Министерство образования и науки Украины
Севастопольский национальный технический университет

МАТЕМАТИЧЕСКИЙ АНАЛИЗ АЛГОРИТМОВ

Экспериментальная проверка теоретической оценки трудоемкости алгоритма

Методические указания
к выполнению лабораторной работы
для студентов, обучающихся по направлению
6.050101“Компьютерные науки”
очной и заочной форм обучения

Севастополь

2014

УДК 510.5

МАТЕМАТИЧЕСКИЙ АНАЛИЗ АЛГОРИТМОВ., Экспериментальная проверка теоретической оценки трудоемкости алгоритма: методические указания / Сост. Ю. В. Коваленко, Е. Н. Заикина. — Севастополь: Изд-во СевНТУ, 2014— 10 с.

Целью лабораторной являетс экспериментальная проверка теоретической оценки трудоемкости алгоритма поиска минимума и включает ознакомление с принципами использования генератора случайных чисел для создания наборов исходных данных.

Методические указания составлены в соответствии с требованиями программы дисциплины «Теория алгоритмов и элементы математической логики» для студентов направления 6.050101 и 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

Лабораторная работа посвящена экспериментальной проверке теоретической оценки трудоемкости алгоритма поиска минимума и включает ознакомление с принципами использования генератора случайных чисел для создания наборов исходных данных.

2. ТЕОРЕТИЧЕСКИЙ РАЗДЕЛ

2.1 Теоретическая оценка среднего количества операций переприсваивания в алгоритме поиска минимума

Алгоритм поиска минимума последовательно перебирает элементы массива, сравнивая текущий элемент массива с текущим значением минимума. На очередном шаге, когда просматривается **к**-ый элемент массива, переприсваивание минимума произойдет, если в подмассиве из первых **к** элементов минимальным элементом является последний. В случае равномерного распределения исходных данных вероятность того, что максимальный из **к** элементов расположен в определенной (последней) позиции равна **1/к.**

Тогда в массиве из **N** элементов среднее количество операций переприсваивания максимума определяется как:

$$\sum_{i=1}^{N} 1/i = Hn \approx Ln(N) + \gamma, \_\gamma = 0{,}57$$

Величина **Hn** называется **n**-ым гармоническим числом. Таким образом, среднее значение (математическое ожидание) среднего количества операций присваивания в алгоритме поиска минимума в массиве из **N** элементов определяется величиной **Hn** (для бесконечно большого количества испытаний).

2.2 Понятие о генераторе псевдослучайных чисел

Для тестирования программ часто необходимо использовать наборы случайных чисел. Такие числа могут быть использованы и для реализации тех или иных алгоритмов. Для создания таких чисел используется генератор случайных чисел. В общем случае алгоритм называется **рандомизированным** (randomized), если его поведение определяется не только набором входных величин, но и значениями, которые выдает генератор случайных чисел.

На практике большинство сред программирования предоставляют в распоряжение программиста **генератор псевдослучайных чисел**, т.е. детерминированный алгоритм, который возвращает числа, которые ведут себя при статистическом анализе как случайные.

3 ХОД РАБОТЫ

1. Ознакомиться с теоретическим разделом настоящих методических указаний и повторить соответствующий лекционный материал.

2. Изучить функции языка Pascal, использующиеся для генерации псевдослучайных чисел, привести их описание в отчете.

3. Составить структурную схему и написать программу поиска минимума в массиве сгенерированных псевдослучайных чисел.

4. Написать программу подсчета *n*-го гармонического числа.

5. Подсчитать количество операций переприсваивания для программной реализации поиска минимума в массиве случайных чисел. Внести изменения в соответствующую программу. Длину массива и максимальное случайное число в последовательности взять в соответствии с вариантом.

6. Сравнить практически полученное значение с теоретическим *n*-м гармоническим числом. Примеры основных функций приведены в разделе 5.

7. Сделать выводы по работе, оформить отчет, подготовить ответы на контрольные вопросы.

4 ВАРИАНТЫ ЗАДАНИЙ

Таблица 1 – Варианты заданий для выполнения лабораторной работы

Вариант	Наибольшее случайное число в последовательности	Количество элементов в массиве случайных чисел
1	100	100, 1000, 5000
2	250	150, 550, 1000
3	150	100, 500, 1000
4	950	250, 300, 400
5	200	200, 300, 500
6	800	100, 1000, 2000
7	250	300, 400, 1400
8	800	300, 450, 1000
9	200	150, 200, 1000
10	350	100, 500, 1500
11	100	300, 900, 3000
12	400	100, 200, 500
13	500	100, 1000, 1500
14	950	250, 500, 750
15	150	100, 1000, 5000
16	200	150, 550, 1000
17	750	100, 500, 1000
18	550	250, 300, 400
19	150	200, 300, 500
20	200	100, 1000, 2000

