

Algorytmy i programowanie - poziom 5-6

Plan przygotowania na 10 tygodni

Cel materiału

Przygotowanie do pierwszych 2-3 miesięcy nauki na poziomie ocen 5-6. Każdy tydzień zawiera lekcję, ćwiczenie, zadanie rozszerzające i odpowiedź.

Podstawa zakresu

Zakres opracowano bezpośrednio na podstawie czterech tabel z dokumentu: prezentacja algorytmu liniowego, podstawowe zasady programowania, techniki algorytmiczne i wybrane algorytmy oraz elementy analizy algorytmów.

Jak pracować

- 2 spotkania po 45-60 minut w tygodniu
- najpierw rozwiąż zadanie bez odpowiedzi, potem popraw własne rozwiązanie
- prowadź zeszyt prób śladowych, błędów i wniosków
- po każdym module zalicz krótki quiz i zapisz wynik

Rozdzielenie wymagań: ocena 5 i 6

Ocena 5 - samodzielne i poprawne zastosowanie	Ocena 6 - dobór, uzasadnienie i rozszerzenie
• zapisywanie i testowanie algorytmów w wybranej notacji oraz poprawianie błędów	• rygor zapisu w zadanej notacji oraz przenoszenie metod na problemy z innych dziedzin
• funkcje i procedury z parametrami, zasięg zmiennych, tablice i napisy	• ocena efektywności programu i porównywanie implementacji w różnych językach
• pętle zagnieżdżone, dziel i zwyciężaj, Horner, Fibonacci, rekurencja i NWD	• świadomy wybór algorytmu i struktury danych oraz samodzielna praca z dokumentacją
• sortowanie przez wstawianie i inne poznane sortowania, algorytmy tekstowe i zachłanne	• rozumienie stosu rekurencji i wybór między iteracją a rekurencją
• ocena pamięci, porównanie algorytmów i rozpoznanie rozwiązania efektywnego	• trudniejsze problemy: wyszukiwanie wzorca, ośmiu hetmanów, trwałe małżeństwo
	• wyznaczanie złożoności czasowej i pamięciowej, także za pomocą sum i wzorów

Mapa pracy

Tydzień 1: Specyfikacja i notacja algorytmu [zakres 5]

Tydzień 2: Od algorytmu do programu [zakres 5]

Tydzień 3: Funkcje, parametry i metoda zstępująca [zakres 5]

Tydzień 4: Tablice, napisy i dobór struktury danych [zakres 5-6]

Tydzień 5: Iteracja, pętle zagnieżdżone i niezmiennik [zakres 5-6]

Tydzień 6: Sortowanie i porównanie metod [zakres 5-6]

Tydzień 7: Rekurencja, stos i algorytm Euklidesa [zakres 5-6]

Tydzień 8: Dziel i zwyciężaj, Fibonacci, Horner i zachłanność [zakres 5]

Tydzień 9: Algorytmy tekstowe i trudniejsze problemy [zakres 5-6]

Tydzień 10: Złożoność czasowa i pamięciowa [zakres 5-6]

Tydzień 1. Specyfikacja i notacja algorytmu

Poziom

Zakres docelowy: ocena 5

Po tym module

- zapiszesz dane, wynik i warunki brzegowe
- utworzysz listę kroków, pseudokod i schemat blokowy
- przeprowadzisz próbę śladową

Lekcja

Algorytm jest skończonym, jednoznacznym przepisem. Specyfikacja odpowiada na trzy pytania: jakie są dane, jaki ma być wynik i jakie założenia muszą spełniać dane.

Dobra notacja nie pozostawia domysłów. W schemacie blokowym owal oznacza start/stop, równoległobok wejście/wyjście, prostokąt operację, a romb warunek. Pseudokod powinien mieć wcięcia i jawne zakończenie pętli lub warunku.

Poprawność wstępnie sprawdzamy próbą śladową: ręcznie zapisujemy wartości zmiennych po każdym kroku dla danych typowych, granicznych i błędnych.

```
DANE: a, b - liczby rzeczywiste
WYNIK: większa liczba lub informacja o równości
JEŻELI a > b: wypisz a
W PRZECIWNYM RAZIE JEŻELI b > a: wypisz b
INACZEJ: wypisz 'równe'
```

Ćwiczenie

Zapisz w pseudokodzie algorytm obliczający cenę po rabacie. Rabat 10% obowiązuje od 200 zł, a dane ujemne są błędne. Wykonaj testy dla -1, 199,99, 200 i 350.

