Сравнение временных интервалов выполнения алгоритмов на разных по количеству элементов последовательностях

В.И. Рыбас, email: rwi@mail.ru

МБОУ лицей №4 г. Россоши Россошанского муниципального района

Аннотация. *Рассмотрено решение задачи на обработку последовательности элементов переборным (неэффективным по количеству операций и по памяти) и эффективным алгоритмами на языках программирования Pascal, Python и Java. Приведены сравнительные характеристики временных затрат при решении с небольшим (до 100) и большим (до 100000) количеством элементов последовательности.*

Ключевые слова: *время выполнения программы, количество исходных данных в последовательности, эффективность алгоритма.*

Введение

ЕГЭ по информатике и ИКТ всегда считался одним из самых сложных, но и одним из наиболее востребованных экзаменов среди выпускников. Такая ситуация связана прежде всего с запросами в качестве обязательного экзамена по информатике среди перспективных направлений подготовки ВУЗов. Самым важным изменением в ЕГЭ по информатике и ИКТ в 2021 году является компьютерная форма проведения экзамена.

1. Факторы, влияющие на выбор алгоритмов при решении задания №27 ЕГЭ по информатике

Одним из самых сложных заданий ЕГЭ по информатике и ИКТ является задание №27, решением которого должна быть программа, написанная на одном из языков программирования, позволяющая обработать последовательность элементов записанных в файле данных. Компьютерная форма проведения экзамена предполагает в качестве ответа одно или несколько чисел, являющихся результатом такой обработки. При выборе алгоритма решения задачи необходимо задуматься о различных факторах:

- выбор языка программирования;
- количество элементов последовательности в файле данных;
- временные затраты при выполнении алгоритма.

Если рассмотреть условие 27 задания в официальном КИМе, представленном на сайте ФИПИ, то одним из немаловажных факторов является дополнение в конце текста условия. Приведем дословно это дополнение – «Предупреждение: для обработки файла B не следует

использовать переборный алгоритм, вычисляющий сумму для всех возможных вариантов, поскольку написанная по такому алгоритму программа будет выполняться слишком долго». В связи с этим возникает несколько вопросов:

- насколько долго;
- на каком языке программирования дольше.

2. Сравнение временных интервалов выполнения алгоритмов на разных по количеству элементов последовательностях

Для проведения эксперимента рассмотрим стандартную задачу, решаемую двумя разными алгоритмами с использованием последовательностей расположенных в текстовых файлах.

Условие задачи: В файле данных находится последовательность положительных чисел. В первой строке файла положительное число N, определяющее количество чисел в последовательности. В остальных N строках содержатся N положительных чисел последовательности. Определить максимальную сумму пары чисел последовательности.

Файл A содержит последовательность из 60 чисел, файл B содержит 60000 чисел.

Будем решать задачу с использованием двух самых популярных среди выпускников языков программирования Pascal, Python и Java.

Решим задачу двумя алгоритмами – неэффективным (переборным) и эффективным (однопроходным).

Рассмотрим неэффективный алгоритм решения по памяти, связанный с организацией хранения последовательности данных в виде одномерного массива и добавим неэффективность решения по количеству операций с перебором всех существующих вариантов и увеличению количества операций пропорционально увеличению количества элементов.

Реализуем неэффективный алгоритм в виде программы, на выбранных языках программирования используя данные, представленные в файле A.

PascalABC.net

```
program p1;
var
a:array [1..100000] of integer;
N, i, j, m, start, finish: integer;
begin
  start := Milliseconds;
  Assign( input, 'A.txt' );
  Readln( N );
  for i:=1 to N do
    Readln( a[i] );
  m:=0;
```

```
  for i:=1 to n-1 do
    for j:=i+1 to n do
      if a[i]+a[j]>m
        then
         m:=a[i]+a[j];
  writeln(m);
  finish := Milliseconds;
  writeln('Time: ', (finish - start)/1000);
end.
```

Python 3.9.1

```
import time
t0 = time.time()
Fin = open('A.txt')
N = int(Fin.readline())
A =[]
m=0
for i in range(N):
    A.append(int(Fin.readline()))
for i in range(len(A)-1):
    for j in range(i+1, len(A)):
        if m < A[i] + A[j]:
            m=A[i]+A[j]
print(m)
Fin.close()
t1 = time.time() - t0
print("Time: ", t1)
```

Java 8

```
import java.io.File;
import java.io.IOException;
import java.util.Scanner;
public class Main {
public static void main(String[] args) throws IOException {
double start = System.currentTimeMillis();
Scanner scanner = new Scanner(new File("A.txt"));
        int[] numbers;
        int n = 0;
        if (scanner.hasNext()) {
            n = scanner.nextInt();
}
        numbers = new int[n];
        for (int i = 0; i < n; i++) {
            numbers[i] = scanner.nextInt();
}
        scanner.close();
        int m = 0;
        for (int i = 0; i < numbers.length - 1; i++) {
        for (int j = i + 1; j < numbers.length; j++) {
         if (m < numbers[i] + numbers[j]) {
                    m = numbers[i] + numbers[j];
}
```

```
            }
        }
System.out.println(m);
double difference = System.currentTimeMillis() - start;
System.out.println("Time: "+ difference / 1000);
}
}
```

Результаты выполнения программ на данных, представленных в файле A.txt (60 элементов последовательности):

- PascalABC.net – 0,017с;
- Python 3.9.1 – 0,011с;
- Java 8 – 0,036с.

