# Projekt 1

Bartosz Kaźmierczak Gr. K31

**Wyszukiwanie Binarne**

Implementacja wyszukiwania binarnego:

```
1 odwołanie
static int BinarneBezInstrumentacji(int[] tab, int Wartosc, int WielkoscTablicy)
{

    int Lewa = 0;
    int Prawa = WielkoscTablicy - 1;
    int Srodek = 0;
    while (Lewa <= Prawa)
    {
        Srodek = (Lewa + Prawa) / 2;
        if (tab[Srodek] == Wartosc)
        {

            return Srodek;
        }
        else if (tab[Srodek] < Wartosc)
        {
            Lewa = Srodek + 1;
        }
        else
        {
            Prawa = Srodek - 1;
        }
    }
    return 0;
}
```

# Przypadek Pesymistyczny:

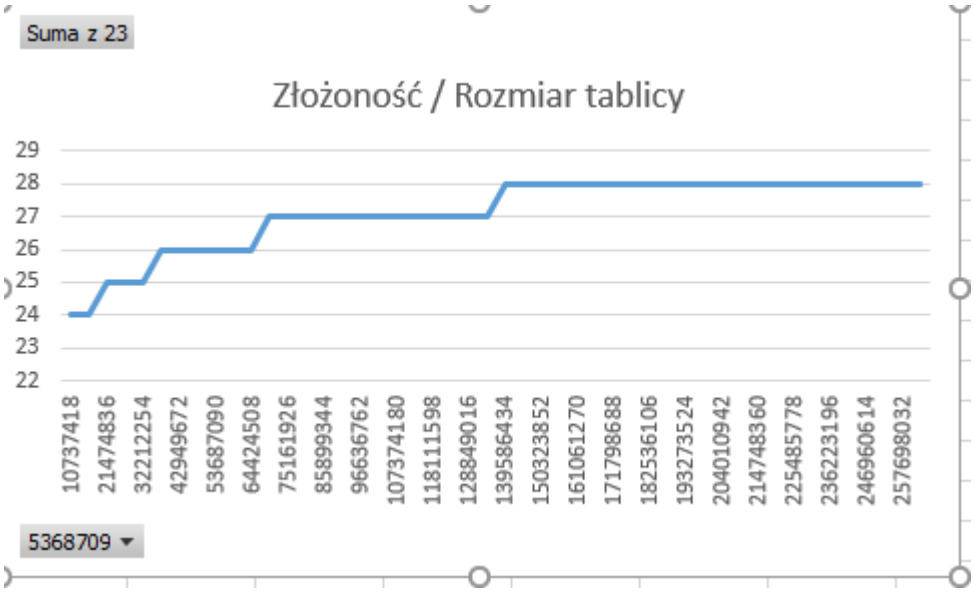
```
class Program
{

    private static object indeksLiczby;
    static ulong suma;
    static double CzasRoznica;
    static ulong dlugosc;
    static long ilosc;
    1 odwołanie
    static int Binarne(int[] tab, int Wartosc, int WielkoscTablicy)
    {
        ilosc = 0;
        suma = 0;
        dlugosc = 0;
        int Lewa = 0;
        int Prawa = WielkoscTablicy - 1;
        int Srodek = 0;
        while (Lewa <= Prawa)
        {
            Srodek = (Lewa + Prawa) / 2;
            if (tab[Srodek] == Wartosc)
            {
                ilosc++;
                suma += (ulong)tab[ilosc];
                dlugosc += (ulong)Math.Pow(2, ilosc - 1);

            }
            else if (tab[Srodek] < Wartosc)
            {
                ilosc++;
                suma += (ulong)tab[ilosc];
                Lewa = Srodek + 1;
                dlugosc += (ulong)Math.Pow(2, ilosc - 1);
            }
            else
            {
                ilosc++;
                suma += (ulong)tab[ilosc];
                Prawa = Srodek - 1;
                dlugosc += (ulong)Math.Pow(2, ilosc - 1);
            }
        }
        return 0;
    }
```