Zadanie na ocenę 6

Zapisz ten sam algorytm w liście kroków i na schemacie blokowym. Porównaj, która notacja lepiej ujawnia walidację.

Sprawdź się

Co jest częścią specyfikacji?

- A. kolor edytora
- B. dane, wynik i założenia
- C. nazwa kompilatora
- D. liczba komentarzy

Odpowiedzi - tydzień 1

Odpowiedź do ćwiczenia

Najpierw odrzuć $cena < 0$. Następnie jeśli $cena \geq 200$, $wynik = 0,9 * cena$, w przeciwnym razie $wynik = cena$. Wyniki: błąd; 199,99; 180; 315.

Quiz

Poprawna odpowiedź: B. dane, wynik i założenia. Specyfikacja jest kontraktem rozwiązania.

Kryterium zaliczenia

Potrafisz odtworzyć rozwiązanie bez zaglądania do odpowiedzi, wyjaśnić każdy krok i podać co najmniej jeden test graniczny.

Tydzień 2. Od algorytmu do programu

Poziom

Zakres docelowy: ocena 5

Po tym module

- odróżnisz kompilację od wykonania
- rozpoznasz błędy składni, wykonania i logiczne
- wyjaśnisz rolę procesora i pamięci

Lekcja

Kod źródłowy jest tłumaczony przez kompilator lub wykonywany z udziałem interpretera. Program wynikowy zawiera instrukcje możliwe do wykonania przez procesor; dane i instrukcje w czasie pracy znajdują się w pamięci.

Błąd składni zatrzymuje tłumaczenie, błąd wykonania ujawnia się podczas pracy, a błąd logiczny daje niepoprawny wynik mimo poprawnego uruchomienia.

Program powinien być czytelny: znaczące nazwy, konsekwentne wcięcia, krótkie funkcje i komentarze wyjaśniające decyzję, nie oczywistość.

```
def pole_prostokata(a, b):  
    if a <= 0 or b <= 0:  
        return None  
    return a * b
```

Ćwiczenie

Wskaż rodzaj błędu: (a) brak dwukropka po if, (b) dzielenie przez zero dla jednego testu, (c) obwód policzony jako a*b.

Zadanie na ocenę 6

Wyjaśnij, dlaczego dwie implementacje tego samego algorytmu mogą mieć inną szybkość, mimo identycznej złożoności asymptotycznej.

Sprawdź się

Błąd logiczny...

- A. zawsze blokuje kompilację
- B. daje zły wynik mimo uruchomienia
- C. dotyczy wyłącznie pamięci
- D. jest komentarzem

Odpowiedzi - tydzień 2

Odpowiedź do ćwiczenia
(a) składni, (b) wykonania, (c) logiczny.

Quiz
Poprawna odpowiedź: B. daje zły wynik mimo uruchomienia. Program działa, ale realizuje złą regułę.

Kryterium zaliczenia

Potrafisz odtworzyć rozwiązanie bez zaglądania do odpowiedzi, wyjaśnić każdy krok i podać co najmniej jeden test graniczny.

Tydzień 3. Funkcje, parametry i metoda zstępująca

Poziom

Zakres docelowy: ocena 5

Po tym module

- podzielisz problem na podproblemy
- zastosujesz parametry i wartość zwracaną
- rozpoznasz zasięg lokalny

Lekcja

Metoda zstępująca zaczyna od opisu całego zadania, a następnie rozbija je na małe funkcje. Każda funkcja powinna mieć jedną odpowiedzialność.

Parametr przez wartość jest kopią argumentu. Przekazanie przez referencję pozwala zmienić obiekt wywołującego i powinno być stosowane świadomie. Zmienna lokalna istnieje tylko w swoim bloku.

Kontrakt funkcji obejmuje warunki wstępne, rezultat i możliwe sytuacje błędne. To ułatwia niezależne testowanie.

```
def srednia(dane):  
    if not dane:  
        return None  
    suma = 0  
    for x in dane:  
        suma += x  
    return suma / len(dane)
```

Ćwiczenie

Podziel program analizujący oceny na funkcje: wczytanie, walidacja, średnia, liczba ocen powyżej średniej i prezentacja.

