Projekt 1

Zweryfikować przedstawioną na wykładzie ocenę średniej i pesymistycznej złożoności wyszukiwania liniowego i binarnego.

Przeprowadzić analizę za pomocą instrumentacji i pomiarów czasu. W porównaniu wykorzystać tablice liczb

całkowitych o rozmiarze rzędu 230 bajtów (228 elementów typu *uint/int*).

W sprawozdaniu przedstawić dla każdego algorytmu:

- kod źródłowy przed instrumentacją
- kod źródłowy po instrumentacji
- zebrane wyniki w postaci tekstu i wykresów
- wnioski z analizy zebranych danych

Kod źródłowy przed instrumentacją:

wyszukiwanie liniowe:

```
static bool Liniowe(int[] tab, int szukana)  //BEZ INSTRUMENTACJI
        {
            for (int i = 0; i < tab.Length; i++)
                if (tab[i] == szukana)
                    return true;
            return false;
        }
```

wyszukiwanie binarne:

```
static int Binarne(int[] tablica, int szukana) //BEZ INSTRUMENTACJI
        {

            for (int i = 0; i < tablica.Length; i++)
            {
                int left = 0, right = tablica.Length - 1, middle;
                while (left <= right)
                {
                    middle = (left + right) / 2;
                    if (tablica[middle] == szukana) return middle + 1;
                    else if (tablica[middle] > szukana) right = middle - 1;
                    else left = middle + 1;


                }
            }
            return -1;

               }
```

Kod źródłowy po instrumentacji:

Wyszukiwanie liniowe z instrumentacją:

```
static void LinioweCzas()
        {
            double elapsed;
            long elapsedTime = 0, MinTime = long.MaxValue, MaxTime = long.MinValue,
czas;
            for (int n = 0; n < (NIter + 1 + 1); ++n)
            {
                long start = Stopwatch.GetTimestamp();
                bool result = Liniowe(tablica, tablica.Length - 1);
                long end = Stopwatch.GetTimestamp();
                czas = end - start;
                elapsedTime += czas;

                if (czas < MinTime) MinTime = czas;
                if (czas > MaxTime) MaxTime = czas;
            }
            elapsedTime -= (MinTime + MaxTime);
            elapsed = elapsedTime * (1.0 / (NIter * Stopwatch.Frequency));
            Console.Write("\t" + elapsed.ToString("F10"));
        }
        static bool LinioweInstrSpr(int[] tab, int szukana)
        {
            for (int i = 0; i < tab.Length; i++)
            {
                Licz++;
                if (tab[i] == szukana) return true;
            }
            return false;
        }
        static void LinioweInstr()
        {
            Licz = 0;
            bool result = LinioweInstrSpr(tablica, tablica.Length - 1);
            Console.Write("\t" + Licz);

        }


        static bool LinioweInstrSrSpr(int[] tab, int szukana)
        {
            for (int i = 0; i < tab.Length; i++)
            {
                Licz++;
                srednia += Licz;
                if (tab[i] == szukana) return true;
            }
            return false;
        }
        static void LinioweInstrSr()
        {
            Licz = 0;
            srednia = 0.00000;
            bool result = LinioweInstrSrSpr(tablica, tablica.Length - 1);
            Console.Write("\t" + srednia / Licz);
}
```

