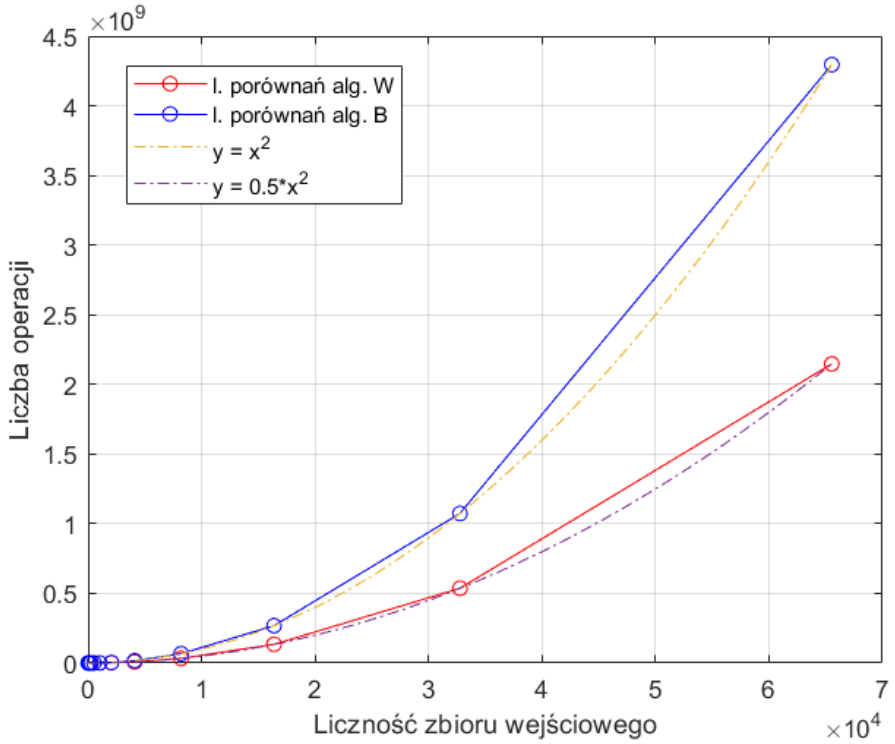
Piotr Mikołajczyk			
AISDE LAB 2	272018 Nr Indeksu	2021.03.09 – 2021.03.11 Data wykonania ćwiczenia	2021.03.14 Nominalna data oddania sprawozdania
	Punkty do wykonania	„Implementujemy sortowanie bąbelkowe i/lub koktajlowe i porównujemy przynajmniej z jedną implementacją sortowania przez wstawianie.”	

SPRAWOZDANIE:
Badanie złożoności obliczeniowej algorytmów na przykładzie sortowania

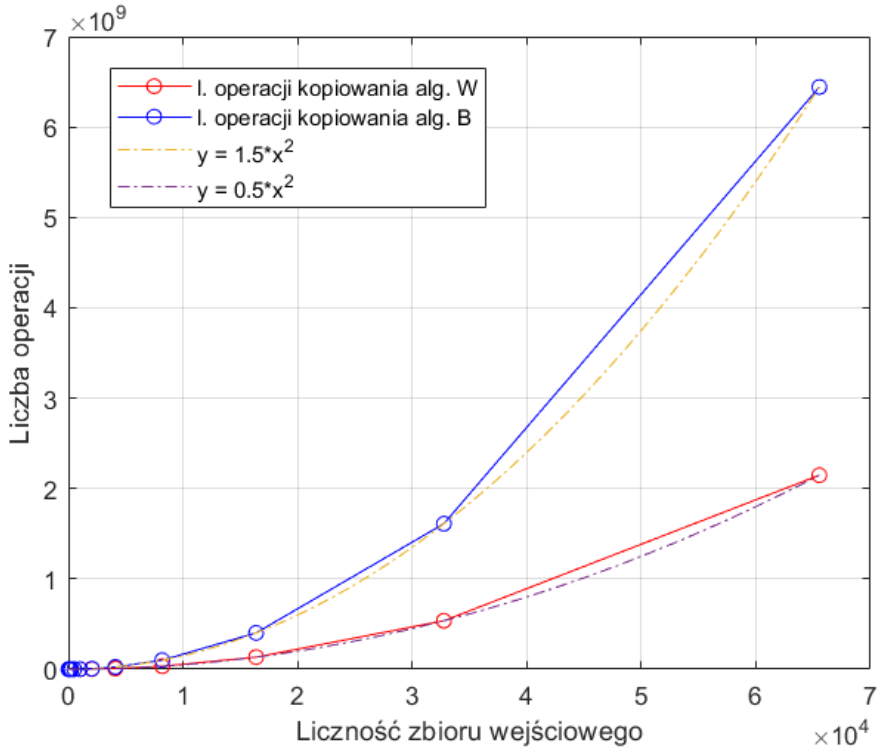
W laboratorium 2 porównano algorytm zaimplementowany w języku C++ sortowania przez wstawianie (insertion sort) oraz implementacja przedstawioną poniżej algorytmu bąbelkowego (bubble sort).

```
void Array<RECORD>::bubbleSort(Iterator first, Iterator last)
{
   int i = 0;
   //string arrsize = " arrSize = ";
   //arrsize += std::to_string(last);
   //cout << arrsize;
   while (i <= last) {
      Iterator curr = last-1;
      for (Iterator j = last; j > first; --j) {
         //string str1 = " j = ";
         //string str2 = " curr = ";
         if (array[curr] > array[j])
            swap(array[curr], array[j]);
         curr = curr - 1;
         /*
         str1 += std::to_string(j);
         str2 += std::to_string(curr);
         cout << str1;
         cout << str2;
         str1.clear();
         str2.clear();
         cout << "\n";
         curr = curr - 1;
         */
      }
      i = i + 1;
   }
}
```