Zadanie na ocenę 6

Porównaj przekazanie dużej tablicy przez wartość i przez stałą referencję w C++. Uwzględnij czas i pamięć.

Sprawdź się

Zmienna lokalna...

- A. jest widoczna wszędzie
- B. istnieje w swoim zakresie
- C. zawsze jest tablicą
- D. musi mieć wartość 0

Odpowiedzi - tydzień 3

Odpowiedź do ćwiczenia

Przykład: `wczytaj_oceny()`, `poprawne(dane)`, `srednia(dane)`, `ile_powyzej(dane, prog)`, `pokaz(wynik)`. Funkcje obliczeniowe nie powinny same wczytywać ani drukować.

Quiz

Poprawna odpowiedź: B. istnieje w swoim zakresie. Zakres ogranicza widoczność i czas życia nazwy.

Kryterium zaliczenia

Potrafisz odtworzyć rozwiązanie bez zaglądania do odpowiedzi, wyjaśnić każdy krok i podać co najmniej jeden test graniczny.

Tydzień 4. Tablice, napisy i dobór struktury danych

Poziom

Zakres docelowy: ocena 5-6

Po tym module

- przetworzysz tablicę i napis
- dobierzesz strukturę do operacji
- unikniesz typowych błędów indeksowania

Lekcja

Tablica lub wektor grupuje elementy jednego typu i umożliwia dostęp przez indeks. Napis jest sekwencją znaków. Typowe wzorce to przejście, filtr, licznik, akumulator, minimum i maksimum.

Dobór struktury wynika z operacji: wektor jest dobry do kolejnego przeglądania i dostępu po indeksie; zbiór do szybkiego sprawdzania obecności; mapa do wiązania klucza z wartością.

Najczęstsze błędy to wyjście poza zakres, inicjalizowanie minimum zerem oraz pominięcie pustego wejścia.

```
def minimum(dane):  
    if not dane:  
        return None  
    m = dane[0]  
    for x in dane[1:]:  
        if x < m:  
            m = x  
    return m
```

Ćwiczenie

Napisz algorytm zliczający częstotliwość znaków w napisie bez rozróżniania wielkości liter.

Zadanie na ocenę 6

Uzasadnij wybór mapy zamiast listy par dla długiego tekstu i dużego alfabetu.

Sprawdź się

Bezpieczna inicjalizacja minimum to...

- A. 0
- B. pierwszy element niepustych danych
- C. 999
- D. długość tablicy

Odpowiedzi - tydzień 4

Odpowiedź do ćwiczenia

Zamień napis na małe litery; dla każdego znaku zwiększ licznik w mapie. Czas $O(n)$, pamięć $O(k)$, gdzie k to liczba różnych znaków.

Quiz

Poprawna odpowiedź: B. pierwszy element niepustych danych. Zero i stałe strażnicze mogą nie należeć do danych.

Kryterium zaliczenia

Potrafisz odtworzyć rozwiązanie bez zaglądania do odpowiedzi, wyjaśnić każdy krok i podać co najmniej jeden test graniczny.

Tydzień 5. Iteracja, pętle zagnieżdżone i niezmiennik

Poziom

Zakres docelowy: ocena 5-6

Po tym module

- zapiszesz pętlę zagnieżdżoną
- dobierzesz warunek zakończenia
- wyjaśnisz niezmiennik pętli

Lekcja

Pętla for jest naturalna, gdy znamy zbiór elementów lub liczbę powtórzeń; while - gdy kończymy po spełnieniu warunku. Każda pętla musi robić postęp ku zakończeniu.

Pętle zagnieżdżone często analizują pary elementów. Dla n elementów pełna para pętli wykonuje około n^2 operacji; jeśli druga pętla zaczyna od $i+1$, analizujemy każdą parę raz.

Niezmiennik to zdanie prawdziwe przed i po każdym obrocie. Pomaga uzasadnić poprawność, np. po i krokach suma zawiera dokładnie pierwsze i elementów.

```
licznik = 0
for i in range(len(a)):
    for j in range(i + 1, len(a)):
        if a[i] == a[j]:
            licznik += 1
```

Ćwiczenie

Określ liczbę porównań dla $n=5$ oraz wzór dla powyższego kodu.