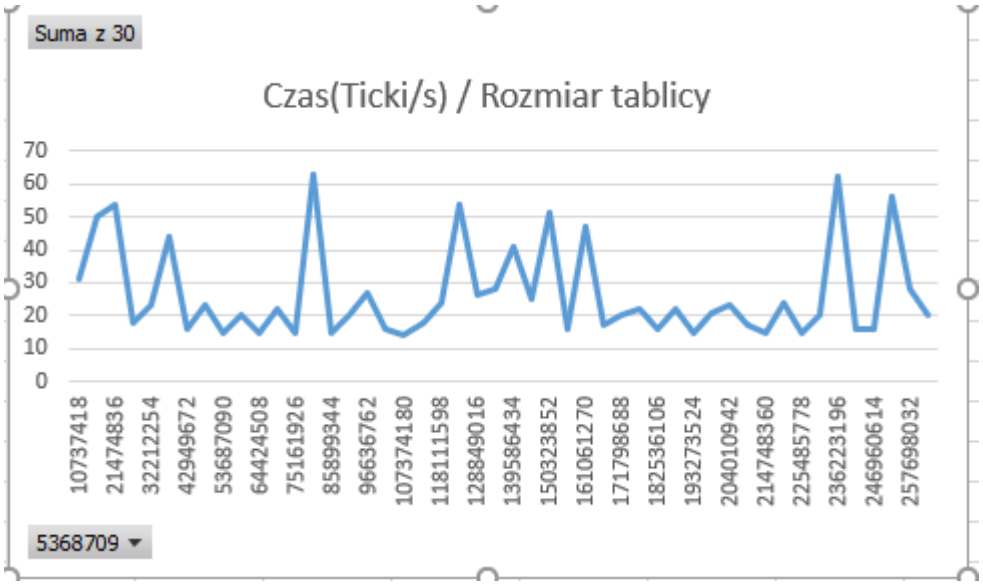
```
Odwołania: 0
static void Main(string[] args)
{
    ilosc = 0;
    suma = 0;
    dlugosc = 0;
    Random r = new Random();
    int LosowaLiczba = r.Next();
    int Liczba = 0;

    Console.WriteLine("Wyszukiwanie binarne - wariant pesymistyczny: ");

    Console.WriteLine();

    for (int i = 5368709; i < 268435456; i += 5368709)
    {
        int szukana = -1;

        Liczba++;
        int[] tablica = new int[i];
        for (int j = 0; j < tablica.Length; j++)
        {
            tablica[j] = j + 1;

        }

        Array.Sort(tablica);

        //var Czas = new System.Diagnostics.Stopwatch();

        //Czas.Start();
        //BinarneBezInstrumentacji(tablica, szukana, tablica.Length);
        //Czas.Stop();
        //CzasRoznica = Czas.ElapsedTicks;
        //indeksLiczby = Binarne(tablica, szukana, tablica.Length);
        //ulong Wynik = suma / (ulong)ilosc;

        Console.WriteLine("Nr porządkowy {0}, indeks  {1} Złożoność {2}, Czas {3}  ", Liczba, indeksLiczby, Wynik, CzasRoznica);
```

Kod użyty przy pomiarze czasu(Tiki procesora na sekundę):

```
var Czas = new System.Diagnostics.Stopwatch();

Czas.Start();
BinarneBezInstrumentacji(tablica, szukana, tablica.Length);
Czas.Stop();
CzasRoznica = Czas.ElapsedTicks;
indeksLiczby = Binarne(tablica, szukana, tablica.Length);
ulong Wynik = suma / (ulong)ilosc;
```

Rezultaty:

