17.01.2021

Algorytmy i Struktury Danych
Sprawozdanie z projektu numer 2

Grupa Projektowa nr. 5
Jakub Matuszko

Numer indeksu: 166673

Treść projektu: Zaimplementuj sortowanie przez wstawianie oraz sortowanie kubełkowe.

Treść projektu:

Zaimplementuj sortowanie przez wstawianie oraz sortowanie kubełkowe.

Wymagane cechy programu:

Cechy programu:

a) program powinien mieć możliwość odczytywania danych wejściowych z pliku tekstowego i zapisu posortowanego ciągu wynikowego również do pliku tekstowego

b) na potrzeby wykonywanych testów należy zaimplementować funkcję generującą ”losowe” ciagi elementów (o zadanej długości) i zapisującą je do pliku tekstowego

c) założyć, że sortowanymi elementami są liczby całkowite z przedziału [0, N], gdzie N powinno być ”odpowiednio dużym” parametrem ustalonym wewnątrz programu

d) kod powinien być opatrzony stosownymi komentarzami

Wybrany język programowania: C++

Wybrane środowisko: Code::Blocks

Wejście programu

Plik input.txt z liczbami całkowitymi mniejszymi od 2’147’483’647 i większymi od – 2’147’483’647. Liczby tę są oddzielone spacją lub enterem.

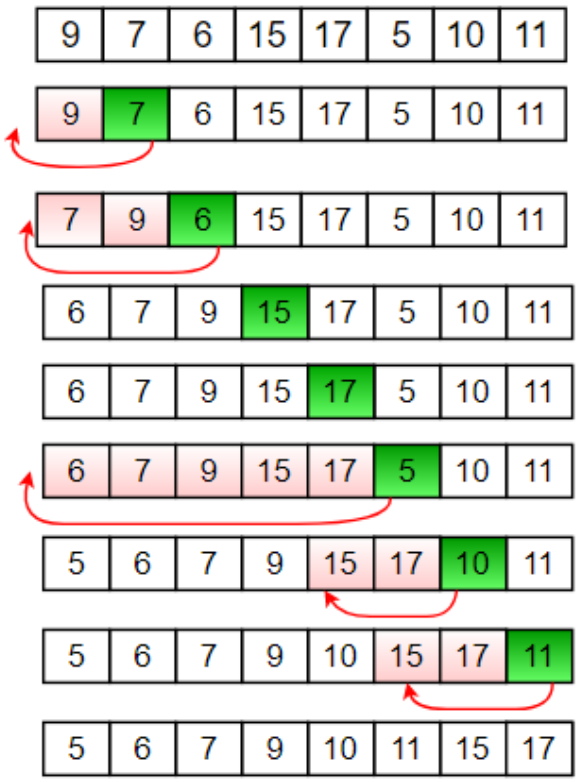
Wyjście programu

Plik output.txt z liczbami całkowitymi mniejszymi od 2’147’483’647 i większymi od – 2’147’483’647. Oddzielone średnikiem i spacją. Pierwsza linijka z nich to dane posortowane sortowaniem przez wstawianie, a druga linijka to dane posortowane sortowaniem kubełkowym.

Problematyka algorytmów:

Sortowanie poprzez wstawianie (insertion sort)

Algorytm sortowania przez wstawianie polega na przestawianiu elementów w zbiorze. Zaczynając od początku i wybierając kolejną liczbę a następnie wstawiając ją w odpowiednie miejsce aż do posortowania całości.



