

Projekt 1

„Wyszukiwanie liniowe/binarne”

Sebastian Krzyżaniak

Wyszukiwanie binarne

(przed instrumentacją)

```
static int BinSearch(int N, int Number) //pesymistyczne dla Number == 0 //N==wielkość tablicy
{
    int Left = 0; //lewa strona tablicy
    int Right = N - 1; //prawa strona tablicy (zakładamy rozmiar tablicy (2^i)-1)
    int Middle; //środek tablicy
    while (Left <= Right)
    {
        Middle = (Left + Right) >> 1; // dzielenie przez 2 // wyznaczanie środka
        if (Tab[Middle] == Number) return Middle; //zwraca pozycję, na jakiej znajduje się szukana liczba
        else if (Tab[Middle] > Number) Right = Middle - 1;
        else Left = Middle + 1;
    }
    return -1; //zwraca wartość -1, jeżeli w tablicy nie ma porządanej wartości
}
```

* n -(zakres w jakim będziemy szukać danej liczby)

[illegible]

```
static void Main()
{
    Tab = new int[Max_Value_Int];
    for (int q = 0; q < Max_Value_Int; q++) Tab[q] = q + 1;
    int n = 100000;
    StreamWriter file_csv = new StreamWriter("bin_pes.csv");
    for (int i = 10; i <= 30; i++)
    {
        int n += 100000;
        Count_Op = 0;
        BinSearch(n, 0);
        file_csv.WriteLine("{0}; {1}", n, Count_Op);
        Console.WriteLine("{0}; {1}", n, Count_Op);
    }
    file_csv.Close();
}
```

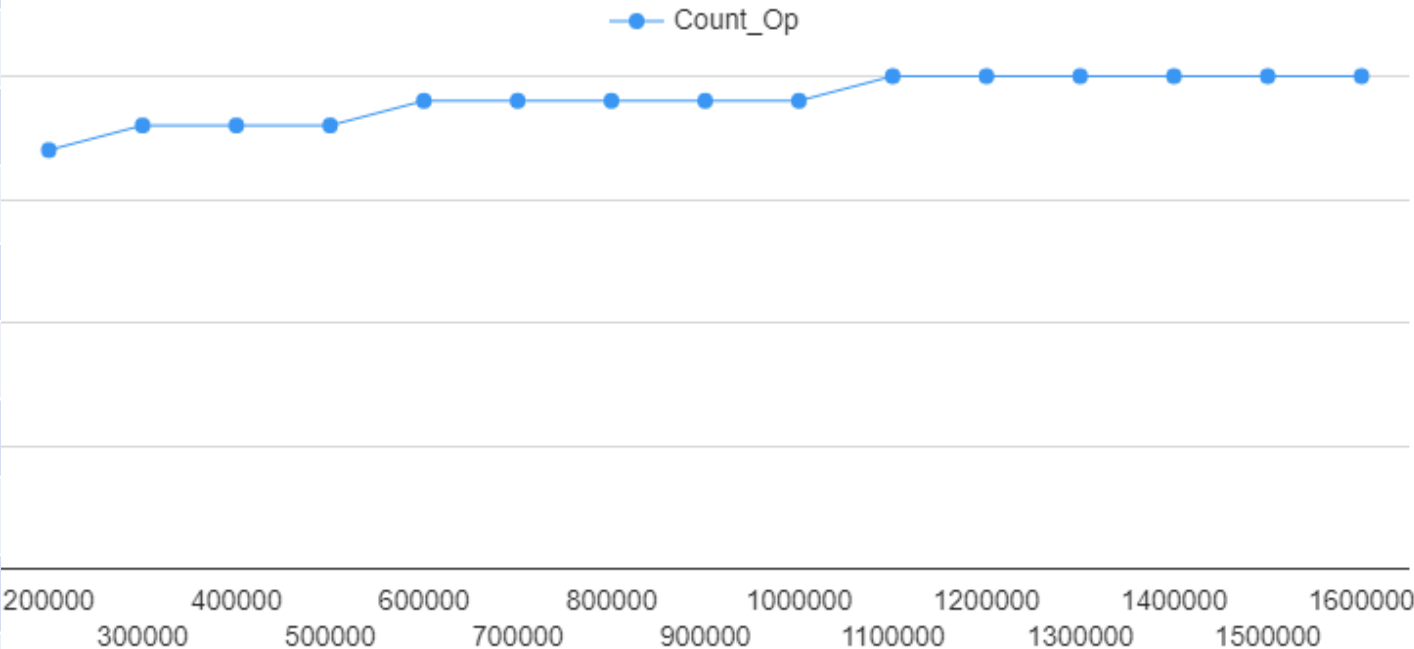
//zapis w pliku

//n –w uporządkowanym szeregu liczb (naturalnych) od 0 do n będziemy szukać danego elementu

//zapis i zamknięcie pliku

Wyszukiwanie binarne pesymistyczne
(zliczanie operacji)

n	Count_Op
200000	17
300000	18
400000	18
500000	18
600000	19
700000	19
800000	19
900000	19
1000000	19
1100000	20
1200000	20
1300000	20
1400000	20
1500000	20
1600000	20




Wyszukiwanie binarne pesymistyczne (instrumentacja czasowa)

