**Министерство образования и науки Украины**
**Севастопольский национальный технический университет**

# МАТЕМАТИЧЕСКИЙ АНАЛИЗ АЛГОРИТМОВ

**Экспериментальное определение количества элементарных операций языка высокого уровня в программной реализации алгоритма**

Методические указания
к выполнению лабораторной работы
для студентов, обучающихся по направлению
**6.050101“Информационные управляющие системы и технологии”**
очной и заочной форм обучения

**Севастополь**
**2014**

УДК 510.5

**МАТЕМАТИЧЕСКИЙ АНАЛИЗ АЛГОРИТМОВ:** методические / Сост. Ю. В. Коваленко, Е. Н. Заикина. — Севастополь: Изд-во СевНТУ, 2014— 14 с.

Целью методических указаний является экспериментальная проверка теоретически полученной функции трудоёмкости для алгоритма точного решения задачи о сумме методом полного перебора. Получение практических навыков расстановки счетчика операций в программе на языке высокого уровня

Методические указания составлены в соответствии с требованиями программы дисциплины «Теория алгоритмов и математическая логика» для студентов направления 6.050101и утверждены на заседании кафедры Информационных систем, протокол № 1 от 29 августа 2014 года.

Допущено учебно-методическим центром СевНТУ в качестве методических указаний.

Рецензент: Скороход Б.А., доктор технических наук, проф. кафедры ТК.

# СОДЕРЖАНИЕ

# 1 ЦЕЛЬ РАБОТЫ

Экспериментальная проверка теоретически полученной функции трудоемкости для алгоритма точного решения задачи о сумме методом полного перебора. Получение практических навыков расстановки счетчика операций в программе на языке высокого уровня.

# 2 ТЕОРЕТИЧЕСКИЙ РАЗДЕЛ

## 2.1 Понятие элементарных операций в формальной системе языка высокого уровня

Будем понимать под **элементарными** операциями языка высокого уровня (на примере языка C++) те, которые могут быть представлены в виде **элементарных конструкций** данного языка (но не обязательно в виде одной машинной команды). Так, за одну элементарную операцию можно считать следующие:

| | | |
|---|---|---|
| 1) | операцию присваивания | a←b |
| 2) | операцию индексации массива | a[i] |
| 3) | арифметические операции | *,/,-,+ |
| 4) | операции сравнения | a < b |
| 5) | логические операции | or,and,not |

Неявно в операцию сравнения входит машинная команда перехода в конструкции if-then-else.

Цикл не является элементарной операцией, т.к. может быть представлен в виде:

```
for i<=1 to N                    i<=1  ←——
                                   тело
      тело            ⇔          i<=i+1
end;                             if i ≤ N——→
```

Таким образом, конструкция цикла требует $1+3*N$ элементарных операций.

В качестве примера построения функции трудоемкости и расстановки элементарных операций рассмотрим задачу суммирования двумерного массива на псевдоязыке:

| **SumM(A,N;Sum);** | **элементарных операций в строке** |
|---|---|
| Sum←0 | 1 |
| for i←1 to N | 1 + 3*N |
| for j←1 to N | 1+3*N |
| Sum←Sum+A[i,j]; | 1(←)+1(+)+1(i)+1(j) |
| end for j | |
| end for i | |
| Return (Sum); | |
| End; | |

Таким образом, $f_A(N) = 1+1+N*(3+1+N*(3+4)) = 7N^2+4*N+2 = \Theta(N^2)$.

### 2.2 Формулировка задачи о сумме

Словесно задача о сумме формулируется как задача нахождения таких чисел из данной совокупности, которые в сумме дают заданное число.

В терминах языка высокого уровня задача формулируется как задача определения таких элементов исходного массива из $N$ чисел, которые в сумме дают число $V$ (задача относится к классу *NPC*).

