Oceń złożoność średnią i pesymistyczną wyszukiwania liniowego i binarnego

1. Wyszukiwanie liniowe ocena złożoności pesymistycznej

Implementacja metody obliczającej wyszukiwanie liniowe bez instrumentacji:

```
private static int SimpleSearch(int[] tab, int lookUpValue) //Wyszukiwanie Liniowe przed instrumentacją
{

    for (int i = 0; i < tab.Length; i++)
    {

        if (tab[i] == lookUpValue)
        {
            return i;
        }
    }
    return -1;
}
```

Implementacja metody obliczającej wyszukiwanie liniowe z instrumentacją:

```
private static int iter = 10;
private static int counter; // zmienna globalna wykorzystywana do iteracji instrumentacji
private static int SimpleSearchOptions(int[] tab, int lookUpValue) //Wyszukiwanie Liniowe po instrumentacją
{
    counter = 0;
    for (int i = 0; i < tab.Length; i++)
    {
        counter++;
        if (tab[i] == lookUpValue)
        {
            return i;
        }
    }
    return -1;
}
```

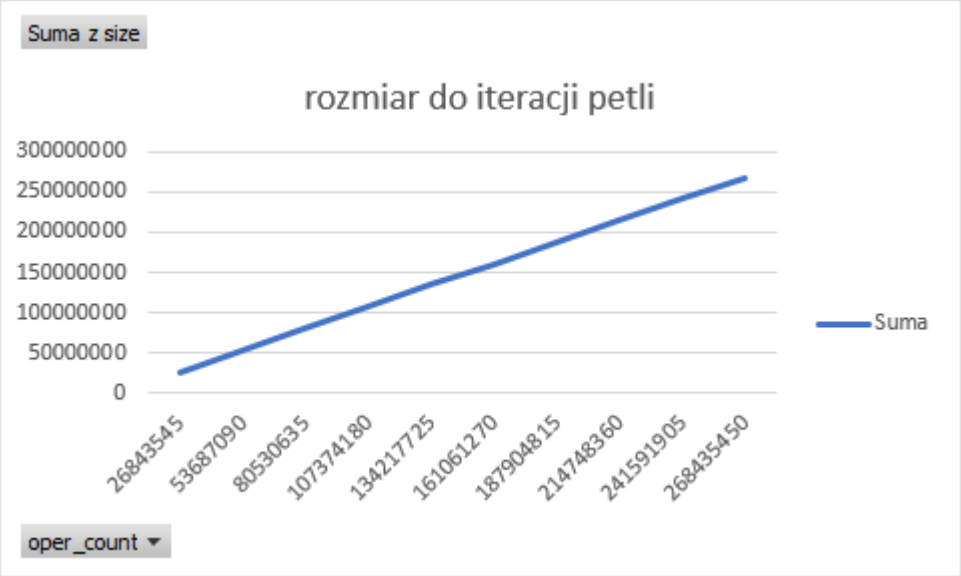
- Metoda main wyszukiwania liniowego w ocenie złożoności pesymistycznej:

```
static void Main(string[] args)
{
    Random random = new Random();
    int lookUpValue = 1001; // wartość szukana
    long min = long.MaxValue;
    long max = long.MinValue;
    long timeElapsed = 0;
    long iterTimeElapsed;

    double ElapsedSeconds;
    int result = 0;
    //Generate tables with random value
    Console.WriteLine("size;lookupvalue;result;time;oper_count");
    //for (int i = 2000000; i < Math.Pow(2,28); i += 100000)
    for (int i = 26843545; i <= 268435450; i += 26843545)
    {
        int[] tab = new int[i];
        for (int k = 0; k < tab.Length; k++)
        {
            tab[k] = random.Next(1, 1000);
        }
        Array.Sort(tab);
        for (int j = 0; j < iter + 2; ++j)
        {
            long start = Stopwatch.GetTimestamp(); // start czasu
            result = SimpleSearch(tab, lookUpValue); // pomiar czasu dla liniowego tutaj robimy bez instrumentacji
            long stop = Stopwatch.GetTimestamp();   // stop czas
            iterTimeElapsed = stop - start;
            timeElapsed += iterTimeElapsed;
            if (iterTimeElapsed < min) min = iterTimeElapsed;
            if (iterTimeElapsed > max) max = iterTimeElapsed;

            result = SimpleSearchOptions(tab, lookUpValue); // wynik instrumentacji
            timeElapsed = (min + max);
        }
        ElapsedSeconds = timeElapsed * (1.0 / (iter * Stopwatch.Frequency));
        Console.WriteLine(i + ";" + lookUpValue + ";" + result + ";" + (ElapsedSeconds.ToString("F4")) + ";" + counter); // tablica; szukana; index; czas; za którym razem
    }
}
```