Wyszukiwanie binarne z instrumentacją:

```
static bool BinarneInstrSpr(int[] tab, int szukana)
        {
            int left = 0, right = tab.Length - 1, mid;
            while (left <= right)
            {
                mid = (left + right) / 2;
                Licz++;
                if (tab[mid] == szukana) return true;
                else
                {

                    if (tab[mid] == szukana) right = mid - 1;
                    else left = mid + 1;
                }
            }
            return false;
        }
        static int BinarneInstrSre(int[] tab, int szukana, int tablicaLength)
        {
            int left = 0;
            int right = tab.Length - 1;
            int middle = 0;
            wielkosc = 0;
            ilosc = 0;
            dlugosc = 0;
            while (left <= right)
            {
                ilosc++;
                middle = (left + right) / 2;
                wielkosc += (ulong)tab[ilosc];
                dlugosc += (ulong)Math.Pow(2, ilosc - 1);
                if (tab[middle] == szukana)
                {
                    ilosc++;
                    wielkosc += (ulong)tab[ilosc];
                    dlugosc += (ulong)Math.Pow(2, ilosc - 1);

                    return middle;
                }
                else if (tab[middle] < szukana)
                {
                    ilosc++;
                    wielkosc += (ulong)tab[ilosc];
                    dlugosc += (ulong)Math.Pow(2, ilosc - 1);

                    left = middle + 1;
                }
                else
                {
                    ilosc++;
                    wielkosc += (ulong)tab[ilosc];
                    dlugosc += (ulong)Math.Pow(2, ilosc - 1);

                    right = middle - 1;

                }
```

```
            }
            return -1;

        }
        static void BinarneInstr()
        {
            Licz = 0;
            bool result = BinarneInstrSpr(tablica, tablica.Length - 1);
            Console.Write("\t" + Licz);
        }

        static void BinarneCzas(int[] tablica, int szukana)
        {

            double elapsedSeconds;
            long elapsedTime = 0, MinTime = long.MaxValue, MaxTime = long.MinValue;
            for (int n = 0; n < (NIter + 1 + 1); ++n)
            {
                long start = Stopwatch.GetTimestamp();
                int result = Binarne(tablica, szukana);
                int left = 0, right = tablica.Length - 1, mid;
                while (left <= right)
                {
                    mid = (left + right) / 2;

                    if (tablica[mid] == szukana) return;

                    else


                        if (tablica[mid] == szukana) right = mid - 1;
                    else left = mid + 1;


                }
                long end = Stopwatch.GetTimestamp();
                long elapsed = end - start;
                elapsedTime += elapsed;
                if (elapsed < MinTime) MinTime = elapsed;
                if (elapsed > MaxTime) MaxTime = elapsed;


            }
            elapsedTime -= (MinTime + MaxTime);
            elapsedSeconds = elapsedTime * (1.0 / (NIter * Stopwatch.Frequency));
            Console.WriteLine("\t{0}", elapsedSeconds.ToString("F10"));
        }
```

Uruchamianie pomiarów (Main)

```
for (int k = 33554432; k <= 268435456; k += 33554432)
            {
                Console.Write(k);
                tablica = new int[k];
                for (int i = 2; i < tablica.Length; i++)
                {

                    tablica[i] = i;
                }
```

```
                LinioweInstr();
                //LinioweInstrSr();
                LinioweCzas();
                BinarneInstr();
                BinarneCzas(tablica, szukana);

            }

            Console.WriteLine();
            Console.WriteLine("Zakończono pomiar pesymistyczny, naciśnij dowolny
klawisz...");
            Console.ReadKey();


            int index = 0;
            ilosc = 0;
            wielkosc = 0;
            dlugosc = 0;

            Console.WriteLine("Średnia złożoność");
            Console.WriteLine("Size \tLinSrInstr \tBinSRInstr");
            //for (int k = 1; k <= 10; k += 1)
            for (int k = 33554432; k <= 268435456; k += 33554432)
            {
                Console.Write(k);
                tablica = new int[k];
                int y = tablica.Length - 1;
                for (int i = 2; i < tablica.Length; i++)
                {

                    tablica[i] = i;
                }


                LinioweInstrSr();
                for (int l = 0; l < NIter + 2; l++)
                {

                    index = BinarneInstrSre(tablica, y, tablica.Length);


                }
                Console.WriteLine("\t" + wielkosc);



            }
            Console.WriteLine("Zakończono pomiar średni, naciśnij dowolny
klawisz...");
            Console.ReadKey();
```

Wyniki:

W celu pomiaru została wykorzystana tablica o 268435456 elementów. Tablica rozpoczynała się od 33554432 i każdy kolejny pomiar był zwiększany o liczbę początkową co dało nam 8 pomiarów.

