

# Podstawowe Struktury Danych i Algorytmy

Poniżej znajdziesz szczegółowe wyjaśnienia najważniejszych tematów związanych ze strukturami danych i algorytmami, wraz z przykładami i schematami blokowymi przedstawionymi w prosty sposób.

## 1. Złożoność obliczeniowa i Notacja asymptotyczna

### Złożoność obliczeniowa

Złożoność obliczeniowa określa, jak czas wykonania algorytmu rośnie wraz ze wzrostem wielkości danych wejściowych. Mierzymy ją w zależności od liczby operacji, które algorytm wykonuje.

### Notacja asymptotyczna

#### Notacja Big O (O)

Opisuje górną granicę złożoności algorytmu, czyli najgorszy możliwy scenariusz.

**Przykład:** Dla algorytmu przeszukiwania liniowego złożoność to  $O(n)$ , gdzie  $n$  to liczba elementów.

#### Notacja Theta ( $\Theta$ )

Opisuje dokładną złożoność algorytmu, zarówno górną, jak i dolną granicę.

**Przykład:** Dla algorytmu sortowania przez wstawianie w przypadku średnim, złożoność to  $\Theta(n^2)$ .

### Przykłady złożoności

Algorytm	Złożoność
Przeszukiwanie liniowe	$O(n)$
Sortowanie bąbelkowe	$O(n^2)$
Sortowanie szybkie	$O(n \log n)$
Wyszukiwanie binarne	$O(\log n)$

## 2. Algorytmy wyszukiwania i sortowania

### Wyszukiwanie

#### Wyszukiwanie liniowe

Przeszukuje każdy element po kolei.

**Przykład:** Szukamy liczby 5 w tablicy [1, 3, 5, 7, 9].

- 1. Sprawdzamy 1 → nie
- 2. Sprawdzamy 3 → nie

3. Sprawdzamy 5 → tak

**Złożoność:**  $O(n)$

### Wyszukiwanie binarne

Działa na posortowanej tablicy, dzieli ją na połowy.

**Przykład:** Szukamy liczby 5 w tablicy [1, 3, 5, 7, 9].

1. Środkowy element to 5 → znaleziono

**Złożoność:**  $O(\log n)$

### Sortowanie

#### Sortowanie bąbelkowe (Bubble Sort)

Porównuje sąsiednie elementy i zamienia je, jeśli są w złej kolejności. Powtarza to, aż cała tablica będzie posortowana.

**Przykład:**

Tablica przed sortowaniem: [5, 2, 9, 1]

1. [2, 5, 9, 1]
2. [2, 5, 1, 9]
3. [2, 1, 5, 9]
4. [1, 2, 5, 9]

**Złożoność:**  $O(n^2)$

#### Sortowanie szybkie (Quick Sort)

Dzieli tablicę na mniejsze podtablice wokół pivotu, a następnie sortuje je rekurencyjnie.

**Przykład:**

Tablica: [5, 2, 9, 1]

1. Wybieramy pivot, np. 5.
2. Dzielimy na [2, 1] i [9].
3. Sortujemy [2, 1] → [1, 2].
4. Łączymy: [1, 2, 5, 9]

**Złożoność:**  $O(n \log n)$

### Inne sortowania

- **Sortowanie przez wstawianie (Insertion Sort):**  $\Theta(n^2)$
- **Sortowanie przez wybieranie (Selection Sort):**  $O(n^2)$
- **Sortowanie przez scalanie (Merge Sort):**  $O(n \log n)$

### 3. Listy z dowiązaniem

Listy z dowiązaniem to struktura danych składająca się z węzłów, gdzie każdy węzeł zawiera dane oraz wskaźnik do następnego węzła.

Przykład listy jednokierunkowej

```
[1 | next] -> [2 | next] -> [3 | next] -> null
```

Operacje na listach

- **Dodawanie elementu:** Wstawienie na początku lub końcu listy.
- **Usuwanie elementu:** Usunięcie węzła z listy.
- **Przeszukiwanie:** Przejście przez listę od początku do końca.

Schemat blokowy listy jednokierunkowej

```
+-----+   +-----+   +-----+   +-----+
| 1 |next| -> | 2 |next| -> | 3 |next| -> |null|
+-----+   +-----+   +-----+   +-----+
```

### 4. Stosy, kolejki, kopiec binarny, kolejki priorytetowe

Stosy (Stacks)

Stos działa na zasadzie LIFO (Last In, First Out). Ostatni dodany element jest pierwszy usuwany.