a) Zebrane dane podczas instrumentacji:

W eksperymencie wykorzystałem 10 punktów pomiarowych

Etykiety wierszy	Suma z size
26843545	26843545
53687090	53687090
80530635	80530635
107374180	107374180
134217725	134217725
161061270	161061270
187904815	187904815
214748360	214748360
241591905	241591905
268435450	268435450



b) Zebrane dane podczas pomiaru czasu:

W eksperymencie wykorzystałem 10 punktów pomiarowych

Etykiety wierszy	Suma z size
0,0157	26843545
0,0238	53687090
0,0318	80530635
0,0394	107374180
0,0465	134217725
0,0556	161061270
0,0629	187904815
0,0719	214748360
0,0782	241591905
0,0855	268435450

2. Wyszukiwanie liniowe ocena złożoności średniej:

Implementacja metody obliczającej wyszukiwanie liniowe bez instrumentacji:

```
private static int SimpleSearch(int[] tab, int lookUpValue) //Wyszukiwanie Liniowe przed instrumentacją
{
    for (int i = 0; i < tab.Length; i++)
    {
        if (tab[i] == lookUpValue)
        {
            return i;
        }
    }
    return -1;
}
```

Implementacja metody obliczającej wyszukiwanie liniowe z instrumentacją:

```
private static int iter = 10;
private static int counter; // zmienna globalna wykorzystywana do iteracji instrumentacji
private static long suma;
private static int SimpleSearchOptions(int[] tab, int lookUpValue) //Wyszukiwanie Liniowe po instrumentacją
{
    counter = 0;
    suma = 0;
    for (int i = 0; i < tab.Length; i++)
    {
        counter++;
        suma += (long)tab[i];
        if (tab[i] == lookUpValue)
        {
            return i;
        }
    }
    return -1;
}
```

