МИНОБРНАУКИ РФ

ФГБОУ ВО «ИжГТУ имени М.Т. Калашникова»

Кафедра «Программное обеспечение»

Отчет

по лабораторной работе №3 по программированию

на тему «Разработка структур данных и алгоритмов на основе векторов и списков. Итераторы для контейнеров последовательностей»

Выполнил студент группы Б03-191-3: Р.А. Гумметов

Принял: В.Г. Тарасов

Ижевск 2019

1. Условие задачи

Задается общее количество элементов N и нужное нам K. Пронумеруем все сочетания из N элементов по K (лексикографически меньшая имеет меньший номер, наименьшая имеет номер 0). Ваша задача – выдать номер заданного сочетания. Нужно выдать на стандартное устройство вывода одно число – номер сочетания.

2. Постановка задачи

Со стандартного устройства ввода в первой строке вводится 2 числа – N и K, 0 ≤ K ≤ N ≤ 50, 1 ≤ N. Во второй строке вводятся элементы выборки через пробел.

3. Идея решение задачи

Создадим вектор number\_soch для хранения сочетания, у которого ищем номер. Будем хранить цифры заданного сочетания на соответствующих этим цифрам индексах, остальные места в векторе заполним нулями. Строим треугольник Паскаля с n + 1 строками. С помощью свойств треугольника Паскаля, в цикле while ищем номер сочетания. В векторе range храним диапазон, в котором находится искомый номер. Создаем счетчик l, начальное значение которого равно 1 и итератор it, который указывает на первый элемент вектора number\_soch и двигаемся по массиву number\_soch. На каждом шаге диапазон, в котором находится искомый номер сужается. Цикл while будет работать до тех пор, пока оба элемента массива range не будут равны. Эти элементы есть искомый номер сочетания.

4. Код программы

#include <iostream>

#include <vector>

#include <string>

#include <math.h>

using namespace std;

int main()

{

int n, k;

cin >> n >> k;

vector <int> number\_soch(n, 0);

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

{

int num;

cin >> num;

number\_soch[num - 1] = num;

}

vector <vector <unsigned long long>> triangle(n + 1);

triangle[0].resize(1, 1);

triangle[1].resize(2, 1);

for (int i = 2; i <= n; i++)

{

triangle[i].resize(i + 1, 0);

for (int j = 0; j < i + 1; j++)

{

if (j == 0)

triangle[i][j] = 1;

else if (j == triangle[i].size() - 1)

triangle[i][j] = 1;

else

{

triangle[i][j] = triangle[i - 1][j - 1] + triangle[i - 1][j];

}

}

}

unsigned long long points[2] = { n, k };

unsigned long long range[2] = { 1, triangle[n][k] };

vector<int>::iterator it = number\_soch.begin();

int l = 1;

while (range[0] != range[1])

{

if (l == \*it)

{

points[0]--;

points[1]--;

range[1] = range[0] + triangle[points[0]][points[1]] - 1;

}

else

{

range[0] = range[0] + triangle[points[0] - 1][points[1] - 1];

points[0]--;

}

l++;

it++;

}

cout << range[0] - 1;

}

5. Вывод

Таким образом написана программа, находящая номер заданного сочетания из n по k.

Временная сложность программы составляет O().

Пространственная сложность программы составляет O().