SPRAWDZIAN · INFORMATYKA

# Sprawdzian ze struktur danych **Grupa A**

Informatyka: stos, kolejka, listy, drzewa, tablice haszujące i grafy

Imię i nazwisko: _____

Klasa: _____

Data: _____

Punkty: _____ / 26

10 zadań, 26 pkt. Czytaj uważnie polecenia. W zadaniach otwartych zapisz obliczenia.

**Zadanie 1.** (0–1 pkt)

Tablica haszująca stosuje otwarte adresowanie liniowe (linear probing). Przy dużym współczynniku wypełnienia ($\alpha > 0{,}7$) pojawia się szczególny problem. Który?

(A) Nadmiar pamięci operacyjnej — tablica rezerwuje zbyt wiele pustych komórek na zapas
(B) Nieodwracalna utrata danych — nowe klucze nadpisują elementy zapisane wcześniej
(C) Klasteryzacja — tworzenie długich ciągów zajętych komórek, spowalniających wyszukiwanie
(D) Automatyczne zmniejszenie tablicy — rozmiar spada, gdy liczba elementów przekroczy próg

*Zaznacz jedną poprawną odpowiedź.*

**Zadanie 2.** (0–1 pkt)

Do początkowo pustego binarnego drzewa poszukiwań (BST) wstawiono kolejno klucze: 50, 30, 70, 20, 40, 60. Który klucz zostanie odwiedzony jako trzeci przy przejściu inorder (lewe poddrzewo, węzeł, prawe poddrzewo)?

(A) 30
(B) 50
(C) 20
(D) 40

*Zaznacz jedną poprawną odpowiedź.*

**Zadanie 3.** (0–3 pkt)

Oceń prawdziwość stwierdzeń o strukturach danych.

| | | | |
|---|---|---|---|
| 1. | Kopiec binarny (binary heap) jest szczególnym przypadkiem drzewa binarnych poszukiwań (BST), w którym klucze są posortowane od korzenia do liści. | **P** | **F** |
| 2. | Drzewo AVL to samobalansujące drzewo BST, w którym czynnik balansu każdego węzła (różnica wysokości poddrzew lewego i prawego) należy do zbioru $\{-1, 0, 1\}$. | **P** | **F** |
| 3. | W tablicy haszującej z metodą łańcuchową (chaining) pesymistyczna złożoność wyszukiwania elementu wynosi $O(n)$, gdy wszystkie klucze trafią do tego samego kubełka. | **P** | **F** |

*Przy każdym zdaniu zakreśl P (prawda) albo F (fałsz).*

**Zadanie 4.** (0–4 pkt)

Dopasuj strukturę danych do jej kluczowej właściwości.

| | | | |
|---|---|---|---|
| 1. | Stos | ... | A. O(1) dostęp do elementu minimum |
| 2. | Kolejka | ... | B. FIFO (First In First Out) |
| 3. | Kopiec minimalny | ... | C. Średnio O(1) wyszukiwanie według klucza |
| 4. | Tablica haszująca | ... | D. LIFO (Last In First Out) |

*Przy każdym numerze wpisz literę pasującego elementu.*

**Zadanie 5.** (0–4 pkt)

Dopasuj strukturę danych do jej typowego zastosowania (pary CELOWO wymieszane):

| | | |
|---|---|---|
| 1. Stos (stack) | … | A. Implementacja kolejki priorytetowej — element o najwyższym priorytecie na wierzchołku |
| 2. Kolejka (queue) | … | B. Obsługa operacji cofnij/ponów (undo/redo) w edytorze tekstu |
| 3. Graf (lista sąsiedztwa) | … | C. Modelowanie połączeń w sieci społecznościowej |
| 4. Kopiec binarny (heap) | … | D. Planowanie zadań procesora — elementy obsługiwane w kolejności zgłoszenia |

*Przy każdym numerze wpisz literę pasującego elementu.*

**Zadanie 6.** (0–2 pkt)

Uzupełnij zdania dotyczące struktur danych: (1) Struktura działająca według zasady FIFO (pierwszy na wejściu, pierwszy na wyjściu) to (1) _____ (2) Operacja dodawania elementu na szczyt stosu nazywa się (2) _____

**Zadanie 7.** (0–2 pkt)

Dany jest pusty stos S. Wykonujemy operacje: push(3), push(7), push(1), pop(), push(5), pop(), pop(). Napisz jakie wartości zostaną pobrane ze stosu (w kolejności pop-ów) i jaki element pozostanie na stosie.

