){

sum = sum + (x.first \* x.second);

}

return sum;

}

void k\_more\_1(vector<pair<int, int>> &values) {

unsigned long last\_element = values.size() - 1;

int except = 2 \* values[last\_element].second;

int add = 1 + values[last\_element].second;

pair<int, int> remain = make\_pair(values[last\_element].first \* values[last\_element].second - add, 1);

// cout << "Исключить: " << except << "\*1" << endl << "Добавить: 1\*"

// << add << endl << "Остаток: " << remain.first << "\*1" << endl;

values[last\_element] = make\_pair(1, add);

values.insert(values.cend(), remain);

if (values[last\_element + 1].first == 0) {

values.pop\_back();

}

}

void k\_equally\_1(vector<pair<int, int>>& values) {

unsigned long len\_vector = values.size() - 1;

auto add\_new\_value = make\_pair(1, values[len\_vector - 1].second + 1);

vector<pair<int, int>> delete\_value;

// cout << "Добавить: " << add\_new\_value.first << "\*" << add\_new\_value.second << endl;

delete\_value.push\_back(values[len\_vector]);

delete\_value.push\_back(values[len\_vector - 1]);

values.pop\_back();

values.pop\_back();

// cout << "Удаление: ";

// cout << delete\_value[0].first << "\*" << delete\_value[0].second << ", ";

// cout << delete\_value[1].first << "\*" << delete\_value[1].second << endl;

values.push\_back(add\_new\_value);

int sum\_delete = calc\_sum(delete\_value);

// cout << "Остаток: ";

if (sum\_delete - add\_new\_value.second != 0) {

values.emplace\_back(make\_pair(sum\_delete - add\_new\_value.second, 1));

// cout << values[len\_vector].first << "\*1" << endl;

} else {

// cout << "0" << endl;

}

}

int main() {

int n, m;

cout << "Введите число n: ";

cin >> n;

if (n <= 0) {

cout << "Число должно быть больше 0" << endl;

return ERR\_INPUT\_N;

}

cout << "Введите число m: ";

cin >> m;

cout << endl;

if (m <= 0) {

cout << "Число должно быть больше 0" << endl;

return ERR\_INPUT\_M;

}

vector<pair<int, int>> multi\_values;

multi\_values.emplace\_back(n, 1);

while (multi\_values[0].second != n) {

if (multi\_values[multi\_values.size() - 1].first == 1) {

k\_equally\_1(multi\_values);

} else {

k\_more\_1(multi\_values);

}

optimize(multi\_values);

int max = 0;

for (auto x : multi\_values){

if (x.second > max) {

max = x.second;

}

}

if (max == m) {

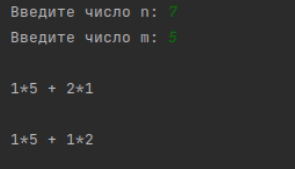
print(multi\_values);

}

}

return 0;

}**Результат работы программы**



**Литература**

1. Курс лекций «Методы комбинаторных вычислений» - Волосатова Т.М., Родионов С.В.
2. bigor.bmstu.ru