Program Studiów Informatycznych na Uniwersytecie Wrocławskim

Studia stacjonarne pierwszego stopnia

zatwierdzony 25 września 2007 roku przez
Radę Wydziału Matematyki i Informatyki Uniwersytetu Wrocławskiego

Dotyczy studentów wpisanych na pierwszy semestr po 30 września 2007

Spis treści

Spis tabel

1 Studia stacjonarne pierwszego stopnia

Program informatycznych studiów stacjonarnych pierwszego stopnia na Uniwersytecie Wrocławskim nazywanych dalej *studiami* jest oparty na założeniach programu studiów informatycznych magisterskich i licencjackich na Uniwersytecie Wrocławskim obowiązującego od 1997 roku. Uwzględnia przy tym zmiany wprowadzone przez ustawę *Prawo o szkolnictwie wyższym* z dnia 27 lipca 2005 roku oraz *Standardy kształcenia* na studiach pierwszego stopnia dla kierunku informatyka nazywane dalej *standardami*. Studia są oparte na systemie punktów kredytowych ECTS.

Absolwenci studiów mogą zdobywać dalsze wykształcenie w kierunku informatycznym na studiach drugiego stopnia. Program studiów informatycznych drugiego stopnia na Uniwersytecie Wrocławskim jest skorelowany z programem studiów opisywanych niniejszym dokumentem. W szczególności wiele przedmiotów stanowi rozwinięcie przedmiotów przeznaczonych dla studiów pierwszego stopnia. Także zaliczenie odpowiednich przedmiotów obowiązkowych ze studiów drugiego stopnia w trakcie studiów pierwszego stopnia zwalnia z egzaminu wstępnego na studia drugiego stopnia. Dlatego w przedstawionym tu programie zamieszczone zostały odniesienia do programu studiów drugiego stopnia na Uniwersytecie Wrocławskim. Odniesienia te są wyróżnione kursywą.

1.1 Klasyfikacja przedmiotów

W programie studiów występują następujące grupy przedmiotów:

Przedmioty obowiązkowe (O) obejmują matematyczne podstawy informatyki oraz kanon wiedzy informatycznej niezbędnej do zrozumienia szerokiego spektrum badań i zastosowań informatycznych. *Grupa ta obejmuje*

łącznie przedmioty wymagane w toku studiów pierwszego i drugiego stopnia. W grupie przedmiotów obowiązkowych wyróżniamy trzy podgrupy:

O.1 Przedmioty obejmujące matematyczne podstawy informatyki, prowadzone na jednym poziomie trudności. Ich zaliczenie jest wymagane do ukończenia studiów. Do grupy tej należą: *Analiza matematyczna (1. semestr)*, *Algebra (2. semestr)*, *Logika dla informatyków (1. semestr)*, *Elementy rachunku prawdopodobieństwa (3. semestr).*

O.2 Przedmioty prowadzone na poziomie zasadniczym (L) i rozszerzonym (M). Do ukończenia studiów wymagane jest zaliczenie przedmiotu na poziomie zasadniczym. Do grupy tej należą: *Matematyka dyskretna (3. semestr)*, *Programowanie (2. semestr)*, *Analiza numeryczna (3. semestr)*, *Algorytmy i struktury danych (4. semestr).*

O.3 *Przedmioty prowadzone tylko na poziomie rozszerzonym. Zaliczenie przedmiotów z tej grupy nie jest wymagane do zaliczenia studiów (są one obowiązkowe dla studiów drugiego stopnia). W grupie tej aktualnie występuje jeden przedmiot:* Języki formalne i złożoność obliczeniowa.

Przedmioty informatyczne (I) obejmują treści informatyczne prezentowane w formie uogólnionej i abstrakcyjnej znajdujące zastosowania w różnych narzędziach i rozwiązaniach informatycznych. W grupie tej wyróżniamy dwie podgrupy:

I.1 Przedmioty obejmujące treści, których znajomość jest konieczna do ukończenia studiów. Do grupy tej należą: *Wstęp do informatyki*, *Architektura systemów komputerowych*, *Bazy danych*, *Systemy operacyjne*, *Sieci komputerowe*, *Inżynieria oprogramowania.*

I.2 Pozostałe przedmioty informatyczne.

Przedmioty z grupy I.1, jako obejmujące podstawowe treści kierunkowe ze standardów, nie mogą być zaliczane przez studentów studiów drugiego stopnia.

Kursy narzędzi informatycznych (K) to przedmioty, których celem jest praktyczna nauka określonego narzędzia informatycznego.

Seminaria (S) to przedmioty prowadzone w formie konwersatorium wymagające od studenta wykazania się umiejętnością samodzielnego opracowania i prezentacji zagadnienia związanego z tematyką seminarium.

Projekty programistyczne (P) polegają na przygotowaniu przez studenta pod opieką prowadzącego zaawansowanego, interdyscyplinarnego, kompletnego projektu programistycznego; z projektami nie muszą być związane planowe zajęcia; projekty mogą być przygotowywane w ramach pracy własnej, w ramach praktyk zawodowych, mogą być indywidualne lub zespołowe, mogą być także kontynuacją i rozwinięciem projektów rozpoczętych w ramach przedmiotów informatycznych lub kursów narzędzi informatycznych.