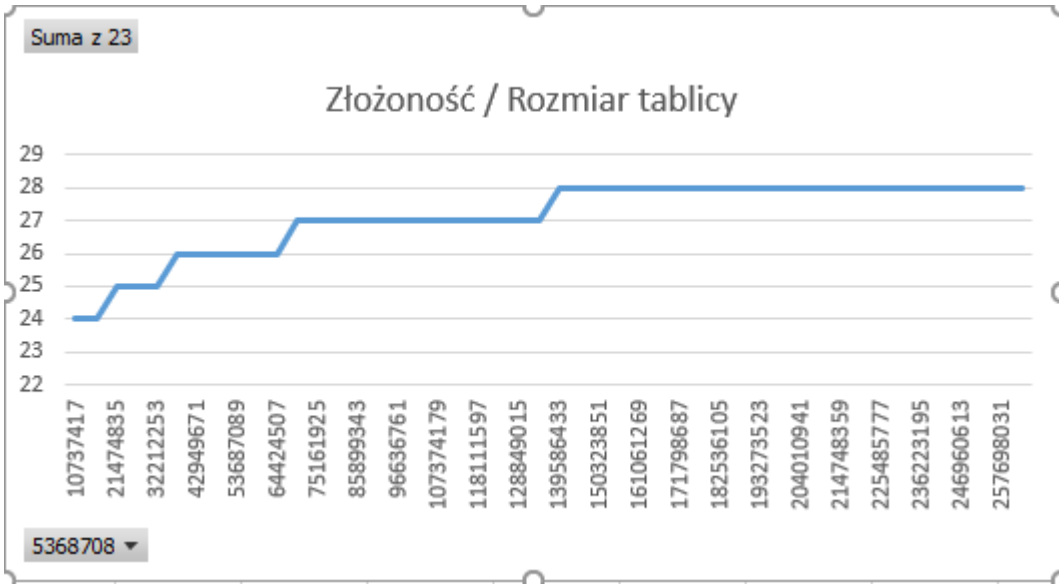


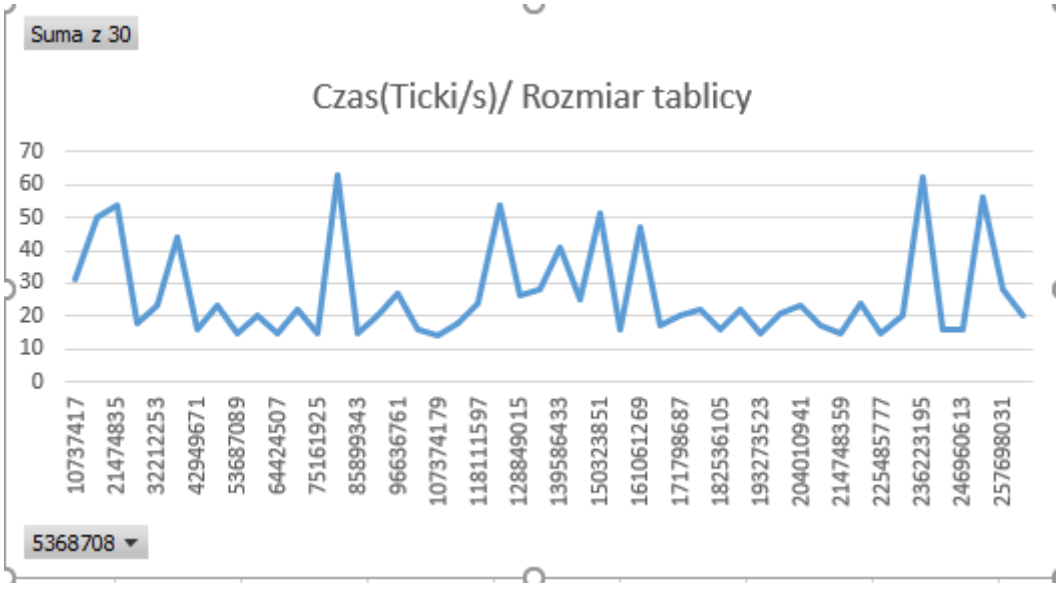



## Przypadek Średni:

Do badania złożoności została użyta dokładnie ta sama metoda co podczas przypadku pesymistycznego

Rezultaty:




# Wyszukiwanie liniowe:

Implementacja wyszukiwania liniowego:

```
static int LinioweBezInstrumentacji(int[] tab, int Szukana)
{
    for (int i = 0; i < tab.Count(); i++)
    {
        if (tab[i] == Szukana)
        {
            return i;
        }

    }
    return 0;
}
```

Przypadek pesymistyczny:

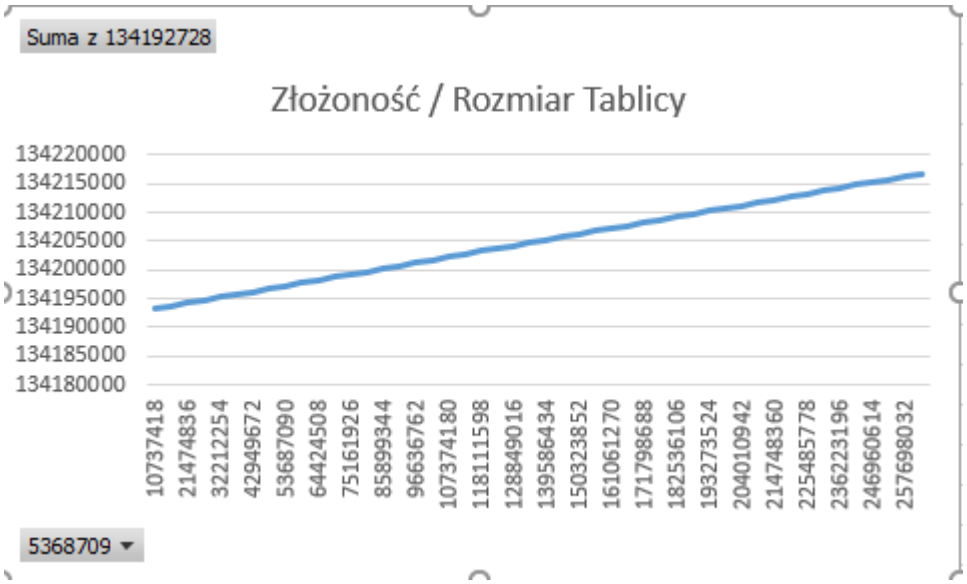
```
private static long CzasStart;
private static object indeksLiczby;
private static long CzasStop;
private static double CzasRoznica;
static long suma;
static long ilosc;

Odwołania: 0
static int LinioweBezInstrumentacji(int[] tab, int Szukana)
{
    for (int i = 0; i < tab.Count(); i++)
    {
        if (tab[i] == Szukana)
        {
            return i;
        }

    }
    return 0;
}
Odwołania: 0
static int Liniowe(int[] tab, int Szukana)
{
    for (int i = 0; i < tab.Count(); i++)
    {
        ilosc++;
        if (tab[i] == Szukana)
        {
            suma += tab[i];
            return i;
        }

    }
    return 0;
}
```