**Дано**: Массив $S[i]$, $i=1, N$ и число $V$.

**Требуется**: определить $S_j$: $\sum S_j = V$

Получим **асимптотическую** оценку сложности решения данной задачи при использовании прямого перебора вариантов:

Поскольку исходный массив содержит $N$ чисел, то проверке на $V$ подлежат варианты:

- V содержит 1 слагаемое $\Rightarrow C_N^1 = N$;
- V содержит 2 слагаемых $\Rightarrow C_N^2 = (N*(N-1))/(1*2)$;
- V содержит 3 слагаемых $\Rightarrow C_N^3 = (N*(N-1)*(N-2))/(1*2*3)$;
- и т.д. до N слагаемых.

Т.е. необходимо рассмотреть сумму всех возможных **сочетаний** элементов массива из $N$ чисел.

Любое подмножество из $k$ элементов множества, содержащего $n$ элементов, называется **сочетанием** из $n$ элементов по $k$.

Число всех различных сочетаний из $n$ элементов по $k$ равно

$$C_n^k = \frac{n!}{k!(n-k)!}$$

Для всех действительных чисел $a$, $b$, отличных от нуля, и для всех натуральных чисел $n$ справедливо следующее соотношение

$$(a+b)^n = \sum_{k=0}^{n} C_n^k a^{n-k} b^k = C_n^0 a^n b^0 + C_n^1 a^{n-1} b^1 + ... + C_n^n a^0 b^n \ .$$

Если в формуле заменить *b* на *–b*, то получим

$$(a-b)^n = \sum_{k=0}^{n} (-1)^k C_n^k a^{n-k} b^k \ .$$

**Биномиальные коэффициенты** формулы $C_n^k$ составляют в **треугольнике Паскаля** строку с номером *n*.

| *n* | $C_n^k$ |
|---|---|
| *0* | *1* |
| *1* | *1 1* |
| *2* | *1 2 1* |
| *3* | *1 3 3 1* |
| *4* | *1 4 6 4 1* |
| *5* | *1 5 10 10 5 1* |
| *6* | *1 6 15 20 15 6 1* |
| *7* | *1 7 21 35 35 21 7 1* |
| … | ..... |

Каждый коэффициент образуется путем сложения двух стоящих над ним (слева и справа). Крайние значения известны для любого *n*: $C_n^0 = C_n^n = 1$. В строке с номером n слева направо стоят значения $C_n^0$, $C_n^1$, $C_n^2$, …, $C_n^n$.

Свойства биномиальных коэффициентов. Для целых $n \geq 0$, $k \geq 0$ справедливо свойство симметрии:

$$C_n^k = C_n^{n-k}$$

При *a = b = 1* получаем

$$\sum_{k=0}^{n} C_n^k = 2^n$$

При *a =1, b = -1* получаем

$$\sum_{k=0}^{n} (-1)^k C_n^k = 0$$

Поскольку сумма биномиальных коэффициентов равна $(1+1)^n=\sum c_n^k=2^N$ и для каждого варианта необходимо выполнить суммирование, то оценка сложности алгоритма решения задачи о сумме имеет вид:

$$f_A(N,V) = O(N*2^N).$$

### 2.3 Точное решение задачи о сумме

Определим вспомогательный массив Counter, хранящий текущее сочетание исходных чисел, подлежащее проверке на V

Вспомогательный массив Cnt[j] → "0"-$S_j$ не входит в V; → "1"-$S_j$ входит в V

Решение получено, если $V=\sum_{j=1}^{N} S[j]*Cnt[j]$

Условия существования решения имеют вид:

$$\text{Min}(S_i) \le V \le \sum S_i$$

Могут быть предложены следующие две реализации полного перебора:

1.Перебор по $C_N^K$ $\quad \sum_{k=1}^{N} C_N^K=2^N-1$ ;

2.Перебор по двоичному счётчику, реализованному в массиве Cnt:

00...01

00...10

Вторая идея алгоритмически более проста, кроме того, для ее реализации можно применить битовые операции. Тогда задача сведется к перебору маски от одного до $2^n$, а разряд с номером j будет показывать j-ый элемент массива cnt. Более простой случай, если он будет равен единице, тогда нужно добавить j-ый элемент в сумму и не добавлять в противном случае. Однако можно реализовать программу, создав массив cnt и просто увеличивая его на единицу.