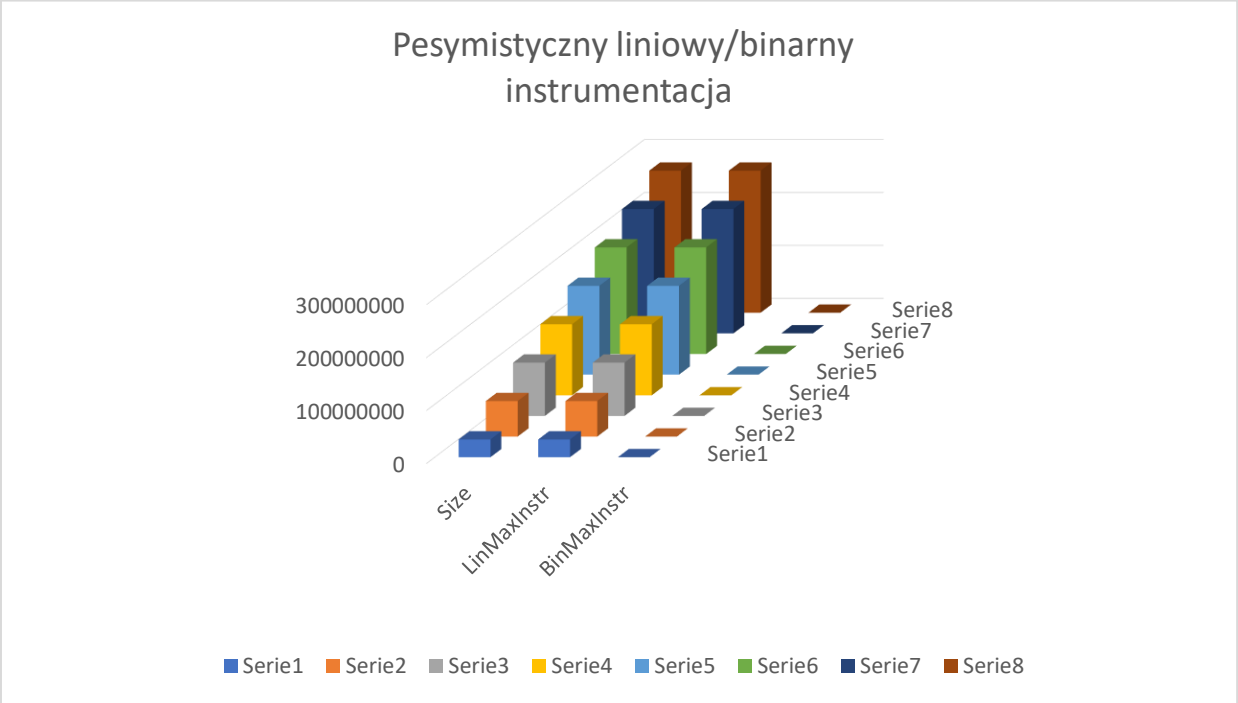
```
Pesymistyczna złożoność
Size    LinMaxInstr     LinMaxTim       BinMaxInstr     BinMaxTim
33554432        33554432        0,0905932900    26      0,0000009900
67108864        67108864        0,1768564200    27      0,0000014300
100663296       100663296       0,2641170800    27      0,0000010700
134217728       134217728       0,3527016200    28      0,0000012000
167772160       167772160       0,4413645700    28      0,0000010500
201326592       201326592       0,5279464300    28      0,0000010800
234881024       234881024       0,6162417500    28      0,0000010900
268435456       268435456       0,7039531100    29      0,0000024300

Zakończono pomiar pesymistyczny, naciśnij dowolny klawisz...
Średnia złożoność
Size    LinSrInstr      BinSRInstr
33554432        16777216,5      1377
67108864        33554432,5      1484
100663296       50331648,5      1484
134217728       67108864,5      1595
167772160       83886080,4      1595
201326592       100663296,333333        1595
234881024       117440512,285714        1595
268435456       134217728,25    1710
Zakończono pomiar średni, naciśnij dowolny klawisz...
```