**Zadanie 8.** (0–3 pkt)

Opisz różnicę między tablicą (array) a listą jednokierunkową (linked list) pod względem dostępu do elementu o zadanym indeksie oraz wstawiania nowego elementu na początek struktury.

**Zadanie 9.** (0–3 pkt)

Wyjaśnij, czym jest drzewo BST (binary search tree) i jaką własność musi spełniać każdy jego węzeł względem wartości w swoim lewym i prawym poddrzewie. Podaj oczekiwaną złożoność czasową wyszukiwania elementu w zrównoważonym drzewie BST.

**Zadanie 10.** (0–3 pkt)

Opisz, jak za pomocą stosu (stack) sprawdzić, czy wyrażenie zawierające nawiasy okrągłe, kwadratowe i klamrowe jest poprawnie domknięte (zbalansowane), np. dla wyrażenia {[a+(b*c)]-d}.

Rozwiąż ten sprawdzian online, z wyjaśnieniem każdej odpowiedzi: matury-online.pl/sprawdzian/informatyka/struktury-danych

SPRAWDZIAN · INFORMATYKA

# Sprawdzian ze struktur danych **Grupa B**

Informatyka: stos, kolejka, listy, drzewa, tablice haszujące i grafy

Imię i nazwisko: _____

Klasa: _____

Data: _____

Punkty: _____ / 26

10 zadań, 26 pkt. Czytaj uważnie polecenia. W zadaniach otwartych zapisz obliczenia.

**Zadanie 1.** (0–1 pkt)

Ile porównań w najgorszym przypadku wykona wyszukiwanie binarne w posortowanej tablicy zawierającej $16$ elementów?

(A) 2
(B) 5
(C) 4
(D) 16

*Zaznacz jedną poprawną odpowiedź.*

**Zadanie 2.** (0–1 pkt)

Dla której struktury danych operacja dostępu do elementu o podanym indeksie ma złożoność $O(1)$?

(A) lista jednokierunkowa
(B) drzewo BST
(C) tablica (array)
(D) stos oparty na liście

*Zaznacz jedną poprawną odpowiedź.*

**Zadanie 3.** (0–3 pkt)

Oceń prawdziwość stwierdzeń o drzewie AVL:

| | | | |
|---|---|---|---|
| 1. | Pesymistyczna złożoność wyszukiwania w drzewie AVL wynosi $O(n)$. | **P** | **F** |
| 2. | Wstawienie elementu do drzewa AVL może wymagać przeprowadzenia rotacji. | **P** | **F** |
| 3. | W drzewie AVL współczynnik wyważenia (balance factor) każdego węzła może wynosić od $-2$ do $2$. | **P** | **F** |

*Przy każdym zdaniu zakreśl P (prawda) albo F (fałsz).*

**Zadanie 4.** (0–4 pkt)

Dopasuj strukturę danych do jej charakterystycznej cechy:

| | | | |
|---|---|---|---|
| 1. | Stos (stack) | … | A. LIFO – ostatni dodany jest pierwszym pobranym |
| 2. | Kolejka (queue) | … | B. Wyszukiwanie w czasie O(log n) – drzewo zrównoważone |
| 3. | Drzewo BST | … | C. FIFO – pierwszy dodany jest pierwszym pobranym |
| 4. | Tablica haszująca | … | D. Średnia złożoność wyszukiwania O(1) |

*Przy każdym numerze wpisz literę pasującego elementu.*

**Zadanie 5.** (0–4 pkt)

Dopasuj strukturę danych do typu zadania, do którego jest najlepiej przystosowana.

