

KARTA KURSU

Nazwa	Algorytmy i Struktury Danych
Nazwa w j. ang.	Algorithms and Data Structures

Koordinator	dr Roman Czapla	Zespół dydaktyczny
		dr inż. Magdalena Andrzejewska dr Roman Czapla dr Leszek Głowacki dr Zdobysław Świerczyński
Punktacja ECTS*	5	

Opis kursu (cele kształcenia)

Celem kursu jest zapoznanie studentów z fundamentalnymi oraz wybranymi zaawansowanymi algorytmami i strukturami danych, które stanowią podstawę współczesnej informatyki. Uczestnicy zdobywają wiedzę z zakresu analizy złożoności obliczeniowej, uczą się projektować, analizować i implementować algorytmy oraz stosować różne struktury danych do efektywnego rozwiązywania problemów obliczeniowych.

Wykłady prowadzone są w oparciu o pseudokod, co pozwala na przedstawienie ogólnych metod i idei niezależnie od języka programowania. Ćwiczenia mają charakter praktyczny i obejmują implementację omawianych algorytmów w wybranym języku programowania (preferowany jest język C), w zależności od wymagań kursu i preferencji studentów.

W kontekście kierunku Cyberbezpieczeństwo szczególny nacisk kładziony jest na zrozumienie roli algorytmów i struktur danych w analizie bezpieczeństwa systemów informatycznych, przetwarzaniu dużych zbiorów danych oraz modelowaniu zagrożeń w sieciach komputerowych. Omawiane techniki stanowią podstawę działania mechanizmów kryptograficznych, systemów detekcji zagrożeń oraz narzędzi analizy ruchu sieciowego.

Warunki wstępne

Wiedza	Student powinien znać podstawowe pojęcia matematyki dyskretnej, w tym elementy logiki, teorię zbiorów i relacji oraz podstawy kombinatoryki. Wymagana jest także ogólna orientacja w zakresie złożoności obliczeniowej i umiejętność analitycznego myślenia.
Umiejętności	Student powinien potrafić programować w co najmniej jednym języku wysokiego poziomu (np. język C) na poziomie podstawowym, obejmującym zmienne, typy danych, instrukcje sterujące, funkcje oraz proste struktury danych (tablice, listy). Powinien także umieć rozwiązywać podstawowe zadania algorytmiczne i korzystać z dokumentacji programistycznej.
Kursy	<u>Wymagane zaliczenie kursu:</u> Teoretyczne Podstawy Informatyki, Programowanie, Matematyka dyskretna, Wstęp do matematyki

Efekty uczenia się

	Efekt uczenia się	Odniesienie do efektów kierunkowych
	Po zakończeniu kursu student:	
	W01: zna podstawowe pojęcia algorytmiki, zasady rekurencji oraz techniki analizy złożoności obliczeniowej (notacja asymptotyczna, analiza teoretyczna i praktyczna).	K_W01 K_W09
	W02: zna fundamentalne algorytmy sortowania i wyszukiwania oraz rozumie ich złożoność i zastosowania.	K_W03 K_W08