**Przykład operacji:**

- **Push:** Dodanie elementu na szczyt stosu.
- **Pop:** Usunięcie elementu ze szczytu stosu.

**Schemat stosu:**

```
+-----+
| 3 | <- Top
+-----+
| 2 |
+-----+
| 1 |
+-----+
```

Kolejki (Queues)

Kolejka działa na zasadzie FIFO (First In, First Out). Pierwszy dodany element jest pierwszy usuwany.

**Przykład operacji:**

- **Enqueue:** Dodanie elementu na koniec kolejki.
- **Dequeue:** Usunięcie elementu z początku kolejki.

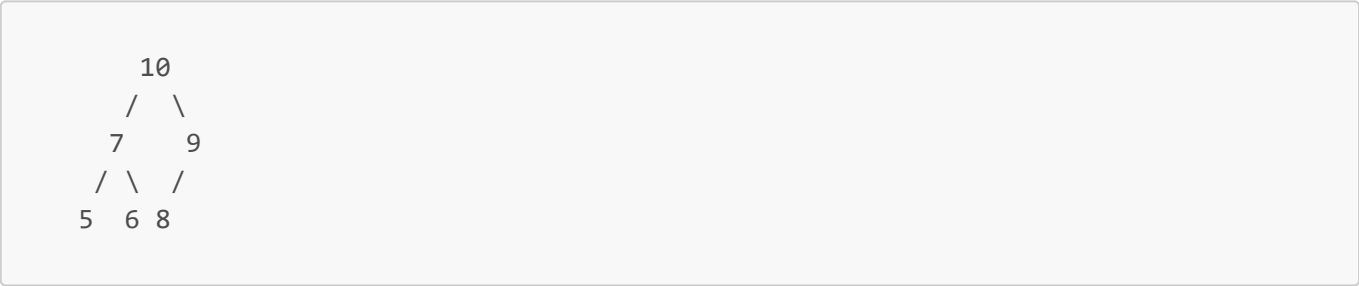
**Schemat kolejki:**



**Kopiec binarny (Binary Heap)**

Struktura danych oparta na drzewie binarnym, gdzie każdy rodzic jest większy (kopiec maksymalny) lub mniejszy (kopiec minimalny) od swoich dzieci.

**Schemat kopca maksymalnego:**



**Kolejki priorytetowe (Priority Queues)**

Kolejka, w której każdy element ma przypisaną wartość priorytetu. Elementy są usuwane zgodnie z priorytetem.

**Przykład:**

Elementy z priorytetami:

- A (priorytet 2)
- B (priorytet 1)
- C (priorytet 3)

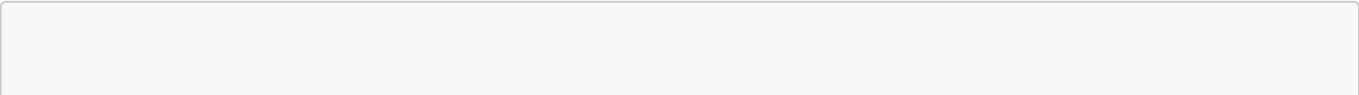
Usuwanie: C, A, B

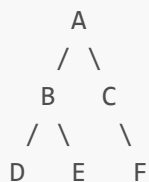
**5. Drzewa**

**Drzewa binarne (Binary Trees)**

Każdy węzeł ma co najwyżej dwóch potomków: lewego i prawego.

**Schemat drzewa binarnego:**

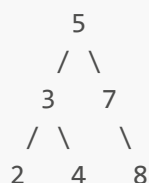




## Drzewa BST (Binary Search Trees)

Drzewo binarne, w którym dla każdego węzła wszystkie elementy w lewym poddrzewie są mniejsze, a w prawym większe od wartości węzła.

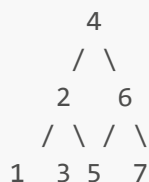
### Schemat BST:



## Drzewa AVL

Są to samobalansujące drzewa BST, które utrzymują różnicę wysokości poddrzew dla każdego węzła nie większą niż 1.

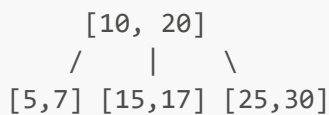
### Schemat drzewa AVL:



## B-drzewa (B-Trees)

Drzewa wielokierunkowe, używane głównie w systemach baz danych i plików, umożliwiające efektywne przeszukiwanie dużych zbiorów danych.