Таким образом, программа поиска суммы на Turbo Pascal имеет вид:

```
Program Test;
var
   vector: array[1..100] of integer;
   n, s: integer;

procedure create_array(n: integer);
          var i: integer;
```

```
begin
     randomize;
     for i := 1 to n do
         vector[i] := random(100);
     for i := 1 to n do
         write(vector[i], ' ');
     writeln;
end;

function task_sum(n, sum: integer): boolean;
var
   mask: integer;
   j, s: integer;
   fl: boolean;
begin
     fl := false;
     for mask := 1 to (1 shl n) do
     begin
          s := 0;
          for j := 1 to n do
              if (mask and (1 shl (j-1) ) <> 0) then
                 s := s + vector[j];
          if s = sum then
          begin
               fl := true;
               break;
          end;
     end;
     task_sum := fl;
end;

begin
writeln('Count elements');
read(n);

create_array(n);

writeln('Summa');
read(s);

writeln(task_sum(n, s));
end.
```

На C# это будет выглядеть так:

| Размерность вектора случайных чисел | Максимальное случайное число в векторе | Значение суммы (V) |
|---|---|---|
| 15 | 60 | 26 |

```
using System;

namespace laba3
{
    class Program
    {
        static protected uint[] init_mas(uint N, uint maxval)
        {
            uint[] a = new uint[N];
            Random rnd = new Random();
            for (int i = 0; i < N; i++)
            {
                a[i] = (uint)rnd.Next(0, (int)maxval);
                Console.Out.Write(a[i] + " ");
            }
            return a;
        }

        static protected bool task(ref uint[] mas, uint v,ref uint fn)
        {
            uint s;
            fn++;
            for (uint mask = 1; mask < (1 << mas.Length); mask++)
            {
                s = 0;
                fn += 6;
                for (int j = 0; j < mas.Length; j++)
                {
                    fn += 7;
                    if ((mask & (1 << j)) != 0)
                    {
                        s += mas[j];
                        fn += 4;
                    }
                    if (s == v)
                    {
                        fn++;
                        return true;
                    }
                }
            }
            return false;
        }

        static protected uint min(ref uint[] mas)
        {
            uint m = mas[0];
            for (int i = 0; i < mas.Length; i++)
            {
                if (m > mas[i])
                    m = mas[i];
```

```
            }
            return m;
        }

        static protected uint summ(ref uint[] mas)
        {
            uint m = 0;
            for (int i = 0; i < mas.Length; i++)
                m += mas[i];
            return m;
        }

        static void Main(string[] args)
        {
            uint vector_length = 0, maxnum = 0, sum = 0;
            uint[] mas;
            Console.Write("Размерность вектора случайных чисел: ");
            vector_length = UInt32.Parse(Console.ReadLine());
            Console.Write("Максимальное случайное число: ");
            maxnum = UInt32.Parse(Console.ReadLine());
            Console.Write("Значение суммы: ");
            sum = UInt32.Parse(Console.ReadLine());
            Console.Write("Массив случайных чисел? (input y): ");
            if (Console.ReadLine() == "y")
            {
                Console.Write("Числа массива: ");
                mas = init_mas(vector_length, maxnum);
                Console.WriteLine();
            }
            else
            {
                Console.WriteLine("Введите {0} элементов массива (че-
рез пробел): ",vector_length);
                String [] vrm = Console.ReadLine().Split(new char[]{'
'});
                mas = new uint[vector_length];
                for (int i = 0; i < vector_length; i++)
                    mas[i] = UInt32.Parse(vrm[i]);
            }
            if (sum >= min(ref mas) && sum <= summ(ref mas))
            {
                uint fn = 0;
                Console.WriteLine("Есть ли такие числа?: {0}",
task(ref mas, sum, ref fn));
                Console.WriteLine("f(n) расчетное = {0}", (16 + 8 *
vector_length) * Math.Pow(2, vector_length) - 3 * vector_length - 12);
                Console.WriteLine("f(n) эксп. = {0}", fn);
            }
            else
              Console.WriteLine("Решение не существует!");
            Console.In.ReadLine();
```

```
        }
    }
}
```

### 2.4 Функция трудоемкости процедуры TaskSum

а) В лучшем случае $f\check{}(S,N,V)=\Theta(N)$, если V=S[N];

б) В худшем случае $f\hat{}(S,N,V)= \Theta (N*2^N)$

Существуют более эффективные алгоритмы решения этой задачи, а также ее частных случаев, которые в данной работе рассмотрены не будут.