Przedmioty nieinformatyczne (N) obejmują treści z dziedzin innych niż informatyka.

Lektoraty języków obcych (L)

Wychowanie fizyczne (WF)

1.2 Przedmioty i treści obowiązkowe

Przedmioty obowiązkowe W programie studiów status obowiązkowych mają przedmioty obejmujące podstawy matematyczne informatyki oraz kanon wiedzy informatycznej niezbędnej do zrozumienia szerokiego spektrum badań i zastosowań informatycznych. Przedmioty te są przypisane w planie studiów do określonych semestrów i wymagane do zaliczenia tychże semestrów. Część tych przedmiotów prowadzona jest na dwóch różnych poziomach trudności: *zasadniczym (L) i rozszerzonym (M)*. Zaliczenie poziomu zasadniczego gwarantuje uzyskanie wiedzy wymaganej na poziomie studiów pierwszego stopnia, a zaliczenie poziomu rozszerzonego daje głębszą wiedzę, przydatną w szczególności na studiach informatycznych drugiego stopnia. Przedmioty występujące na dwóch różnych stopniach trudności są prowadzone równolegle (w tym samym semestrze), mogą mieć wspólne zajęcia wykładowe i pomocnicze, różnią się zakresem wiedzy wymaganej na egzaminie i liczbą punktów ECTS przyznawanych za zaliczenie przedmiotu.

Za zaliczenie przedmiotu na poziomie rozszerzonym przewidziane jest przyznanie punktów w procedurze rekrutacyjnej na studia drugiego stopnia. Zaliczenie takie zwalnia również z konieczności zaliczania tego przedmiotu na studiach drugiego stopnia, jeśli jest on w ich programie uznany za obowiązkowy bądź obejmujący treści obowiązkowe.

Treści obowiązkowe Zakres wiedzy informatycznej wyznaczony przez standardy a nieobjęty programem zasadniczych wersji przedmiotów obowiązkowych, występuje w programie studiów jako treści obowiązkowe. Są one wy-

kładane w ramach przedmiotów informatycznych z grupy I.1, które nie mają statusu obowiązkowych, jednak są gwarantowane przez Instytut. Studenci na zakończenie studiów muszą wykazać się znajomością treści obowiązkowych:

- zaliczając odpowiedni przedmiot gwarantowany,
- zaliczając inne przedmioty obejmujące odpowiednie treści lub
- wykazując się znajomością treści na egzaminie licencjackim.

1.3 Oferta dydaktyczna, plan studiów i indywidualny plan studiów studenta

Oferta dydaktyczna dla studiów, to ogłaszana przez dyrekcję corocznie przed rozpoczęciem roku akademickiego lista przedmiotów zawierająca opis techniczny (nazwę, wymiar godzin, sposób zaliczenia, typ przedmiotu) oraz merytoryczny (umiejętności wstępne, cele i umiejętności, program, źródła wiedzy) przedmiotu. W skład oferty wchodzą przedmioty prowadzone przez pracowników Instytutu lub zaproszonych specjalistów i oferowane studentom informatyki. W ofercie, obok przedmiotów obowiązkowych i gwarantowanych (O.1-O.3 i I.1), występuje wiele przedmiotów informatycznych, kursów narzędzi informatycznych, seminariów oraz kilka przedmiotów nieinformatycznych. Przedmioty te mogą być stałe bądź okazjonalne: w przypadku przedmiotów stałych dyrekcja gwarantuje ponowne umieszczenie przedmiotu w ofercie w ciągu następnych dwóch lat, podczas gdy przedmioty okazjonalne mogą być usuwane z oferty bez uprzedzenia.

Plan studiów na rok akademicki jest tworzony z oferty, na podstawie opinii studentów, wymogów programu oraz kierunków badań naukowych pracowników. W planie występują wszystkie przedmioty obowiązkowe z grup O.1-O.2 przynajmniej w wersji zasadniczej oraz przedmioty z grupy I.1. Poza wymienionymi, w planie występuje wiele przedmiotów informatycznych, kursów narzędzi informatycznych, seminariów oraz kilka przedmiotów nieinformatycznych. Każdy przedmiot może być prowadzony raz na dwa lata, w dowolnym semestrze. Lektoraty języków obcych oraz zajęcia wychowania fizycznego są prowadzone niezależnie od Instytutu przez wydzielone jednostki Uniwersytetu.

Indywidualny plan studiów studenta powstaje poprzez wybór przez studenta przedmiotów z planu studiów. Wybór ten dokonywany jest na początku każdego semestru i jest zobowiązaniem studenta do zaliczenia wybranych przedmiotów. Student ponosi odpowiedzialność za skonstruowanie indywidualnego planu studiów w ten sposób, by umożliwił mu zaliczenie odpowied-

niego semestru studiów oraz ukończenie studiów, czyli spełnienie wymagań opisanych w rozdziale 2.