### Schemat B-drzewa stopnia 3:



## 6. Grafy i podstawowe algorytmy grafowe

## Grafy

Grafy składają się z wierzchołków (punktów) i krawędzi (połączeń między nimi).

### Rodzaje grafów:

- **Nieskierowane:** Krawędzie nie mają kierunku.
- **Skierowane:** Krawędzie mają kierunek.

### Schemat grafu nieskierowanego:

```
A -- B
|    |
C -- D
```

## Podstawowe algorytmy grafowe

### Algorytm BFS (Breadth-First Search)

Przeszukiwanie wszerek grafu, odwiedzając najpierw wszystkie sąsiadujące wierzchołki.

#### Przykład:

Graf:

```
A -- B -- C
|    |
D -- E
```

BFS zaczynając od A: A, B, D, C, E

### Algorytm DFS (Depth-First Search)

Przeszukiwanie w głąb grafu, eksplorując jak najdalej w jednym kierunku przed powrotem.

#### Przykład:

Graf:

```
A -- B -- C
|    |
D -- E
```

DFS zaczynając od A: A, B, C, E, D

### Algorytm Dijkstry

Znajduje najkrótszą ścieżkę między dwoma wierzchołkami w grafie z wagami na krawędziach.

**Przykład:**

Graf z wagami:

```
A --2-- B --1-- C
|       |
4       3
|       |
D -----E
```

Najkrótsza ścieżka z A do C:  $A \rightarrow B \rightarrow C$  (koszt 3)

## Podsumowanie

Powyższe struktury danych i algorytmy są fundamentem informatyki i programowania. Zrozumienie ich pozwala na efektywne rozwiązywanie problemów oraz optymalizację kodu. Pamiętaj, że wybór odpowiedniej struktury danych oraz algorytmu ma kluczowe znaczenie dla wydajności aplikacji.

## Pytania Egzaminacyjne z Struktur Danych i Algorytmów

Poniżej znajdziesz zestaw pytań do każdego z sześciu głównych tematów związanych ze strukturami danych i algorytmami. Pod każdym pytaniem znajduje się prawidłowa odpowiedź, która pomoże Ci w przygotowaniu się do egzaminu.

### 1. Złożoność obliczeniowa i Notacja asymptotyczna

#### Pytanie 1.1

**Co opisuje notacja Big O (O) w analizie złożoności algorytmów?**

**Odpowiedź:** Notacja Big O opisuje górną granicę złożoności algorytmu, czyli najgorszy możliwy scenariusz wzrostu czasu wykonania w zależności od wielkości danych wejściowych.

#### Pytanie 1.2

**Jaka jest złożoność czasowa algorytmu sortowania szybkim (Quick Sort) w średnim przypadku?**

**Odpowiedź:** Średnia złożoność czasowa Quick Sort wynosi  $O(n \log n)$ .

#### Pytanie 1.3

**Czym różni się notacja Theta ( $\Theta$ ) od notacji Big O (O)?**

**Odpowiedź:** Notacja Theta ( $\Theta$ ) opisuje dokładną złożoność algorytmu, zarówno górną, jak i dolną granicę, podczas gdy notacja Big O (O) opisuje tylko górną granicę.

### 2. Algorytmy wyszukiwania i sortowania

## Pytanie 2.1

**Opisz działanie algorytmu sortowania bąbelkowego (Bubble Sort).**

**Odpowiedź:** Algorytm sortowania bąbelkowego porównuje sąsiednie elementy tablicy i zamienia je miejscami, jeśli są w złej kolejności. Proces ten powtarza się wielokrotnie, aż cała tablica będzie posortowana.

## Pytanie 2.2

**Jaka jest złożoność czasowa algorytmu sortowania przez wstawianie (Insertion Sort) w najgorszym przypadku?**

**Odpowiedź:** Najgorszy przypadek dla Insertion Sort ma złożoność czasową  $\Theta(n^2)$ .

## Pytanie 2.3

**W jakich warunkach algorytm sortowania szybkiego (Quick Sort) osiąga najlepszą wydajność?**

**Odpowiedź:** Quick Sort osiąga najlepszą wydajność, gdy pivot dzieli tablicę na mniej więcej równe części, co prowadzi do złożoności czasowej  $O(n \log n)$ .

