Sortowanie

Co to jest sortowanie?

Sortowanie to proces układania danych w określonym porządku – rosnącym lub malejącym.
Przykład: uporządkowanie listy ocen, nazwisk, numerów.

Dlaczego sortowanie jest ważne

- Ułatwia przeszukiwanie tablic,
- Porządkuje dane do analizy,
- W Javie – często używane w listach i tablicach.

Typy sortowania

1. Wewnętrzne – sortowanie w pamięci (tablice, ArrayList),

2. Zewnętrzne – duże zbiory danych (pliki, bazy danych).

Podejścia do sortowania

- Porównawcze – Bubble, Selection, Quick, Merge, Insertion
- Nieporównawcze – Counting Sort, Radix Sort

Sortowanie bąbelkowe (Bubble Sort)

Nazwa pochodzi od „bąbelków” – największe elementy wypływają na górę listy.

Idea

Porównuj sąsiednie elementy:

- jeśli są w złej kolejności – zamień je miejscami,
- powtarzaj aż do końca listy.

Przykład działania

Dane: [5, 3, 8, 4, 2]

1. przebieg: [3, 5, 8, 4, 2]
2. przebieg: [3, 5, 4, 8, 2]
3. przebieg: [3, 4, 5, 2, 8]
4. przebieg: [3, 4, 2, 5, 8]
5. przebieg: [3, 2, 4, 5, 8]
6. przebieg: [2, 3, 4, 5, 8]

Zalety i wady

Zalety:

- Prosty do zrozumienia i napisania,
- Dobry do nauki.

Wady:

- Bardzo wolny przy dużych zbiorach danych.

Przykład

Selection Sort

Sortowanie przez wybór. W każdym przebiegu znajdowany jest najmniejszy element i przenoszony na początek tablicy.

Idea

- Dzielimy tablicę na część posortowaną i nieposortowaną.
- W nieposortowanej części szukamy najmniejszego elementu.
- Zamieniamy go z pierwszym elementem nieposortowanej części.
- Powtarzamy dla pozostałej części tablicy.

Przykład działania

Tablica: [5, 3, 8, 4, 2]

1. Znajdź min w [5,3,8,4,2] → 2 → zamień z 5 → [2,3,8,4,5]
2. Znajdź min w [3,8,4,5] → 3 → już na miejscu
3. Znajdź min w [8,4,5] → 4 → zamień z 8 → [2,3,4,8,5]
4. Znajdź min w [8,5] → 5 → zamień → [2,3,4,5,8]

Zalety i wady

Zalety:

- prosty
- przewidywalny czas działania
- niewiele zamian.

Wady:

- powolny dla dużych zbiorów ($O(n^2)$)
- nieefektywny w porównaniu do szybszych algorytmów

Przykład

Insertion Sort

Sortowanie przez wstawianie. Wstawia każdy element w odpowiednie miejsce w już posortowanej części tablicy.

Idea

- Pierwszy element traktujemy jako posortowany.
- Kolejne elementy wstawiamy w odpowiednie miejsce w posortowanej części.
- Przesuwamy większe elementy w prawo, aby zrobić miejsce dla nowego elementu.

Przykład działania

Tablica: [5, 3, 8, 4, 2]

1. Posortowana [5], wstaw 3 → [3,5]
2. Wstaw 8 → [3,5,8]
3. Wstaw 4 → [3,4,5,8]
4. Wstaw 2 → [2,3,4,5,8]

Zalety i wady

Zalety:

- szybki dla prawie posortowanych danych
- prosty do implementacji.

Wady:

- $O(n^2)$ w najgorszym przypadku
- nieefektywny dla dużych zbiorów.

Przykład

Quick Sort

Sortowanie szybkie (Quick Sort). Dzieli tablicę na mniejsze i większe elementy względem pivotu.

Idea

- Wybierz pivot (np. środkowy element).
- Podziel tablicę na dwie części: elementy mniejsze od pivot i większe od pivot.
- Rekurencyjnie sortuj obie części.
- Połącz wyniki.

Przykład działania

Tablica: [5,3,8,4,2], pivot = 5

- Mniejsze: [3,4,2]
- Większe: [8]
- Rekurencyjnie sortujemy [3,4,2] → [2,3,4]
- Łączymy

Zalety i wady

Zalety:

- bardzo szybki O(n log n) w średnim przypadku
- bardzo wydajny.

Wady:

- $O(n^2)$ w najgorszym przypadku (np. tablica już posortowana i złe pivoty).

Przykład

Merge Sort

Sortowanie przez scalanie. Dzieli tablicę na pół, sortuje rekurencyjnie i scala w jedną posortowaną tablicę.

Idea

- Dziel tablicę na mniejsze części aż do jednoelementowych.
- Scalaj po dwie części, porównując elementy i tworząc posortowaną tablicę.

Przykład działania

Tablica: [5,3,8,4,2]

- Dzielimy: [5,3,8] i [4,2]
- Dzielimy dalej: [5,3] i [8] → [5],[3]
- Scalanie [3,5] + [8] → [3,5,8]
- [4,2] → [2,4]
- Scalamy [3,5,8] + [2,4] → [2,3,4,5,8]

Zalety i wady

Zalety:

- stabilny
- zawsze O(n log n)
- dobrze działa dla dużych zbiorów.

Wady:

- wymaga dodatkowej pamięci (tablice pomocnicze).

Przykład

Counting Sort (sortowanie zliczaniem)

Nieporównawcze sortowanie, działa dla liczb całkowitych w ograniczonym zakresie. Liczy wystąpienia każdego elementu.

Idea

- Tworzymy tablicę liczników dla wszystkich wartości.
- Zliczamy ile razy każdy element występuje.
- Generujemy posortowaną tablicę na podstawie liczników.

Przykład działania

Tablica: [4,2,2,8,3,3,1]

- Liczniki: [1:1, 2:2, 3:2, 4:1, 8:1]
- Posortowana: [1,2,2,3,3,4,8]

Zalety i wady

Zalety:

- O(n+k) bardzo szybkie dla małych zakresów liczb
- stabilne.

Wady:

- tylko liczby całkowite
- wymaga pamięci dla tablicy liczników.

Przykład

Radix Sort

Sortowanie cyfrowe, działa dla liczb całkowitych, sortując po kolejnych cyfrach od najmniej znaczącej do najbardziej znaczącej.

Idea

- Sortujemy tablicę według najmniej znaczącej cyfry (LSD).
- Następnie według kolejnej cyfry itd. aż do cyfry najbardziej znaczącej.
- Można użyć Counting Sort jako podsortowanie dla każdej cyfry.

Przykład działania

Tablica: [170, 45, 75, 90, 802, 24, 2, 66]

- Sortujemy po jedności → [170, 90, 802, 2, 24, 45, 75, 66]
- Po dziesiątkach → [802, 2, 24, 45, 66, 170, 75, 90]
- Po setkach → [2, 24, 45, 66, 75, 90, 170, 802]

Zalety i wady

Zalety:

- szybki O(nk) dla liczb całkowitych
- stabilny.

Wady:

- działa tylko dla liczb całkowitych lub znakowanych
- wymaga dodatkowej pamięci

Przykład