1.4 Wymiar godzinowy i punktowy przedmiotów

Wszystkie przedmioty są semestralne. Zajęcia do każdego przedmiotu mogą odbywać się w formie: *wykładu*, *ćwiczeń*, *pracowni*, *repetytorium*, *seminarium*. Przedmiot może kończyć się egzaminem bądź zaliczeniem. Tabela 1 przedstawia minimalną liczbę godzin zajęć i punkty ECTS dla przedmiotów obowiązkowych. Każdy z przedmiotów może być prowadzony w nieco innym, większym wymiarze godzin niż przedstawiony w tabeli, przy czym zwiększony wymiar obejmuje jedynie zajęcia nieobowiązkowe. Celem tych zajęć może być ułatwienie studentom opanowania materiału (*repetytoria*). Wszystkie przedmioty obowiązkowe kończą się egzaminem.

Semestr	Przedmiot	wykład	ćw.	prac.	ECTS
1	Analiza matematyczna	60	45		10
1	Logika dla informatyków	30	30		7
2	Algebra	45	30		7
2	Programowanie (L)	45	30	15	9
2	Programowanie (M)	60	30	15	12
3	Matematyka dyskretna (L)	30	30		6
3	Matematyka dyskretna (M)	45	45		9
3	Analiza numeryczna (L)	45	30		8
3	Analiza numeryczna (M)	60	30	15	12
3	Elementy rachunku prawdopodobieństwa	15	15		3
4	Algorytmy i struktury danych (L)	45	30	15	9
4	Algorytmy i struktury danych (M)	60	30	30	13
-	*Języki formalne i złożoność obliczeniowa*	60	30		9

Tabela 1: Przedmioty obowiązkowe

Liczby godzin zajęć oraz punkty ECTS dla przedmiotów z pozostałych grup podane są w Tabeli 2. W wyjątkowych sytuacjach oferta może zawierać inne przedmioty, dla których wymiar godzinowy i punktowy ustalany jest indywidualnie przez dyrekcję w porozumieniu z dziekanem.

Ponadto punkty przyznaje się za:

Typ przedmiotu	wykł.	ćw./prac./sem.	egz.	ECTS
Informatyczny	30	30	tak	6
Kurs (1)	30	30	nie	5
Kurs (2)	15	45	nie	5
Seminarium	0	30	nie	3
Projekt	0	0–60	nie	4
Lektorat(1)	0	60	nie	2
Lektorat(2)	0	60	tak	3
Nieinformatyczny (1)	30	0	nie	2
Nieinformatyczny (2)	30	0–30	tak	4
WF	0	30	nie	1

Tabela 2: Pozostałe przedmioty

- *praktykę zawodową* — za zaliczenie tygodnia praktyki student otrzymuje 1 punkt.
- *egzamin licencjacki* — za przygotowanie się do egzaminu i jego zdanie przyznaje się 10 punktów.

2 Tok studiów

Studia trwają sześć semestrów i kończą się egzaminem licencjackim. Absolwenci studiów uzyskują dyplom licencjacki z informatyki.

2.1 Przyjęcie na studia

Szczegółowe zasady przyjęć na studia ogłaszane są corocznie przez Senat Uniwersytetu Wrocławskiego.

2.2 Zaliczenie semestru

Na zaliczenie kolejnych semestrów studiów studenci są zobowiązani do zaliczenia przedmiotów obowiązkowych przypisanych do danego semestru oraz do uzyskania odpowiedniej liczby punktów ECTS. Liczba ta wynosi $30k$, gdzie k jest numerem semestru dla $k = 1, 2, 3, 4, 5$, oraz 170 dla ostatniego, szóstego semestru.

Punkty mogą być uzyskiwane za:

- przedmioty obowiązkowe, informatyczne, nieinformatyczne, kursy i seminaria z planu studiów,
- projekty programistyczne i praktyki,
- egzamin licencjacki,
- lektoraty języków obcych i zajęcia wychowania fizycznego.

Dodatkowo, do zaliczenia szóstego semestru jest konieczne:

- zaliczenie języka angielskiego na poziomie B2 oraz innego języka obcego na poziomie co najmniej A1 II lub zaliczenie języka angielskiego na poziomie co najmniej A2 II oraz innego języka obcego na poziomie B2,
- zaliczenie jednego seminarium,
- zaliczenie jednego projektu programistycznego,
- uzyskanie co najmniej czterech punktów za przedmioty nieinformatyczne,
- uzyskanie co najmniej trzech punktów za praktyki zawodowe,
- uzyskanie co najmniej 54 punktów za przedmioty informatyczne (I),
- uzyskanie łącznie co najmniej 140 punktów za przedmioty obowiązkowe, informatyczne, kursy narzędzi informatycznych i projekty programistyczne (O+I+K+P)

2.3 Ukończenie studiów

Warunkiem ukończenia studiów jest zaliczenie szóstego semestru studiów, uzyskanie co najmniej 180 punktów, znajomość treści obowiązkowych oraz zdanie egzaminu licencjackiego.