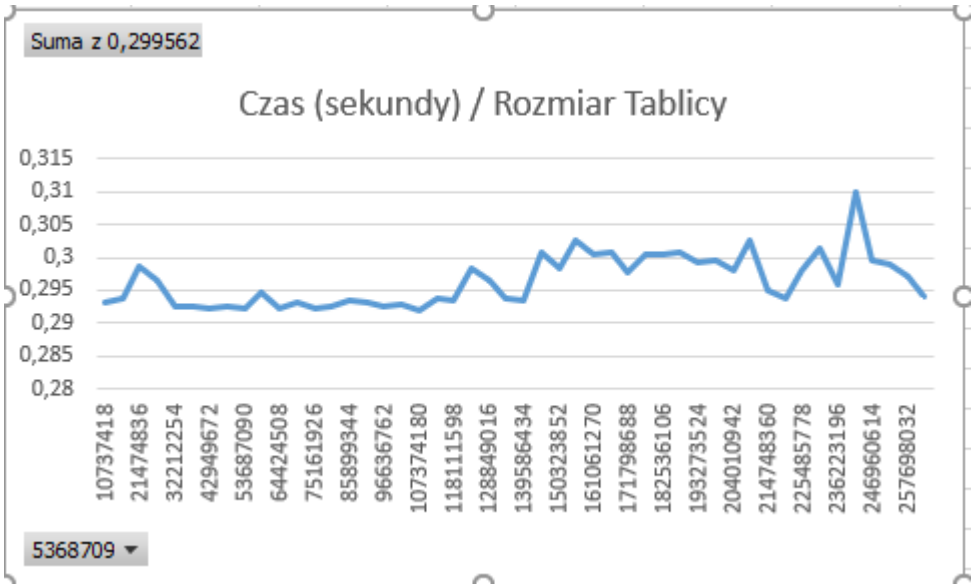
```
static void Main(string[] args)
{
    Random r = new Random();
    int LosowaLiczba = r.Next();
    int szukana = -1;
    int Liczba = 0;
    Console.WriteLine("Wyszukiwanie liniowe - wariant pesymistyczny: ");
    Console.WriteLine();
    for (int i = 5368709; i < 268435456; i += 5368709)
    {

        Liczba++;
        int[] tablica = new int[i];

        for (int j = 0; j < tablica.Length; j++)
        {
            tablica[j] = j;
            suma += tablica[j];
        }



        //CzasStart = Stopwatch.GetTimestamp();
        //LinioweBezInstrumentacji(tablica, szukana);
        //CzasStop = Stopwatch.GetTimestamp();
        //indeksLiczby = Liniowe(tablica, szukana);
        //CzasRoznica = (CzasStop - CzasStart) * (1.0 / Stopwatch.Frequency);
        //Console.WriteLine("Nr porządkowy {0}, indeks  {1} Złożoność {2}, Czas {3}  ", Liczba, indeksLiczby, ilosc, CzasRoznica);

    }
    Console.ReadKey();
```

Kod użyty przy pomiarze czasu:

```
CzasStart = Stopwatch.GetTimestamp();
LinioweBezInstrumentacji(tablica, szukana);
CzasStop = Stopwatch.GetTimestamp();
indeksLiczby = Liniowe(tablica, szukana);
CzasRoznica = (CzasStop - CzasStart) * (1.0 / Stopwatch.Frequency);
Console.WriteLine("Nr porządkowy {0}, indeks  {1} Złożoność {2}, Czas {3}  ", Liczba, indeksLiczby, ilosc, CzasRoznica);
```