| | | | |
|---|---|---|---|
| 1. | Stos (stack) | … | A. szybkie wyszukiwanie wartości po kluczu |
| 2. | Kolejka (queue) | … | B. utrzymywanie posortowanych danych z szybkim wyszukiwaniem |
| 3. | Tablica mieszająca (hash table) | … | C. sprawdzanie poprawności nawiasowania w wyrażeniu |
| 4. | Drzewo BST (binary search tree) | … | D. obsługa zadań w kolejności zgłoszeń (np. drukarka) |

*Przy każdym numerze wpisz literę pasującego elementu.*

**Zadanie 6.** (0–2 pkt)

Uzupełnij zdania dotyczące struktur danych: (1) Struktura, w której nowe elementy dodaje się na końcu, a usuwa z początku, nazywa się (1) _____ (2) Operacja dodania elementu na stos nazywana jest (2) _____

**Zadanie 7.** (0–2 pkt)

Wyjaśnij, na czym polega zasada działania stosu (stack), i podaj jeden przykład zastosowania tej struktury danych w informatyce.

**Zadanie 8.** (0–3 pkt)

Opisz operację extract-min na kopcu minimalnym (min-heap) zaimplementowanym jako tablica. Wyjaśnij każdy krok i podaj złożoność czasową tej operacji.

**Zadanie 9.** (0–3 pkt)

Opisz działanie operacji insert oraz extract-min na kolejce priorytetowej zaimplementowanej jako min-kopiec binarny (min-heap). Podaj złożoność czasową każdej operacji i zilustruj insert elementu 3 do kopca [1, 5, 4, 8, 9].

**Zadanie 10.** (0–3 pkt)

Porównaj dwie metody reprezentacji grafu: macierz sąsiedztwa i listę sąsiedztwa. Dla każdej metody podaj: złożoność pamięciową, złożoność sprawdzenia istnienia krawędzi $(u, v)$ oraz złożoność wylistowania wszystkich sąsiadów wierzchołka $v$.

Rozwiąż ten sprawdzian online, z wyjaśnieniem każdej odpowiedzi: matury-online.pl/sprawdzian/informatyka/struktury-danych

KLUCZ ODPOWIEDZI · DLA NAUCZYCIELA

# Sprawdzian ze struktur danych

## Grupa A

**1.** (1 pkt) **C**

**2.** (1 pkt) **D**

**3.** (3 pkt) **1. F; 2. P; 3. P** — po 1 pkt za zdanie

**4.** (4 pkt) **1–D, 2–B, 3–A, 4–C** — po 1 pkt za parę

**5.** (4 pkt) **1–B, 2–D, 3–C, 4–A** — po 1 pkt za parę

**6.** (2 pkt) **(1) kolejka; (2) push** — po 1 pkt za lukę

**7.** (2 pkt) Przykładowa odpowiedź: Push 3 → S=[3] Push 7 → S=[3,7] Push 1 → S=[3,7,1] Pop → zwraca 1, S=[3,7] Push 5 → S=[3,7,5] Pop → zwraca 5, S=[3,7] Pop → zwraca 7, S=[3] Pobrane wartości: 1, 5, 7 Pozostaje na stosie: 3
Punktacja: Poprawne wartości pobrane przez pop (1 pkt). Poprawny stan końcowy stosu (1 pkt).

**8.** (3 pkt) Przykładowa odpowiedź: Tablica przechowuje elementy w ciągłym obszarze pamięci, dzięki czemu dostęp do elementu o dowolnym indeksie ma złożoność O(1) — obliczany jest bezpośredni adres w pamięci. W liście jednokierunkowej, aby dotrzeć do n-tego elementu, trzeba przejść kolejno przez wszystkie poprzedzające węzły, co daje złożoność O(n). Z kolei wstawienie nowego elementu na początek listy jednokierunkowej wymaga jedynie zmiany jednego wskaźnika — O(1), natomiast w tablicy statycznej trzeba przesunąć wszystkie istniejące elementy o jedną pozycję — O(n). Tablica jest więc korzystniejsza przy częstym dostępie po indeksie, a lista — przy częstym wstawianiu na początek.
Punktacja: Poprawne wskazanie, że dostęp do elementu tablicy po indeksie ma złożoność O(1), a w liście O(n) (1 pkt) + poprawne wskazanie, że wstawienie na początek listy ma złożoność O(1), a w tablicy statycznej wymaga przesunięcia wszystkich elementów O(n) (1 pkt) + poprawny wniosek/podsumowanie, kiedy która struktura jest korzystniejsza (1 pkt). Razem 3 pkt.