$Time *= 1000$

```
const int Max_Value_Int = 268435455;
static int[] Tab;

static int BinSearch(int N, int Number){slajd nr. 2}

static void Main()
{
    Tab = new int[Max_Value_Int];
    for (int q = 0; q < Max_Value_Int; q++) Tab[q] = q + 1;
    int n = 100;
    Stopwatch Stoper = new Stopwatch(); //inicjacja klasy Stopwatch() ~~tworzymy stoper
    StreamWriter file = new StreamWriter("bin_pes_czas.csv"); //tworzymy plik "bin_pes_czas" z rozszerzeniem csv
    for (int i = 10; i <= 18; i++)
    {
        n = n * 5;
        Stoper.Restart(); //ustawienie czasu stopera na 0 i włączenie
        for (int j = 0; j < 100000000; j++) BinSearch(n, 0); //wywołanie szukania binarnego (1mln dla dokł. wyników)
        Stoper.Stop(); //zatrzymanie stopera
        file.WriteLine("{0}; {1}", n, Stoper.ElapsedMilliseconds); //zapis danych w pliku
        Console.WriteLine("{0}; {1}", n, Stoper.ElapsedMilliseconds); //wyświetlenie danych na konsoli
    }
    file.Close();
}
```

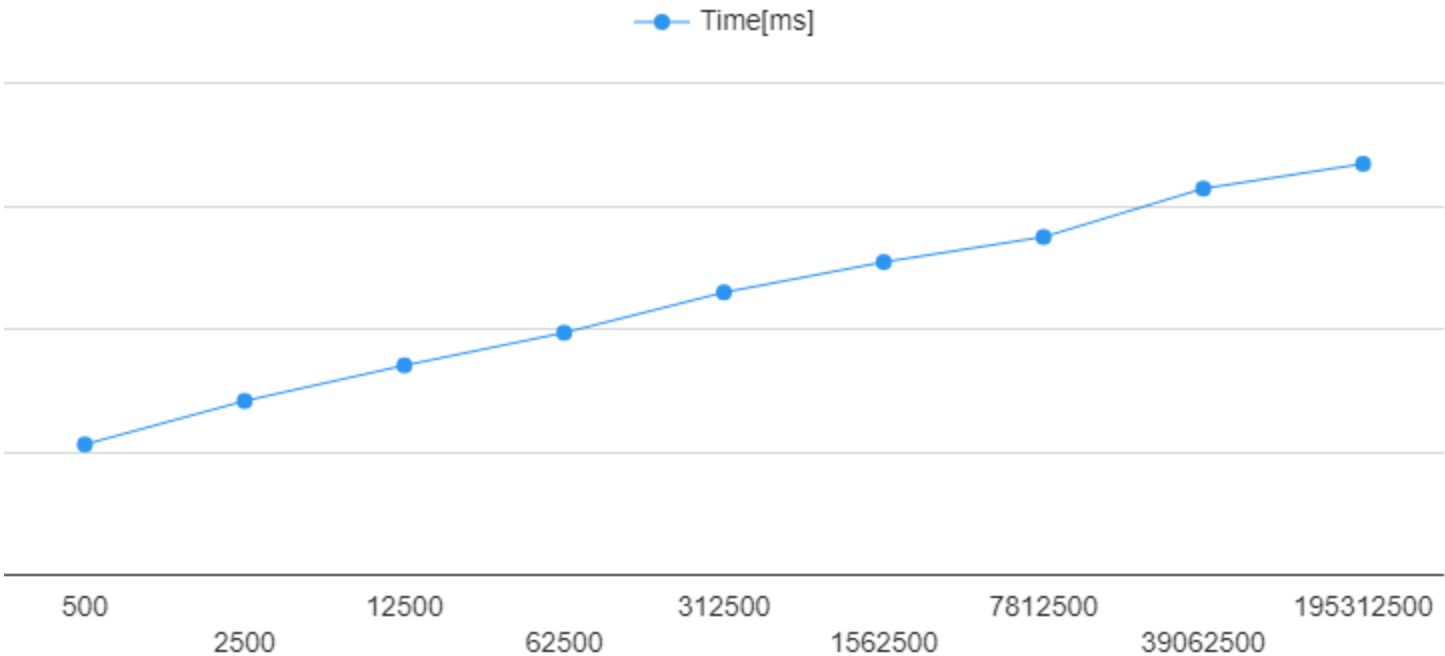


//Time[ms]

Wyszukiwanie binarne pesymistyczne
(instrumentacja czasowa)

Time=1000*

n	Time[ms]
500	7,96
2500	10,61
12500	12,78
62500	14,77
312500	17,23
1562500	19,08
7812500	20,61
39062500	23,56
195312500	25,07



Wyszukiwanie binarne średnie (zliczanie operacji)

