Projekt 0

Zadanie:
Ocenić złożoność algorytmu obliczania *n*-tej liczby Fibonacciego empirycznie:
a) przy użyciu instrumentacji
b) przy wykorzystaniu pomiaru czasu wykonania

Implementacja metody obliczającej n-tą liczbą Fibonacciego:

```
uint fib(uint n)
{
  switch (n)
  {
     case 0: return 1;
     case 1: return 1;
     default: return fib(n - 1) + fib(n - 2);
  }
} /* fib() */
```

a) ocena przy wykorzystaniu instrumentacji

Do oceny złożoności algorytmu wybrano zliczanie liczby dodawań jako operacji dominującej. Liczby: wywołań funkcji, odejmowań (2 x liczba dodawań) i dodawań są liniowo powiązane ze sobą.

Kod wykorzystany w eksperymencie:

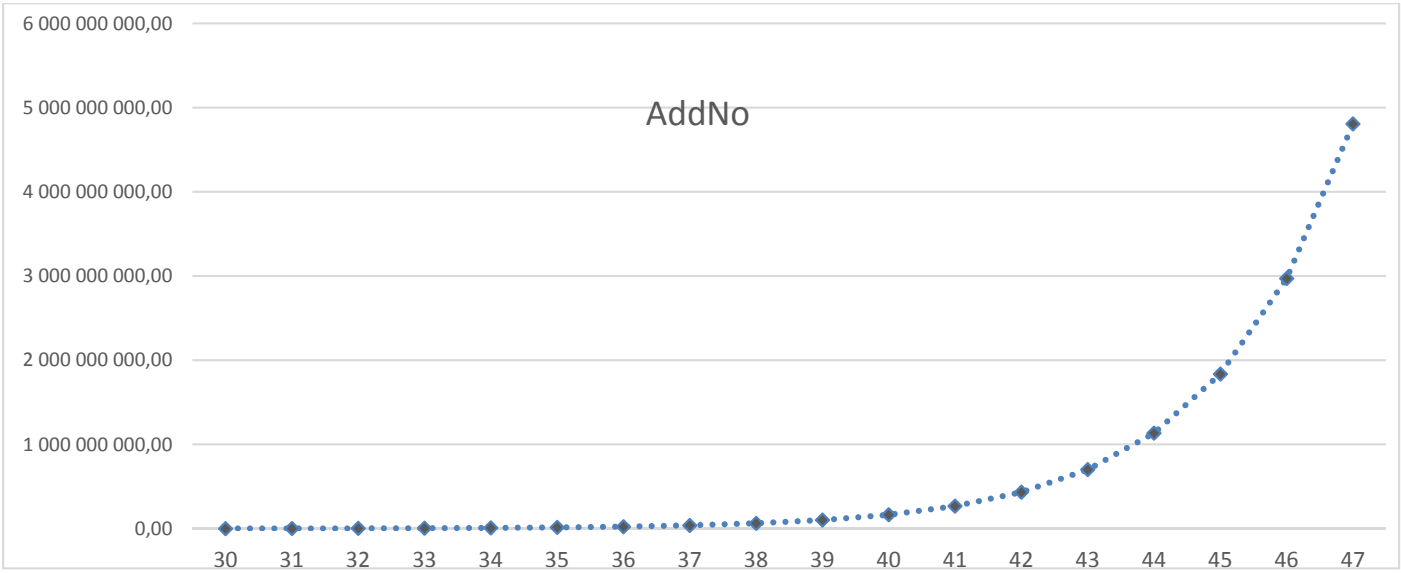
```
using System;

namespace ProjektWzorcowy
{
    class Program
    {
        static ulong AddNumber;

        static uint fib(uint n)
        {
            switch (n)
            {
                case 0: return 1;
                case 1: return 1;
                default: AddNumber++; return fib(n - 1) + fib(n - 2);
            }
        } /* fib() */

        static void Main(string[] args)
        {
            Console.WriteLine("n\tAddNo");
            for (uint u = 30; u < 48; u++)
            {
                AddNumber = 0;
                ulong r = fib(u);
                Console.WriteLine("{0}\t{1}", u, AddNumber);
            }
        }
    }
}
```

W eksperymencie wykorzystano 18 punktów pomiarowych o wartościach od 30 do 47 (maksymalna przy której w typie *uint* mieści się poprawny rezultat funkcji).

Zebrane dane

n	AddNo
30	1346268
31	2178308
32	3524577
33	5702886
34	9227464
35	14930351
36	24157816
37	39088168
38	63245985
39	102334154
40	165580140
41	267914295
42	433494436
43	701408732
44	1134903169
45	1836311902
46	2971215072
47	4807526975

Zależność liczby dodawań od numeru liczby Fibonacciego



b) ocena przy wykorzystaniu pomiaru czasu wykonania

Kod wykorzystany w eksperymencie (uwaga: bez instrumentacji !!!):

```
using System;
using System.Diagnostics;

namespace ProjektWzorcowy
{
  class Program
  {
    static uint fib(uint n)
    {
      switch (n)
      {
        case 0: return 1;
        case 1: return 1;
        default: return fib(n - 1) + fib(n - 2);
      }
    } /* fib() */

    static void Main(string[] args)
    {
      const int NIter = 10; // liczba powtórzeń testu

      Console.WriteLine("n\tt[s]");
      for (uint u = 30; u < 48; u++)
      {
        double ElapsedSeconds;
        long ElapsedTime = 0, MinTime = long.MaxValue, MaxTime = long.MinValue, IterationElapsedTime;
        for (int n = 0; n < (NIter + 1 + 1); ++n) // odejmujemy wartości skrajne
        {
          long StartingTime = Stopwatch.GetTimestamp();
          ulong r = fib(u);
          long EndingTime = Stopwatch.GetTimestamp();
          IterationElapsedTime = EndingTime - StartingTime;
          ElapsedTime += IterationElapsedTime;
          //Console.Write("Iter[" + n + "]:" + IterationElapsedTime + "\t");
          if (IterationElapsedTime < MinTime) MinTime = IterationElapsedTime;
          if (IterationElapsedTime > MaxTime) MaxTime = IterationElapsedTime;
        }
        ElapsedTime -= (MinTime + MaxTime);
        ElapsedSeconds = ElapsedTime * (1.0 / (NIter * Stopwatch.Frequency));
        Console.WriteLine("{0}\t{1}", u, ElapsedSeconds.ToString("F4"));
      }
    }
  }
}
```

W eksperymencie wykonano pomiary na komputerze wyposażonym w procesor Intel® i5-480M z wykorzystaniem Visual Studio 2013 Ultimate. Projekt został skompilowany z ustawieniami Release/Any CPU.

Zebrane dane

n	t[s]
30	0,0070
31	0,0114
32	0,0191
33	0,0295
34	0,0476
35	0,0772
36	0,1245
37	0,2005
38	0,3245
39	0,5266
40	0,8497
41	1,3744
42	2,2253
43	3,5988
44	5,8232
45	9,4324
46	15,2546
47	24,6712

Zależność czasu wykonania od numeru liczby Fibonacciego