KARTA OPISU MODUŁU KSZTAŁCENIA

Nazwa modułu/przedmiotu	Kod
Algorytmika praktyczna	**1010514321010508880**

Kierunek studiów	Profil kształcenia (ogólnoakademicki, praktyczny)	Rok / Semestr
Informatyka	**ogólnoakademicki**	**1 / 2**
Ścieżka obieralności/specjalność	Przedmiot oferowany w języku:	Kurs (obligatoryjny/obieralny)
-	**polski**	**obieralny**

Stopień studiów:	**Forma studiów** (stacjonarna/niestacjonarna)
I stopień	**niestacjonarna**

Godziny	Liczba punktów
Wykłady: **20** Ćwiczenia: **-** Laboratoria: **20** Projekty/seminaria: **-**	**6**

Status przedmiotu w programie studiów (podstawowy, kierunkowy, inny)	(ogólnouczelniany, z innego kierunku)
kierunkowy	**z danego kierunku**

Obszar(y) kształcenia i dziedzina(y) nauki i sztuki	**Podział ECTS (liczba i %)**
nauki techniczne	**100 6%**
nauki techniczne	**100 6%**

Odpowiedzialny za przedmiot / wykładowca:

dr inż. Szymon Wąsik
email: szymon.wasik@cs.put.poznan.pl
tel. 665-3032
Wydział Informatyki
ul. Piotrowo 2, 60-965 Poznań

Odpowiedzialny za przedmiot / wykładowca:

dr inż. Maciej Antczak
email: maciej.antczak@cs.put.poznan.pl
tel. 665-3256
Wydział Informatyki
ul. Piotrowo 2, 60-965 Poznań

Wymagania wstępne w zakresie wiedzy, umiejętności, kompetencji społecznych:

1	**Wiedza:**	Student rozpoczynający ten przedmiot powinien posiadać podstawową wiedzę z zakresu implementacji algorytmów w języku C, C++ lub Pascal oraz analizy ich złożoności.
2	**Umiejętności:**	Powinien posiadać umiejętność rozwiązywania podstawowych problemów programistycznych wykorzystujących techniki sortowania danych, kopce, metody zachłanne oraz programowanie dynamiczne. Dodatkowo student powinien być zaznajomiony z koncepcją programowania zespołowego zgodnie z formułą ACM ICPC oraz posiadać umiejętność czytania ze zrozumieniem prostych tekstów w języku angielskim. Student powinien posiadać umiejętność pozyskiwania informacji ze wskazanych źródeł. Powinien również rozumieć konieczność poszerzania swoich kompetencji.
3	**Kompetencje społeczne**	W zakresie kompetencji społecznych student musi prezentować takie postawy jak uczciwość, odpowiedzialność, wytrwałość, ambicja i ciekawość poznawcza, kreatywność, kultura osobista, szacunek dla innych ludzi.

Cel przedmiotu:

1. Przekazanie podstawowej wiedzy o złożoności obliczeniowej w zakresie jej analizy, działania deterministycznej i niedeterministycznej maszyny Turinga, maszyny RAM, klasyfikacji problemów i algorytmów, klas złożoności P i NP, strukturach danych obejmujące sposób działania drzew, drzew BST, AVL, drzew przedziałowych, grafów, w tym grafów dwudzielnych oraz analizę ich złożoności oraz wiedzy z algorytmiki w zakresie dynamicznego zarządzania pamięcią, algorytmów przeszukiwania drzew i grafów, znajdowania najkrótszych ścieżek, maksymalnego przepływu w grafach, wyznaczania drzew rozpinających, geometrii obliczeniowej i algorytmów tekstowych oraz o zaawansowanych technikach programistycznych usprawniających proces implementacji algorytmów oraz optymalizujących ich wydajność.
2. Rozwijanie umiejętności dowodzenia NP-zupełności problemów, implementacji programistycznej poznanych algorytmów oraz struktur danych, doboru odpowiedniego algorytmu i struktury danych do rozwiązywanego problemu oraz ocenę złożoności obliczeniowej i pamięciowej ich implementacji, programowania zespołowego zgodnie z formułą ACM ICPC, wykorzystania zewnętrznych bibliotek z gotowymi implementacjami algorytmów i struktur danych.
3. Przygotowanie studentów do startów w konkursach w programowaniu zespołowym.