Znajomość treści obowiązkowych można wykazać:

- zaliczając gwarantowany przedmiot informatyczny (z grupy I.1) obejmujący odpowiednie treści,
- zaliczając wskazany przedmiot (przedmioty) zawierający równoważne treści bądź wymagający znajomości odpowiednich treści,
- wykazując znajomość odpowiednich treści w projekcie programistycznym — lub

- wykazując znajomość odpowiednich treści na egzaminie licencjackim.

Egzamin licencjacki ma formę pisemną. Na egzaminie obowiązuje zakres wiedzy określony w standardach.

3 Programy i treści obowiązkowe

3.1 Logika dla informatyków

Wymagane przygotowanie studentów

Matematyka w zakresie szkoły średniej.

Program wykładu

1. Podstawowe pojęcia teoriomnogościowe i operacje na zbiorach: suma, iloczyn, iloczyn kartezjański, zbiór potęgowy, relacje, funkcje, relacje równoważności, klasy abstrakcji, zbiór ilorazowy.

2. Moce zbiorów. Zbiory skończone i nieskończone. Zbiory przeliczalne i zbiory mocy continuum. Twierdzenia Cantora i Cantora-Bernsteina.

3. Częściowe porządki, elementy minimalne i najmniejsze, kresy. Porządki liniowe. Twierdzenia o punkcie stałym. Dobre porządki. Indukcja noetherowska.

4. Składnia i semantyka rachunku zdań i rachunku predykatów. Pojęcie spełniania i prawdziwości formuł. Niesprzeczność zbioru formuł.

5. Unifikacja termów. Informacja o metodzie rezolucji.

6. Dowodzenie twierdzeń. Informacja o systemie naturalnej dedukcji.

Literatura

[1] Wojciech Guzicki, Piotr Zakrzewski, *Wykłady ze wstępu do matematyki. Wprowadzenie do teorii mnogości*, PWN, Warszawa 2005.

[2] Wojciech Guzicki, Piotr Zakrzewski, *Wstęp do matematyki. Zbiór zadań*, PWN, Warszawa 2005.

[3] Kazimierz Kuratowski, *Wstęp do teorii mnogości i topologii*, PWN, Warszawa 2004.

[4] Wiktor Marek, Janusz Onyszkiewicz, *Elementy logiki i teorii mnogości w zadaniach*, PWN, Warszawa 2005.

[5] Helena Rasiowa, *Wstęp do matematyki współczesnej*, PWN, Warszawa 2007.

[6] Jerzy Tiuryn, *Wstęp do teorii mnogości i logiki*, Skrypt Uniw. Warszawskiego, 1994.

Opracowali Witold Charatonik i Jerzy Marcinkowski

3.2 Analiza matematyczna

Wymagane przygotowanie studentów

Matematyka w zakresie szkoły średniej.

Program wykładu

1. Liczby rzeczywiste i zespolone (4 godz.)
   (a) kresy (1 godz.)
   (b) aksjomat ciągłości (1 godz.)
   (c) liczby zespolone jako punkty płaszczyzny (1 godz.)
   (d) postać biegunowa (1 godz.)
2. Ciągi i szeregi liczbowe rzeczywiste i zespolone (10 godz.)
   (a) ciągi zbieżne (1 godz.)
   (b) warunek Cauchy'ego zbieżności (1 godz.)
   (c) ciągi rekurencyjne (przykłady) (2 godz.)
   (d) twierdzenie Bolzano–Weierstrassa (2 godz.)
   (e) kryteria zbieżności szeregów (2 godz.)
   (f) szeregi potęgowe (2 godz.)
3. Funkcje jednej zmiennej (6 godz.)
   (a) granica funkcji w puncie, granice jednostronne (1 godz.)
   (b) ciągłość funkcji. Definicja Cauchy'ego i Heinego (1 godz.)
   (c) własności funkcji ciągłej na odcinku domkniętym (2 godz.)
   (d) własność Darboux (2 godz.)
4. Pochodna funkcji (10 godz.)
   (a) interpretacja geometryczna pochodnej (1 godz.)

(b) pochodna funkcji złożonej i odwrotnej (2 godz.)

(c) twierdzenie o wartości średniej (1 godz.)

(d) pochodne wyższych rzędów (2 godz.)

(e) wzór Taylora (2 godz.)

(f) ekstrema i badanie przebiegu funkcji (2 godz.)

5. Całkowanie (6 godz.)

   (a) funkcja pierwotna (2 godz.)

   (b) całka oznaczona. Interpretacja geometryczna funkcji pierwotnej (2 godz.)

   (c) całka Riemanna (2 godz.)

6. Ciągi i szeregi funkcyjne (10 godz.)

   (a) zbieżność jednostajna (norma jednostajna) (2 godz.)

   (b) szeregi potęgowe (3 godz.)

   (c) szereg Taylora (3 godz.)

   (d) funkcje analityczne (wielomiany, funkcja wykładnicza itp.) (2 godz.)

7. Funkcje wielu zmiennych (14 godz.)

   (a) pochodne cząstkowe, pochodne kierunkowe (2 godz.)

   (b) wzór Taylora (2 godz.)

