|  |  |
| --- | --- |
|  | **Министерство образования и науки 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 Сидоров Михаил Михайлович

Группа РК6-56Б

Тип задания Лабораторная работа

Тема лабораторной работы Разбиение целых чисел

Вариант 20

Студент **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Сидоров М.М.**

*подпись, дата фамилия, и.о.*

Преподаватель **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Волосатова Т.М.**

*подпись, дата фамилия, и.о.*

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Москва, 2024 г.*

# Оглавление

[Оглавление 2](#_Toc90495022)

[Задание на лабораторную работу 3](#_Toc90495023)

[Описание комбинаторного алгоритма 3](#_Toc90495024)

[Листинг программы 6](#_Toc90495025)

[Результаты работы программы 9](#_Toc90495026)

[Источники 10](#_Toc90495027)

# Задание на лабораторную работу

Перечислить все разбиения заданного целого числа n>0 на слагаемые, где максимальное из них не входит в заданный интервал [L;U]. Для генерации разбиений следует применить **алгоритм Гинденбурга**, а слагаемые каждого разбиения должны быть записаны в порядке **не убывания** своих величин слева направо и разделены знаком +.

# Описание комбинаторного алгоритма

Цель алгоритма: Алгоритм Гинденбурга генерирует все разбиения заданного натурального числа n на слагаемые.

Разбиения строятся в порядке:

1. Увеличения количества слагаемых.

2. Лексикографического порядка (слагаемые упорядочены по неубыванию).

## Шаг 1: Начальная инициализация

Начать с разбиения, состоящего из одного слагаемого, равного числу n:

p₁ = n

Количество слагаемых на данном этапе m = 1.

## Шаг 2: Генерация разбиения с текущим числом слагаемых m

Просмотреть текущее разбиение справа налево и найти наибольшее слагаемое pᵢ, которое можно увеличить на 1, соблюдая условие:

pₘ - pᵢ > 2

Увеличить pᵢ на 1 и присвоить это значение всем слагаемым справа от pᵢ до предпоследнего pₘ₋₁:

pⱼ = pᵢ + 1 для всех j = i, ..., m-1

Пересчитать последнее слагаемое pₘ как:

pₘ = n - ∑(p₁ + p₂ + ... + pₘ₋₁)

## Шаг 3: Проверка завершения серии разбиений на m слагаемых

Если достигнуто лексикографически наибольшее разбиение (разница между наибольшим и наименьшим слагаемым не превышает 1):

pₘ - p₁ ≤ 1

Переходим к построению следующей серии с количеством слагаемых m + 1.

## Шаг 4: Переход к следующему числу слагаемых m + 1

Формируем новое начальное разбиение с m + 1 слагаемыми:

p₁ = p₂ = ... = pₘ = 1, pₘ₊₁ = n - m

## Шаг 5: Завершение алгоритма

Алгоритм завершает работу, когда все слагаемые равны 1:

n = p₁ + p₂ + ... + pₙ, где pᵢ = 1 для всех i.

## Пример разбиения числа n = 5

Шаг 1: Начальное разбиение: p₁ = 5.

Шаг 2: Разбиения на два слагаемых: 1 + 4, 2 + 3.

Шаг 3: Разбиения на три слагаемых: 1 + 1 + 3, 1 + 2 + 2.

Шаг 4: Разбиение на четыре слагаемых: 1 + 1 + 1 + 2.

Шаг 5: Разбиение на пять слагаемых: 1 + 1 + 1 + 1 + 1.

Таким образом, алгоритм Гинденбурга позволяет генерировать все разбиения числа в порядке увеличения количества слагаемых и лексикографического порядка.

## Дополнительное условие из задания

В задании требуется, чтобы максимальное из слагаемых разбиения не входило в заданный интервал [L; U]. Для этого на каждом шаге алгоритма необходимо добавить проверку максимального слагаемого pₘ и исключать разбиения, в которых:

L ≤ pₘ ≤ U

Если максимальное слагаемое pₘ не удовлетворяет этому условию, разбиение исключается из результата.

## Пример с дополнительным условием

Рассмотрим пример с числом n = 5 и интервалом [2; 3]:

1. Разбиения на два слагаемых:  
 1 + 4 (допустимо), 2 + 3 (исключается, так как 3 ∈ [2; 3]).

2. Разбиения на три слагаемых:  
 1 + 1 + 3 (исключается), 1 + 2 + 2 (исключается).

3. Разбиение на четыре слагаемых:  
 1 + 1 + 1 + 2 (исключается).

4. Разбиение на пять слагаемых:  
 1 + 1 + 1 + 1 + 1 (допустимо).

# Листинг программы

#include <iostream>

#include <vector>

#include <string>

using namespace std;

bool check(vector<int> part, int size, int l, int u) {

if (l > part[size - 1] || part[size - 1] > u)

return 1;

return 0;

}

void print\_num(int n, vector<int> part, int len) {

cout << n << " = ";

for (int i = 0; i < len - 1; i++)

cout << part[i] << " + ";

cout << part[len - 1] << endl;

}

vector<int> init(int n, int iter) {

vector<int> a;

a.reserve(iter);

for (int i = 0; i < iter - 1; i++)

a.push\_back(1);

a.push\_back(n - iter + 1);

return a;

}

int find(vector<int> part, int len) {

for (int i = len - 1; i >= 0; i--)

if (part[i] + 2 <= part[len - 1])

return i;

return -1;

}

int last(vector<int> part, int n) {

int a = n;

for (int i = 0; i < part.size() - 1; i++)

a -= part[i];

return a;

}

void gindenburg(int n, int l, int u) {

vector<int> partition;

int a;

for (int i = 1; i <= n; i++) {

partition = init(n, i);

if (check(partition, i, l, u))

print\_num(n, partition, i);

while ((a = find(partition, i)) >= 0) {

partition[a]++;

for (int j = a + 1; j < i - 1; j++)

partition[j] = partition[a];

partition[i - 1] = last(partition, n);

if (check(partition, i, l, u)) print\_num(n, partition, i);

}

}

}

int main(int argc, char\*\* argv) {

if (argc != 4) {

cout << "Incorrect number of arguments. Use n l u";

return 1;

}

int n = stoi(argv[1]);

int l = stoi(argv[2]);

int u = stoi(argv[3]);

if (n <= 0 || l <= 0 || u <= 0) {

cout << "Invalid arguments. Must be > 0";

return 1;

}

if (l > u) {

cout << "Invalid l. Must be <= u";

return 1;

}

gindenburg(n, l, u);

}

# Результаты работы программы

Input: 5 1 2

Output:

5 = 5

5 = 1 + 4

5 = 2 + 3

5 = 1 + 1 + 3

# Источники

1. http://bigor.bmstu.ru/?cnt/?doc=Comby/int\_numb\_div.mod/?bck=Comby/comtz.the#T1121473007