Министерство образования и науки Российской Федерации

САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

Факультет программной инженерии и компьютерной техники
Направление подготовки 09.03.04 Программная инженерия

Дисциплина «Алгоритмы и структуры данных»

ОТЧЁТ

по лабораторной работе Введение (Stepic)

Студенка Кузенкова Елизавета группы P3217

Преподаватель Муромцев Дмитрий Ильич

Санкт-Петербург

2019 г.

Содержание

Задача 1: небольшое число Фибоначчи

Дано целое число $1 \le n \le 40$

, необходимо вычислить n-е число Фибоначчи (напомним, что $F_0 = 0$, $F_1 = 1$ и $F_n = F_{n-1} + F_{n-2}$ при $n \ge 2$

).

Sample Input:

```
3
```

Sample Output:

```
2
```

Исходный код к задаче 1

```
class Lab1_1
    {
        private void DoWork(string[] args)
        {
            var n = int.Parse(Console.ReadLine());
            Console.WriteLine(this.Fib(n));
        }

        private int Fib(int n)
        {
            if (n == 1 || n == 2)
            {
                return 1;
            }
            else
            {
                return this.Fib(n - 1) + this.Fib(n - 2);
            }

        }

        public static void Main(string[] args)
        {
            var app = new Lab1_1();
            app.DoWork(args);
        }
    }
```

Задача 2: последняя цифра большого числа Фибоначчи

Дано число $1 \le n \le 10^7$, необходимо найти последнюю цифру n-го числа Фибоначчи.

Как мы помним, числа Фибоначчи растут очень быстро, поэтому при их вычислении нужно быть аккуратным с переполнением. В данной задаче, впрочем, этой проблемы можно избежать, поскольку нас интересует только последняя цифра числа Фибоначчи: если $0 \le a, b \le 9$

— последние цифры чисел F_i и F_{i+1} соответственно, то $(a+b) \bmod 10$ — последняя цифра числа F_{i+2}

Sample Input:

```
317457
```

Sample Output:

```
2
```

Исходный код к задаче 2

```
class Lab1_2
    {
        private void DoWork(string[] args)
        {
            var n = int.Parse(Console.ReadLine());
            Console.WriteLine(this.Fib(n));
        }

        private int Fib(int n)
        {
            int[] f = new int[2];
            f[0] = 1; f[1] = 1;
            for (int i = 2; i <= (n - 1); i++)
                f[i % 2] = (f[i % 2] + f[(i + 1) % 2]) % 10;
            return f[(n - 1) % 2];
        }

        public static void Main(string[] args)
        {
            var app = new Lab1_2();
            app.DoWork(args);
        }
    }
```

Задача 3: огромное число Фибоначчи по модулю

Даны целые числа $1 \le n \le 10^{18}$ и $2 \le m \le 10^5$, необходимо найти остаток от деления n-го числа Фибоначчи на m

Sample Input:

```
10 2
```

Sample Output:

```
1
```

Исходный код к задаче 3

```
class Lab1_3
    {
        /**
         * Метод для определения элемента в последовательности периодов Пизано
         *
         * @param divider делитель
         * @return элемента последовательности периода Пизано
         */
        private static long CalcPisanoPeriods(long divider)
        {
            long a = 0;
            long b = 1;
            long c;

            for (long i = 0L; i < divider * divider; i++)
            {
                c = (a + b) % divider;
```

```
                a = b;
                b = c;

                if (a == 0 && b == 1)
                {
                    return i + 1;
                }
            }
            return a;
        }

        private long GetHugeFibonacci(long elementNumber, long divider)
        {
            long remainder = elementNumber % CalcPisanoPeriods(divider);

            long first = 0L;
            long second = 1L;

            long result = remainder;

            for (int i = 1; i < remainder; i++)
            {
                result = (first + second) % divider;
                first = second;
                second = result;
            }

            return result % divider;
        }

        private void DoWork(string[] args)
        {
            var input = Console.ReadLine().Split(' ').Select(long.Parse).ToArray();
            Console.WriteLine(this.GetHugeFibonacci(input[0], input[1]));
        }

        public static void Main(string[] args)
        {
            var app = new Lab1_3();
            app.DoWork(args);
        }
    }
```

Задача 4: наибольший общий делитель

По данным двум числам $1 \le a, b \le 2 \cdot 10^9$ найдите их наибольший общий делитель.