Rysunek 1 Poglądowy sposób działania algorytmu sortowania poprzez wstawianie
https://www.geeksforgeeks.org/recursive-insertion-sort/

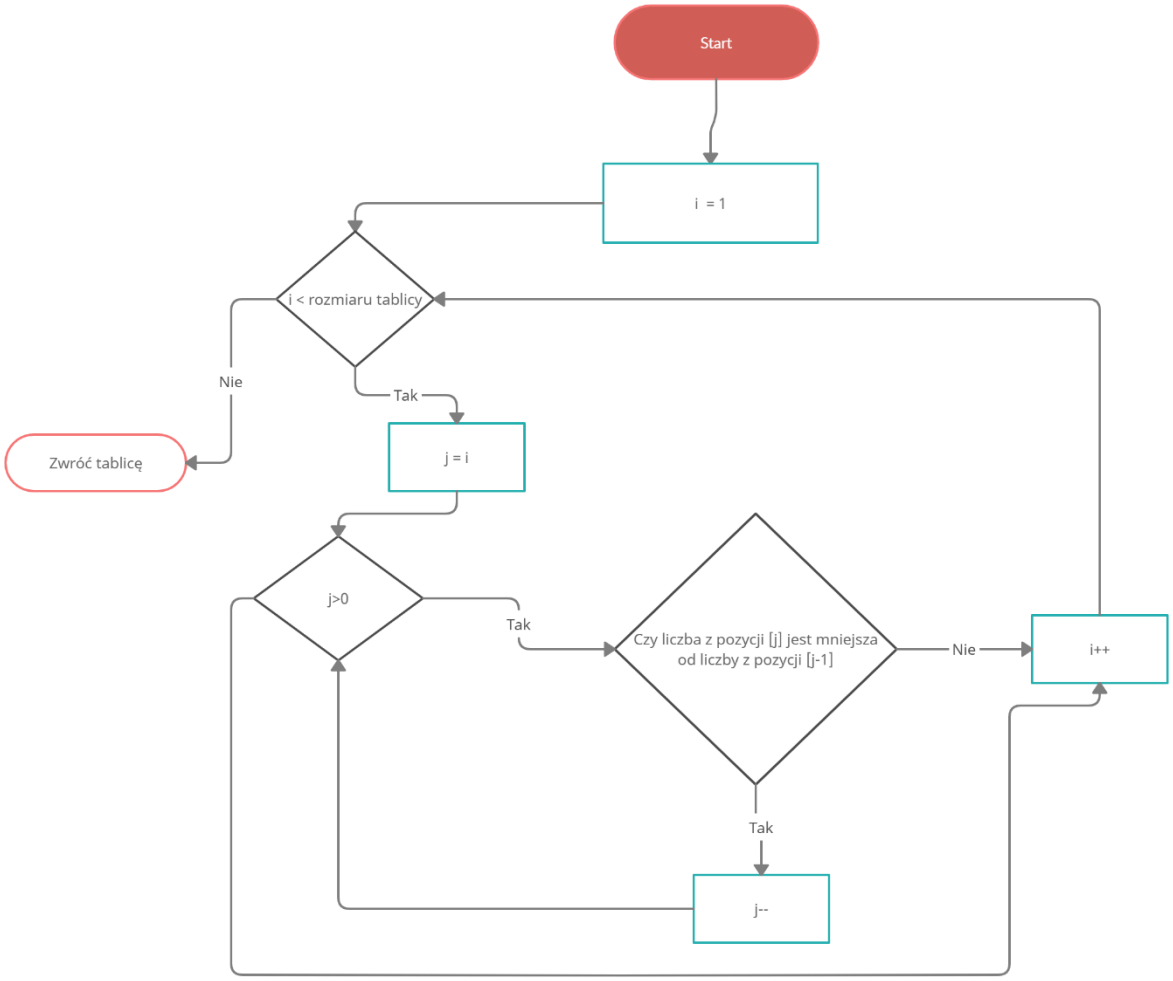
Złożoność czasowa:

- Najgorszy przypadek: $O(n^2)$

- Oczekiwany przypadek: $O(n^2)$

- Najlepszy przypadek: $O(n)$

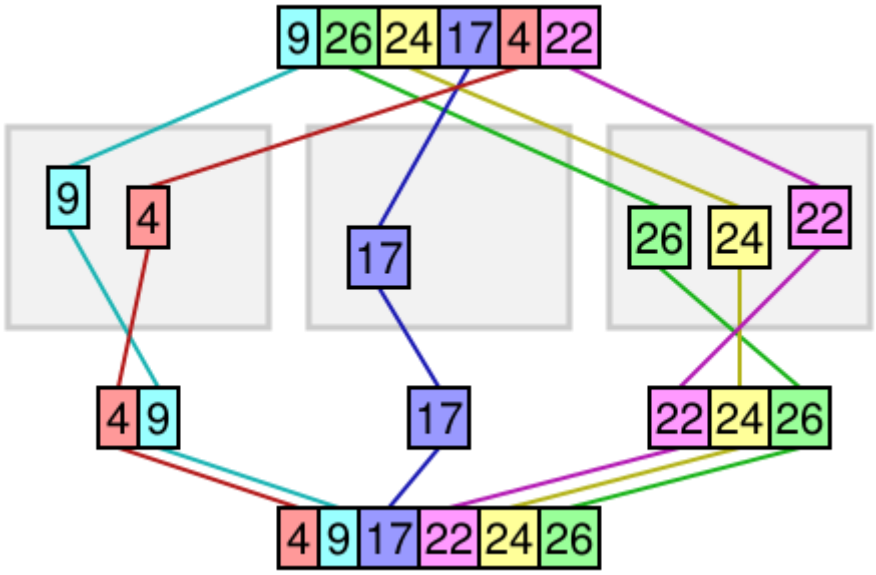
Krok 01: Zacznij od drugiego elementu:
Krok 02: Sprawdź czy element poprzedni jest mniejszy od wybranego: jeżeli tak to przejdź do kroku 3, Jeżeli to nie koniec liczb to przejdź to wybierz kolejny element i przejdź do kroku 2. Jeżeli to koniec liczb to przejdź do kroku 4
Krok 03: Jeżeli element poprzedni jest mniejszy to zamień go miejscem z obecnym i przejdź do kroku 2
Krok 04: Zwróć tablicę liczb.



Rysunek 2 Schemat blokowy algorytmu sortowania przez wstawianie

Sortowanie kubełkowe (bucket sort)

Dane sortowanie polega na dzieleniu dużego zbioru na mniejsze „kubełki”. W danym algorytmie tworzymy tyle samo kubełków co jest liczb, a następnie poprzez odpowiednie obliczenia wrzucamy liczbę do wybranego kubełka. Wykorzystany sposób obliczania indeksu kubełka dla liczby to wzór: $\left[\frac{Wartość\ liczby}{Największa\ liczba\ w\ zbiorze}\right] * Ilość\ elementów$ i ten wynik jest zaokrąglany do dołu. W taki oto sposób wiele liczb może trafić do jednego kubełka jak i niektóre mogą zostać puste. Kolejnym krokiem jest posortowanie każdego kubełka osobno – przy wykorzystaniu innego algorytmu sortowania. Gdy wszystko zostało posortowane, ostatnim krokiem jest wyczyszczenie pierwotnego wektora liczb i nadpisanie go liczbami z kubełków.



Rysunek 3 Wizualizacja przedstawiająca sposób działania sortowania kubełkowego
https://medium.com/karuna-sehgal/an-introduction-to-bucket-sort-62aa5325d124