## Rezultaty:



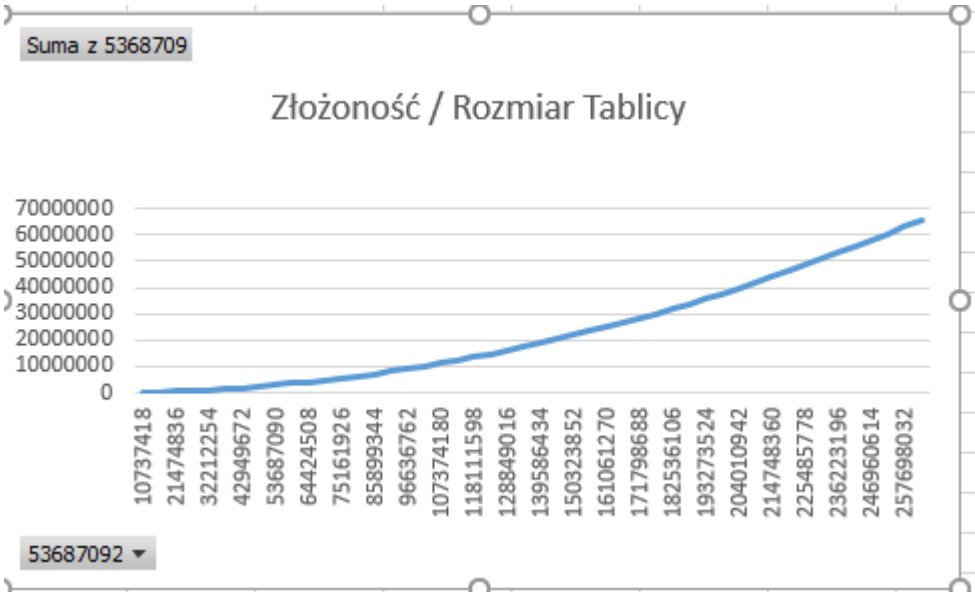


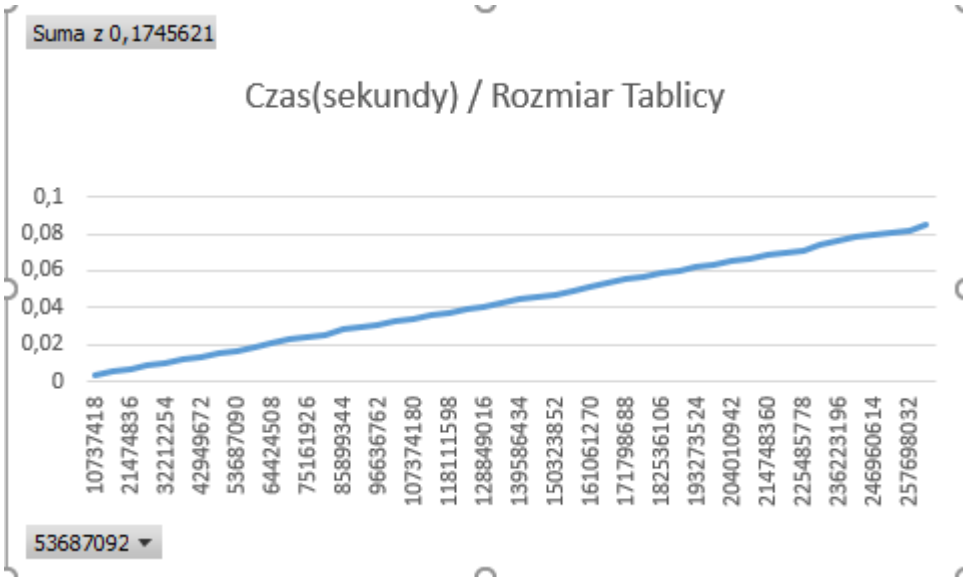

## Przypadek Średni:

Do badania złożoności została użyta dokładnie ta sama metoda co podczas przypadku pesymistycznego.

Złożoność została zbadana przy pomocy średniej arytmetycznej.

Rezultaty:






Wnioski :

- Złożoność w przypadku wyszukiwania pesymistycznym i średnim jest szybsze niż wyszukiwanie liniowe w obu przypadkach.
- Średnia złożoność wyszukiwania binarnego w obu sposobach to O(log n) natomiast w liniowym pesymistyczny to O(n) i średnim O(1/2n+1)
- Algorytm wyszukiwania liniowego może mieć podobną wydajność, jeżeli zostanie użyty do rozwiązania mniejszego rozmiaru problemów.