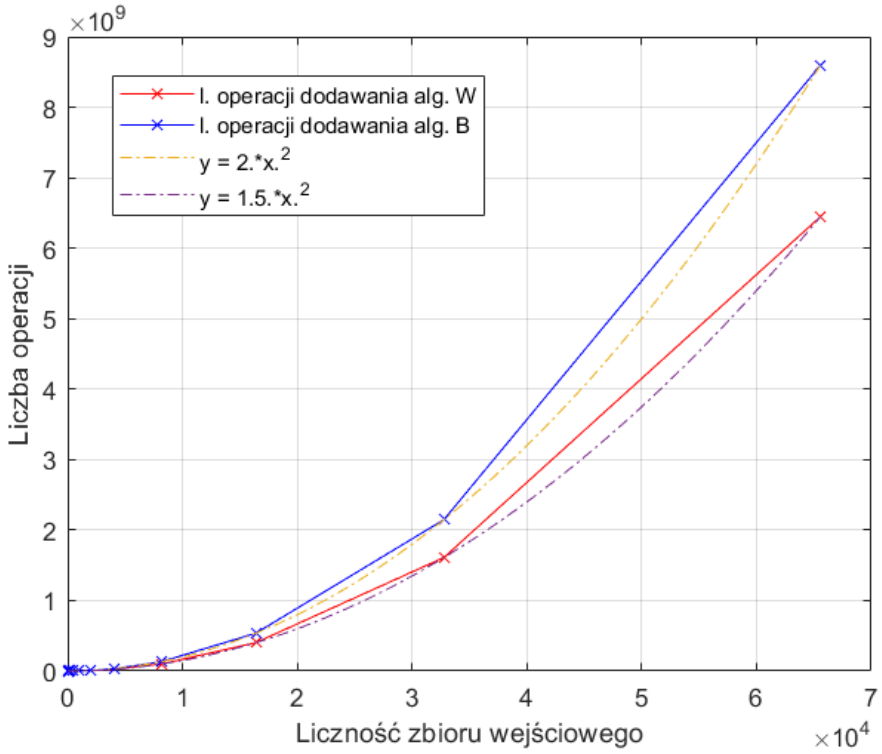
Zbadano i porównano algorytmy ciągami malejącym oraz rosnącym , badając złożoności pesymistyczne i optymistyczne. Dane zadane są jako ciąg kolejnych potęg dwójki: [2,4,8,16 … 65536]
Na kolejnych rysunkach zamieszczono wykresy liczby operacji w zależności od liczności zbioru oraz złożoności czasowe algorytmów dla optymistycznych i pesymistycznych przypadków.