Przypadek pesymistyczny

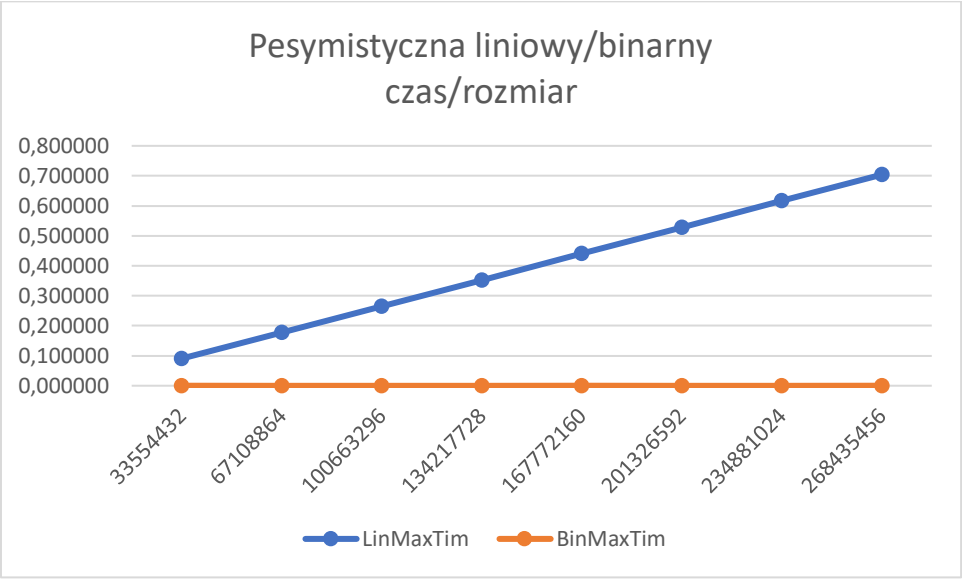
Tabela przedstawiająca przypadek pesymistyczny:

Size	LinMaxInstr	LinMaxTim	BinMaxInstr	BinMaxTim
33554432	33554432	0,090593	26	0,0000010
67108864	67108864	0,176856	27	0,0000014
100663296	100663296	0,264117	27	0,0000011
134217728	134217728	0,352702	28	0,0000012
167772160	167772160	0,441365	28	0,0000011
201326592	201326592	0,527946	28	0,0000011
234881024	234881024	0,616242	28	0,0000011
268435456	268435456	0,703953	29	0,0000024

Wykres porównujący ilość powtórzeń w przypadku pesymistycznym, podczas wyszukiwania liniowego (LinMaxInstr) i binarnego (BinMaxInstr) względem rozmiaru tabeli (Size).



Wykres porównujący czas potrzebny na wyszukanie w przypadku pesymistycznym, pomiędzy wyszukiwaniem liniowym (LinMaxTim), a wyszukiwaniem binarnym (BinMaxTim).