Wiedza	W03: posiada wiedzę na temat podstawowych i zaawansowanych struktur danych (tablice, listy, stosy, kolejki, drzewa, haszowanie) oraz potrafi wskazać ich typowe użycia.	K_W01 K_W03
	W04: zna podstawowe algorytmy grafowe (BFS, DFS, algorytmy najkrótszej ścieżki, minimalne drzewa rozpinające) oraz rozumie ich zastosowania.	K_W01 K_W04
	W05: rozumie główne paradygmaty projektowania algorytmów (dziel i zwyciężaj, programowanie dynamiczne, algorytmy zachłanne, metoda z nawrotami).	K_W03 K_W08
	W06: zna podstawowe algorytmy tekstowe oraz wybrane techniki przetwarzania tekstu.	K_W03
	W07: zna podstawowe klasy problemów obliczeniowych (P, NP) oraz przykłady problemów NP-trudnych i NP-zupełnych.	K_W01 K_W09
	W08: zna metody dowodzenia poprawności algorytmów (np. indukcja matematyczna, niezmienniki pętli) oraz rozumie pojęcie amortyzowanej analizy złożoności.	K_W01 K_W08
Umiejętności	Efekt uczenia się	Odniesienie do efektów kierunkowych
	Po zakończeniu kursu student:	
	U01: potrafi analizować poprawność algorytmów oraz oszacować ich złożoność czasową i pamięciową.	K_U06 K_U13
	U02: umie implementować w wybranym języku programowania podstawowe algorytmy sortowania, wyszukiwania oraz operacji na kolekcjach danych.	K_U03 K_U04
	U03: potrafi dobrać odpowiednią strukturę danych do rozwiązania określonego problemu i zaimplementować ją praktycznie.	K_U04
	U04: umie stosować algorytmy grafowe w rozwiązywaniu problemów praktycznych i poprawnie je implementować.	K_U04 K_U07
	U05: potrafi rozwiązywać problemy algorytmiczne przy użyciu różnych technik projektowania (backtracking, algorytmy zachłanne, programowanie dynamiczne).	K_U04
	U06: umie zaimplementować i zastosować podstawowe algorytmy tekstowe.	K_U04
	U07: potrafi krytycznie ocenić trudność obliczeniową problemu oraz wskazać możliwości jego przybliżonego rozwiązania.	K_U06 K_U13
Kompetencje społeczne	U08: potrafi porównywać algorytmy w ujęciu teoretycznym i eksperymentalnym oraz interpretować wyniki pomiarów ich efektywności	K_U06
	Efekt uczenia się	Odniesienie do efektów kierunkowych
	Po zakończeniu kursu student:	
Kompetencje społeczne	K01: potrafi współpracować w zespole przy rozwiązywaniu problemów algorytmicznych, dzieląc się zadaniami i odpowiedzialnością.	K_K01
	K02: dostrzega znaczenie efektywności algorytmów i struktur danych w praktyce inżynierskiej oraz odpowiedzialność za jakość projektowanych rozwiązań.	K_K04

	K03: rozumie konieczność systematycznego poszerzania wiedzy w zakresie algorytmów i struktur danych ze względu na dynamiczny rozwój technologii informatycznych.	K_K02
--	---	-------

Studia stacjonarne

Organizacja											
Forma zajęć	Wykład (W)	Ćwiczenia w grupach									
		A		K		L		S		P	E
Liczba godzin	30					30					

Studia niestacjonarne

Organizacja											
Forma zajęć	Wykład (W)	Ćwiczenia w grupach									
		A		K		L		S		P	E
Liczba godzin	20					20					

Opis metod prowadzenia zajęć

Podczas wykładów omawiane są fundamentalne koncepcje algorytmiki oraz własności podstawowych i zaawansowanych struktur danych, ilustrowane pseudokodem i przykładami zastosowań. Zajęcia ćwiczeniowe mają charakter praktyczny - studenci rozwiązują zadania algorytmiczne, implementują wybrane algorytmy i struktury danych w wybranym języku programowania, a następnie prezentują i omawiają swoje rozwiązania.

Część zajęć może przybierać formę pracy z pseudokodem, analizy poprawności algorytmów, obliczania i porównywania ich złożoności obliczeniowej oraz dyskusji nad wyborem optymalnych rozwiązań. Szczególny nacisk kładziony jest na łączenie podejścia teoretycznego z praktyczną implementacją.

Formy sprawdzania efektów uczenia się

	E – learning	Gry dydaktyczne	Ćwiczenia w szkole	Zajęcia terenowe	Praca laboratoryjna	Projekt indywidualny	Projekt grupowy	Udział w dyskusji	Referat	Praca pisemna (esej)	Egzamin ustny	Egzamin pisemny	Zadania problemowe
W01					x			x				x	
W02					x			x				x	
W03					x			x				x	
W04					x			x				x	
W05					x			x				x	
W06					x			x				x	
W07					x			x				x	
W08					x			x				x	
U01					x			x				x	
U02					x			x				x	
U03					x			x				x	

U04					x			x				x	
U05					x			x				x	
U06					x			x				x	
U07					x			x				x	
U08					x			x				x	
K01								x					
K02								x					
K03								x					

Osiągnięcie efektów kształcenia podanych powyżej uprawnia studentów do uzyskania oceny nie wyższej niż dostateczna.

Zaliczenie ćwiczeń

Warunkiem zaliczenia ćwiczeń jest uzyskanie co najmniej 50% punktów z każdego z dwóch kolokwii sprawdzających znajomość zagadnień teoretycznych oraz umiejętność praktycznej implementacji algorytmów i struktur danych.

