# ЛАБОРАТОРНАЯ РАБОТА №3

# ЭКСПЕРИМЕНТАЛЬНОЕ ОПРЕДЕЛЕНИЕ КОЛИЧЕСТВА

# ЭЛЕМЕНТАРНЫХ ОПЕРАЦИЙ ЯЗЫКА ВЫСОКОГО УРОВНЯ В

# ПРОГРАММНОЙ РЕАЛИЗАЦИИ АЛГОРИТМА

Цель работы

Экспериментальные исследования теоретически полученной функции трудоёмкости для алгоритма точного решения задачи о сумме методом полного перебора. Получение практических навыков пооперационного анализа и построения функции трудоёмкости.

Задания

1. Ознакомиться с программой, реализующей алгоритм решения задачи о сумме;
2. Дополнить функцию TaskSum конструкциями подсчёта элементарных операций в соответствии с изложенной в методических указаниях методикой;
3. Для заданных по варианту значений (Таблица 1) использовать два способа построения массива: заполнения вручную и заполнения с помощью генератора псевдослучайных чисел. При использовании генератора обеспечить смену псевдослучайных последовательностей. Провести порядка 50 экспериментов;

|  |  |  |  |
| --- | --- | --- | --- |
| Вариант | Размерность вектора случайных чисел | Максимальное  случайное число в векторе | Значение суммы (V) |
| 12 | 11 | 100 | 58 |

Таблица 1 – Вариант задания

1. Представить результаты экспериментов, проведённых с использованием генератора псевдослучайных чисел, в виде таблицы. Сделать выводы по работе;

Текст программы

#include <iostream>

#include <ctime>

#include <cmath>

using namespace std;

const int N = 11; // длина массива

const int V = 58; // искомая сумма

int vector[N]; // исходный массив

int counter[N]; // вспомогательный массив

bool flag; // флаг нахождения искомых слагаемых

int n; // число слагаемых в сумме

int ops; // количество элементарных операций

double W = 212873; // эл. операции для худшего случая

int sum\_ops = 0; // сумма эл. операций

int sum\_n = 0; // общее кол-во элементов, обр. сумму

double sum\_perc = 0; // сумма процентов от худшего случая

// Поиск элементом массива, образующих сумму V

void TaskSum(void)

{

int MaxN = int(pow(2, N) - 1); // количество вариантов

flag = false;

for (int i = 0; i < N; i++) counter[i] = 0;

ops = 7 + N \* 5;

// перебор вариантов по двоичному счётчику counter

for (int cnt = 1; cnt <= MaxN; cnt++)

{

int sum = 0; // сумма элементов соответствия

n = 0; // количество слагаемых

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

{

sum = sum + counter[i] \* vector[i];

n += counter[i];

}

ops += 4 + N \* 8;

if (sum == V) // проверка результата суммирования

{

flag = true;

ops++;

return;

}

n = 0;

// увеличение двоичного счётчика на единицу

int j = N - 1;

while ((counter[j] == 1) && (j != 0))

{

counter[j] = 0;

j--;

ops += 7;

}

counter[j] = 1;

ops += 5;

}

}

// Заполнение генератором псевдослучайных чисел

void fillVectorRand()

{

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

vector[i] = rand() % 101;

return;

}

// Ввод массива вручную

void fillVector()

{

cout << "Enter the array: ";

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

{

cin >> vector[i];

if (vector[i]) cout << "The int read was: " << vector[i] << endl;

}

}

// Решение задачи о поиске элементов массива, дающих в сумме число V

int main()

{

srand(time(0));

for (int l = 0; l < 50; l++)

{

cout << endl;

fillVectorRand(); // заполнение массива

//fillVector();

TaskSum(); // поиск слагаемых

if (flag)

{

cout << "Result found:\n";

for (int k = 0; k < N; k++) cout << counter[k] << " ";

cout << endl;

for (int k = 0; k < N; k++) cout << vector[k] << " ";

cout << endl;