Zadanie na ocenę 6

Zaproponuj wersję $O(n)$ wykrywającą, czy istnieje duplikat, używając zbioru. Omów koszt pamięci.

Sprawdź się

Pętla while jest szczególnie dobra, gdy...

- liczba kroków nie jest z góry znana
- nie ma warunku
- program nic nie powtarza
- używamy tylko napisów

Odpowiedzi - tydzień 5

Odpowiedź do ćwiczenia
$4+3+2+1 = 10$. Ogólnie $n(n-1)/2$, czyli $O(n^2)$.

Quiz
Poprawna odpowiedź: A. liczba kroków nie jest z góry znana. Warunek steruje liczbą powtórzeń.

Kryterium zaliczenia

Potrafisz odtworzyć rozwiązanie bez zaglądania do odpowiedzi, wyjaśnić każdy krok i podać co najmniej jeden test graniczny.

Tydzień 6. Sortowanie i porównanie metod

Poziom

Zakres docelowy: ocena 5-6

Po tym module

- wykonasz sortowanie przez wybór, wstawianie i bąbelkowe
- porównasz liczbę operacji
- rozpoznasz stabilność

Lekcja

Wybór znajduje minimum nieuporządkowanej części i zamienia je z kolejną pozycją. Wstawianie umieszcza element w uporządkowanym prefiksie. Bąbelkowe zamienia sąsiadów i przesuwa największy element na koniec.

W najgorszym przypadku wszystkie trzy metody mają $O(n^2)$. Wstawianie może działać blisko $O(n)$ dla danych prawie uporządkowanych. Sortowanie stabilne zachowuje kolejność elementów o równych kluczach.

Ocena 6 wymaga nie tylko zapisu, ale uzasadnienia wyboru. Dla dużych danych zwykle wybieramy algorytm $O(n \log n)$ z biblioteki, jeśli zadanie nie narzuca implementacji szkolnej.

```
for i in range(1, len(a)):
    x = a[i]
    j = i - 1
    while j >= 0 and a[j] > x:
        a[j + 1] = a[j]
        j -= 1
    a[j + 1] = x
```

Ćwiczenie

Posortuj przez wstawianie [5, 2, 4, 2]. Zapisz stan po każdej iteracji zewnętrznej.

Zadanie na ocenę 6

Dobierz metodę dla: 20 prawie uporządkowanych ocen; miliona rekordów; danych z ograniczonym zakresem 0-100.

Sprawdź się

Sortowanie przez wstawianie buduje...

- A. uporządkowany prefiks
- B. drzewo
- C. stos wywołań
- D. wyłącznie kopię

Odpowiedzi - tydzień 6

Odpowiedź do ćwiczenia

$[2,5,4,2] \rightarrow [2,4,5,2] \rightarrow [2,2,4,5]$. Przy porównaniu $>$ algorytm jest stabilny.

Quiz

Poprawna odpowiedź: A. uporządkowany prefiks. Każdy kolejny element trafia do uporządkowanej części.

Kryterium zaliczenia

Potrafisz odtworzyć rozwiązanie bez zaglądania do odpowiedzi, wyjaśnić każdy krok i podać co najmniej jeden test graniczny.

Tydzień 7. Rekurencja, stos i algorytm Euklidesa

Poziom

Zakres docelowy: ocena 5-6

Po tym module

- zapiszesz przypadek bazowy
- prześledzisz stos wywołań
- porównasz wersję iteracyjną i rekurencyjną

Lekcja

Rekurencja rozwiązuje problem przez wywołanie tej samej funkcji dla mniejszego przypadku. Musi mieć przypadek bazowy i krok zbliżający do niego.

Każde wywołanie zajmuje ramkę stosu z parametrami i miejscem powrotu. Dlatego rekurencja liniowa ma zwykle $O(n)$ pamięci, choć jej wersja iteracyjna może używać $O(1)$.

Algorytm Euklidesa korzysta z własności $NWD(a,b)=NWD(b,a \bmod b)$. Liczba kroków rośnie logarytmicznie względem mniejszej liczby.

```
def nwd(a, b):  
    if b == 0:  
        return abs(a)  
    return nwd(b, a % b)
```

Ćwiczenie

Rozpisz stos wywołań $nwd(48,18)$ i zwroty.

