Przykład 10. Złożoność czasowa algorytmu sortowania bąbelkowego

Podobnie jak w przypadku algorytmu z przykładu 9, rozważać będziemy złożoność czasową algorytmu sortowania bąbelkowego ze względu na liczbę operacji porównania. W pierwszym przejściu algorytmu musimy porównać n liczb, czyli wykonać $n-1$ porównań. Po pierwszym przejściu ostatni element tablicy będzie największym elementem, w następnym przejściu musimy wykonać $n-2$ porównań, potem $n-3$, a w ostatnim przejściu już tylko jedną operację porównania.
Liczby operacji porównania w kolejnych przejściach tworzą ciąg arytmetyczny: $(a_n) = (1, 2, ..., n-2, n-1)$ o $n-1$ elementach.
Ponieważ suma skończonego ciągu arytmetycznego wyraża się wzorem:

$$S_k = \frac{a_1 + a_k}{2} \cdot k, \qquad [1]$$

a w naszym przypadku:
$a_1 = 1$
$a_k = n-1$
$k = n-1$,
po podstawieniu do wzoru [1] otrzymujemy:

$$L_n = S_k = \frac{1+n-1}{2} \cdot (n-1) = \frac{n}{2} \cdot (n-1) = \frac{n^2-n}{2}.$$

Liczba operacji porównań algorytmu sortowania bąbelkowego wynosi L_n. Nie interesuje nas jednak dokładna wartość tej liczby, ale rząd jej wielkości. Wystarczy nam zatem ustalenie, że dla algorytmu sortowania bąbelkowego złożoność czasowa wynosi $O(n^2)$.

Przykład 11. Złożoność czasowa algorytmu sortowania przez wybór

Prowadząc rozumowanie podobne jak dla algorytmów podanych w poprzednich dwóch przykładach, obliczymy liczbę porównań w algorytmie sortowania przez wybór.
W pierwszym przejściu, aby znaleźć element najmniejszy, należy wykonać $n-1$ porównań. W następnym przejściu porównań tych będzie $n-2$, ponieważ mamy do przeszukania o jeden element mniej. W ostatnim przejściu wykonamy tylko jedno porównanie. Liczby porównań w kolejnych przejściach algorytmu tworzą zatem ciąg arytmetyczny: $(a_n) = (1, 2, ..., n-2, n-1)$ o $n-1$ elementach.
Suma tego ciągu wynosi $L_n = \frac{n^2-n}{2}$, więc algorytm sortowania przez wybór również posiada złożoność czasową rzędu $O(n^2)$.

Ćwiczenie 8.

Uzupełnij program utworzony w ćwiczeniu 2, tak by obliczał i wypisywał liczbę przeprowadzonych porównań. Sprawdź działanie programu dla różnych wielkości zbioru danych n.

5.2. Złożoność pamięciowa

Złożoność pamięciowa algorytmu to wielkość pamięci (operacyjnej lub masowej) niezbędnej do wykonania algorytmu. Złożoność pamięciowa to zwykle po prostu wielkość pamięci zajmowanej przez wszystkie zmienne (także tworzone dynamicznie). Jest jednak