Zaliczenie ćwiczeń jest warunkiem koniecznym dopuszczenia do egzaminu końcowego. Ćwiczenia mają charakter zaliczeniowy i nie kończą się wystawieniem odrębnej oceny.

Prowadzący może uwzględnić aktywność studenta na zajęciach ćwiczeniowych, w szczególności systematyczne rozwiązywanie zadań, udział w dyskusji oraz zaangażowanie w pracę zespołową, przy podejmowaniu decyzji o zaliczeniu.

Egzamin końcowy

Egzamin obejmuje treści wykładów i ma formę testu teoretyczno-praktycznego. Sprawdza wiedzę dotyczącą analizy złożoności algorytmów, znajomości technik projektowania oraz rozumienia własności struktur danych i algorytmów grafowych.

Kryteria oceny

Ocena końcowa

Ocena końcowa pochodzi wyłącznie z egzaminu końcowego. Ocena z egzaminu końcowego ustalana jest na podstawie liczby uzyskanych punktów oraz poziomu poprawności i kompletności rozwiązań.

- Ocena dostateczna - student uzyskuje pozytywny wynik egzaminu końcowego i zalicza ćwiczenia, osiągając efekty kształcenia na poziomie podstawowym.
- Oceny dobre i bardzo dobre – student, oprócz spełnienia warunków zaliczenia na ocenę dostateczną, dodatkowo:
 - analizuje i porównuje algorytmy pod względem złożoności oraz praktycznej efektywności,
 - potrafi dobrać i zastosować zaawansowane struktury danych do rozwiązywania problemów,
 - świadomie stosuje różne paradygmaty projektowania algorytmów (dziel i zwyciężaj, zachłanne, dynamiczne, backtracking),
 - prezentuje wysoki poziom poprawności, przejrzystości i czytelności implementowanych rozwiązań.

Obecność na wykładach jest warunkiem koniecznym zaliczenia tej części kursu i dopuszczenia do egzaminu.

Uwagi

Treści merytoryczne (wykaz tematów)

1. Podstawy algorytmiki
 - podstawowe definicje i pojęcia, rekurencja i elementy matematyki dyskretnej, przykłady algorytmów rekurencyjnych, analiza złożoności obliczeniowej, notacja asymptotyczna, praktyczna i teoretyczna analiza algorytmów,
 - metody dowodzenia poprawności algorytmów (indukcja matematyczna, niezmienniki pętli)
2. Algorytmy wyszukiwania i sortowania
 - proste algorytmy sortowania (bąbelkowe, przez wybieranie, przez wstawianie),
 - metoda „dziel i zwyciężaj” (sortowanie przez scalanie, sortowanie szybkie),
 - sortowanie przez kopcowanie i kopce binarne,
 - sortowanie o złożoności liniowej (przez zliczanie, kubełkowe, pozycyjne),
 - algorytm wyboru k-tego elementu, wyszukiwanie binarne i interpolacyjne, statystyki pozycyjne.
3. Struktury danych
 - tablice i listy,
 - abstrakcyjne struktury danych i ich implementacje (stos, kolejka, tablica asocjacyjna),
 - tablice z haszowaniem,
 - drzewa binarne i drzewa wyszukiwania,
 - drzewa zrównoważone (AVL, czerwono-czarne, B-drzewa).
4. Algorytmy grafowe
 - reprezentacja grafów,
 - algorytmy przeszukiwania (BFS, DFS) i ich zastosowania: sortowanie topologiczne, silnie spójne składowe, mosty,
 - algorytmy najkrótszej ścieżki (Dijkstra, Bellman-Ford, Floyd-Warshall),
 - minimalne drzewa rozpinające (Kruskal, Prim).
5. Techniki projektowania algorytmów
 - metoda z nawrotami (backtracking), np. problem n-hetmanów,
 - algorytmy zachłanne: Huffman, problem plecakowy (wersja zachłanna), problem wyboru zajęć, pokrycie zbioru, wydawanie reszty,
 - programowanie dynamiczne: ciąg Fibonacciego, najdłuższy rosnący podciąg, plecak (wersja dynamiczna), zbiór wierzchołków niezależnych w drzewie, wydawanie reszty.
6. Algorytmy tekstowe
 - wyszukiwanie wzorca w tekście (naiwne, Rabin-Karp, Knuth-Morris-Pratt),
 - drzewa i tablice sufiksów, inne techniki przetwarzania tekstu.
7. Problemy trudne obliczeniowo
 - klasy złożoności P i NP,
 - problemy NP-zupełne i NP-trudne,
 - algorytmy aproksymacyjne.