   (c) ekstrema funkcji wielu zmiennych (3 godz.)

   (d) pochodne cząstkowe funkcji złożonej (1 godz.)

   (e) całki wielokrotne (4 godz.)

   (f) twierdzenie o zamianie zmiennych dla całek. Jakobian (2 godz.)

Literatura

[1] Grigorij. M. Fichtenholz, *Rachunek różniczkowy i całkowy*, tom 1–3, PWN, Warszawa 2003–2005.

[2] Włodzimierz Krysicki, Lech Włodarski, *Analiza matematyczna w zadaniach*, tom 1–2, PWN, Warszawa 2005.

[3] Kazimierz Kuratowski, *Rachunek różniczkowy i całkowy. Funkcje jednej zmiennej*, PWN, Warszawa 1979.

[4] Helena i Julian Musielakowie, *Analiza matematyczna*, tom 1, cz. 1–2, Wydawnictwo Uniwersytetu Adama Mickiewicza, Poznań 1993.

[5] Walter Rudin, *Podstawy analizy matematycznej*, PWN, Warszawa 1982.

[6] Walter Rudin, *Analiza rzeczywista i zespolona*, PWN, Warszawa 1999.

Opracował Adam Szustalewicz przy współpracy Tadeusza Pytlika

3.3 Algebra

Program wykładu

1. **Grupy i grupy permutacji.** Podstawowe pojęcia: rząd grupy, rząd elementu grupy, podgrupa. Grupy permutacji. Rozkład permutacji na cykle. Znak permutacji.

2. **Homomorfizmy grup.** Kongruencje. Dzielniki normalne. Grupa ilorazowa. Uogólnienie na przypadek innych algebr. Wzmianka o algebrach początkowych.

3. **Zagadnienia kombinatoryczne.** Twierdzenie Lagrange'a. Działanie grupy na zbiorze. Orbity i stabilizatory. Lemat Burnside'a.

4. **Arytmetyka modularna.** Relacja podzielności. Pierścienie i pierścienie Z_n. Algorytm Euklidesa. Chińskie twierdzenie o resztach. Własności grup cyklicznych.

5. **Wielomiany.** Pierścienie wielomianów. Podzielność wielomianów. Przykład konstrukcji ciała skończonego. Cykliczność grupy multiplikatywnej ciała skończonego.

6. **Przestrzenie liniowe i moduły.** Zbiory liniowo niezależne. Bazy. Macierze i przekształcenia liniowe. Rząd macierzy. Algorytm eliminacji Gaussa.

7. **Wyznaczniki.** Własności wyznaczników. Rozwinięcie Laplace'a.

8. **Równania liniowe.** Zbiór rozwiązań układu równań liniowych. Dopełnienie ortogonalne podprzestrzeni. Wzory Cramera.

9. **Elementy geometrii.** Iloczyn skalarny. Odległość punktów. Równania prostych i płaszczyzn. Izometrie i przekształcenia ortogonalne. Wielomian charakterystyczny. Obroty. Wzmianka o kwaternionach.

10. **Nierówności liniowe.** Lemat Farkasa. Zbiór rozwiązań układu nierówności liniowych a uwypukleniem zbioru rozwiązań bazowych.

11. **Formy dwuliniowe i kwadratowe.** Równoważne formy kwadratowe (w pełnej grupie przekształceń i grupie ortogonalnej). Metoda Lagrange'a sprowadzania formy kwadratowej do postaci kanonicznej. Sprowadzanie formy kwadratowej do postaci kanonicznej w grupie ortogonalnej.

Literatura

[1] G. Birkhoff, S. Mac Lane, *Przegląd algebry współczesnej*, PWN, 1966.

[2] L. Gårding, T. Tambour, *Algebra for Computer Science*, Springer-Verlag, 1988.

[3] B. Gleigchgewicht, *Algebra*, Oficyna Wydawnicza GiS, 2002.

[4] J. Rutkowski, *Algebra abstrakcyjna w zadaniach*, PWN, 2000.

Literatura uzupełniająca

[5] A. Białynicki-Birula, *Algebra*, Biblioteka Matematyczna 40, PWN, 1980.

[6] Aleksiej I. Kostyrkin, *Wstęp do algebry*, Cz. 1-3, Wydawnictwo Naukowe PWN, 2004.

[7] S. Lang, *Algebra*, PWN, 1984.

[8] A. Schrijver, Theory of linear and integer programming, Wiley, 2001.

Opracowali Witold Karczewski i Antoni Kościelski

3.4 Programowanie (L)

Przedmiot *Programowanie (L)* jest okrojoną wersją przedmiotu *Programowanie (M)*. Programy obu przedmiotów, z wyjątkiem zagadnień oznaczonych przez „M:", są takie same. Zagadnienia oznaczone ciągiem „M:" są wymagane na *Programowaniu (M)* i nie są wymagane na *Programowaniu (L)*.

3.5 Programowanie (M)

Wymagane przygotowanie studentów

Zakłada się, że słuchacze wykładu posiedli umiejętność programowania na co najmniej elementarnym poziomie. Ponadto zakłada się wiedzę z zakresu wykładu *Logika dla informatyków* i *Wstęp do informatyki*.

