ЛАБОРАТОРНАЯ РАБОТА № 3.
“Экспериментальное определение количества элементарных операций языка высокого уровня в программной реализации алгоритма”

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 С++) 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

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

Если в формуле (1.5) 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*.

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 $n\ge 0$, $k\ge 0$ справедливо свойство симметрии:

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

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
…	…………………………………………………………

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

“0”-S_j не входит в V

Вспомогательный массив Cnt[j]

“1”-S_j входит в V

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

Условия существования решения имеют вид: Min (S_i) $\le V \le \sum S_i$

Могут быть предложены следующие две реализации полного перебора:

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

2.Перебор по двоичному счётчику реализованному в массиве Cnt:

00...01

00...10

Вторая идея алгоритмически более проста и сводится к задаче о увеличении счётчика в массиве Cnt на «1».

при 00...0111 увеличение на «1» приводит к сбросу всех правых «1»

при 00...1000 увеличение на «1» приводит к переустановке «0» в «1»

Таким образом процедура перебора на псевдоязыке имеет вид:

```
#include <conio.h>
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <iomanip.h>
```

```
#define SIZE ;

int vector[4];
int counter[4];
int N=4, V=5;
int flag;
FILE *stream;

void TaskSum(void){
  int i,j,sum, cnt, MaxN;
  MaxN = int(pow(2,N)-1);

  flag = 0; i=0;
  while(i<N){ counter[i]=0;  i++; }//end while(i<N)
  cnt = 1;
  do{
    sum = 0;
    i=0;
    while(i<N){
      sum = sum+counter[i]*vector[i];
      i++; }//end while (i<N)
    if (sum== V) {
      flag=1;
      return;
    }//end if
    j=N-1;
    while((counter[j]==1)&&(j!=0)){
      counter[j]=0;
      j = j-1;
    }//end while counter[j]==1
    counter[j]=1;
    cnt = cnt + 1;
  }//end do
  while(cnt<=MaxN);
}//end TaskSum

void main(){
      int       i,min,cnt, power, sum,j, temp, c;
      double  result;
      for(i=0; i<N; i++){
         if (fscanf(stdin, "%d", &temp))
           printf("The integer read was: %i\n",temp);
         vector[i]=temp;
      }
      fclose(stream);
```

```
    flag=0;
    TaskSum();
    if (flag==1){
     cout<<"OK\n";
     for(int k = 0; k<N; k++) printf("%d%s",counter[k]," ");
     printf("%s\n"," ");
     for(k = 0; k<N; k++) printf("%d%s",vector[k]," ");
     }//end if
    else cout<<"NO elements giving the sum\n";
}//end main
```

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 $f\hat{}(S,N,V)= \Theta (N*2^N)$, если решения вообще нет

$$f\hat{}_A(N) =8*N*2^N+16*2^N-3*N-12,$$

что согласуется с асимптотической оценкой.

3. ХОД РАБОТЫ

1. Изучить теоретический раздел настоящих методических указаний и повторить соответствующий лекционный материал.
2. Ознакомится с программой, реализующей алгоритм точного решения задачи о сумме – Приложение 1. Составить блок-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 2). 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

Вариант	Размерность вектора случайных чисел	Максимальное случайное число в векторе	Значение суммы (V)
1	10	100	10
2	11	80	15
3	12	70	20
4	13	90	25
5	9	25	60
6	8	35	55
7	10	40	50
8	15	60	45
9	14	50	40
10	13	45	30
11	12	55	35
12	11	100	58
13	9	65	48
14	7	85	38
15	9	75	68
16	8	100	17
17	12	90	29
18	11	50	37
19	10	40	41
20	14	30	43

6. 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.

7. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Определение трудоемокости алгоритма.
2. Обозначения лучшего, худшего, среднего случая в анализе трудоемкости алгоритма.
3. Классификация алгоритмов по виду функции трудоемокости.
4. Перечень элементарных операций языка высокого уровня.
5. Формулировка задачи о сумме.
6. Основные комбинаторные формулы, использующиеся в задаче о сумме.