5 ПРИМЕРЫ ОСНОВНЫХ ФУНКЦИЙ ПРОГРАММЫ

```
Program Test;
var vector: array [1..10] of integer;
    { Создание массива из 10 псевдослучайных целых чисел  величиной
от 0
    до 100, массив записывается в файл Example.TXT, на экран выводит-
ся
    минимальное целое число}
         i,
         N,
         min,           {значение минимума}
         cnt:integer;   {счетчик операций переприсваивания}
         result:integer;
         out_file: text;

Procedure create_array(Nmax:integer);
         var i:integer;
Begin
     {Nmax = 10; соответствует размерности массива}
     randomize;

     writeln(out_file, Nmax, ' random numbers from 0 to 100');
     writeln(Nmax, ' random numbers from 0 to 100');
     for i := 1 to Nmax do
         begin
              vector[i] := random(100);
          writeln(vector[i]);
          writeln(out_file, vector[i]);
      end;
end;


Begin
      assign(out_file,''Example_TA2.TXT');
      rewrite(out_file);
      writeln('Input amount of numbers');
      read(N);
      create_array(N); {генерация массива псевдослучайных чисел}
      min := vector[1];
      cnt := 1;
      for i := 1 to N do
          if vector[i] < min then
          begin
               min := vector[i];
               cnt := cnt+1;
          end;
      writeln('Minimal ', min, ', Count operations ', cnt);
      writeln(out_file, 'Minimal ', min, ', Count operations', cnt);
      close(out_file);
End.
```

Или для на C#:

Наибольшее случайное число в последовательности	Количество элементов в массиве случайных чисел
850	300, 450, 1000

```
using System;
using System.IO;

namespace laba2
{
    class Program
    {
        /// <summary>
        /// Инициализация массива с псевдослучайными числами
        /// </summary>
        /// <param name="N">Количество элементов</param>
        /// <param name="maxval">Максимальное значения элемента</param>
        /// <param name="r">Открытый StreamWriter для вывода в файл (ссылка)</param>
        /// <returns>Массив псевдослучайных чисел</returns>
        static private int[] init_mas(int N, int maxval,ref StreamWriter r)
        {
            int []a = new int[N];
            Random rnd = new Random();
            for (int i = 0; i < N; i++)
            {
                a[i] = rnd.Next(0, maxval);
                r.Write(a[i] + " ");
            }
            return a;
        }

        /// <summary>
        /// Процедура поиска первого(с начала массива) минимального элемента
        /// </summary>
        /// <param name="mas">ссылка на массив</param>
        /// <param name="min">Возвращает значение минимального элемента(ссылка, иниц. перем. в 0)</param>
        /// <param name="count">Возвращает количество операций переприсваивания(ссылка, иниц. перем. в 0)</param>
        static private void poiskmin(ref int[] mas, ref int min, ref int count)
        {
            min = mas[0];
            for (int i = 0; i < mas.Length; i++)
            {
                if (mas[i] < min)
```

```
{
                    min = mas[i];
                    count++;
                }
            }
        }

        /// <summary>
        /// n-ое гармоническое число (через сумму)
        /// </summary>
        /// <param name="N">Количество элементов</param>
        /// <returns></returns>
        static private double harm_sum(int N)
        {
            double result = 0;
            for (int i = 1; i <= N; i++)
                result += 1.0 / i;
            return result;
        }

        /// <summary>
        /// n-ое гармоническое число (через логарифм)
        /// </summary>
        /// <param name="N">Количество элементов</param>
        /// <returns></returns>
        static private double harm_ln(int N)
        {
            return (Math.Log(N) + 0.57d);
        }

        static void Main(string[] args)
        {
            StreamWriter potok = new StreamWriter("out.txt");

            Console.Write("Введите количество элементов: ");
            int N = Int32.Parse(Console.ReadLine());
            Console.Write("Введите наибольшее значения случайного числа: ");
            int MV = Int32.Parse(Console.ReadLine());

            int[] masiv = init_mas(N, MV, ref potok);

            int min = 0, count = 0;
            poiskmin(ref masiv, ref min, ref count);
            Console.WriteLine("MIN: {0}  COUNT: {1}  Hn: {2}  Hnln: {3}",
min, count, harm_sum(N), harm_ln(N));
            potok.WriteLine("/n{0} {1} {2} {3}", min, count, harm_sum(N),
harm_ln(N));
            Console.In.Read();
            potok.Close();
        }
```

```
  }
}
```

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. Понятие о трудоемкости алгоритма.
2. Понятие о сложности алгоритма.
3. Определение детерминированного и рандомизированного алгоритмов.
4. Определение генератора псевдослучайных чисел.
5. Описать основные функции и константы, использующиеся для создания последовательности псевдослучайных чисел в языке С++.

8 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.