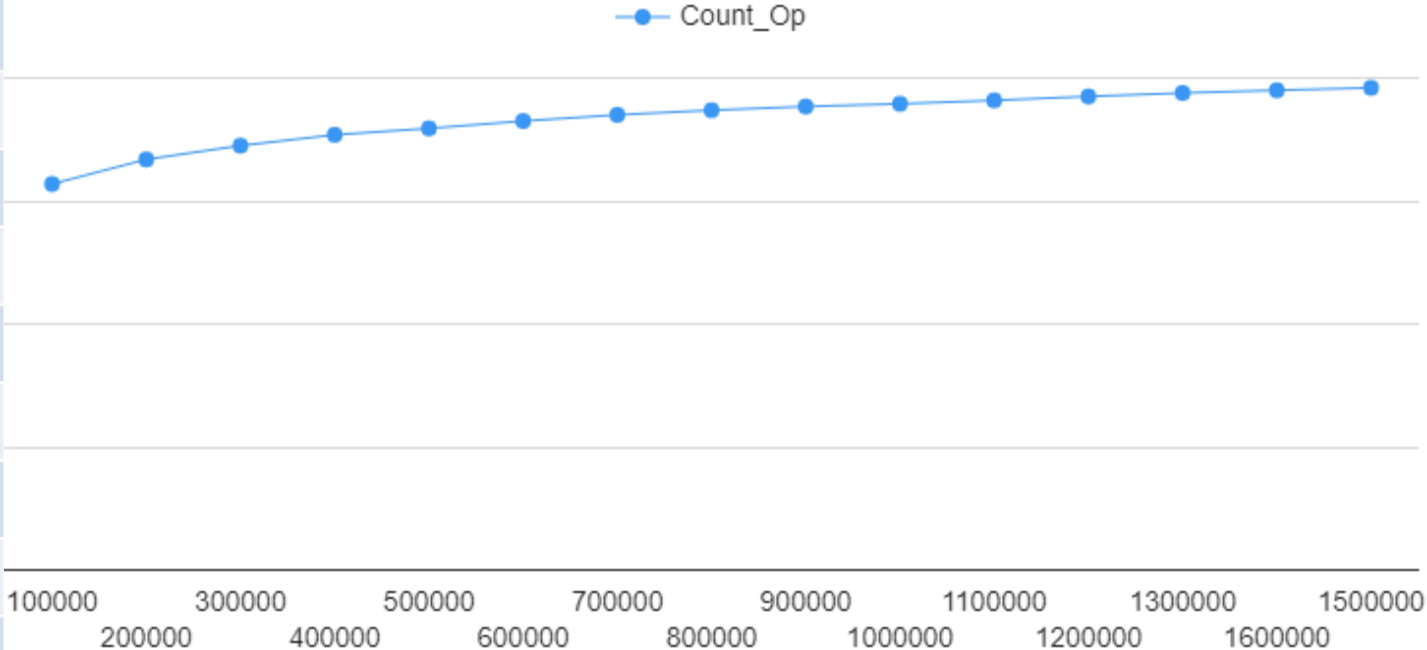
```
const int Max_Value_Int = 268435455;
static int[] Tab;
static long Count_Op;

static int BinSearch(int N, int Number)
{
    int Left = 0;
    int Right = N - 1;
    int Middle;
    while (Left <= Right)
    {
        Count_Op++;
        Middle = (Left + Right) >> 1; //
        if (Tab[Middle] == Number) return Middle;
        else if (Tab[Middle] > Number) Right = Middle - 1;
        else Left = Middle + 1;
    }
    return -1;
}
```

```
//
//
// static void Main()
// {
//     Tab = new int[Max_Value_Int];
//     for (int q = 0; q < Max_Value_Int; q++) Tab[q] = q + 1;
//     int n = 0;
//     StreamWriter file = new StreamWriter("bin_avg.csv");
//     for (int i = 10; i <= 21; i++)
//     {
//         n += 100000;
//         Count_Op = 0;
//         for (int j = 0; j < n; j++)
//         {
//             BinSearch(n, Tab[j]);
//         }
//         file.WriteLine("{0}; {1}", n, ((double)Count_Op) / n);
//         Console.WriteLine("{0}; {1}", n, ((double)Count_Op) / n);
//     }
//     file.Close();
// }
//
//
//
//
//
```

Wyszukiwanie binarne średnie
(zliczanie operacji)

n	Count_Op
100000	15,69
200000	16,69
300000	17,25
400000	17,69
500000	17,95
600000	18,25
700000	18,50
800000	18,69
900000	18,84
1000000	18,95
1100000	19,09
1200000	19,25
1300000	19,39
1400000	19,50
1500000	19,60



Wyszukiwanie binarne średnie (instrumentacja czasowa)