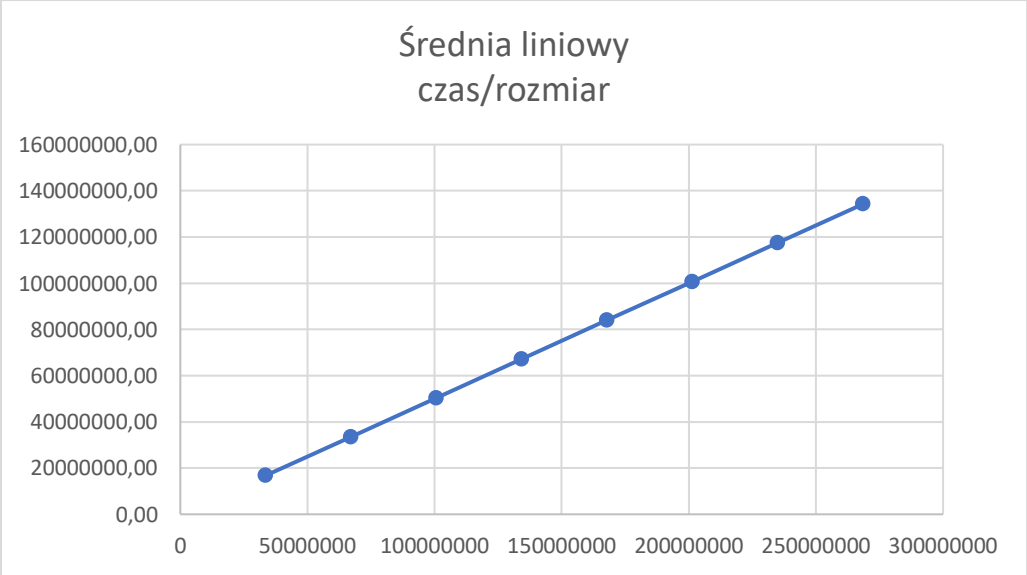


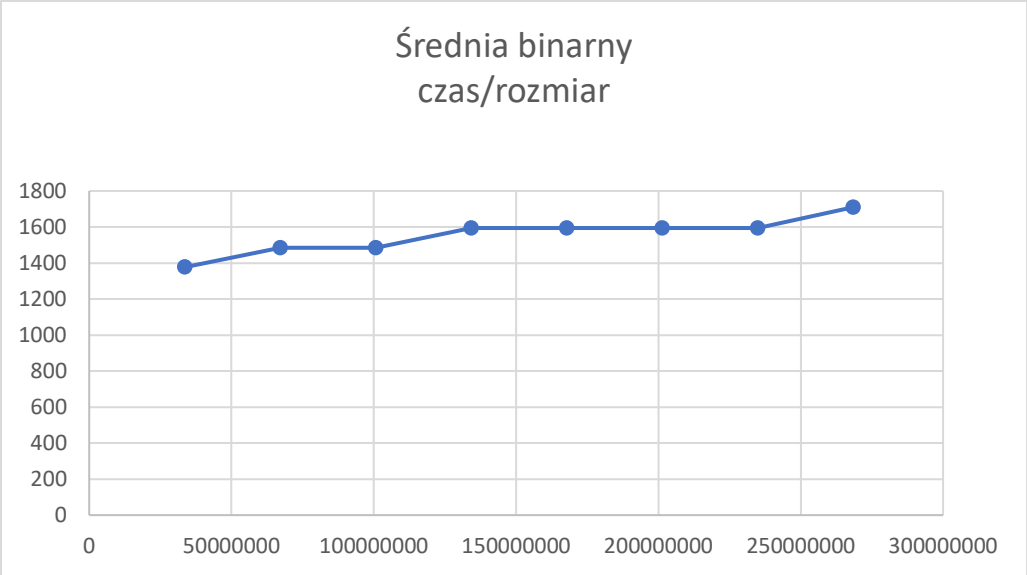
Przypadek średni

Tabela przedstawiająca różnicę ilości porównań podczas korzystania z wyszukiwania liniowego (LinSrInstr) a wyszukiwania binarnego (BinSRInstr) przy wykorzystaniu takiej samej tablicy jak w przykładzie z przypadkiem pesymistycznym.

Size	LinSrInstr	BinSRInstr
33554432	16777216,50	1377
67108864	33554432,50	1484
100663296	50331648,50	1484
134217728	67108864,50	1595
167772160	83886080,40	1595
201326592	100663296,33	1595
234881024	117440512,29	1595
268435456	134217728,25	1710

Powyższe dane przedstawione na wykresach:





Wnioski:

Wyszukiwanie binarne jest szybsze od wyszukiwania liniowego zarówno w przypadku pesymistycznym, jak i średnim.

Czas potrzebny na wykonanie wyszukiwania liniowego jest znacząco dłuższy od czasu potrzebnego na wyszukiwanie binarne.

Wyszukiwanie liniowe może mieć porównywalną wydajność do binarnego jedynie podczas przeszukiwania niewielkich zbiorów.