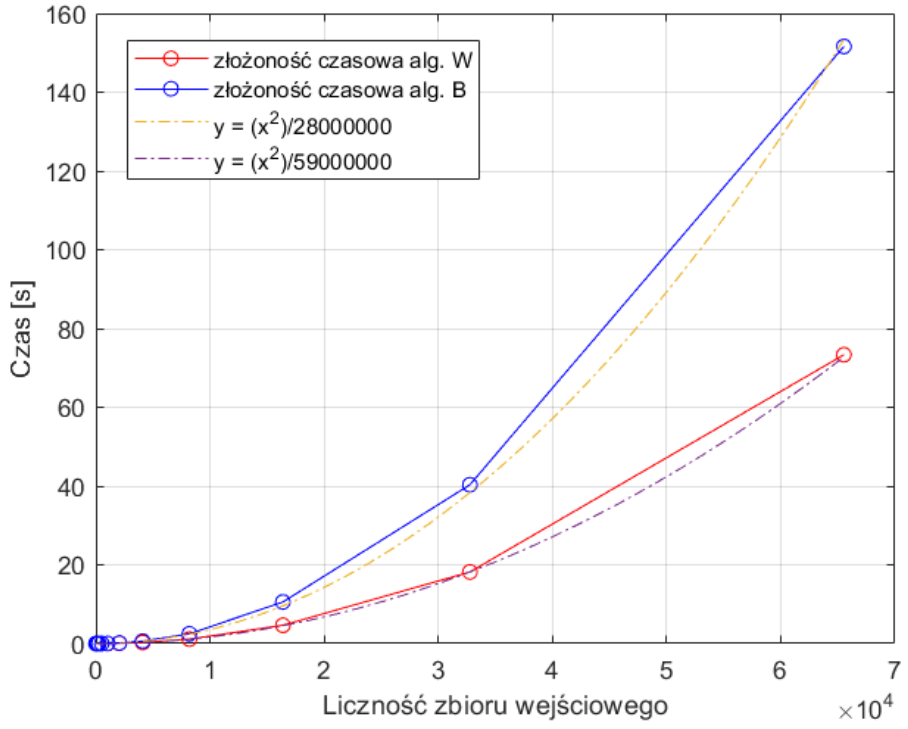
Rys. 1 – Liczba porównań algorytmu wstawiania i bąbelkowego w zależności od liczności zbioru – przypadek pesymistyczny – wykres liniowy



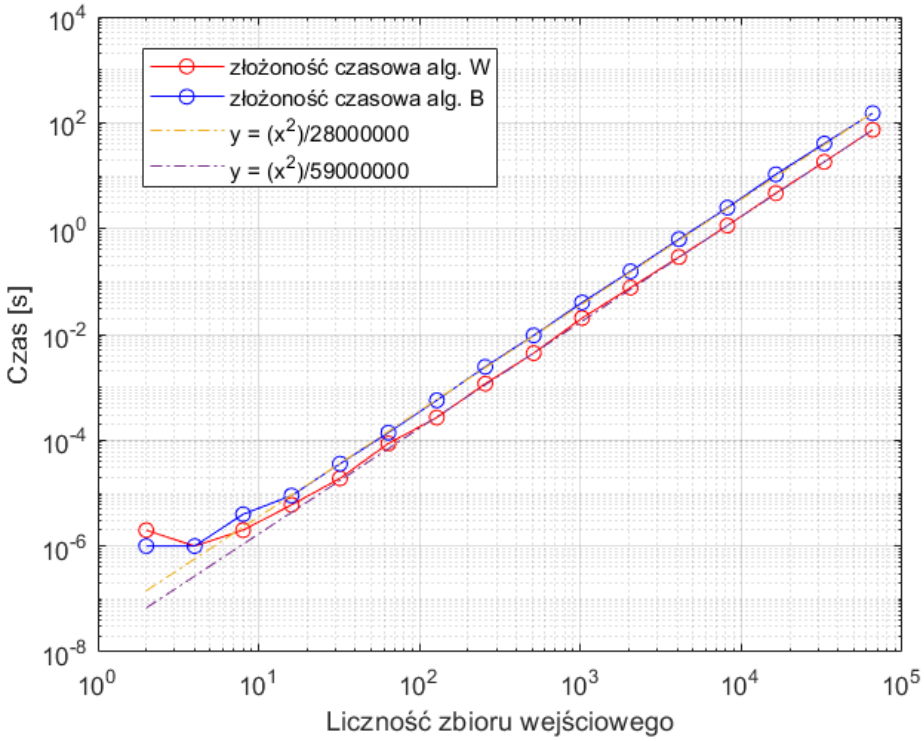
Rys.2– Liczba operacji kopiowania algorytmu wstawiania i bąbelkowego w zależności od liczności zbioru – przypadek pesymistyczny – wykres liniowy