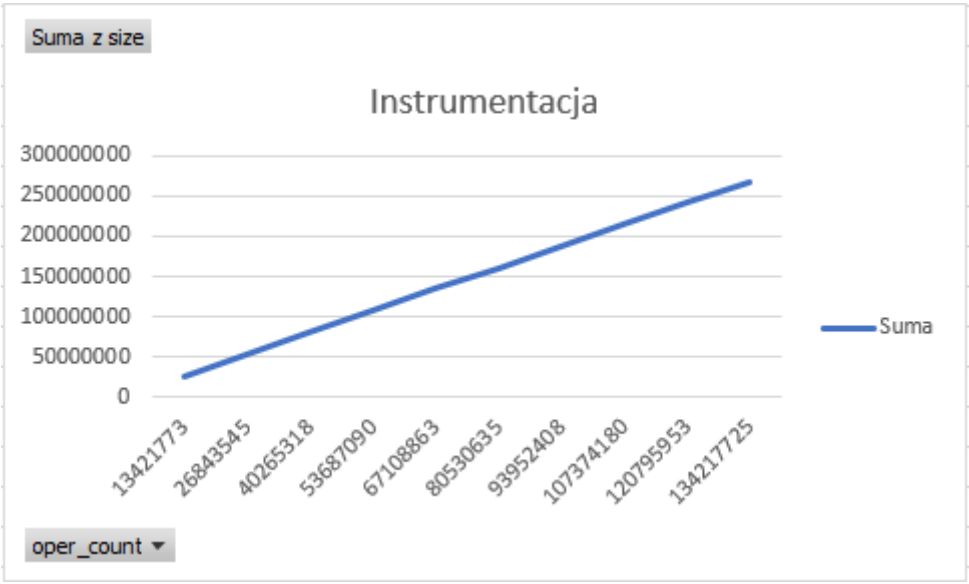
- Metoda main wyszukiwania liniowego w ocenie złożoności średniej:

```
static void Main(string[] args)
{
    Random random = new Random();

    int result = 0; //zmienna ktora przechowuje index liczby znalezionej
    long min = long.MaxValue;
    long max = long.MinValue;
    long timeElapsed = 0;
    long iterTimeElapsed;
    double ElapsedSeconds;
    Console.WriteLine("size;lookupvalue;result;time;oper_count");
    for (int i = 26843545; i <= 268435450; i += 26843545)
    {
        int[] tab = new int[i];
        for (int k = 0; k < tab.Length; k++)
        {
            tab[k] = k + 1;
        }
        Array.Sort(tab); // sortowanie tablicy rosnąco
        int lookUpValue = tab.Length; // wartość szukana
        for (int j = 0; j < iter + 2; ++j)
        {
            long start = Stopwatch.GetTimestamp(); // start czasu
            result = SimpleSearch(tab, lookUpValue); // pomiar czasu dla liniowego tutaj robimy bez instrumentacji
            long stop = Stopwatch.GetTimestamp();   // stop czas
            iterTimeElapsed = stop - start;
            timeElapsed += iterTimeElapsed;
            if (iterTimeElapsed < min) min = iterTimeElapsed;
            if (iterTimeElapsed > max) max = iterTimeElapsed;
            result = SimpleSearchOptions(tab, lookUpValue); // wynik instrumentacji
            timeElapsed = (min + max);
        }
        long wynik = suma / counter;
        ElapsedSeconds = timeElapsed * (1.0 / (iter * Stopwatch.Frequency));
        Console.WriteLine(i + ";" + lookUpValue + ";" + result + ";" + (ElapsedSeconds.ToString("F4")) + ";" + wynik); // tablica; szukana; index; czas; za którym razem
    }
}
```

a) Zebrane dane podczas instrumentacji:

W eksperymencie wykorzystałem 10 punktów pomiarowych

Etykiety wierszy	Suma z size
13421773	26843545
26843545	53687090
40265318	80530635
53687090	107374180
67108863	134217725
80530635	161061270
93952408	187904815
107374180	214748360
120795953	241591905
134217725	268435450



b) Zebrane dane podczas pomiaru czasu:

W eksperymencie wykorzystałem 10 punktów pomiarowych

Etykiety wierszy	Suma z size
0,0156	26843545
0,0247	53687090
0,0311	80530635
0,0396	107374180
0,0496	134217725
0,0512	161061270
0,0624	187904815
0,0704	214748360
0,0783	241591905
0,0876	268435450

3. Wyszukiwanie binarne ocena złożoności pesymistycznej

Implementacja metody obliczającej wyszukiwanie binarne bez instrumentacji:

```
public static int BinarySearchPesymistic(int[] vector, int search)
{
    int left = 0;
    int right = vector.Length - 1;

    while (left <= right)
    {
        int currentPosition = left + (right - left) / 2;

        if (vector[currentPosition] == search)
        {
            return currentPosition;
        }

        if (vector[currentPosition] < search)
        {
            left = currentPosition + 1;
        }

        if (vector[currentPosition] > search)
        {
            right = currentPosition - 1;
        }
    }
    return -1;
}
```