Time/=100,000,000

```
const int Max_Value_Int = 268435455;
static int[] Tab;

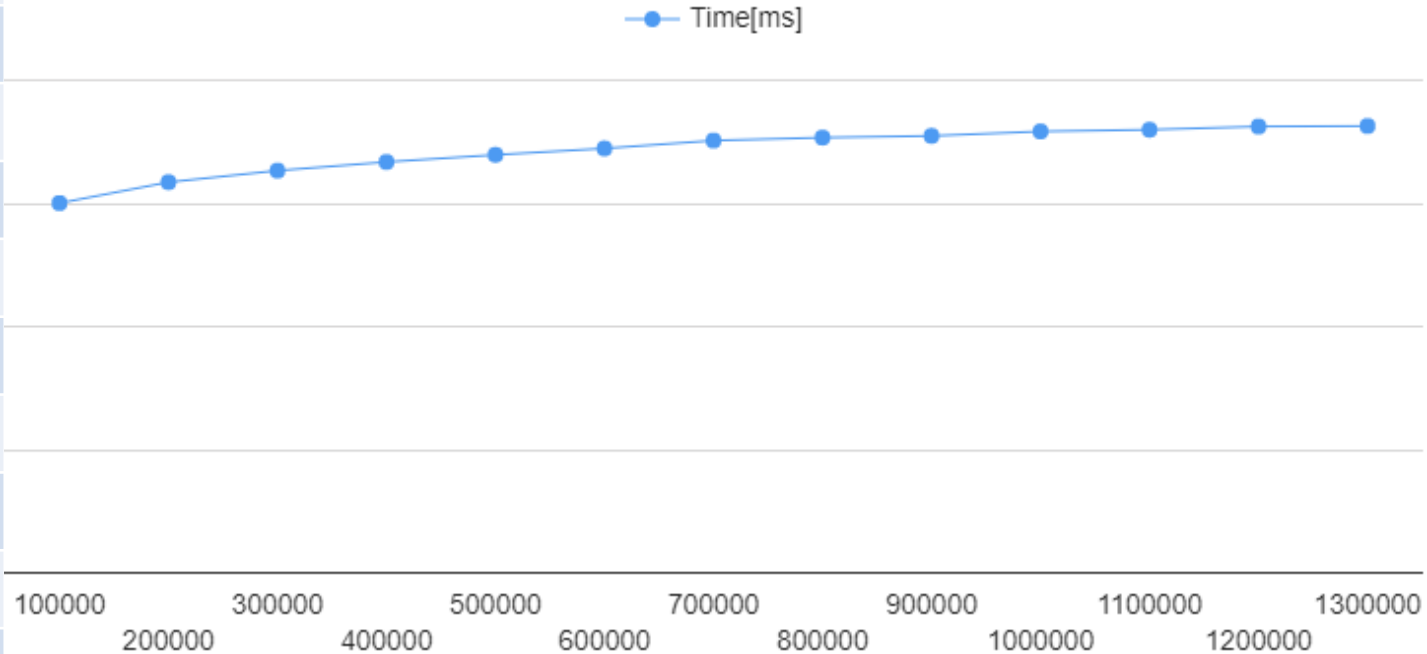
static int BinSearch(int N, int Number){slajd nr. 2}

static void Main()
{
    Tab = new int[Max_Value_Int];
    for (int q = 0; q < Max_Value_Int; q++) Tab[q] = q + 1;
    int n = 0;
    Stopwatch Stoper = new Stopwatch(); //inicjacja klasy Stopwatch() ~~tworzymy stoper
    StreamWriter file = new StreamWriter("bin_avg_czas.csv"); //tworzymy plik "bin_avg_czas" z rozszerzeniem csv
    for (int i = 10; i <= 14; i++)
    {
        n += 100000;
        Stoper.Restart(); //ustawienie czasu stopera na 0 i włączenie
        for (int k = 0; k < 100; k++) for (int j = 0; j < n; j++) BinSearch(n, Tab[j]);
        Stoper.Stop(); //zatrzymanie stopera
        double czas = Stoper.ElapsedMilliseconds;
        file.WriteLine("{0}; {1}", n, (czas/n)*1000000); //zapis danych w pliku
        Console.WriteLine("{0}; {1}", n, (czas/n)*1000000); //wyświetlenie danych na konsoli
    }
    file.Close();
}
```

Wyszukiwanie binarne średnie
(instrumentacja czasowa)

Time/=100,000,000

n	Time[ms]
100000	15,01
200000	15,86
300000	16,33
400000	16,68
500000	16,97
600000	17,23
700000	17,55
800000	17,67
900000	17,74
1000000	17,92
1100000	17,99
1200000	18,12
1300000	18,14



Wyszukiwanie liniowe

(przed instrumentacją)

```
static int LinSearch(int N, int Number)
{
    for (int i = 0; i < N; i++)
    {
        if (Tab[i] == Number) return i;    //zwraca pozycje szukanej liczby
    }
    return -1;
}
```

Wyszukiwanie liniowe pesymistyczne (zliczanie operacji)

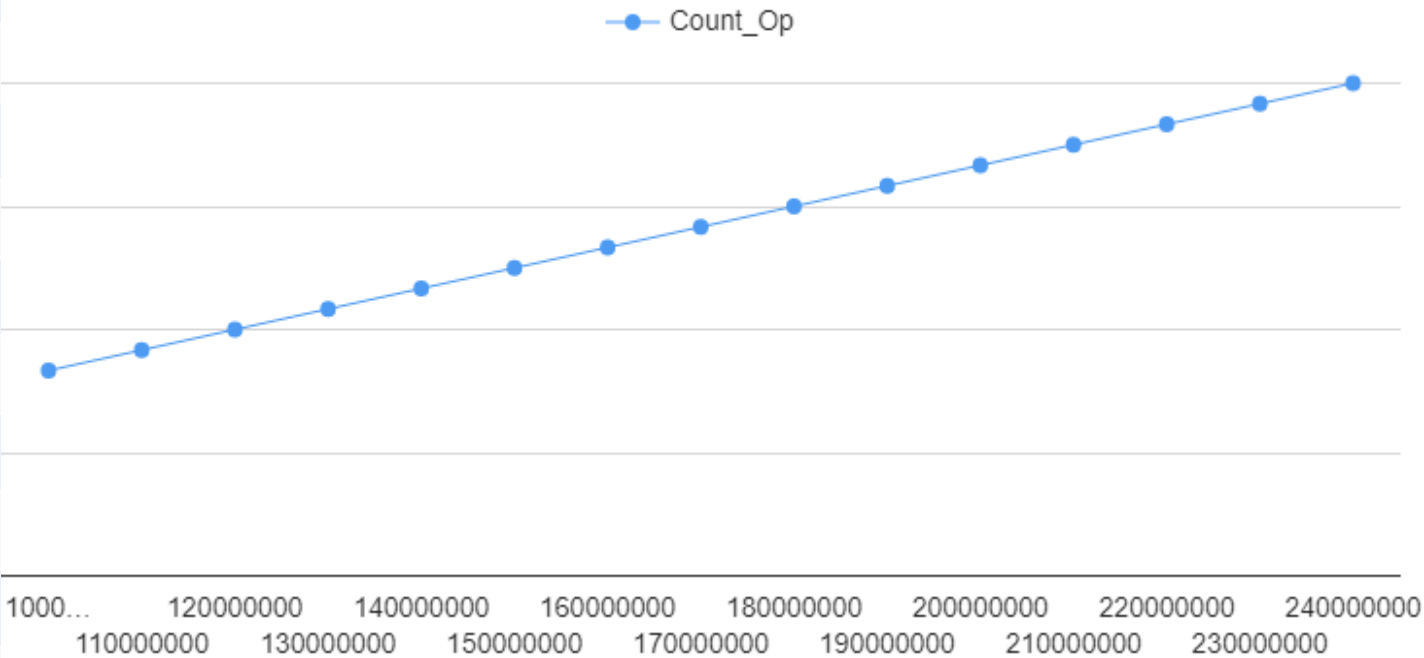
```
const int Max_Value_Int = 268435455;
static int[] Tab;
static long Count_Op;

static int LinSearch(int N, int Number)
{
    for (int i = 0; i < N; i++)
    {
        Count_Op++;
        if (Tab[i] == Number) return i;
    }
    return -1;
}
```

```
//
//
//
// static void Main()
// {
//     Tab = new int[Max_Value_Int];
//     for (int q = 0; q < Max_Value_Int; q++) Tab[q] = q + 1;
//     StreamWriter file = new StreamWriter("lin_pes.csv");
//     for (int i = 100; i <= 200; i += 10)
//     {
//         int n = i * 1000000;
//         Count_Op = 0;
//         LinSearch(n, 0);
//         file.WriteLine("{0}; {1}", n, Count_Op);
//         Console.WriteLine("{0}; {1}", n, Count_Op);
//     }
//     file.Close();
// }
//
//
//
//
```