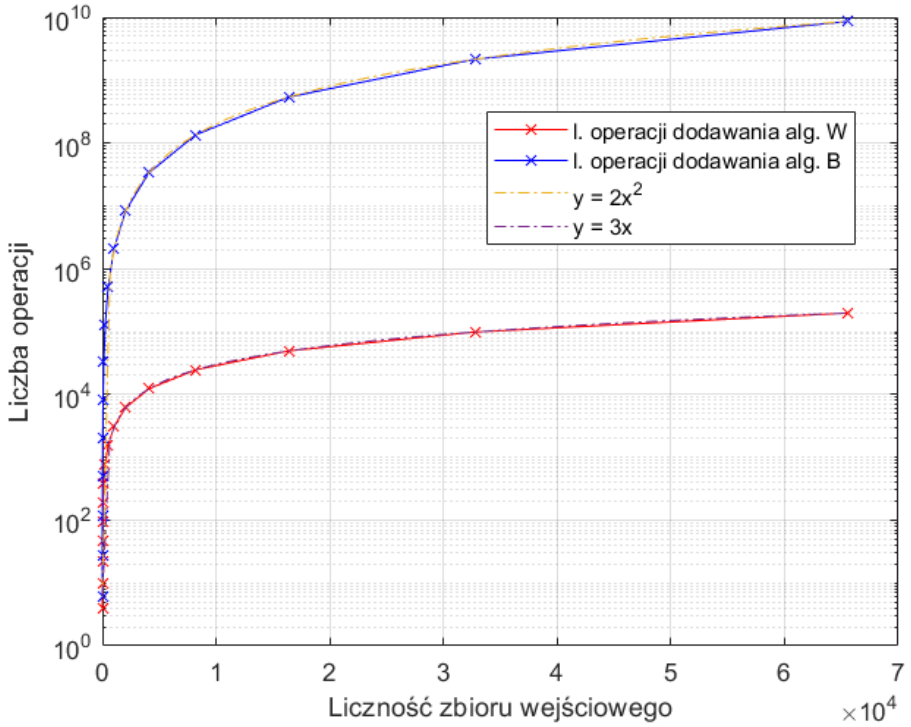
Rys. 3 – Liczba operacji dodawania algorytmu wstawiania i bąbelkowego w zależności od liczności zbioru – przypadek pesymistyczny – wykres liniowy



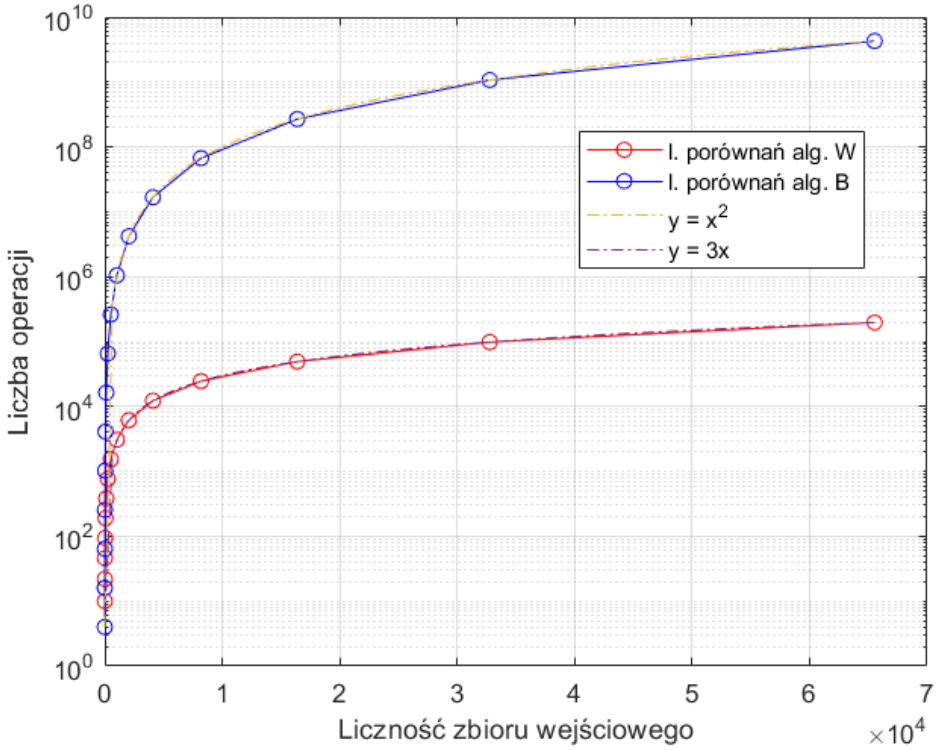
Rys. 4 – Złożoność czasowa algorytmu wstawiania i bąbelkowego w zależności od liczności zbioru – przypadek pesymistyczny – wykres liniowy