}

else cout << "NO elements giving the sum\n";

cout << "n - " << n << endl;

sum\_n += n;

cout << "Ops - " << ops << endl;

sum\_ops += ops;

cout << "Per - " << ops / W \* 100.0 << endl;

sum\_perc += ops / W \* 100.0;

}

cout << endl << sum\_n / 50.0 << endl; // среднее кол-во слагаемых

cout << sum\_ops / 50.0 << endl; // среднее кол-во эл. операций

cout << sum\_perc / 50.0 << endl; // средний процент от худшего

return 0;

}

Ход работы

Программа из методических указаний была переработана таким образом, чтобы выводить результаты для 50 различных значений элементов массива и подсчитывать для них среднее значение количества слагаемых в сумме, количества использованных элементарных операций и его средний процент от количества операций для наихудшего случая, полученного теоретическим путём.

Полученные значения были перенесены в таблицу (Таблица 2).

|  |  |  |  |
| --- | --- | --- | --- |
| № эксперимента | Экспериментальные данные | | Сравнение с теорией |
| Количество элементов, образующих сумму | Использованное количество элементарных операций | Процент от теоретически наихудшего случая |
| 1 | 0 | 212873 | 100 |
| 2 | 2 | 6901 | 3.24 |
| 3 | 0 | 212873 | 100 |
| 4 | 2 | 3677 | 1.73 |
| 5 | 3 | 20034 | 9.83 |
| 6 | 1 | 13460 | 6.32 |
| 7 | 3 | 63366 | 29.767 |
| 8 | 0 | 212873 | 100 |
| 9 | 0 | 212873 | 100 |
| 10 | 0 | 212873 | 100 |
| 11 | 0 | 212873 | 100 |
| 12 | 0 | 212873 | 100 |
| 13 | 0 | 212873 | 100 |
| 14 | 0 | 212873 | 100 |
| 15 | 0 | 212873 | 100 |
| 16 | 0 | 212873 | 100 |
| 17 | 0 | 212873 | 100 |
| 18 | 0 | 212873 | 100 |
| 19 | 2 | 26869 | 12.62 |
| 20 | 0 | 212873 | 100 |
| 21 | 0 | 212873 | 100 |
| 22 | 1 | 564 | 0.26 |
| 23 | 4 | 6159 | 2.89 |
| 24 | 0 | 212873 | 100 |
| 25 | 0 | 212873 | 100 |
| 26 | 0 | 212873 | 100 |
| 27 | 2 | 3573 | 1.67 |
| 28 | 2 | 1389 | 0.65 |
| 29 | 0 | 212873 | 100 |
| 30 | 1 | 26772 | 12.57 |
| 31 | 0 | 212873 | 100 |
| 32 | 0 | 212873 | 100 |
| 33 | 2 | 109965 | 51.66 |
| 34 | 3 | 5334 | 2.51 |
| 35 | 2 | 159885 | 75.11 |
| 36 | 2 | 1389 | 0.65 |
| 37 | 1 | 252 | 0.11 |
| 38 | 0 | 212873 | 100 |
| 39 | 0 | 212873 | 100 |
| 40 | 1 | 1812 | 0.85 |
| 41 | 3 | 93318 | 43.83 |
| 42 | 0 | 212873 | 100 |
| 43 | 0 | 212873 | 100 |
| 44 | 3 | 56814 | 26.68 |
| 45 | 1 | 356 | 0.17 |
| 46 | 2 | 7005 | 3.29 |
| 47 | 1 | 1812 | 0.85 |
| 48 | 2 | 765 | 0.36 |
| 49 | 0 | 212873 | 100 |
| 50 | 2 | 27597 | 12.96 |
| Среднее по проведённым экспериментам | 0.96 | 123493 | 58.01 |

Таблица 2 – Результаты экспериментов

Вывод

В ходе работы была исследована функция решения задачи о сумме путём полного перебора с помощью двоичного счётчика. Для массива длиной 11, максимального элемента 100 и суммы 58 были получены следующие результаты: в худшем случае функция производит 212873 элементарные операции, а в среднем функция производит 58.01% операций от данного числа.

Были получены навыки анализа алгоритмов и построения их функций трудоёмкости путём пооперационного анализа.