Sample Input 1:

```
18 35
```

Sample Output 1:

```
1
```

Sample Input 2:

```
14159572 63967072
```

Sample Output 2:

```
4
```

Исходный код к задаче 4

```
class Lab1_4
    {
        private int GetNOD(int firstValue, int secondValue)
        {
            if (firstValue == 0)
            {
                return secondValue;
            }
            else if (secondValue == 0)
            {
                return firstValue;
            }
            else if (firstValue >= secondValue)
            {
                return GetNOD(firstValue % secondValue, secondValue);
            }
            else
            {
                return GetNOD(firstValue, secondValue % firstValue);
            }
        }

        private void DoWork(string[] args)
        {
            var input = Console.ReadLine().Split(' ').Select(int.Parse).ToArray();
            Console.WriteLine(this.GetNOD(input[0], input[1]));
        }

        public static void Main(string[] args)
        {
            var app = new Lab1_4();
            app.DoWork(args);
        }
    }
```

Тест: правила работы с логарифмами

Отметьте **все** верные утверждения.

- $n^{\log_2 n} = n$
+ $\log_3(2n) = \log_3 2 + \log_3 n$
- $\log_2 n \cdot \log_2 3 = \log_3 n$
- $(\log_5 n)^2 = 2\log_5 n$

Тест: правильная скорость роста

Отметьте **все** верные утверждения.

+ $\log_3(2n) = \Theta(\log_2(3n))$
+ $2^n = \Theta(2^{n+1})$
+ $n^2/\log_3 n = \Omega(n(\log_2 n)^2)$
+ $n! = \Omega(2^n)$
+ $\sqrt{n} = \Omega((\log_2 n)^3)$
+ $2^n = O(2^{n+1})$
+ $n^{1/2} = O\left(5^{\log_2 n}\right)$

Чтобы проверить равенства вида $f(n)=O(g(n))$ нужно найти $\lim_{n\to\infty} f(n)/g(n)$. Если он равен константе, равенства верны и для O, и для Ω, и для Θ. Если бесконечности, то только для Ω. Если нулю, то только для O.

Limit[Log[3,2 * n]/Log[2,3 * n] , n → Infinity]

$$\frac{\text{Log}[2]}{\text{Log}[3]}$$

Limit[$2^n/2^{(n+1)}$, n → Infinity]

$$\frac{1}{2}$$

Limit[$(n^2)/(\text{Log}[3,n])$, n → Infinity]

$$\infty$$

Limit[Factorial[n]/2^n , n → Infinity]

$$\infty$$

Limit[Sqrt[n]/(Log[2,n])^3 , n → Infinity]

$$\infty$$

Limit[$(2^n)/2^{(n+1)}$, n → Infinity]

$$\frac{1}{2}$$

Limit[Sqrt[n]/$5^{(\text{Log}[2,n])}$, n → Infinity]

$$0$$

Тест: правильная скорость роста

Упорядочите данные семь функций по возрастанию скорости роста (сверху — медленнее всего растущая функция, снизу — быстрее всего растущая).

$$\log_5 n$$

$$n^{0.3}$$

$$\sqrt{n}$$

$$n \log_2 n$$

$$n(\log_2 n)^3$$

$$n^3$$

$$4^n$$

```
ff = Inactivate[{ log_5 n , n log_2 n , n(log_2 n)^3, n^3, 4^n}];
sf = Sort[ff, {f1, f2} => Limit[Activate[f1/f2], n -> Infinity] != Infinity];
```

$$\{\frac{\text{Log}[n]}{\text{Log}[5]}, n^{0.3}, \sqrt{n}, \frac{n\text{Lo }[n]}{\text{Log}[2]}, \frac{n\text{Log}[n]^3}{\text{Log}[2]^3}, n^3, 4^n\}$$

```
Grid[tableData = Transpose[Prepend[Transpose[Prepend[
        (f1 => ((f2 => If[Limit[Activate[f1/f2], n -> Infinity] != Infinity, ">", "<"]) /@ sf)) /@ sf,
        Rotate[TraditionalForm[#], Pi/2] & /@ sf]], Prepend[TraditionalForm /@ sf, ""]]]]
```

	$\frac{\log(n)}{\log(5)}$	$n^{0.3}$	$\sqrt{n}$	$\frac{n\log(n)}{\log(2)}$	$\frac{n\log^3(n)}{\log^3(2)}$	n^3	4^n
$\frac{\log(n)}{\log(5)}$	>	>	>	>	>	>	>
$n^{0.3}$	<	>	>	>	>	>	>
$\sqrt{n}$	<	<	>	>	>	>	>
$\frac{n\log(n)}{\log(2)}$	<	<	<	>	>	>	>
$\frac{n\log^3(n)}{\log^3(2)}$	<	<	<	<	>	>	>
n^3	<	<	<	<	<	>	>
4^n	<	<	<	<	<	<	>