Wykaz literatury podstawowej

Wskazane przez prowadzącego rozdziały:

1. T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, *Wprowadzenie do algorytmów*, Wydawnictwo Naukowe PWN, Warszawa 2023;
2. J. Kubica, *Struktury danych z przymrużeniem oka. Zabawna przygoda z przykładami pachnącymi kawą*, Helion, Gliwice 2024;
3. G. Heineman, *Nauka algorytmów. Poradnik pisania lepszego kodu*, Helion, Gliwice 2022.
4. P. Wróblewski, *Algorytmy, struktury danych i techniki programowania. Wydanie VI*, Helion, Gliwice 2019.
5. L. Banachowski, W. Rytter, K. M. Diks, *Algorytmy i struktury danych*, Wydawnictwo Naukowe PWN, Warszawa 2018;
6. S. Dasgupta, C. Papadimitriou, U. Vazirani, *Algorytmy*, Wydawnictwo Naukowe PWN, Warszawa 2010;
7. P. Kotowski, *Algorytmy + Struktury Danych = Abstrakcyjne Typy Danych*, Wydawnictwo BTC, Warszawa 2006.
8. D. Knuth, *Sztuka programowania, tom 1 i 3*, WNT, 2002

Wykaz literatury uzupełniającej

1. A. Bhargava, *Algorytmy. Ilustrowany przewodnik*, Helion, Gliwice 2017.
2. T. H. Cormen, *Algorytmy bez tajemnic*, Helion, Gliwice 2013.
3. K. Pieńkosz, J. Wojciechowski, *Grafy i sieci*, Wydawnictwo Naukowe PWN, Warszawa 2013;
4. Z.J. Czech, S. Deorowicz, P. Fabian, *Algorytmy i Struktury Danych. Wybrane zagadnienia*, Wydawnictwo Politechniki Śląskiej, Gliwice 2010;
5. A. Debudaj-Grabysz, S. Deorowicz, J. Widuch, *Algorytmy i Struktury Danych. Wybór zaawansowanych metod*, Wydawnictwo Politechniki Śląskiej, Gliwice 2012;
6. Drozdek A., *C++. Algorytmy i struktury danych*. Helion, Gliwice 2004.

Bilans godzinowy zgodny z CNPS (Całkowity Nakład Pracy Studenta) - studia stacjonarne

Liczba godzin w kontakcie z prowadzącymi	Wykład	30
	Konwersatorium (ćwiczenia, laboratorium itd.)	30
	Pozostałe godziny kontaktu studenta z prowadzącym	2
Liczba godzin pracy studenta bez kontaktu z prowadzącymi	Lektura w ramach przygotowania do zajęć	33
	Przygotowanie krótkiej pracy pisemnej lub referatu po zapoznaniu się z niezbędną literaturą przedmiotu	0
	Przygotowanie projektu lub prezentacji na podany temat (praca w grupie)	0
	Przygotowanie do egzaminu	30
Ogółem bilans czasu pracy		125
Liczba punktów ECTS w zależności od przyjętego przelicznika		5

Bilans godzinowy zgodny z CNPS (Całkowity Nakład Pracy Studenta) - **studia niestacjonarne**

Liczba godzin w kontakcie z prowadzącymi	Wykład	20
	Konwersatorium (ćwiczenia, laboratorium itd.)	20
	Pozostałe godziny kontaktu studenta z prowadzącym	1
Liczba godzin pracy studenta bez kontaktu z prowadzącymi	Lektura w ramach przygotowania do zajęć	39
	Przygotowanie krótkiej pracy pisemnej lub referatu po zapoznaniu się z niezbędną literaturą przedmiotu	0
	Przygotowanie projektu lub prezentacji na podany temat (praca w grupie)	0
	Przygotowanie do egzaminu	45
Ogółem bilans czasu pracy		125
Liczba punktów ECTS w zależności od przyjętego przelicznika		5