Implementacja metody obliczającej wyszukiwanie binarne z instrumentacją:

```
private static int iter = 10;
private static int counter;
public static int BinarySearchPesymisticOptions(int[] vector, int search)
{
    int left = 0;
    int right = vector.Length - 1;
    counter = 0;
    while (left <= right)
    {
        counter++;
        int currentPosition = left + (right - left) / 2;

        if (vector[currentPosition] == search)
        {

            return currentPosition;
        }

        if (vector[currentPosition] < search)
        {
            counter++;
            left = currentPosition + 1;
        }

        if (vector[currentPosition] > search)
        {
            right = currentPosition - 1;
            counter++;
        }
    }

    return -1;
}
```

- Metoda main wyszukiwania binarnego w ocenie złożoności pesymistycznej:

```
static void Main(string[] args)
{
    Console.WriteLine("size;lookupvalue;result;time;oper_count");
    int result = 0;
    long min = long.MaxValue;
    long max = long.MinValue;
    long timeElapsed = 0;
    long iterTimeElapsed;
    int lookUpValue;
    double ElapsedSeconds;
    //for (int i = 2000000; i < Math.Pow(2, 28); i += 100000)
    for (int i = 26843545; i <= 268435450; i += 26843545)
    {

        int[] tab = new int[i];
        for (int k = 0; k < tab.Length; k++)
        {
            tab[k] = k + 1;
        }
        Array.Sort(tab); // sortowanie tablicy rozsnąco
        lookUpValue = (tab.Length + 2); // wartość szukana
        for (int j = 0; j < iter + 2; ++j)
        {
            long start = Stopwatch.GetTimestamp(); // start czasu
            result = BinarySearchPesymistic(tab, lookUpValue); // pomiar czasu dla binarnego tutaj robimy bez instrumentacji
            long stop = Stopwatch.GetTimestamp();   // stop czas
            iterTimeElapsed = stop - start;
            timeElapsed += iterTimeElapsed;
            if (iterTimeElapsed < min) min = iterTimeElapsed;
            if (iterTimeElapsed > max) max = iterTimeElapsed;
            result = BinarySearchPesymisticOptions(tab, lookUpValue); // wynik instrumentacji
        }
        timeElapsed = (min + max);
        ElapsedSeconds = timeElapsed * (1.0 / (iter * Stopwatch.Frequency));
        Console.WriteLine(i + ";" + lookUpValue + ";" + result + ";" + (ElapsedSeconds.ToString("F0")) + ";" + counter); // tablica; szukana; index; czas; za którym razem
    }
}
```

a) Zebrane dane podczas instrumentacji:

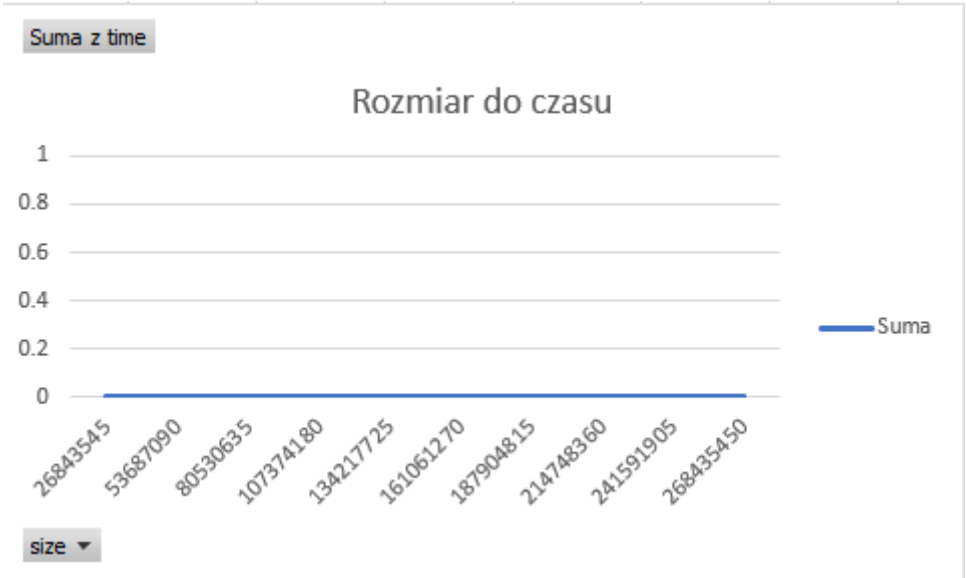
W eksperymencie wykorzystałem 10 punktów pomiarowych