Zadanie na ocenę 6

Zapisz iteracyjną wersję NWD i porównaj czas oraz pamięć obu wersji.

Sprawdź się

Rekurencja musi mieć...

- A. przypadek bazowy
- B. dwie tablice
- C. zmienną globalną
- D. sortowanie

Odpowiedzi - tydzień 7

Odpowiedź do ćwiczenia

$\text{nwd}(48,18) \rightarrow \text{nwd}(18,12) \rightarrow \text{nwd}(12,6) \rightarrow \text{nwd}(6,0) \rightarrow 6$; następnie każda ramka zwraca 6.

Quiz

Poprawna odpowiedź: A. przypadek bazowy. Bez bazy wywołania nie kończą się poprawnie.

Kryterium zaliczenia

Potrafisz odtworzyć rozwiązanie bez zaglądania do odpowiedzi, wyjaśnić każdy krok i podać co najmniej jeden test graniczny.

Tydzień 8. Dziel i zwyciężaj, Fibonacci, Horner i zachłanność

Poziom

Zakres docelowy: ocena 5

Po tym module

- rozpoznasz technikę dziel i zwyciężaj
- obliczysz wielomian schematem Hornera
- ocenisz metodę zachłanną

Lekcja

Dziel i zwyciężaj dzieli problem na mniejsze części, rozwiązuje je i scala wyniki. Klasyczny przykład to wyszukiwanie binarne lub sortowanie przez scalanie.

Schemat Hornera oblicza wielomian $((((a_n \cdot x + a_{n-1}) \cdot x + a_{n-2}) \dots) + a_0)$, używając liniowej liczby mnożeń i dodawań.

Algorytm zachłanny podejmuje lokalnie najlepszą decyzję. Dla polskich nominałów wydawanie reszty działa, ale dla dowolnego zestawu monet może nie dać minimum liczby monet.

```
def horner(wsp, x):
    wynik = wsp[0]
    for a in wsp[1:]:
        wynik = wynik * x + a
    return wynik
```

Ćwiczenie

Oblicz $2x^3 - 3x^2 + 4x - 5$ dla $x=2$ schematem Hornera.

Zadanie na ocenę 6

Dla monet $[1,3,4]$ pokaż, że zachłanne wydanie 6 nie jest optymalne.

Sprawdź się

Schemat Hornera dla n -tego stopnia ma czas...

- A. $O(n)$
- B. $O(n^2)$
- C. $O(2^n)$
- D. $O(1)$ dla dowolnego n

Odpowiedzi - tydzień 8

Odpowiedź do ćwiczenia
Kolejno: 2; $2*2-3=1$; $1*2+4=6$; $6*2-5=7$.

Quiz
Poprawna odpowiedź: A. $O(n)$. Jedno przejście po współczynnikach.

Kryterium zaliczenia

Potrafisz odtworzyć rozwiązanie bez zaglądania do odpowiedzi, wyjaśnić każdy krok i podać co najmniej jeden test graniczny.

Tydzień 9. Algorytmy tekstowe i trudniejsze problemy

Poziom

Zakres docelowy: ocena 5-6

Po tym module

- sprawdzisz palindrom i anagram
- zapiszesz proste wyszukiwanie wzorca
- poznasz ideę nawrotu

Lekcja

Palindrom sprawdzamy parami od końców lub przez porównanie z odwróceniem. Anagramy można rozpoznać po posortowaniu znaków $O(n \log n)$ albo tablicy liczników $O(n)$.

Naiwne wyszukiwanie wzorca porównuje wzorzec w każdym możliwym położeniu i w najgorszym przypadku ma $O(nm)$. Algorytmy KMP lub Boyera-Moore'a ograniczają powtarzanie pracy.

Problem ośmiu hetmanów rozwiązuje się nawrotem: wybieramy bezpieczne pole, schodzimy głębiej, a przy braku możliwości cofamy wybór. Trwałe małżeństwo pokazuje, że poprawność zależy od niezmienników i definicji stabilności.

```
def znajdz(tekst, wzorzec):
    for i in range(len(tekst)-len(wzorzec)+1):
        if tekst[i:i+len(wzorzec)] == wzorzec:
            return i
    return -1
```

Ćwiczenie

Porównaj dwie metody anagramu dla alfabetu łacińskiego: sortowanie i 26 liczników.