Efekty kształcenia i odniesienie do kierunkowych efektów kształcenia

Wiedza:

1. Ma uporządkowaną, podbudowaną teoretycznie wiedzę ogólną w zakresie algorytmów i złożoności. - [K1st_W4]
2. Zna podstawowe metody, techniki i narzędzia stosowane przy rozwiązywaniu zadań informatycznych z zakresu analizy złożoności obliczeniowej algorytmów i problemów. - [K1st_W7]

Umiejętności:

1. Potrafi planować i przeprowadzać eksperymenty, w tym pomiary czasu działania algorytmów, interpretować uzyskane wyniki i wyciągać wnioski o poprawności doboru i złożoności algorytmów. - [K1st_U3]

2. Potrafi wykorzystać do formułowania i rozwiązywania zadań informatycznych metody analityczne i eksperymentalne w celu dobrania odpowiednich algorytmów i struktur danych. - [K1st_U4]

3. Potrafi ocenić złożoność obliczeniową elementarnych algorytmów i problemów. - [K1st_U8]

4. Potrafi pozyskiwać informacje o algorytmach i strukturach danych oraz sposobach ich wykorzystania w konkretnych problemach z literatury oraz Internetu (w języku ojczystym i angielskim), integrować je, dokonywać ich interpretacji i krytycznej oceny, wyciągać wnioski oraz formułować i wyczerpująco uzasadniać opinie. - [K1st_U9]

5. Ma umiejętność formułowania algorytmów i ich programowania z użyciem języka C++ oraz wykorzystania zaawansowanych możliwości tego języka. - [K1st_U11]

6. Potrafi organizować, współdziałać i pracować w grupie w celu opracowania i implementacji efektywnych wersji algorytmów, przyjmując w niej różne role oraz potrafi odpowiednio określić priorytety służące realizacji określonego przez siebie lub innych zadania. - [K1st_U18]

Kompetencje społeczne:

1. Rozumie na podstawie historii standardów C++ ('11, '14, '17), że w informatyce wiedza i umiejętności bardzo szybko stają się przestarzałe. - [K1st_K1]

2. Ma świadomość znaczenia wiedzy w rozwiązywaniu problemów wymagających projektowania nowych algorytmów oraz zna przykłady i rozumie przyczyny wadliwie zaimplementowanych algorytmów, które doprowadziły do poważnych strat finansowych. - [K1st_K2]

Sposoby sprawdzenia efektów kształcenia

Ocena formująca:

a) w zakresie wykładów weryfikowanie założonych efektów kształcenia realizowane jest przez:

- monitorowanie i premiowanie pracy własnej studentów polegającej na samodzielnym rozwiązywaniu zadań algorytmicznych

- premiowanie aktywności studentów na wykładach

b) w zakresie laboratoriów weryfikowanie założonych efektów kształcenia realizowane jest przez:

- ocenę rozwiązań zadań demonstrujących sposób działania algorytmów prezentowanych przez studentów na tablicy

- kontrolę postępów studenta w rozwiązywaniu problemów algorytmicznych na zajęciach laboratoryjnych oraz poza nimi, które oceniane są przez automatyczny system sprawdzający

Ocena podsumowująca:

a) w zakresie wykładów weryfikowanie założonych efektów kształcenia realizowane jest przez:

- dwa kolokwia wymagające rozwiązania 4-8 praktycznych problemów z zakresu algorytmiki,

- zadania punktowane są w skali 0-5 punktów (maksymalna liczba punktów zależna od stopnia trudności), ze skokiem co 0,25 punktu,

- punkty za kolokwia przeliczane są na procenty poprzez podzielenie przez maksymalną liczbę punktów do zdobycia i każde z nich wliczane z wagą 0.25 do końcowej oceny przedmiotu

b) w zakresie laboratoriów weryfikowanie założonych efektów kształcenia realizowane jest przez:

- analizę wyników zadań rozwiązywanych w czasie semestru - ok. 30 zadań obowiązkowych, z czego na 3.0 trzeba rozwiązać 50% zadań, na 3.5 60% zadań, itd. - ocena wlicza się do końcowej oceny przedmiotu z wagą 0.5

Dodatkowe elementy oceny:

- za rozwiązanie dodatkowych zadań wybranych przez prowadzącego student może dostać do 1,5% dodatkowych punktów za zadanie