Cel zajęć i wskazówki metodyczne

Celem zajęć jest przedstawienie studentom możliwie szerokiego kręgu zagadnień związanych z programowaniem komputerów, ze szczególnym uwzględnieniem podstawowych konstrukcji występujących w językach programowania i związanych z nimi technik tworzenia programów.

Różne koncepcje i konstrukcje językowe przedstawiane są na przykładzie używanych w praktyce języków programowania. Dzięki temu konstrukcje te są prezentowane w dostosowanym do nich formalizmie. Takie podejście zmusza studentów do większego wysiłku związanego z opanowaniem zmieniającej się w czasie wykładu składni, ale również rozwija u słuchaczy umiejętność abstrahowania istotnych koncepcji od mało ważnych detali składniowych.

Wykład nie zakłada u studenta znajomości żadnego z wykorzystywanych języków programowania. Nie ma także na celu nauczenia studentów żadnego z nich. Powinien dać natomiast zrozumienie i umiejętność świadomego korzystania z mechanizmów spotykanych w językach programowania, które studenci opanują samodzielnie lub w ramach Kursów Narzędzi Informatyki.

Treści przedstawione na wykładzie stanowią podstawę do studiowania wielu przedmiotów związanych z programowaniem komputerów.

Program wykładu, z wyjątkiem zagadnień oznaczonych przez „M:", jest taki sam jak program wykładu *Programowanie (L)*. Zagadnienia oznaczone ciągiem „M:" są wymagane na *Programowaniu (M)* i nie są wymagane na *Programowaniu (L)*.

Program wykładu

1. Wstęp do różnych paradygmatów programowania i wspierających je języków programowania. Pojęcia leksyki, składni i semantyki języka programowania. (1 godz.)

2. Programowanie funkcjonalne. Programowanie deklaratywne a imperatywne. Polimorfizm. Skutki uboczne. Porządek wartościowania. Trwałe i ulotne struktury danych. (8 godz.)

3. Abstrakcyjne typy danych. Typy definiowane rekurencyjnie (listy, drzewa, stosy, kolejki itd.) Indukcyjne dowody ich własności. (2 godz.) M: Algebraiczne specyfikacje typów danych. (2 godz.)

4. Gramatyki regularne i bezkontekstowe. Automaty skończone i wyrażenia regularne. Analiza leksykalna i składniowa. Lekser. Parser. Konkretne i abstrakcyjne drzewa rozbioru. (6 godz.) M: Języki kontekstowe i hierarchia Chomsky'ego. (1 godz.)

5. Metody opisu semantyki języków programowania. Semantyka operacyjna. (4 godz.) M: Semantyka denotacyjna i równoważność semantyk. (4 godz.)

6. Specyfikacje programów. Logika Hoare'a i dowodzenie poprawności specyfikacji. Metoda niezmienników. Synteza programów metodą wstępującą i zstępującą. (4 godz.) M: Poprawność i zupełność aksjomatyki Hoare'a. (2 godz.)

7. Struktura współczesnych języków programowania. Nazwy, komórki i wartości, struktury sterujące, funkcje i procedury, dynamiczne struktury danych, wyjątki. Typy danych, mocna typizacja, rekonstrukcja typów. (6 godz.)

8. Pojęcie translatora. Zasięg zmiennych, wywoływanie funkcji i zarządzanie pamięcią. Języki o strukturze blokowej, rekordy aktywacji, metody przydziału pamięci dla zmiennych lokalnych i globalnych. Funkcje wyższego rzędu, funarg problem. Automatyczne zarządzanie pamięcią i odśmiecanie. (6 godz.)

9. Moduły, rodzajowość i abstrakcja danych. Ukrywanie (encapsulation) danych. Rozwiązania w konkretnych językach (moduły, pakiety, itp.). Rodzajowość (funktory, pakiety rodzajowe, wzorce (templates)). (4 godz.)

10. Programowanie obiektowe, metodologia OO, obiekty i klasy, abstrakcyjne typy danych a klasy, podtypowanie a dziedziczenie, języki bezklasowe. Obiektowy styl programowania. (4 godz.)

11. M: Programowanie w logice. (6 godz.)

Literatura

[1] Harold Abelson, Gerald Jay Sussman, Julie Sussman, *Structure and Interpretation of Computer Programs*, MIT Press, 1985.

[2] Michael A. Arbib, *Projektowanie programów poprawnych i dobrze zbudowanych*, WNT, Warszawa 1982.

[3] M. Ben-Ari, *Podstawy programowania współbieżnego*, WNT, Warszawa 1989.

[4] Edsger W. Dijkstra, *Umiejetność programowania*, WNT, Warszawa 1985.

[6] Michael Marcotty, Henry Ledgard, *W kręgu języków programowania*, WNT, Warszawa 1991.

[7] Peter Van Roy, Seif Haridi, *Programowanie. Koncepcje, techniki, modele*, Helion 2005.