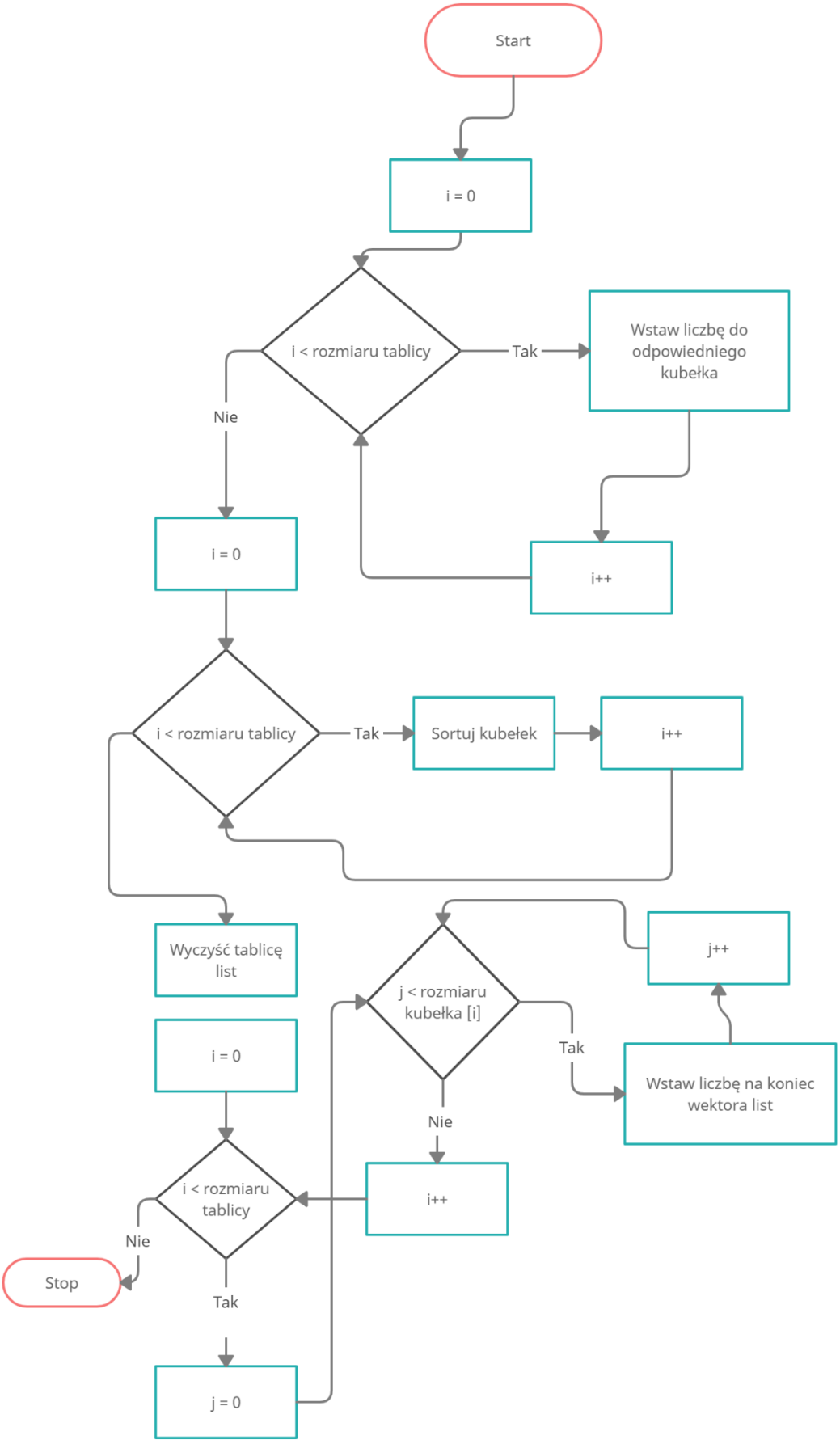
Złożoność czasowa:

- Najgorszy przypadek: $O(n^2)$

- Oczekiwany przypadek: $O(n + \frac{n^2}{k} + k)$, gdzie k to ilość kubełków

- Najlepszy przypadek: $O(n + k)$, gdzie k to ilość kubełków

Krok 01: Utwórz kubełek dla każdej liczby
Krok 02: Znajdź największą liczbę w zbiorze
Krok 03: Wrzuć liczbę do odpowiedniego kubełka: [(liczba / (max zbioru)) * (ilość elementów)] (Wynik zaokrąglony w dół)
Krok 04: Posortuj wszystkie liczby w kubełkach
Krok 05: Zwróć liczby z kubełków do wektora



Rysunek 4 Schemat blokowy sortowania kubełkowego

Testy:

Przeprowadzone testy dla losowo wygenerowanych liczb w zakresie od 0 do 10000.