## Pytanie 2.4

**Jak działa algorytm wyszukiwania binarnego i jaka jest jego złożoność czasowa?**

**Odpowiedź:** Wyszukiwanie binarne działa na posortowanej tablicy, dzieląc ją na połowy i sprawdzając, w której połowie znajduje się szukany element. Proces ten powtarza się rekurencyjnie na odpowiedniej połowie, aż element zostanie znaleziony lub tablica zostanie całkowicie przeszukana. Złożoność czasowa wynosi  $O(\log n)$ .

# 3. Listy z dowiązaniem

## Pytanie 3.1

**Czym różni się lista jednokierunkowa od dwukierunkowej?**

**Odpowiedź:** Lista jednokierunkowa posiada wskaźnik tylko do następnego elementu, co umożliwia przeglądanie listy w jednym kierunku. Lista dwukierunkowa posiada dodatkowo wskaźnik do poprzedniego elementu, co pozwala na przeglądanie listy w obu kierunkach.

## Pytanie 3.2

**Jakie są podstawowe operacje na liście z dowiązaniem?**

**Odpowiedź:** Podstawowe operacje to:

- **Dodawanie elementu:** Wstawienie na początku lub końcu listy.
- **Usuwanie elementu:** Usunięcie węzła z listy.
- **Przeszukiwanie:** Przejście przez listę od początku do końca w celu znalezienia elementu.

# 4. Stosy, kolejki, kopiec binarny, kolejki priorytetowe



## Pytanie 4.1

**Wyjaśnij zasadę działania stosu (Stack) i podaj przykład zastosowania.**

**Odpowiedź:** Stos działa na zasadzie LIFO (Last In, First Out), co oznacza, że ostatni dodany element jest pierwszy usuwany. Przykładem zastosowania stosu jest zarządzanie wywołaniami funkcji w programie, gdzie najnowsze wywołanie jest obsługiwane jako pierwsze.

## Pytanie 4.2

**Czym różni się kolejka priorytetowa od zwykłej kolejki?**

**Odpowiedź:** W kolejce priorytetowej każdy element ma przypisany priorytet, a elementy są usuwane zgodnie z ich priorytetem, niezależnie od kolejności ich dodania. W zwykłej kolejce elementy są usuwane w kolejności ich dodania (FIFO).

## Pytanie 4.3

**Jakie są główne cechy kopca binarnego?**

**Odpowiedź:** Kopiec binarny to drzewo binarne, w którym każdy rodzic jest większy (kopiec maksymalny) lub mniejszy (kopiec minimalny) od swoich dzieci. Dodatkowo, kopiec binarny jest kompletnym drzewem binarnym, co oznacza, że wszystkie poziomy są w pełni wypełnione, z wyjątkiem ostatniego, który jest wypełniony od lewej.

# 5. Drzewa

## Pytanie 5.1

**Czym różni się drzewo BST (Binary Search Tree) od zwykłego drzewa binarnego?**

**Odpowiedź:** Drzewo BST (Binary Search Tree) to drzewo binarne, w którym dla każdego węzła wszystkie elementy w lewym poddrzewie są mniejsze, a w prawym poddrzewie większe od wartości węzła. W zwykłym drzewie binarnym nie ma takiego ograniczenia dotyczącego wartości przechowywanych w węzłach.

## Pytanie 5.2

**Jakie właściwości charakteryzują drzewa AVL?**

**Odpowiedź:** Drzewa AVL to samobalansujące drzewa BST, które utrzymują różnicę wysokości poddrzew dla każdego węzła nie większą niż 1. Dzięki temu zapewniają zrównoważoną strukturę, co gwarantuje złożoność operacji w  $O(\log n)$ .

## Pytanie 5.3

**W jakich zastosowaniach używa się B-drzew?**

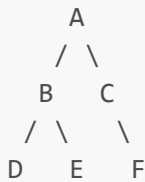
**Odpowiedź:** B-drzewa są używane głównie w systemach baz danych i systemach plików do efektywnego przechowywania i przeszukiwania dużych zbiorów danych. Ich struktura umożliwia szybki dostęp do danych na dyskach twardych, gdzie operacje odczytu i zapisu są kosztowne.

## Pytanie 5.4

**Opisz schemat drzewa binarnego i podaj przykład.**

**Odpowiedź:** Drzewo binarne to struktura danych, w której każdy węzeł ma co najwyżej dwóch potomków: lewego i prawego.

**Przykład:**



## 6. Grafy i podstawowe algorytmy grafowe

Pytanie 6.1

**Czym różni się graf skierowany od grafu nieskierowanego?**

**Odpowiedź:** Graf skierowany ma krawędzie z określonym kierunkiem, co oznacza, że połączenie między dwoma wierzchołkami jest jednokierunkowe. W grafie nieskierowanym krawędzie nie mają kierunku, co oznacza, że połączenie jest dwukierunkowe.