Zadanie na ocenę 6

Zapisz pseudokod nawrotu dla n hetmanów i podaj warunek bezpieczeństwa kolumny i przekątnych.

Sprawdź się

Naiwne wyszukiwanie wzorca ma w najgorszym przypadku...

- A. $O(nm)$
- B. $O(1)$
- C. $O(\log n)$
- D. zawsze $O(n)$

Odpowiedzi - tydzień 9

Odpowiedź do ćwiczenia

Sortowanie: $O(n \log n)$, zwykle $O(n)$ pamięci zależnie od implementacji. Liczniki: $O(n)$ czasu i $O(1)$ pamięci, bo tablica ma stałe 26 pól.

Quiz

Poprawna odpowiedź: A. $O(nm)$. Może porównywać m znaków w n położeniach.

Kryterium zaliczenia

Potrafisz odtworzyć rozwiązanie bez zaglądania do odpowiedzi, wyjaśnić każdy krok i podać co najmniej jeden test graniczny.

Tydzień 10. Złożoność czasowa i pamięciowa

Poziom

Zakres docelowy: ocena 5-6

Po tym module

- rozpoznasz $O(1)$, $O(\log n)$, $O(n)$, $O(n \log n)$, $O(n^2)$
- policzysz operacje przez sumę
- porównasz rozwiązania tego samego zadania

Lekcja

Złożoność czasowa opisuje wzrost liczby operacji wraz z rozmiarem danych, a pamięciowa - wzrost dodatkowej pamięci. Pomijamy stałe i składniki niższego rzędu, np. $3n^2 + 2n + 7$ to $O(n^2)$.

Pętla po n elementach daje $O(n)$. Dwie pełne pętle zagnieżdżone dają $O(n^2)$. Pętla dzieląca problem przez 2 daje $O(\log n)$. Sortowanie przez scalanie ma $O(n \log n)$.

Dla trójkątnej pętli liczba iteracji to $1 + 2 + \dots + (n-1) = n(n-1)/2$. Ocena 6 wymaga powiązania wzoru z kodem oraz omówienia przypadku najlepszego, średniego i najgorszego, gdy się różnią.

```
# para liczb o zadanej sumie
# A: każda para ->  $O(n^2)$ , pamięć  $O(1)$ 
# B: zbiór widzianych ->  $O(n)$  średnio, pamięć  $O(n)$ 
```

Ćwiczenie

Określ czas i pamięć: (a) suma tablicy, (b) kopia odwróconej tablicy, (c) wyszukiwanie binarne, (d) wszystkie pary.

Zadanie na ocenę 6

Zaprojektuj dwa algorytmy wykrywania duplikatu, porównaj czas/pamięć i wskaż lepszy dla $n=10^6$ przy ograniczonej pamięci.

Sprawdź się

$3n^2 + 2n + 7$ należy do...

- A. $O(n^2)$
- B. $O(n)$
- C. $O(\log n)$
- D. $O(2^n)$

Odpowiedzi - tydzień 10

Odpowiedź do ćwiczenia
(a) $O(n)/O(1)$; (b) $O(n)/O(n)$; (c) $O(\log n)/O(1)$ iteracyjnie; (d) $O(n^2)/O(1)$, jeśli tylko przetwarzamy pary.

Quiz
Poprawna odpowiedź: A. $O(n^2)$. Dominuje składnik kwadratowy.

Kryterium zaliczenia

Potrafisz odtworzyć rozwiązanie bez zaglądania do odpowiedzi, wyjaśnić każdy krok i podać co najmniej jeden test graniczny.

Test końcowy

Najpierw rozwiąż bez zaglądania do klucza. Za każde pytanie: 1 punkt. Następnie wybierz jedno zadanie projektowe.