[8] Ravi Sethi, *Programming Languages: Concepts and Constructs*, Addison-Wesley, 1996.

[9] Niklaus Wirth, *Algorytmy + struktury danych = programy*, WNT, Warszawa 1989.

Opracowali Marek Piotrów, Paweł Rychlikowski i Tomasz Wierzbicki

3.6 Matematyka dyskretna (L)

Cele nauczanego przedmiotu

Matematyka dyskretna obejmuje zagadnienia matematyczne, które są przydatne informatykowi w jego pracy zawodowej jako programisty, projektanta i wykonawcy projektów informatycznych, administratora sieci komputerowych. Celem tego przedmiotu jest przygotowanie słuchaczy w zakresie tych zagadnień matematycznych z jednoczesnym ich odniesieniem do dziedzin informatyki, w których te zagadnienia znajdują zastosowanie. Trzon programu tego przedmiotu stanowią matematyczne metody reprezentowania i analizowania algorytmów, odnoszących się do zbiorów skończonych, liczb całkowitych i grafów.

Treści programowe

1. **Algorytmy - przykłady algorytmów klasycznych i ich własności**

   Wymienione tutaj algorytmy i ich własności pojawiają się jako ilustracja ogólniejszych rozważań na temat różnych technik informatycznego rozwiązywania problemów i ich własności, głównie złożoności obliczeniowej i efektywności obliczeń.

- Specyfikacja problemu i algorytmu.
- Opis algorytmu w postaci: listy kroków, schematu blokowego, drzewa obliczeń, drzewa algorytmu.
- Przykłady algorytmów: znajdowanie najmniejszej lub największej liczby w ciągu; jednoczesne znajdowanie najmniejszej i największej liczby w ciągu; porządkowanie kilku liczb (na drzewie); algorytmy porządkowania ciągu n liczb: przez wybór, metodą bąbelkową, przez wstawianie.
- Algorytmy rekurencyjne: zagadka Wież Hanoi, liczby Fibonacciego, porządkowanie przez scalanie.
- Schemat Hornera i jego zastosowania: obliczanie dziesiętnej wartości liczby danej w innym systemie, szybkie obliczanie wartości potęgi.
- Techniki algorytmiczne: przeszukiwanie liniowe, przeszukiwanie binarne, metoda dziel i zwyciężaj, rekurencja.
- Złożoność i praktyczna efektywność algorytmów.

2. **Elementy teorii liczb**

- Funkcje całkowitoliczbowe: powała i podłoga.
- Asymptotyka funkcji liczbowych, symbole: o, O, ?, ?. Funkcje wielomianowe i logarytmiczne.
- Podzielność liczb - algorytm Euklidesa z odejmowaniem i dzieleniem. Złożoność algorytmu Euklidesa. Zastosowania algorytmu Euklidesa: równanie diofantyczne.
- Liczby Fibonacciego - definicja, przykłady występowania w matematyce, w przyrodzie i w sztuce, własności, związek ze złotym podziałem.
- Liczby pierwsze i rozkład liczb na czynniki. Podstawowe twierdzenie arytmetyki. Twierdzenie Euklidesa. Sposoby otrzymywania liczb pierwszych: sito Eratostenesa, wzory na liczby pierwsze (liczby Euklidesa, Fermata, Mersenea, Eulera). Złożoność problemów związanych z liczbami pierwszymi.
- Arytmetyka modularna (kongruencje): dodawanie, odejmowanie, mnożenie, warunek na istnienie odwrotności. Zastosowania: kwadraty łacińskie, ortogonalne kwadraty łacińskie. Chińskie twierdzenie o resztach. Twierdzenie Eulera i Małe Twierdzenie Fermata. Zastosowania: kryptografia z kluczem publicznym, podstawy konstrukcji szyfru RSA.

3. **Elementy kombinatoryki**

   - Permutacje, kombinacje. Symbol Newtona. Trójkąt Pascala. Wzór Newtona.
   - Generowania obiektów kombinatorycznych: podzbiorów, permutacji, kombinacji.
   - Zasada gołębnika (szuflada Dirichleta).
   - Zasada włączania i wyłączania. Funkcja Eulera.
   - Funkcje tworzące obiektów kombinatorycznych.

4. **Zależności rekurencyjne**

   - Źródło zależności rekurencyjnych: technika dziel i zwyciężaj, złożoność algorytmów rekurencyjnych, budowa obiektów kombinatorycznych.
   - Rozwiązywanie zależności rekurencyjnych: przez podstawianie, za pomocą równania charakterystycznego, metodą anihilatorów.
   - Wykorzystanie funkcji tworzących do rozwiązywania zależności rekurencyjnych.

5. **Zbiory częściowo uporządkowane**

   - Relacje, całkowity porządek, częściowy porządek i quasi porządek.
   - Diagram Hassego.
   - Kraty. Algebry Boole'a.