Treści programowe

Treści programowe zawarte w ramach tego przedmiotu oprócz przedstawienia treści algorytmicznych wymaganych od absolwenta studiów I stopnia zawierają obszerny zakres tematów, który ma przygotować studentów do startów w algorytmicznych konkursach w programowaniu zespołowym zgodnych z formułą ACM ICPC. Zakłada się, że przedmiot ten, jako przedmiot obieralny, wybrany zostanie głównie przez studentów zainteresowanych tego typu zawodami, posiadających już pewne podstawowe umiejętności i wiedzę algorytmiczną. Dlatego wiele aspektów omawiane jest na poziomie średniozaawansowanym i nastawione na ich praktyczne wykorzystanie przy rozwiązywaniu problemów na zawodach algorytmicznych.

Wykład podzielony jest na dwie główne części. Pierwsza część jest bardziej teoretyczna i dotyczy złożoności obliczeniowej. Przedstawiony zostaje podział problemów na decyzyjne i optymalizacyjne, wraz z charakterystyką tych klas i przykładami należących do nich problemów. Przed przystąpieniem do omawiania implementacji algorytmów we współczesnych językach programowania omawiana jest deterministyczna i niedeterministyczna maszyna Turinga oraz maszyna RAM, jako przykłady abstrakcyjnego modelu komputera służącego do wykonywania algorytmów. W oparciu o ten materiał wyjaśniona zostaje idea i definicja klas problemów decyzyjnych P oraz NP, wraz z podklasami problemów NP-zupełnych i silnie NP-zupełnych oraz przedstawione sposoby dowodzenia przynależności problemów do tych klas. Druga część związana jest z prezentacją konkretnych algorytmów oraz ich zastosowań do rozwiązywania praktycznych problemów algorytmicznych (stąd nazwa przedmiotu). Część ta zaczyna się od omówienia dynamicznych struktur danych wraz ze szczegółowym omówieniem działania i wykorzystania wskaźników i dynamicznego przydziału pamięci w języku C++. Poruszone pozostają problemy wycieków pamięci, błędnych wskaźników, niepoprawnej arytmetyki wskaźnikowej (przepełnienie bufora) oraz wskaźników zagnieżdżonych (wskaźnik do wskaźnika, itd.).

Wykorzystując dynamiczną alokację pamięci przedstawiona zostaje implementacja list jedno i dwu kierunkowych oraz drzew. Poruszony zostaje problem równoważenia drzewa i przedstawione drzewa AVL, jako przykład drzew zrównoważonych. Ostatnim tematem dotyczącym drzew są drzewa przedziałowe, jako przykład efektywnej struktury danych do przechowywania informacji o danych reprezentowanych przez przedziały liczb na osi liczb całkowitych. Dla każdej struktury danych przedstawiony zostaje algorytm dodawania i usuwania elementów oraz możliwe sposoby ich wyszukiwania. Kolejne są grafy i wykorzystujące je algorytmy. Omówienie ich rozpoczyna się od przedstawienia możliwych reprezentacji grafu (macierz incydencji, lista sąsiedztwa, lista incydencji) oraz ich doboru do rozwiązywanego problemu.

Dalej omawiane są algorytmy wyszukiwania BFS i DFS oraz ich zastosowanie do sortowania topologicznego, wyznacza silnie spójnych i dwuspójnych składowych, mostów i punktów artykulacji. Kolejnymi są algorytmy znajdowania najkrótszych ścieżek za pomocą algorytmów Floyda-Warshala, Bellmana-Forda, Dijsktry i Johnsona, minimalnych drzew rozpinających za pomocą algorytmów Prima i Kruskala oraz maksymalnego przepływu za pomocą algorytmu Forda-Fulkersona i Dinica. Algorytmy przepływowe wykorzystywane są do prezentacji rozwiązania problemu maksymalnego skojarzenia w grafie dwudzielnym. Następnie omawiane jest zastosowanie algorytmów programowania dynamicznego do problemów, których reprezentacja komputerowa wykorzystuje drzewa lub grafy. Kolejna jest geometria obliczeniowa i problemy takie jak rachunek wektorowy, przecinanie się odcinków, znajdowania pary najbliższych i najdalszych punktów i otoczka wypukła. Omówienie algorytmów tekstowych obejmuje algorytmy wyszukiwania wzorca w jednym i dwóch wymiarach, znajdowania cyklów w słowach oraz znajdowania maksymalnych palindromów. Poruszony zostaje temat efektywnego wykorzystania istniejących bibliotek z implementacjami algorytmów i struktur danych na podstawie biblioteki STL oraz zaawansowane możliwości języka C++ na podstawie operacji bitowych. Ostatnim tematem jest konstruowanie algorytmów z powracaniem oraz ich optymalizacja za pomocą metody podziału i ograniczeń.