Etykiety wierszy	Suma z oper_count
26843545	50
53687090	52
80530635	54
107374180	54
134217725	54
161061270	56
187904815	56
214748360	56
241591905	56
268435450	56

b) Zebrane dane podczas pomiaru czasu:

W eksperymencie wykorzystałem 10 punktów pomiarowych

Etykiety wierszy	Suma z time
26843545	0
53687090	0
80530635	0
107374180	0
134217725	0
161061270	0
187904815	0
214748360	0
241591905	0
268435450	0



4. Wyszukiwanie binarne ocena złożoności średniej

Implementacja metody obliczającej wyszukiwanie binarne bez instrumentacji:

```
public static int BinarySearchAVG(int[] vector, int search)
{
    int left = 0;
    int right = vector.Length - 1;

    while (left <= right)
    {
        int currentPosition = left + (right - left) / 2;

        if (vector[currentPosition] == search)
        {
            return currentPosition;
        }

        if (vector[currentPosition] < search)
        {
            left = currentPosition + 1;
        }

        if (vector[currentPosition] > search)
        {
            right = currentPosition - 1;
        }
    }
    return -1;
}
```

Implementacja metody obliczającej wyszukiwanie binarne z instrumentacją:

```
private static int iter = 10;
private static int counter;
static ulong wynik;
static ulong dlugosc;
public static int BinarySearchAVGOptions(int[] vector, int search)
{
    int left = 0;
    int right = vector.Length - 1;
    counter = 0;
    wynik = 0;
    dlugosc = 0;

    while (left <= right)
    {
        counter++;
        int currentPosition = left + (right - left) / 2;
        wynik += (ulong)counter * (ulong)Math.Pow(2, counter - 1);
        dlugosc += (ulong)Math.Pow(2, counter - 1);
        if (vector[currentPosition] == search)
        {
            counter++;
            wynik += (ulong)counter * (ulong)Math.Pow(2, counter - 1);
            dlugosc += (ulong)Math.Pow(2, counter - 1);
            wynik = wynik / dlugosc;
            return currentPosition;
        }
        if (vector[currentPosition] < search)
        {
            counter++;
            wynik += (ulong)counter * (ulong)Math.Pow(2, counter - 1);
            dlugosc += (ulong)Math.Pow(2, counter - 1);
            left = currentPosition + 1;
        }
        if (vector[currentPosition] > search)
        {
            counter++;
            wynik += (ulong)counter * (ulong)Math.Pow(2, counter - 1);
            dlugosc += (ulong)Math.Pow(2, counter - 1);
            right = currentPosition - 1;
        }
    }
    return -1;
}
```

- Metoda main wyszukiwania binarnego w ocenie złożoności średniej:

```
static void Main(string[] args)
{
    Console.WriteLine("size;lookupvalue;result;time;oper_count");
    int result = 0;
    long min = long.MaxValue;
    long max = long.MinValue;
    long timeElapsed = 0;
    long iterTimeElapsed;
    int lookUpValue;
    double ElapsedSeconds;
    //for (int i = 2000000; i < Math.Pow(2, 28); i += 100000)
    for (int i = 26843545; i <= 268435450; i += 26843545)
    {

        int[] tab = new int[i];
        for (int k = 0; k < tab.Length; k++)
        {
            tab[k] = k + 1;
        }
        Array.Sort(tab); // sortowanie tablicy rozsnąco
        lookUpValue = (tab.Length); // wartość szukana
        for (int j = 0; j < iter + 2; ++j)
        {
            long start = Stopwatch.GetTimestamp(); // start czasu
            result = BinarySearchAVG(tab, lookUpValue); // pomiar czasu dla binarnego tutaj robimy bez instrumentacji
            long stop = Stopwatch.GetTimestamp();   // stop czas
            iterTimeElapsed = stop - start;
            timeElapsed += iterTimeElapsed;
            if (iterTimeElapsed < min) min = iterTimeElapsed;
            if (iterTimeElapsed > max) max = iterTimeElapsed;
            result = BinarySearchAVGOptions(tab, lookUpValue); // wynik instrumentacji
        }
        timeElapsed = (min + max);
        ElapsedSeconds = timeElapsed * (1.0 / (iter * Stopwatch.Frequency));
        Console.WriteLine(i + ";" + lookUpValue + ";" + result + ";" + (ElapsedSeconds.ToString("F0")) + ";" + wynik); // tablica; szukana; index; czas; za którym razem
    }
}
```