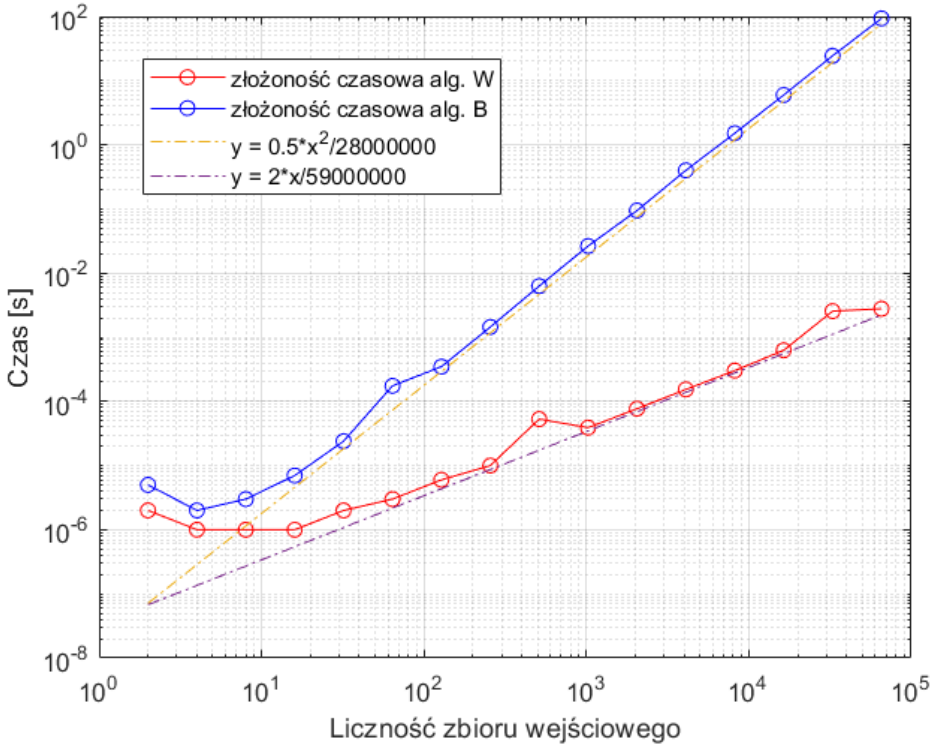
Rys. 5 – Złożoność czasowa algorytmu wstawiania i bąbelkowego w zależności od liczności zbioru – przypadek pesymistyczny – wykres loglog



Rys. 6 – Liczba operacji dodawania algorytmu wstawiania i bąbelkowego w zależności od liczności zbioru – przypadek optymistyczny – wykres półlogarytmiczny



Rys. 7 - Liczba porównań algorytmu wstawiania i bąbelkowego w zależności od liczności zbioru – przypadek optymistyczny – wykres półlogarytmiczny



Rys. 8 – Złożoność czasowa algorytmu wstawiania i bąbelkowego w zależności od liczności zbioru – przypadek optymistyczny – wykres loglog

Wnioski:

Badanie złożoności algorytmów wykazało że dla obydwu granic pesymistycznej oraz optymistycznej algorytm sortowania przez wstawianie jest szybszy i efektywniejszy – dla przypadku pesymistycznego wykonuje dla danych malejących połowę operacji porównania mniej, 3 razy mniej operacji kopiowania oraz połowę operacji dodawania niż algorytm bąbelkowy (rysunki 1,2,3). Na rysunku 4 przedstawiono złożoność czasową algorytmów. Algorytm Insertion Sort wykonuje się średnio ok. 2 razy szybciej dla zbioru pesymistycznego niż algorytm Bubble Sort. Na rysunku 5 przedstawiono logarytmiczny wykres złożoności czasowej od liczności danych – linie proste świadczą o kwadratowej zależności czasowej algorytmów, natomiast przesunięcie stałe przesunięcie oznacza szybsze wykonywanie się jednego z algorytmów. Niestety , algorytmy sortujące badano w środowisku wirtualizowanym VMWare, w środowisku linux oraz na maszynie o skończonej szybkości operacji – dla bardzo małych zbiorów (poniżej 32 elementów) , czasy wykonywania algorytmów odbiegają od idealnych charakterystyk kwadratowych (na poziomie mikrosekund).

Dla przypadku optymistycznego, algorytm Bubble Sort w podstawowej wersji, wykonuje tyle samo operacji porównań oraz dodawań w tej implementacji, jednak, nie są wykonywane operacjie kopiowania, co polepsza złożoność czasową o połowę, jednak algorytm nadal ma charakter kwadratowy (Rys. 8). Dla algorytmu Insertion Sort przeprowadzono również badanie dla przypadku optymistycznego. Algorytm dla zbioru rosnącego, z kwadratowego , zmienia charakter na liniowy co widać szczególnie dla dużego rozmiaru danych liczności zbioru. Z badania wynika wprost że algorytm Insertion Sort jest lepszym algorytmem od podstawowej implementacji algorytmu Bubble Sort.