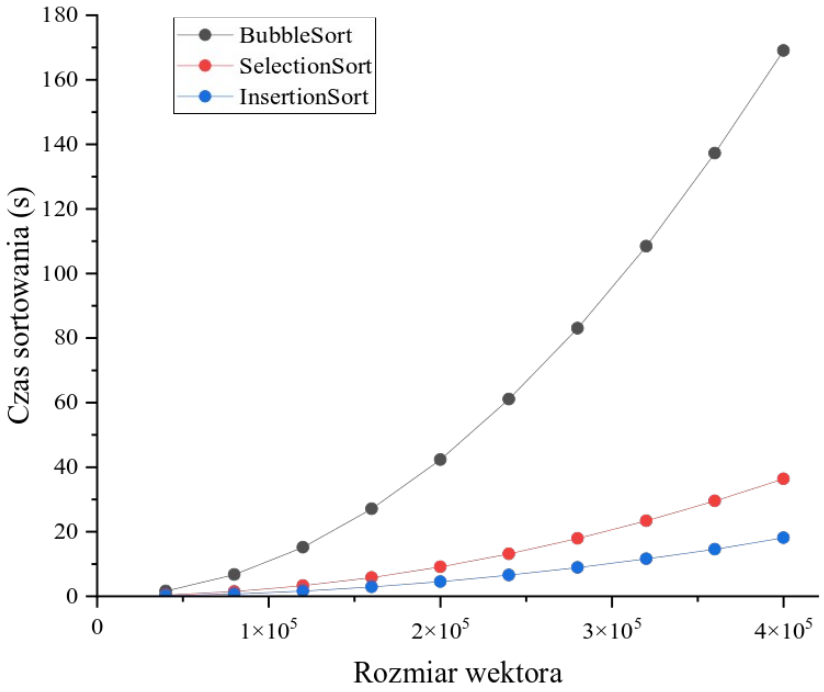
Algorytmy i Struktury Danych 2022/23

Krzysztof Czarnowus, grupa nr 2

Zestaw 5

Zadanie 1.



Rysunek 1. Zależność czasu sortowania wektora każdą z trzech zastosowanych technik od jego rozmiaru wraz z dopasowanymi za pomocą programu Origin funkcjami kwadratowymi.

Tabela 1. Parametry funkcji kwadratowych dopasowanych do uzyskanych zależności.

funkcja sortująca	współczynnik przy x^2	współczynnik przy x	wyraz wolny
Bubble Sort	1.05×10^{-9}	2.98×10^{-6}	-0.255
Selection Sort	2.26×10^{-10}	8.89×10^{-7}	-0.0606
Insertion Sort	1.12×10^{-10}	2.64×10^{-7}	-0.0158

Zadanie 2.

Zmodyfikowano algorytm programu BubbleSort.cpp tak, aby zliczał on dominujące operacje. Przyjęto, że będą nimi porównania. Program nazwany został OperationCount.cpp.

Według oczekiwań, optymistyczna złożoność obliczeniowa dla wyjściowo posortowanego wektora powinna wynieść $O(n)$, ponieważ wykonane zostanie dokładnie n-1 porównań, po czym funkcja stwierdzi, że wektor został uporządkowany.

Pesymistyczna złożoność obliczeniowa dla wektora uporządkowanego malejąco powinna wynieść $O(n^2)$ i wykonać dokładnie $\frac{n \times (n-1)}{2}$ porównań. Wynika to z tego, że pętla w pierwszej iteracji wykona n porównań, w drugiej n-1, itd., kończąc na jednym porównaniu w ostatniej iteracji. Ilość porównań będzie więc sumą szeregu:

$$\sum_{k=1}^{n} n = \frac{n \times (n-1)}{2}$$

Dokładne oszacowanie średniej złożoności obliczeniowej zaimplementowanego algorytmu jest zadaniem trudnym; gdyby nie wprowadzona możliwość przerwania sortowania, kiedy w trakcie jednego obiegu program nie przeprowadzi żadnej zamiany, byłaby ona równa pesymistycznej oraz optymistycznej złożoności obliczeniowej i wynosiła $O(n^2)$. Można jednak założyć, że wyniesie ona $O(n^2)$, chociaż współczynnik będzie mniejszy niż dla pesymistycznej złożoności.

Napisany program zliczył operacje w wektorze posortowanym, wektorze posortowanym odwrotnie oraz w stu wektorach wypełnionych losowymi liczbami – dla tego ostatniego wyciągając średnią ze stu powtórzeń (uruchamiając program ponownie można zauważyć, że wartość ta nie ulega dużym zmianom dla kolejnych wywołań – można więc uznać ją za wystarczające oszacowanie). Rozmiar każdego wektora wynosił 10 000. Wyniki przedstawiono w tabeli 2.

Tabela 2. Zestawienie oczekiwanej oraz rzeczywistej ilości porównań wykonanych podczas wywołania funkcji BubbleSort na odpowiednich wektorach o rozmiarze 10 000.

	oczekiwana ilość operacji	rzeczywista ilość operacji
wektor posortowany	9999	9999
wektor posortowany odwrotnie	49 995 000	49 995 000
uśredniony wektor losowy	-	ok. 7 000 000