Wyszukiwanie liniowe pesymistyczne
(zliczanie operacji)

n	Count_Op
100000000	100000000
110000000	110000000
120000000	120000000
130000000	130000000
140000000	140000000
150000000	150000000
160000000	160000000
170000000	170000000
180000000	180000000
190000000	190000000
200000000	200000000
210000000	210000000
220000000	220000000
230000000	230000000
240000000	240000000



Wyszukiwanie liniowe pesymistyczne (instrumentacja czasowa)

Time*=40

```
const int Max_Value_Int = 268435455;
static int[] Tab;

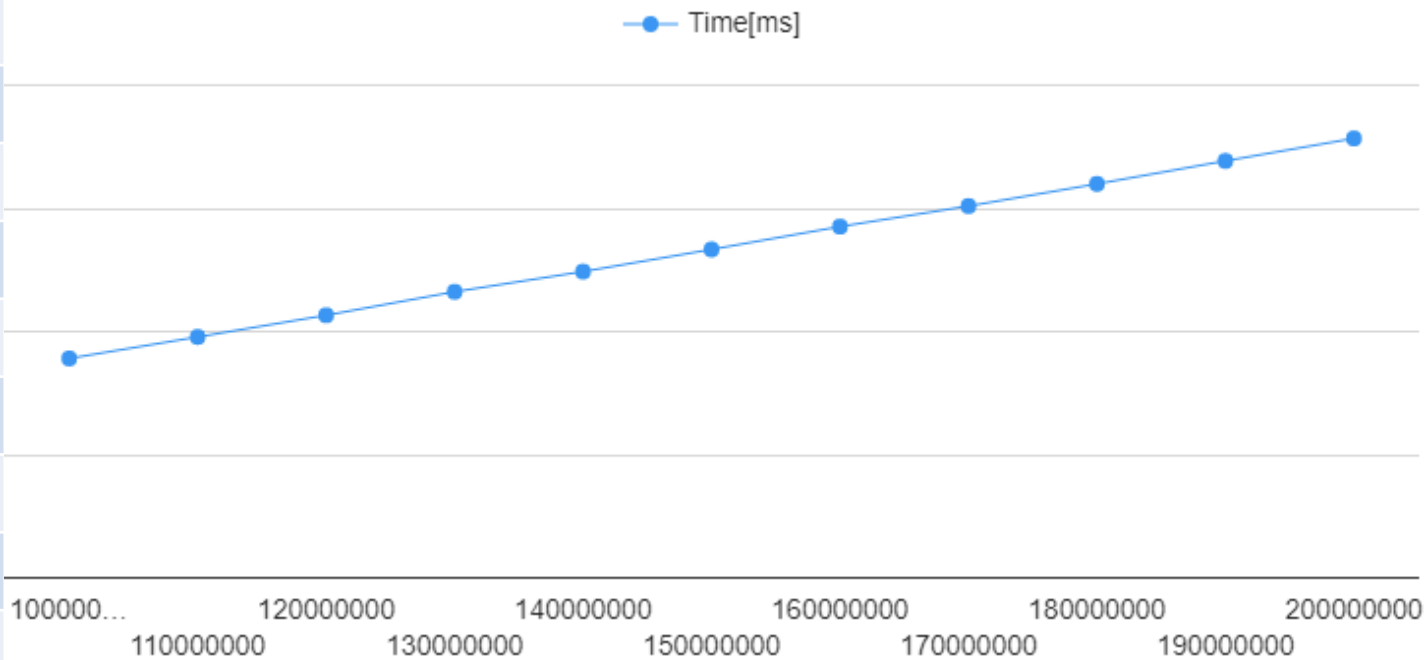
static int LinSearch(int N, int Number){slajd nr. 7}

static void Main()
{
    Tab = new int[Max_Value_Int];
    for (int q = 0; q < Max_Value_Int; q++) Tab[q] = q + 1;
    Stopwatch Stoper = new Stopwatch();
    StreamWriter file = new StreamWriter("lin_pes_czas.csv");
    for (int i = 100; i <= 200; i += 10)
    {
        int n = i * 1000000; //wielkość tablicy, w której pesymistycznie będziemy szukać elementu
        Stoper.Restart();
        for (int j = 0; j < 25; j++) LinSearch(n, 0);
        Stoper.Stop();
        file.WriteLine("{0}; {1}", n, Stoper.ElapsedMilliseconds);
        Console.WriteLine("{0}; {1}", n, Stoper.ElapsedMilliseconds);
    }
    file.Close();
}
```