**9.** (3 pkt) Przykładowa odpowiedź: Drzewo BST (binary search tree) to drzewo binarne, w którym dla każdego węzła wszystkie wartości znajdujące się w jego lewym poddrzewie są mniejsze od wartości tego węzła, a wszystkie wartości w prawym poddrzewie są od niej większe. Dzięki tej własności wyszukiwanie elementu polega na porównywaniu go z kolejnymi węzłami i schodzeniu w lewo lub w prawo, co przy zrównoważonym drzewie o wysokości $\log_2 n$ daje złożoność czasową $O(\log n)$.
Punktacja: Poprawna definicja BST jako drzewa binarnego przechowującego wartości (1 pkt) + poprawne sformułowanie własności porządkującej: wszystkie wartości w lewym poddrzewie mniejsze, w prawym większe od wartości węzła (1 pkt) + poprawna złożoność $O(\log n)$ dla drzewa zrównoważonego wraz z uzasadnieniem (1 pkt). Maksymalnie 3 pkt.

**10.** (3 pkt) Przykładowa odpowiedź: Przechodzimy przez wyrażenie znak po znaku. Gdy napotkamy nawias otwierający ( ( , [ , { ), odkładamy go na stos. Gdy napotkamy nawias zamykający, zdejmujemy element ze szczytu stosu i sprawdzamy, czy jest to nawias otwierający tego samego typu — jeśli nie (lub stos jest pusty), wyrażenie jest niepoprawne. Po przetworzeniu całego wyrażenia sprawdzamy, czy stos jest pusty — jeśli tak, wszystkie nawiasy zostały poprawnie domknięte i wyrażenie jest zbalansowane. Dla {[a+(b*c)]-d}: kolejno odkładane są {, [, (, a przy napotkaniu ), ], } zdejmowane są w odwrotnej kolejności, aż stos jest pusty — wyrażenie jest poprawne.
Punktacja: Poprawny opis zasady: napotkanie nawiasu otwierającego powoduje odłożenie go na stos (1 pkt) + poprawny opis: napotkanie nawiasu zamykającego powoduje zdjęcie elementu ze szczytu stosu i sprawdzenie, czy pasuje typem do zamykanego nawiasu (1 pkt) + poprawny warunek końcowy: wyrażenie jest zbalansowane, gdy po przetworzeniu całego napisu stos jest pusty (1 pkt). Razem 3 pkt.

## Grupa B

**1.** (1 pkt) **B**

**2.** (1 pkt) **C**

**3.** (3 pkt) **1. F; 2. P; 3. F** — po 1 pkt za zdanie

**4.** (4 pkt) **1–A, 2–C, 3–B, 4–D** — po 1 pkt za parę

**5.** (4 pkt) **1–C, 2–D, 3–A, 4–B** — po 1 pkt za parę

**6.** (2 pkt) **(1) kolejką; (2) push** — po 1 pkt za lukę

**7.** (2 pkt) Przykładowa odpowiedź: Stos działa zgodnie z zasadą LIFO (Last In, First Out) — element dodany do stosu jako ostatni (operacją `push`) jest zawsze pierwszym elementem usuwanym (operacją `pop`); dostęp do pozostałych elementów bez usunięcia elementów znajdujących się nad nimi nie jest możliwy. Przykładem zastosowania stosu jest zarządzanie stosem wywołań funkcji w programie — gdy funkcja wywołuje inną funkcję, informacja o miejscu powrotu jest odkładana na stos, a po zakończeniu wywołanej funkcji zdejmowana z jego wierzchołka.
Punktacja: Poprawne wyjaśnienie zasady LIFO (Last In, First Out) — ostatni dodany element jest pierwszym usuwanym (1 pkt). Podanie trafnego przykładu zastosowania, np. obsługa wywołań funkcji (stos wywołań), historia przeglądarki (cofanie), sprawdzanie poprawności nawiasów (1 pkt). Łącznie 2 pkt.