Заменим файл A на файл B в котором находится последовательность из 60000 элементов и рассмотрим результаты выполнения программ:

- PascalABC.net – 19,286с;
- Python 3.9.1 – 1299,842с;
- Java 8 – 1,885с.

Рассмотрим эффективный алгоритм решения, используя один просмотр всей последовательности элементов без хранения в виде массива с поиском двух самых больших чисел.

Реализуем эффективный алгоритм в виде программы, на выбранных языках программирования используя данные, представленные в файле A.

PascalABC.net

```
program p1;
var
    m1,m2,start, finish, a, b: integer;
    N, i: integer;
begin
  start := Milliseconds;
  Assign( input, 'A.txt' );
  readln(N);
  readln(a);
  readln(b);
  m1:=max(a,b);
  m2:=min(a,b);
  for i:=3 to n do
  begin
    readln(a);
    if a>m1 then
    begin
      m2:=m1;
      m1:=a;
    end
    else
      if a>m2 then
```

```
        m2:=a;
  end;
  writeln(m1+m2);
  finish := Milliseconds;
  writeln('Time: ', (finish - start)/1000);
end.
```

Python 3.9.1

```
import time
t0 = time.time()
Fin = open("A.txt")
N=int( Fin.readline() )
a=int( Fin.readline() )
b=int( Fin.readline() )
m1=max(a,b)
m2=min(a,b)
for i in range(N-2):
    a=int( Fin.readline() )
    if a>m1:
        m2=m1
        m1=a
    else:
        if a>m2:
            m2=a
print(m1+m2)
Fin.close()
t1 = time.time() - t0
print("Time: ", t1 )
```

Java 8

```
import java.io.File;
import java.io.IOException;
import java.util.Scanner;
public class Main {
public static void main(String[] args) throws IOException {
double start = System.currentTimeMillis();
Scanner scanner = new Scanner(new File("A.txt"));
        int n = 0;
        if (scanner.hasNext()) {
            n = scanner.nextInt();
}
int a = scanner.nextInt();
        int b = scanner.nextInt();
        int m1 = Math.max(a, b);
        int m2 = Math.min(a, b);
        for (int i = 0; i < n - 2; i++) {
            a = scanner.nextInt();
            if (a > m1) {
                m2 = m1;
                m1 = a;
                       } else {
                        if (a > m2) {
```

```
                        m2 = a;
                                }
                }
            }
        scanner.close();
        System.out.println(m1 + m2);
        double difference = System.currentTimeMillis() - start;
        System.out.println("Time: "+ difference / 1000);
        }
        }
```

Результаты выполнения программ на данных, представленных в файле A.txt (60 элементов последовательности):

- PascalABC.net – 0,015с;
- Python 3.9.1 – 0,011с;
- Java 8 – 0,039с.

Заменим файл A на файл B в котором находится последовательность из 60000 элементов и рассмотрим результаты выполнения программ:

- PascalABC.net – 0,031с;
- Python 3.9.1 – 0,031с;
- Java 8 – 0,252с.

Запишем результаты в виде таблицы и выполним их анализ.

Таблица 1

Временные затраты

Языки программирования	Неэффективный алгоритм (с)		Эффективный алгоритм (с)	
	Файл A.txt	Файл B.txt	Файл A.txt	Файл B.txt
PascalABC.net	0,017	19,286	0,015	0,031
Python 3.9.1	0,011	1299,842	0,011	0,031
Java 8	0,039	1,885	0,039	0,252

Анализируя временные затраты для эффективного алгоритма можно прийти к выводу что выбор языка программирования вряд ли повлияет на скорость обработки даже очень больших последовательностей.

С другой стороны, если использовать неэффективный алгоритм (он проще, его чаще всего знают выпускники, реализуется в виде программы очень быстро) то можно увидеть парадоксальные результаты:

- для PascalABC.net – придется подождать результатов обработки второго файла целых 20 секунд;

- для Python - 20 с лишним минут – это очень большой промежуток времени и на ЕГЭ такие временные затраты не возможны.

Наиболее оптимальные результаты на всех этапах показал язык программирования Java.

Временные затраты также зависят от характеристик компьютера и устаревшие модели показывали результаты в 10 раз хуже.

Заключение

В данной статье были рассмотрены временные затраты эффективного и неэффективного способа решения задач №27 ЕГЭ по информатике. Время выполнения переборных алгоритмов является очень большим и возрастает пропорционально количеству элементов последовательности в совокупности со сложностью операций по определению необходимых результатов. При подготовке к экзамену выпускник должен четко понимать, что неэффективные алгоритмы могут привести к очень длительному ожиданию результатов, о чем и предупреждают разработчики КИМов. Для эффективного решения необходимо применять оптимальные способы хранения данных, алгоритмы обработки с минимальным количеством операций.

Список литературы

1. Тематический сайт для подготовки к ЕГЭ по информатике. – Режим доступа : https://www.kpolyakov.spb.ru/school/ege.htm

2. Тематический канал видематериалов для подготовки к ЕГЭ по информатике. – Режим доступа : https://www.youtube.com/channel/UCmUcjDHUkIMhfqBfyHYXYuA

3. Федеральный институт педагогических ихмерений. – Режим доступа :http://fipi.ru/EGE-I-GVE-11/demoversii-specifikacii-kodifikatory