Zajęcia laboratoryjne kładą nacisk na zastosowanie w praktyce algorytmów i struktur danych prezentowanych na wykładzie poprzez rozwiązywanie i implementacje problemów algorytmicznych na komputerze w języku C++. W celu dobrego przygotowania do startu w zawodach programistycznych oraz nauczenia rozwiązywania konkretnych problemów praktycznych, wykorzystywane są zadania i problemy ilustrujące przedstawiane treści, w dużym stopniu pochodzące z historycznych konkursów oraz zbiorów zadań. Do każdego tematu na wykładzie przyporządkowywane jest około 10 zadań, o trzech różnych i podanych do wiadomości stopniach trudności, obrazujących przedstawiane treści. Zadaniem studentów jest rozwiązanie jak największej liczby problemów na zajęciach laboratoryjnych, gdy możliwa jest konsultacja i pomoc prowadzącego, a następnie ich dokończenie pracy w domu. Początek kolejnych zajęć służy rozwianiu wątpliwości i wyjaśnieniu rozwiązań tych problemów. Do oceniania zadań używany jest automatyczny system sprawdzający, który w przeciągu minuty jest w stanie ocenić poprawność rozwiązania. Dzięki temu student może natychmiast, bez straty czasu rozpocząć proces poszukiwania błędu i poprawiania błędnego rozwiązania. Nauka rozwiązywania problemów algorytmicznych na komputerze dopełniana jest poprzez rozwiązywanie zadań prezentujących sposób działania algorytmów na tablicy. Część zajęć laboratoryjnych przeznaczona jest na zespołowe rozwiązywanie przez studentów problemów zgodnie z formułą ACM ICPC w celu zdobycia doświadczenia w pracy zespołowej nad zestawem problemów. Odbywa się to w sytuacji wzajemnej rywalizacji kilku grup studentów w taki sposób, jak w rzeczywistych zawodach.

Część wyżej wymienionych treści programowych jest realizowana w ramach pracy własnej studenta.

Literatura podstawowa:

1. Wprowadzenie do algorytmów, T.H. Cormen, Ch.E.Leiserson, R.L. Rivest, C. Stein, PWN, W-wa, 2012
2. Algorytmika praktyczna nie tylko dla mistrzów, P. Stańczyk, PWN, 2009
3. Kombinatoryka dla programistów, W. Lipski, WNT, 2007
4. Tablice matematyczne, W. Mizerski, Adamantan, 2008
5. Złożoność obliczeniowa problemów kombinatorycznych, J. Błażewicz, WNT, W-wa, 1988

Literatura uzupełniająca:

1. The CRC Concise Encyclopedia of Mathematics, E. W. Weisstein, CRC Press, 1998
2. C++ Biblioteka standardowa, Podręcznik programisty, N. M. Josuttis, Helion, 2003

Bilans nakładu pracy przeciętnego studenta

Czynność	Czas (godz.)
1. Udział w zajęciach laboratoryjnych	20
2. Przygotowanie do ćwiczeń laboratoryjnych	25
3. Udział w konsultacjach związanych z realizacją procesu kształcenia: ćwiczeń laboratoryjnych i zadań algorytmicznych.	2
	45
4. Rozwiązywanie zadań algorytmicznych oraz ich implementacja poza zajęciami laboratoryjnymi	20
5. Przygotowanie do kolokwiów	20
6. Udział w wykładach	15
7. Przygotowanie do egzaminu i obecność na egzaminie: 13 godz. + 2 godz.	

Obciążenie pracą studenta

forma aktywności	godzin	ECTS
Łączny nakład pracy	145	6
Zajęcia wymagające bezpośredniego kontaktu z nauczycielem	44	2
Zajęcia o charakterze praktycznym	90	4