Wyszukiwanie liniowe pesymistyczne
(instrumentacja czasowa)

Time*=40

n	Time[ms]
100000000	13,36
110000000	14,67
120000000	15,99
130000000	17,42
140000000	18,65
150000000	20,00
160000000	21,39
170000000	22,64
180000000	23,99
190000000	25,39
200000000	26,76



Wyszukiwanie liniowe średnie (zliczanie operacji)

```
const int Max_Value_Int = 268435455;
static int[] Tab;
static long Count_Op;

static int LinSearch(int N, int Number)
{
    for (int i = 0; i < N; i++)
    {
        Count_Op++;
        if (Tab[i] == Number) return i;
    }
    return -1;
}
```

```
static void Main()
{
    Tab = new int[Max_Value_Int];
    for (int q = 0; q < Max_Value_Int; q++) Tab[q] = q + 1;
    StreamWriter file = new StreamWriter("lin_avg.csv");
    for (int i = 100; i <= 200; i += 10)
    {
        int n = i * 100; //sizeTab
        Count_Op = 0;
        for (int j = 0; j < n; j++)
        {
            LinSearch(n, Tab[j]);
        }
        file.WriteLine("{0}; {1}", n, ((double)Count_Op) / n);
        Console.WriteLine("{0}; {1}", n, ((double)Count_Op) / n);
    }
    file.Close();
}
```


Wyszukiwanie liniowe średnie
(zliczanie operacji)

n	Count_Op
10000	5000,5
11000	5500,5
12000	6000,5
13000	6500,5
14000	7000,5
15000	7500,5
16000	8000,5
17000	8500,5
18000	9000,5
19000	9500,5
20000	10000,5



Wyszukiwanie liniowe średnie (instrumentacja czasowa)

Time/=400

```
const int Max_Value_Int = 268435455;
static int[] Tab;

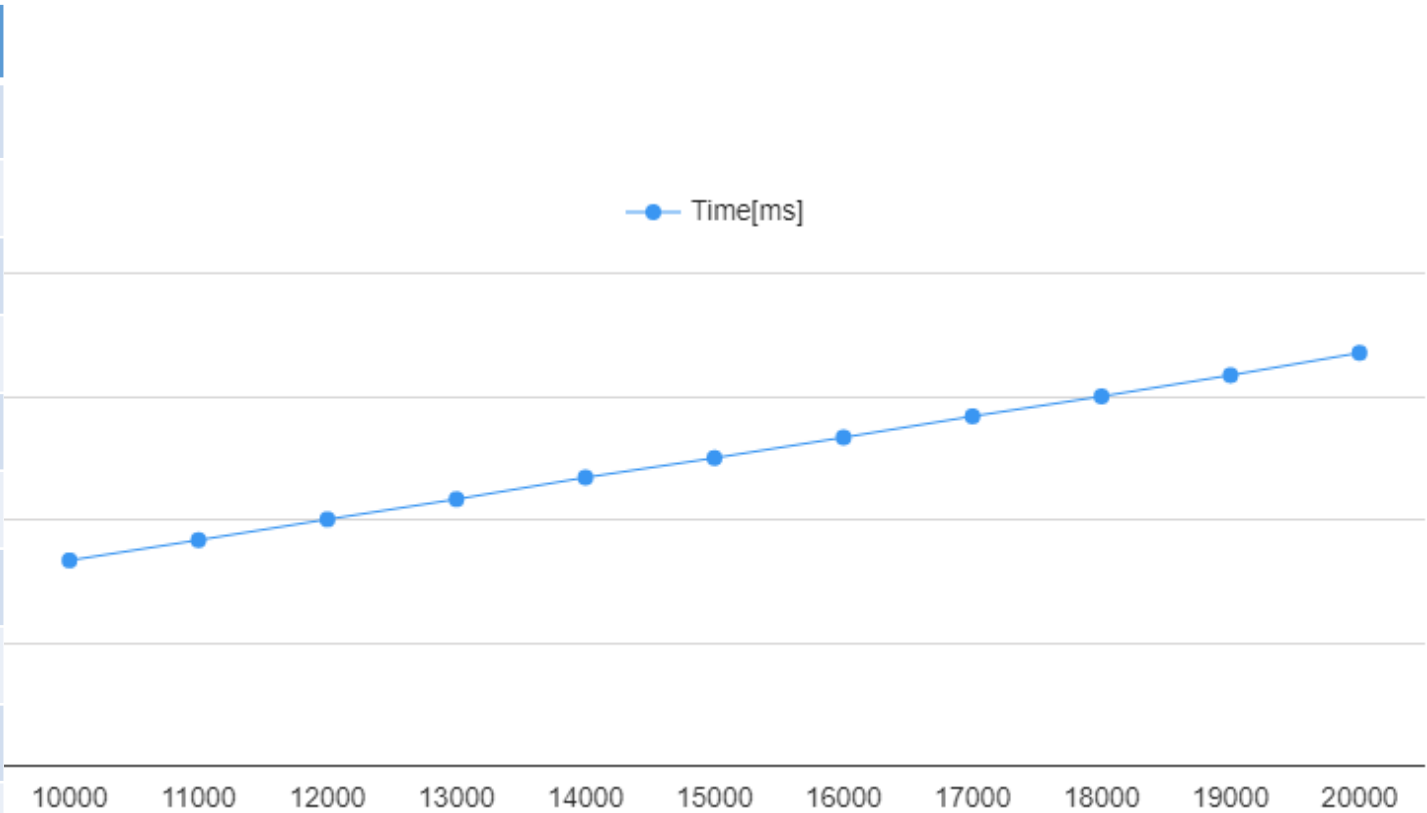
static int LinSearch(int N, int Number){slajd nr. 7}

static void Main()
{
    Tab = new int[Max_Value_Int];
    for (int q = 0; q < Max_Value_Int; q++) Tab[q] = q + 1;
    Stopwatch Stoper = new Stopwatch();
    StreamWriter file = new StreamWriter("lin_avg_czas.csv");
    for (int i = 100; i <= 200; i += 10)
    {
        int n = i * 100; //wielkość tablicy, w której będziemy szukać elementu
        Stoper.Restart();
        for (int k = 0; k < 40; k++) for (int j = 0; j < n; j++) LinSearch(n, Tab[j]);
        Stoper.Stop();
        double czas = Stoper.ElapsedMilliseconds;
        file.WriteLine("{0}; {1}", n, (czas/n)*10);
        Console.WriteLine("{0}; {1}", n, (czas/n)*10);
    }
    file.Close();
}
```

