Temat: Porządkowanie ciągu liczb za pomocą algorytmu sortowania bąbelkowego

Omówienie zagadnienia

Sortowanie bąbelkowe to prosty i popularny algorytm sortowania, który działa poprzez wielokrotne przechodzenie przez sortowany ciąg liczb. Podczas każdego przejścia porównywane są sąsiednie elementy, a jeśli są one w niewłaściwej kolejności, są zamieniane miejscami. Proces ten jest powtarzany aż do momentu, gdy lista zostanie w pełni posortowana.
Algorytm sortowania bąbelkowego jest łatwy do zrozumienia i implementacji, jednak jego efektywność w porównaniu z bardziej zaawansowanymi algorytmami jest stosunkowo niska. Jego złożoność czasowa wynosi $O(n^2)$, gdzie n to liczba elementów w sortowanym ciągu.

Przykłady sortowania bąbelkowego

Weźmy przykładowy ciąg liczb: [5, 3, 8, 6]. Poniżej pokazano kolejne kroki sortowania:

1. **Pierwsze przejście:**
   - Porównujemy 5 i 3. Zamieniamy miejscami: [3, 5, 8, 6].
   - Porównujemy 5 i 8. Kolejność poprawna, brak zmiany.
   - Porównujemy 8 i 6. Zamieniamy miejscami: [3, 5, 6, 8].
2. **Drugie przejście:**
   - Porównujemy 3 i 5. Kolejność poprawna, brak zmiany.
   - Porównujemy 5 i 6. Kolejność poprawna, brak zmiany.
   - Porównujemy 6 i 8. Kolejność poprawna, brak zmiany.

Lista jest teraz posortowana: [3, 5, 6, 8].
Inny przykład:

$$[\underbrace{4,2}_{4>2},5,1,7] \rightarrow [2,\underbrace{4,5}_{4<5},1,7] \rightarrow [2,4,\underbrace{5,1}_{5>1},7] \rightarrow [2,4,1,\underbrace{5,7}_{5<7}]$$

$$[\underbrace{2,4}_{2<4},1,5,7] \rightarrow [2,\underbrace{4,1}_{4>1},5,7] \rightarrow [2,1,\underbrace{4,5}_{4<5},7]$$

$$[\underbrace{2,1}_{2>1},4,5,7] \rightarrow [1,\underbrace{2,4}_{2<4},5,7]$$

$$[\underbrace{1,2}_{1<2},4,5,7]$$

Algorytm w formie kroków

1. Pobierz ciąg liczb do posortowania.
2. Powtarzaj poniższe kroki dla wszystkich elementów w ciągu:
   - Przejdź przez listę, porównując sąsiednie elementy.
   - Jeśli element lewy jest większy od prawego, zamień je miejscami.
3. Jeśli podczas przejścia nie wykonano żadnej zamiany, zakończ sortowanie.
4. Wypisz posortowany ciąg liczb.

Algorytm w pseudojęzyku

```
Wejście: lista liczb L o długości n
Powtarzaj
  zamieniono = fałsz
  Dla i od 0 do n - 2
    Jeśli L[i] > L[i + 1] wtedy
      Zamień L[i] z L[i + 1]
```

zamieniono = prawda
Jeśli zamieniono = fałsz wtedy
Zakończ
Zwróć L

Implementacja w Pythonie

```
def bubble_sort(lista):
    n = len(lista)
    for _ in range(n):
        zamieniono = False
        for i in range(n - 1):
            if lista[i] > lista[i + 1]:
                lista[i], lista[i + 1] = lista[i + 1], lista[i]
                zamieniono = True
        if not zamieniono:
            break
    return lista

# Przykładowa lista
lista = [34, 7, 23, 32, 5, 62, 3, 12, 43, 9]
print("Posortowana lista:", bubble_sort(lista))
```

ZADANIA DO ROZWIĄZANIA

1. **Sortowanie listy wprowadzonej przez użytkownika**: Napisz program, który poprosi użytkownika o podanie listy liczb oddzielonych spacjami, a następnie posortuje je rosnąco za pomocą algorytmu bąbelkowego.
2. **Sortowanie losowej listy**: Napisz program, który wygeneruje losową listę 20 liczb całkowitych z zakresu od 1 do 100 i posortuje ją malejąco algorytmem bąbelkowym.
3. **Sortowanie listy z pliku tekstowego**: Wczytaj listę liczb zapisanych w pliku tekstowym *liczby1.txt* (każda liczba w nowej linii) i posortuj je rosnąco algorytmem bąbelkowym.
4. **Porównanie liczby zamian**: Napisz program, który obliczy i wypisze liczbę zamian dokonanych podczas sortowania losowo wygenerowanej listy 15 liczb.
5. **Wielokrotne sortowanie**: Napisz program, który odczyta 10 różnych list liczb z pliku tekstowego *liczby2.txt* (każdy wiersz zawiera liczby do listy) i posortuje malejąco każdą z nich oddzielnie, zapisując wyniki do nowego pliku tekstowego *wyniki2.txt*.