## 3 ХОД РАБОТЫ

1. Изучить теоретический раздел настоящих методических указаний и повторить соответствующий лекционный материал.
2. Ознакомиться с программой, реализующей алгоритм точного решения задачи о сумме – пункт 2.3. Составить структурную схему алгоритма точного решения задачи о сумме.
3. Включить в функцию TaskSum строки счетчика элементарных операций в соответствии с принятой методикой.
4. Для заданных значений по варианту задания заполнить таблицу (см. Приложение А). Рассмотреть два варианта массива – 1) заполненный вручную; 2) заполненный с помощью генератора случайных чисел (см. лабораторную работу №2).
5. Сделать выводы по работе, оформить отчет, подготовить ответы на контрольные вопросы.

## 4 ВАРИАНТЫ ЗАДАНИЙ

Таблица 1 – Варианты заданий для выполнения лабораторной работы

| Вариант | Размерность вектора случайных чисел | Максимальное случайное число в векторе | Значение суммы (V) |
|---|---|---|---|
| 1 | 10 | 100 | 95 |
| 2 | 11 | 80 | 46 |
| 3 | 12 | 70 | 124 |
| 4 | 13 | 90 | 151 |
| 5 | 9 | 25 | 35 |
| 6 | 8 | 35 | 134 |
| 7 | 10 | 40 | 224 |
| 8 | 16 | 70 | 26 |
| 9 | 14 | 50 | 45 |
| 10 | 13 | 45 | 310 |
| 11 | 12 | 55 | 350 |
| 12 | 11 | 100 | 514 |
| 13 | 9 | 65 | 324 |
| 14 | 7 | 85 | 124 |
| 15 | 9 | 75 | 321 |
| 16 | 8 | 100 | 420 |
| 17 | 12 | 90 | 456 |
| 18 | 11 | 50 | 324 |
| 19 | 10 | 40 | 452 |
| 20 | 14 | 30 | 895 |

## 5 СОДЕРЖАНИЕ ОТЧЁТА

1. Цель работы.
2. Вариант задания.
3. Тексты программы, реализующих расчеты по соответствующему варианту, описания функций, используемых для генерации наборов исходных данных, структурные схемы основных функций, используемых в программе.
4. Результаты работы программы.
5. Развернутый вывод по работе.

## 6 КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Определение трудоемкости алгоритма.
2. Обозначения лучшего, худшего, среднего случая в анализе трудоемкости алгоритма.
3. Классификация алгоритмов по виду функции трудоемкости.
4. Перечень элементарных операций языка высокого уровня.
5. Формулировка задачи о сумме.
6. Основные комбинаторные формулы, использующиеся в задаче о сумме.
7. Каким образом получить индексы элементов, входящие в множество для получения суммы.

## 7 БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Т. Кормен, Ч. Лейзерсон, Р. Ривест Алгоритмы: построение и анализ. М.: МЦИМО, 2001.-960 с.,263 ил.
2. Гулд Х., Тобочник Я. Компьютерное моделирование в физике: в 2-х частях. Часть 2: Пер.с англ. - М.: Мир, 1990. – 400 с., ил.
3. Ульянов М.В., Шептунов М.В. Математическая логика и теория алгоритмов, часть 2: Теория алгоритмов. – М.: МГАПИ, 2003. – 80 с.

## ПРИЛОЖЕНИЕ А – Таблица результатов работы программы

**N =**
**V =**

| N эксперимента | $2^n$ | $f(n) = 8*n*2^n + 16*2^n - 3*n - 12$ | $f(n)$ по данным программного подсчета |
|---|---|---|---|
| | | | |