Pytanie 6.2

**Opisz algorytm BFS (Breadth-First Search) i podaj jego zastosowanie.**

**Odpowiedź:** Algorytm BFS przeszukuje graf wszerz, odwiedzając najpierw wszystkie sąsiadujące wierzchołki, a następnie przechodząc do następnego poziomu. Jest używany m.in. do znajdowania najkrótszej ścieżki w grafach nieskierowanych oraz w problemach związanych z połączeniami i sieciami.

Pytanie 6.3

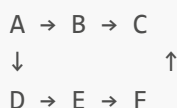
**Jak działa algorytm Dijkstry i do czego jest używany?**

**Odpowiedź:** Algorytm Dijkstry znajduje najkrótszą ścieżkę między dwoma wierzchołkami w grafie z dodatnimi wagami na krawędziach. Jest szeroko stosowany w nawigacji, routingu sieciowym oraz w problemach optymalizacyjnych związanych z trasami.

Pytanie 6.4

**Podaj przykład grafu skierowanego i opisz jego strukturę.**

**Odpowiedź:** Przykład grafu skierowanego:



Struktura tego grafu zawiera wierzchołki A, B, C, D, E, F oraz skierowane krawędzie od A do B, A do D, B do C, D do E, E do F, i F do C.

## Dodatkowe Pytania: Typy Sortowań i Drzew

### Sortowania

#### Pytanie S1

**Wyjaśnij, jak działa sortowanie przez scalanie (Merge Sort) i jaka jest jego złożoność czasowa.**

**Odpowiedź:** Merge Sort działa na zasadzie dzielenia tablicy na mniejsze podtablice, sortowania ich rekurencyjnie, a następnie scalania posortowanych podtablic w jedną posortowaną tablicę. Jego złożoność czasowa wynosi  $O(n \log n)$ .

#### Pytanie S2

**Czym jest sortowanie przez wybieranie (Selection Sort) i jaka jest jego złożoność czasowa?**

**Odpowiedź:** Sortowanie przez wybieranie polega na wielokrotnym znajdowaniu najmniejszego (lub największego) elementu z nieposortowanej części tablicy i zamienianiu go z pierwszym nieposortowanym elementem. Złożoność czasowa to  $O(n^2)$ .

### Drzewa

#### Pytanie D1

**Czym charakteryzuje się drzewo AVL i jakie operacje muszą być wykonywane, aby utrzymać jego właściwości?**

**Odpowiedź:** Drzewo AVL to samobalansujące się drzewo BST, które utrzymuje różnicę wysokości między lewym a prawym poddrzewem każdego węzła nie większą niż 1. Aby utrzymać te właściwości podczas dodawania lub usuwania węzłów, stosuje się rotacje (lewą, prawą, lewo-prawą lub prawo-lewą).

#### Pytanie D2

**Co to jest drzewo B i jakie są jego główne zalety?**

**Odpowiedź:** Drzewo B jest wielokierunkowym drzewem samobalansującym się, które pozwala na przechowywanie wielu kluczy w każdym węźle. Główne zalety to efektywne zarządzanie dużymi zbiorami danych oraz minimalizacja liczby operacji dyskowych dzięki szerokim węzłom.

#### Pytanie D3

**Opisz różnice między drzewem binarnym a drzewem BST.**

**Odpowiedź:** Drzewo binarne to struktura, w której każdy węzeł ma co najwyżej dwóch potomków bez dodatkowych ograniczeń. Drzewo BST (Binary Search Tree) to drzewo binarne, w którym dla każdego węzła wszystkie elementy w lewym poddrzewie są mniejsze, a w prawym poddrzewie większe od wartości węzła.

**Pytanie D4**

**Jakie są zalety używania drzewa AVL w porównaniu do zwykłego drzewa BST?**

**Odpowiedź:** Drzewa AVL są zawsze zrównoważone, co zapewnia gwarantowaną złożoność operacji w  $O(\log n)$ . Zwykłe drzewa BST mogą stać się niezrównoważone, co prowadzi do pogorszenia wydajności operacji do  $O(n)$  w najgorszym przypadku.