Тест повышенной сложности: правильная скорость роста

Упорядочите данные функции по возрастанию скорости роста (сверху — медленнее всего растущая функция, снизу — быстрее всего растущая):

$$\sqrt{\log_4 n}, \log_3 n, \log_2(n!),$$
$$\frac{n}{\log_5 n}, \sqrt{n}, (\log_2 n)^2,$$
$$\log_2 \log_2 n, 3^{\log_2 n}, 2^{3n},$$
$$n^2, n^{\log_2 n}, n^{\sqrt{n}},$$
$$4^n, 2^n, 7^{\log_2 n},$$
$$n!, (\log_2 n)^{\log_2 n}, 2^{2^n}$$

```
ff = Inactivate[{
        Sqrt[Log[4, n]], Log[3, n], Log[2, n!],
        n/Log[5, n], Sqrt[n], Log[2, n]^2,
        Log[2, Log[2, n]], 3^Log[2, n],    2^(3 n), n^2,
        n^Log[2, n], n^Sqrt[n], 4^n,
        2^n, 7^Log[2, n], n!,
        Log[2, n]^Log[2, n], 2^2^n
}];
sf = Sort[ff, {f1, f2} => Limit[Activate[f1/f2], n -> Infinity] != Infinity];

{
        Log[2,Log[2,n]],Sqrt[Log[4,n]],Log[3,n],
        Log[2,n]^2,Sqrt[n],n*Log[5,n]^(1),
        Log[2,n!],3^Log[2,n],n^2,
        7^Log[2,n],Log[2,n]^Log[2,n],n^Log[2,n],
        n^Sqrt[n],2^n,4^n,
        2^(3*n),n!,2^2^n
}

Grid[tableData = Transpose[Prepend[Transpose[Prepend[
        (f1 => ((f2 => If[Limit[Activate[f1/f2], n -> Infinity] != Infinity, ">", "<"]) /@ sf)) /@ sf,
        Rotate[TraditionalForm[#], Pi/2] & /@ sf]], Prepend[TraditionalForm /@ sf, ""]]]]
```

	log(2, log(2, n))	Sqrt(log(4, n))	log(3, n)	log(2, n) ^2	Sqrt(n)	n * log(5, n) ^ (-1)	log(2, n !)	3^log(2, n)	n^2	7^log(2, n)	log(2, n) ^log(2, n)	n^log(2, n)	n^Sqrt(n)	2^n	4^n	2^(3 * n)	n !	2^2^n
log(2, log(2, n))	>	>	>	>	>	>	>	>	>	>	>	>	>	>	>	>	>	>
Sqrt(log(4, n))	<	>	>	>	>	>	>	>	>	>	>	>	>	>	>	>	>	>
log(3, n)	<	<	>	>	>	>	>	>	>	>	>	>	>	>	>	>	>	>
log(2, n) ^2	<	<	<	>	>	>	>	>	>	>	>	>	>	>	>	>	>	>
Sqrt(n)	<	<	<	<	>	>	>	>	>	>	>	>	>	>	>	>	>	>
n * log(5, n) ^ (-1)	<	<	<	<	<	>	>	>	>	>	>	>	>	>	>	>	>	>
log(2, n !)	<	<	<	<	<	<	>	>	>	>	>	>	>	>	>	>	>	>
3^log(2, n)	<	<	<	<	<	<	<	>	>	>	>	>	>	>	>	>	>	>
n^2	<	<	<	<	<	<	<	<	>	>	>	>	>	>	>	>	>	>
7^log(2, n)	<	<	<	<	<	<	<	<	<	>	>	>	>	>	>	>	>	>
log(2, n) ^log(2, n)	<	<	<	<	<	<	<	<	<	<	>	>	>	>	>	>	>	>
n^log(2, n)	<	<	<	<	<	<	<	<	<	<	<	>	>	>	>	>	>	>
n^Sqrt(n)	<	<	<	<	<	<	<	<	<	<	<	<	>	>	>	>	>	>
2^n	<	<	<	<	<	<	<	<	<	<	<	<	<	>	>	>	>	>
4^n	<	<	<	<	<	<	<	<	<	<	<	<	<	<	>	>	>	>
2^(3 * n)	<	<	<	<	<	<	<	<	<	<	<	<	<	<	<	>	>	>
n !	<	<	<	<	<	<	<	<	<	<	<	<	<	<	<	<	>	>
2^2^n	<	<	<	<	<	<	<	<	<	<	<	<	<	<	<	<	<	>