a) Zebrane dane podczas instrumentacji:

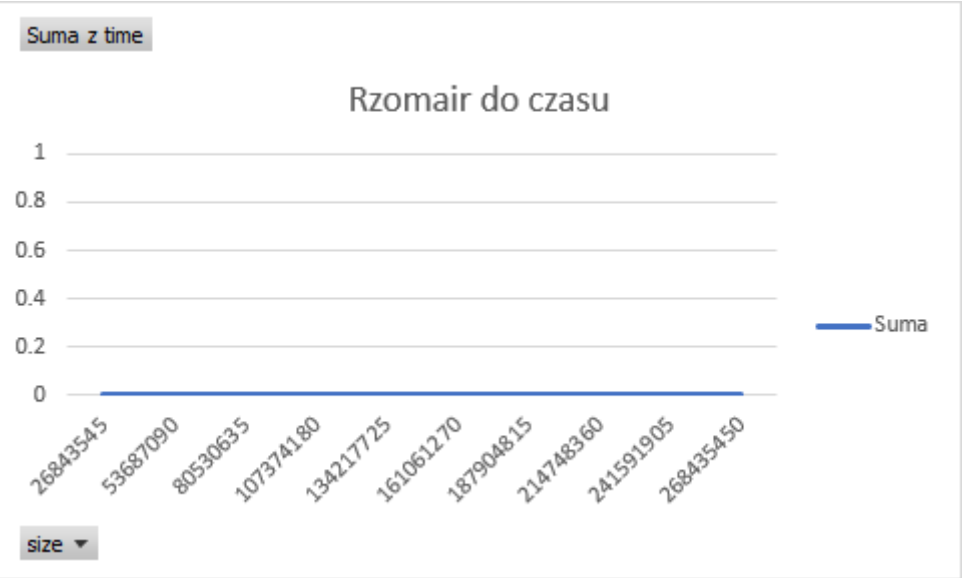
W eksperymencie wykorzystałem 10 punktów pomiarowych

Etykiety wierszy	Suma z oper_count
26843545	49
53687090	51
80530635	53
107374180	53
134217725	53
161061270	55
187904815	55
214748360	55
241591905	55
268435450	55

b) Zebrane dane podczas pomiaru czasu:

W eksperymencie wykorzystałem 10 punktów pomiarowych

Etykiety wierszy	Suma z time
26843545	0
53687090	0
80530635	0
107374180	0
134217725	0
161061270	0
187904815	0
214748360	0
241591905	0
268435450	0



Eksperymenty zostały przeprowadzone na komputerze wyposażonym w procesor Intel Core i7-5600U, w Visual Studio 2017.

Podsumowanie badań:

- wyszukiwanie binarne jest dużo szybsze niż wyszukiwanie liniowe

- w wyszukiwaniu liniowym średni czas nie zmienia się za bardzo stosunku do czasu pesymistycznego,

natomiast różnicę można zauważyć w instrumentacji

- w wyszukiwaniu binarnym średni czas jest identyczny w stosunku do pesymistycznego,

natomiast różnica w instrumentacji jest minimalna.