Wyniki przedstawione są w milisekundach

Numer testu:	**Ilość liczb w zbiorze**	**Czas sortowania kubełkowego (ms)**	**Czas sortowania przez wstawianie (ms)**
1	100	<1ms	<1ms
2	500	<1ms	<1ms
3	1000	<1ms	<1ms
4	2000	<1ms	16ms
5	5000	<1ms	78ms
6	10000	<1ms	312ms
7	20000	16ms	1250ms
8	40000	31ms	4891ms
9	80000	47ms	19469ms
10	100000	47ms	30422ms
11	120000	47ms	44250ms
12	160000	62ms	78031ms
13	200000	78ms	122157ms
14	400000	140ms	492157ms

Tabela 1 Tabela wyników testów

Najgorszy przypadek – liczby poukładane od największego do najmniejszego

Numer testu:	**Ilość liczb w zbiorze**	**Czas sortowania kubełkowego (ms)**	**Czas sortowania przez wstawianie (ms)**
1	5000	<1ms	110ms
2	10000	<1ms	422ms
3	20000	16ms	1766ms
4	40000	16ms	6797ms
5	80000	31ms	27047ms
6	120000	47ms	60750ms

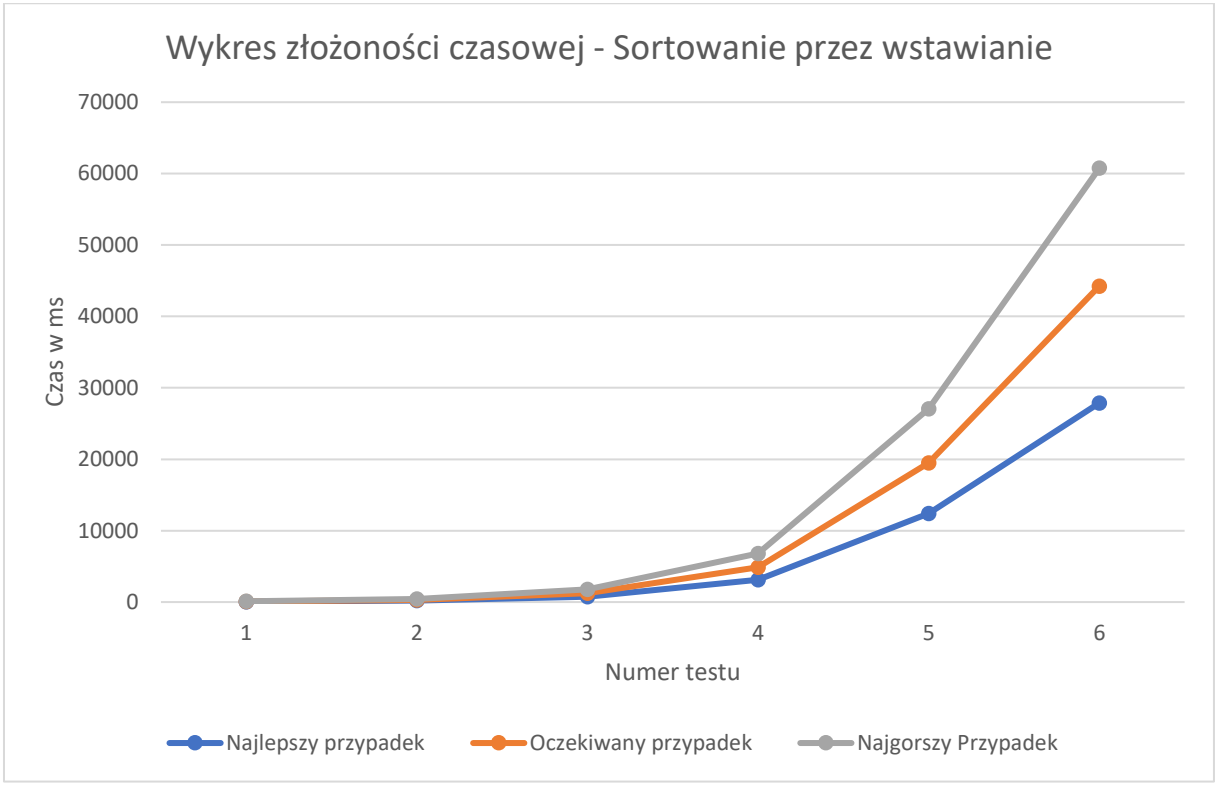
Tabela 2 Tabela wyników pesymistycznych

Najlepszy przypadek – liczby posortowane

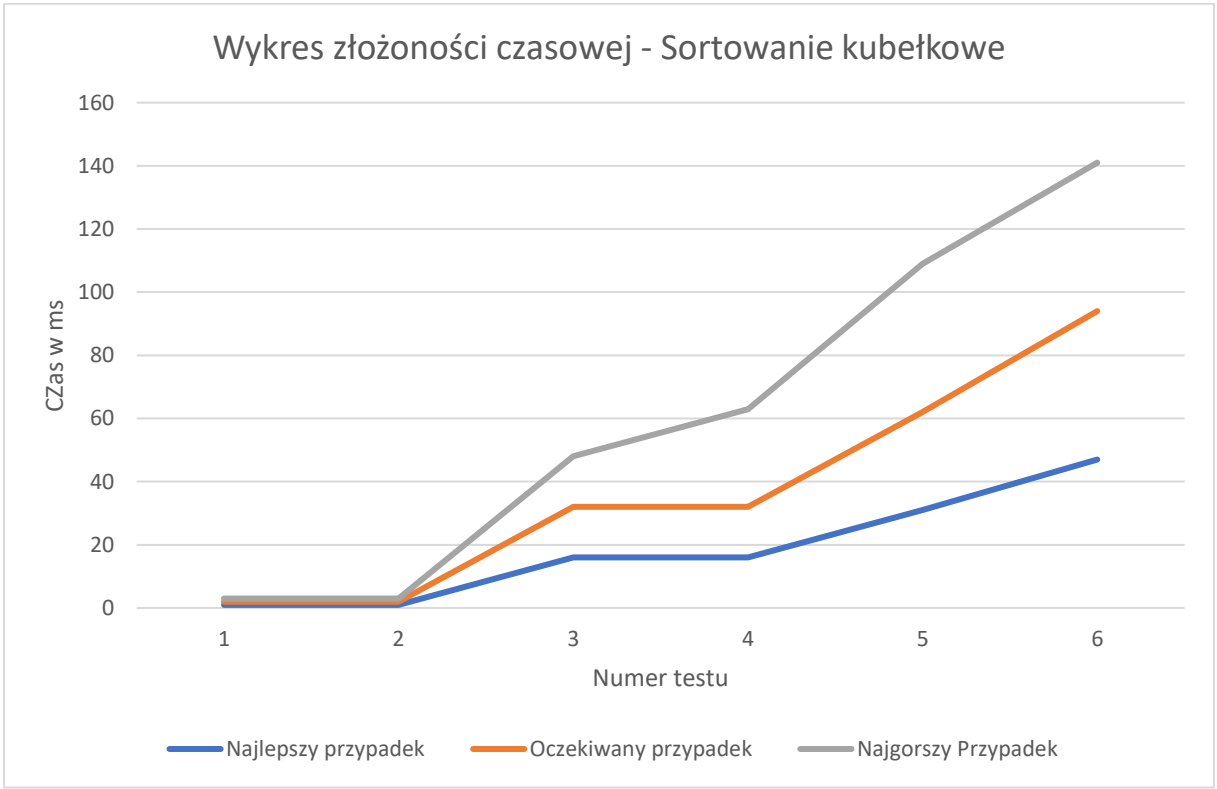
Numer testu:	**Ilość liczb w zbiorze**	**Czas sortowania kubełkowego (ms)**	**Czas sortowania przez wstawianie (ms)**
1	5000	<1ms	47ms
2	10000	<1ms	204ms
3	20000	16ms	765ms
4	40000	16ms	3125ms
5	80000	31ms	12406ms
6	120000	47ms	27875ms

Tabela 3 Tabela wyników optymistycznych

Wykresy złożoności czasowej



Rysunek 5 Wykres złożoności czasowej - sortowanie przez wstawianie



Rysunek 6 Wykres złożoności czasowej - sortowanie kubełkowe