Wyszukiwanie liniowe pesymistyczne
(instrumentacja czasowa)

Time/=400

n	Time[ms]
10000	10,01
11000	11,00
12000	12,01
13000	12,99
14000	14,05
15000	15,00
16000	16,00
17000	17,03
18000	18,00
19000	19,03
20000	20,12



Wnioski

Wyszukiwanie binarne

Jest algorytmem opierającym się na metodzie dziel i zwyciężaj. Posiada klasę złożoności obliczeniowej $O(\log n)$. Przy każdym obiegu, skraca ilość elementów do przeszukania o połowę.

Dla pesymistycznego wyszukiwania jego złożoność wzgl. ilości elementów do przeszukania rośnie logarytmicznie, natomiast średnia potrzebna ilość operacji rośnie spadkowo (większy zbiór hamuje wzrost ilości obiegów).

Algorytm ten jest jednym z najlepszym dla wyszukiwania, z każdym wzrostem elementów do przeszukania wydajniejszy od liniowego, którego opiszę w następnym kroku.

Jedyną wadą jest warunek uporządkowania liczb.

Wnioski

Wyszukiwanie liniowe

Najprostszy algorytm wyszukiwania informacji w ciągu danych, wyszukiwanie zwane również sekwencyjnym.

Algorytm polega na przeglądaniu kolejnych elementów zbioru. Jego „zaletą” jest brak potrzeby sortowania zbioru. Natomiast ogromna wada, to pesymistyczna złożoność. Przy „n” elementów do przeszukania musi wykonać „n” operacji, co diametralnie wpływa na jego wydajność.

Złożoność algorytmu rośnie liniowo wzgl. wielkości zbioru [klasa złożoności obliczeniowej jest równa $O(n)$].

Wyszukiwanie liniowe może być jedynym sposobem wyszukiwania, gdy nie wiadomo niczego na temat kolejności kluczy, lecz stosunkowo jest mało wydajny dla dużej liczby danych.

Dla wyszukiwania binarnego

Kody i ocena złożoności dla $n = (1 \ll i) - 1$

Kolejnych potęg liczby „2”

* n -(zakres w jakim będziemy szukać danej liczby)

[illegible]

```

static void Main()
{
    Tab = new int[Max_Value_Int];
    for (int i = 0; i < Max_Value_Int; i++) Tab[i] = i + 1;
    StreamWriter file_csv = new StreamWriter("bin_pes.csv");
    for (int i = 10; i <= 28; i++)
    {
        int n = (1 << i) - 1; // 2^i - 1 //i- (2^i)-1=n
        Count_Op = 0;
        BinSearch(n, 0);
        file_csv.WriteLine("{0}; {1}", i, n, Count_Op);
        Console.WriteLine("{0}; {1}", i, n, Count_Op);
    }
    file_csv.Close();
}

```

//utworzenie pliku csv

//zapis w pliku

//n –w uporządkowanym szeregu liczb (naturalnych) od 0 do n będziemy szukać danego elementu

//zapis i zamknięcie pliku

*Wyszukiwanie binarne pesymistyczne
(zliczanie operacji)

Przy zbiorach 2^i złożoność algorytmu = i

i	n	Count_Op
10	1023	10
11	2047	11
12	4095	12
13	8191	13
14	16383	14
15	32767	15
16	65535	16
17	131071	17
18	262143	18
19	524287	19
20	1048575	20
21	2097151	21