**8.** (3 pkt) Przykładowa odpowiedź: Extract-min z min-heap: (1) Pobieramy korzeń (indeks 1) — to minimum. (2) Przenosimy ostatni element tablicy na pozycję korzenia i zmniejszamy rozmiar kopca o 1. (3) Uruchamiamy sift-down od korzenia: porównujemy bieżący węzeł z lewym (indeks 2i) i prawym dzieckiem (2i+1); jeśli dziecko jest mniejsze od rodzica, zamieniamy je z mniejszym z dzieci i kontynuujemy rekurencyjnie w dół. Proces trwa najdłużej przez wysokość drzewa h = $\lfloor \log_2 n \rfloor$, więc złożoność to O(log n).
Punktacja: 1. Krok 1: zamiana korzenia (minimum) z ostatnim elementem tablicy, zmniejszenie rozmiaru kopca o 1 (1 pkt). 2. Krok 2: uruchomienie procedury sift-down (min-heapify) od korzenia — porównanie z dziećmi i zamiana z mniejszym dzieckiem, rekurencyjnie do liścia lub do spełnienia własności kopca (1 pkt). 3. Złożoność O(log n) wynikająca z wysokości drzewa $\lfloor \log_2 n \rfloor$ (1 pkt).

**9.** (3 pkt) Przykładowa odpowiedź: Min-heap: każdy rodzic jest ≤ swoich dzieci. Insert(3): dodaj 3 na końcu → [1,5,4,8,9,3]. Sift-up: 3 < parent(4) na indeksie 2 → swap → [1,5,3,8,9,4]. Teraz parent(3)=1 ≤ 3 — koniec. Wynik: [1,5,3,8,9,4]. Extract-min: zdejmij 1, wstaw 4 na korzeń → [4,5,3,8,9]; sift-down: 4 > min(5,3)=3 → swap → [3,5,4,8,9].
Punktacja: Definicja min-heap: rodzic ≤ dzieci, drzewo kompletne (1 pkt). Insert: dodaj element na końcu, sift-up do momentu spełnienia własności kopca — $O(\log n)$ (1 pkt). Extract-min: zdejmij korzeń, wstaw ostatni element na korzeń, sift-down — $O(\log n)$; ilustracja: [1,5,4,8,9,3], 3<4 → swap, [1,5,3,8,9,4] (1 pkt). Razem: 3 pkt.

**10.** (3 pkt) Przykładowa odpowiedź: Macierz sąsiedztwa: pamięć $O(V^2)$, sprawdzenie krawędzi $O(1)$, sąsiedzi $O(V)$. Lista sąsiedztwa: pamięć $O(V + E)$, sprawdzenie krawędzi $O(\deg v)$, sąsiedzi $O(\deg v)$. Macierz lepsza dla gęstych grafów, lista dla rzadkich.
Punktacja: Macierz sąsiedztwa: $O(V^2)$, $O(1)$, $O(V)$ (1 pkt). Lista sąsiedztwa: $O(V + E)$, $O(\deg v)$, $O(\deg v)$ (1 pkt). Rekomendacja: macierz dla grafów gęstych ($E \approx V^2$), lista dla rzadkich ($E \ll V^2$) (1 pkt).

## Skala ocen (orientacyjna, 26 pkt w każdej grupie)

| | |
|---|---|
| bardzo dobry (5) | 24–26 pkt |
| dobry (4) | 20–23 pkt |
| dostateczny (3) | 13–19 pkt |
| dopuszczający (2) | 8–12 pkt |
| niedostateczny (1) | 0–7 pkt |

Progi: 90%, 75%, 50%, 30%. Progi w Twojej szkole mogą być inne. W zadaniach otwartych uznaj każdą odpowiedź poprawną merytorycznie, także sformułowaną inaczej niż przykład.

Ten sprawdzian online, z wyjaśnieniem każdej odpowiedzi: **matury-online.pl/sprawdzian/informatyka/struktury-danych**. Nauczyciele i korepetytorzy ułożą własną kartę z banku ponad 20 tys. zadań: **matury-online.pl/dla-korepetytorow**.