1. Schemat blokowy warunku używa...

- A. rombu
- B. owalu
- C. prostokąta
- D. koła

2. Test graniczny dla progu 50 powinien obejmować...

- A. 49, 50, 51
- B. 1, 2, 3
- C. tylko 50
- D. tylko 100

3. Pętla trójkątna wykonuje około...

- A. $n^2/2$ operacji
- B. $\log n$
- C. jedną operację
- D. 2^n

4. NWD Euklidesa ma liczbę kroków rzędu...

- A. $\log n$
- B. n^2
- C. 2^n
- D. $n!$

5. Stabilne sortowanie zachowuje...

- A. kolejność równych kluczy
- B. wyłącznie liczby
- C. stałą pamięć zawsze
- D. odwrotny porządek

6. Rekurencja liniowa zwykle wymaga pamięci...

- A. $O(n)$ na stos
- B. $O(1)$ zawsze
- C. $O(n^2)$ zawsze
- D. zero

7. Horner oblicza wielomian w czasie...

- A. $O(n)$
- B. $O(n^2)$
- C. $O(2^n)$
- D. $O(1)$

8. Zbiór do wykrywania duplikatów daje średnio...

- A. $O(n)$ czasu i $O(n)$ pamięci
- B. $O(n^2)/O(1)$
- C. $O(\log n)/O(1)$
- D. $O(1)/O(1)$

9. Metoda zachłanna...

- A. nie zawsze daje optimum
- B. zawsze daje optimum
- C. jest rekurencją
- D. wymaga sortowania bąbelkowego

10. Ocena 6 w analizie algorytmów wymaga m.in....

- A. wyznaczania czasu i pamięci ze wzorami
- B. samego uruchomienia programu
- C. znajomości koloru bloków
- D. wyłącznie listy kroków

Zadania projektowe

Projekt 1: Analizator wyników

Zaprojektuj program obliczający min, max, średnią, medianę i liczbę wyników powyżej średniej. Podaj specyfikację, funkcje, pięć testów oraz złożoność.

Projekt 2: Porównanie sortowań

Zaimplementuj wybór i wstawianie. Policz porównania i przesunięcia dla danych rosnących, malejących i losowych.

Projekt 3: Wyszukiwanie wzorca

Zapisz wersję naiwną, przetestuj przypadki brzegowe i uzasadnij $O(nm)$.

Rubryka oceny projektu

Kryterium	5	6
Poprawność	działa dla testów typowych i granicznych	uzasadnione niezmienniki i pełna obsługa błędów
Struktura	czytelny podział na funkcje/klasy	uzasadniony dobór struktur i odpowiedzialności
Analiza	czas i pamięć głównych operacji	porównanie alternatyw, wzory i kompromisy
Samodzielność	dokumentacja i czytelny kod	rozszerzenie lub drugi wariant rozwiązania

Klucz odpowiedzi

1. A - rombu. Romb rozdziela wykonanie na gałęzie.
2. A - 49, 50, 51. Wartości tuż przy granicy wykrywają pomyłki $> / \geq$.
3. A - $n^2/2$ operacji. Suma $1+\dots+(n-1)=n(n-1)/2$.
4. A - $\log n$. Reszty szybko zmniejszają argumenty.
5. A - kolejność równych kluczy. To definicja stabilności.
6. A - $O(n)$ na stos. Każde wywołanie ma ramkę stosu.
7. A - $O(n)$. Jedno przejście po n współczynnikach.
8. A - $O(n)$ czasu i $O(n)$ pamięci. Każdy element sprawdzamy i zapamiętujemy.
9. A - nie zawsze daje optimum. Poprawność zachłanności trzeba udowodnić dla danego problemu.
10. A - wyznaczania czasu i pamięci ze wzorami. Tak wynika z tabeli wymagań.

Wskazówki do projektów

Analizator wyników
Sortowanie kopii ułatwia medianę: $O(n \log n)$ czasu i $O(n)$ pamięci. Min, max, suma i licznik są $O(n)$. Oddziel wejście, walidację, obliczenia i prezentację.
Porównanie sortowań
Wybór ma około $n(n-1)/2$ porównań niezależnie od danych. Wstawianie ma blisko n porównań dla danych rosnących i około $n^2/2$ dla malejących.
Wyszukiwanie wzorca
Sprawdź pusty wzorzec, wzorzec dłuższy niż tekst, dopasowanie na początku/końcu i brak. W $n-m+1$ pozycjach może wystąpić do m porównań.

Interpretacja wyniku

- 9-10 pkt + dobry projekt: gotowość do pracy na poziomie 5-6
- 7-8 pkt: dobra podstawa; powtórz moduły powiązane z błędami
- 5-6 pkt: wróć do tygodni 1-6 i wykonaj ćwiczenia ponownie
- 0-4 pkt: przejdź materiał od początku w wolniejszym tempie