*Wyszukiwanie binarne pesymistyczne (instrumentacja czasowa)

Time/=100.000.000

```
const int Max_Value_Int = 268435455;
static int[] Tab;

static int BinSearch(int N, int Number){slajd nr. 2}

static void Main()
{
    Tab = new int[Max_Value_Int];
    for (int q = 0; q < Max_Value_Int; q++) Tab[q] = q + 1;
    Stopwatch Stoper = new Stopwatch(); //inicjacja klasy Stopwatch() ~~tworzymy stoper
    StreamWriter file = new StreamWriter("bin_pes_czas.csv"); //tworzymy plik "bin_pes_czas" z rozszerzeniem csv
    for (int i = 10; i <= 18; i++)
    {
        int n = (1 << i) - 1; // 2^i - 1 //wielkość tablicy (2^i)-1
        Stoper.Restart(); //ustawienie czasu stopera na 0 i włączenie
        for (int j = 0; j < 100000000; j++) BinSearch(n, 0); //wywołanie szukania binarnego (1mln dla dokł. wyników)
        Stoper.Stop(); //zatrzymanie stopera
        file.WriteLine("{0}; {1}; {2}", i, n, Stoper.ElapsedMilliseconds); //zapis danych w pliku
        Console.WriteLine("{0}; {1}; {2}", i, n, Stoper.ElapsedMilliseconds); //wyświetlenie danych na konsoli
    }
    file.Close();
}
```



//czas w ms

*Wyszukiwanie binarne średnie
(instrumentacja czasowa)

Time/=100.000.000

Przy zbiorach 2^i koszt czasu algorytmu $\approx i$

i	n	Time[s]
10	1023	9,81
11	2047	10,66
12	4095	11,57
13	8191	12,97
14	16383	13,83
15	32767	14,95
16	65535	16,14
17	131071	16,70
18	262143	17,84

*Wyszukiwanie binarne średnie (zliczanie operacji)

```
const int Max_Value_Int = 268435455;
static int[] Tab;
static long Count_Op;

static int BinSearch(int N, int Number)
{
    int Left = 0;
    int Right = N - 1;
    int Middle;
    while (Left <= Right)
    {
        Count_Op++;
        Middle = (Left + Right) >> 1; //
        if (Tab[Middle] == Number) return Middle;
        else if (Tab[Middle] > Number) Right = Middle - 1;
        else Left = Middle + 1;
    }
    return -1;
}
```

```
//
//
// static void Main()
// {
//     Tab = new int[Max_Value_Int];
//     for (int i = 0; i < Max_Value_Int; i++) Tab[i] = i + 1;
//     StreamWriter file = new StreamWriter("bin_avg.csv");
//     for (int i = 10; i <= 28; i++)
//     {
//         int n = (1 << i) - 1; // 2^i - 1
//         Count_Op = 0;
//         for (int j = 0; j < n; j++)
//         {
//             BinSearch(n, Tab[j]);
//         }
//         file.WriteLine("{0}; {1}; {2}", i, n, ((double)Count_Op) / n);
//         Console.WriteLine("{0}; {1}; {2}", i, n, ((double)Count_Op) / n);
//     }
//     file.Close();
// }
//
//
//
//
//
//
```

*Wyszukiwanie binarne średnie
(zliczanie operacji)

Przy zbiorach 2^i złożoność algorytmu = $i-1$

i	n	Count_Op
10	1023	9
11	2047	10
12	4095	11
13	8191	12
14	16383	13
15	32767	14
16	65535	15
17	131071	16
18	262143	17
19	524287	18
20	1048575	19
21	2097151	20

*Wyszukiwanie binarne średnie (instrumentacja czasowa)

Time/=10.000.000

```
const int Max_Value_Int = 268435455;
static int[] Tab;

static int BinSearch(int N, int Number){slajd nr. 2}

static void Main()
{
    Tab = new int[Max_Value_Int];
    for (int i = 0; i < Max_Value_Int; i++) Tab[i] = i + 1;
    Stopwatch Stoper = new Stopwatch(); //inicjacja klasy Stopwatch() ~~tworzymy stoper
    StreamWriter file = new StreamWriter("bin_avg_czas.csv"); //tworzymy plik "bin_avg_czas" z rozszerzeniem csv
    for (int i = 15; i <= 26; i++)
    {
        int n = (1 << i) - 1; // 2^i - 1    //wielkość tablicy (2^i)-1
        Stoper.Restart(); //ustawienie czasu stopera na 0 i włączenie
        for (int j = 0; j < n; j++) BinSearch(n, Tab[j]);
        Stoper.Stop(); //zatrzymanie stopera
        double czas = Stoper.ElapsedMilliseconds;
        file.WriteLine("{0}; {1}; {2}", n, (czas/n)*10000000); //zapis danych w pliku
        Console.WriteLine("{0}; {1}; {2}", n, (czas/n)*10000000); //wyświetlenie danych na konsoli
    }
    file.Close();
}
```

*Wyszukiwanie binarne średnie
(zliczanie operacji)

Przy zbiorach 2^i koszt czasu algorytmu $\approx i$

i	n	Count_Op
15	32767	12,21
16	65535	15,26
17	131071	16,02
18	262143	17,17
19	524287	18,31
20	1048575	19,55
21	2097151	20,84
22	4194303	22,22
23	8388607	23,32
24	16777215	24,55
25	33554431	25,86
26	67108863	27,24