6. **Elementy teorii grafów**

   - Grafy symetryczne i grafy skierowane (digrafy) - definicje. Stopnie wierzchołków i ich własności; komputerowe reprezentacje grafów - macierzowa i listowa; rodzaje grafów: pełne, dwudzielne (tw. Koeniga).
   - Przeszukiwanie grafów: metoda w głąb i wszerz. Użycie stosu i kolejki.
   - Drzewa - własności i równoważne definicje. Stopnie wierzchołków w drzewach. Zastosowanie drzew w informatyce: drzewa wyrażeń, drzewa algorytmów, dolne oszacowanie złożoności algorytmów na drzewach i jego zastosowania. Nieskończony Lemat Koeniga.
   - Najkrótsze drzewa rozpinające w grafie. Algorytmy Kruskala i Prima-Dijkstry. Zastosowania.

- Przechodnie domknięcie digrafu. Algorytm Warshalla i jego efektywne realizacje.
- Problemy najkrótszych dróg: algorytm Dijkstry znajdowania najkrótszych dróg w digrafie z wybranego wierzchołka, algorytm Warshalla-Floyda znajdowania najkrótszych dróg między każdą parą wierzchołków w grafie.
- Drogi i cykle w grafach. Grafy Eulera: kryteria, uogólnienia i problem chińskiego listonosza. Grafy Hamiltona: transformacja wielomianowa między różnymi wersjami problemu, kryteria i problem komiwojażera oraz przybliżone metody jego rozwiązywania.
- Kolorowanie grafów - liczba chromatyczna, jej wartości dla wybranych klas grafów, oszacowanie jej wartości, algorytm sekwencyjny, algorytm sekwencyjny w wersji LF. Optymalne kolorowanie metodą przeszukiwania z nawrotami.
- Grafy planarne i grafy płaskie. Wzór Eulera. Oszacowanie liczby krawędzi w grafie planarnym. Kryteria planarności (bez dowodów): Kuratowskiego i Harary'ego-Tutte'a. Przykłady grafów nieplanarnych. Kolorowanie grafów płaskich pięcioma kolorami.

Literatura

[1] Donald. E. Knuth, *Sztuka programowania*, tomy I–III, WNT, Warszawa 2000.

[2] Witold Lipski, *Kombinatoryka dla programistów*, WNT, Warszawa 1982.

[3] Kenneth A. Ross, Charles R. B. Wright, *Matematyka dyskretna*, PWN, 1996.

[4] Maciej M. Sysło, Narsingh Deo, Janusz S. Kowalik, *Algorytmy optymalizacji dyskretnej z programami w języku Pascal*, PWN, Warszawa 1993, 1995, 1997.

[5] Maciej M. Sysło, *Algorytmy*, WSiP, Warszawa 1997, 2002.

[6] Maciej M. Sysło, *Piramidy, szyszki i inne konstrukcje algorytmiczne*, WSiP, Warszawa 1998.

[7] Robin J. Wilson, *Wprowadzenie do teorii grafów*, PWN, Warszawa 1998.

Opracował Maciej M. Sysło

3.7 Matematyka dyskretna (M)

Wymagane przygotowanie studentów

Wiedza z zakresu przedmiotów *Logika dla informatyków*, *Analiza matematyczna* oraz *Algebra.*

Cel wykładu i wskazówki metodyczne

Nauczenie studentów matematyki przydatnej w analizie algorytmów.

Program wykładu

1. Asymptotyka funkcji liczbowych z uwzględnieniem zastosowań w szacowaniu złożoności czasowej algorytmów.
2. Rozwiązywanie prostych równań rekurencyjnych. Liczby Fibonacciego.
3. Operacje sufit i podłoga. Algorytm mergesort i algorytm Karatsuby jako przykłady algorytmów wykorzystujących rekursję.
4. Podzielność liczb, pierścień Z_n, algorytm Euklidesa. Wyliczanie odwrotności w Z_n.
5. Liczby pierwsze. Rozkład na czynniki. Gęstość liczb pierwszych.
6. Chińskie twierdzenie o resztach. Funkcja Eulera i twierdzenie Eulera.
7. Metoda szufladkowa Dirichleta.
8. Rozmieszczenia, permutacje, kombinacje. Wzór dwumienny.
9. Zasada włączania i wyłączania.
10. Twierdzenie Lagrange'a. Lemat Burnside'a.
11. Rozwiązywanie równań rekurencyjnych — metoda anihilatorów.
12. Funkcje tworzące. Liczby Catalana. Podziały liczby.
13. Definicja i przykłady grafów, stopień wierzchołka. Drogi i cykle w grafach: grafy spójne i dwudzielne.
14. Drzewa — równoważność różnych definicji.
15. Komputerowa reprezentacja grafów.

16. Metody BFS i DFS przeszukiwania grafów.
17. Minimalne drzewa rozpinające — algorytmy Kruskala i Prima-Dijkstry.
18. Najkrótsze drogi i przechodnie domknięcie: algorytmy Dijkstry i Warshalla. Złożoność problemu.
19. Cykle i drogi Eulera. Cykle i drogi Hamiltona, tw. Ore i wielomianowa redukcja problemu drogi do cyklu i odwrotnie.
20. Grafy planarne. Tw. Kuratowskiego i wzór Eulera.
21. Kolorowanie grafów. Algorytm sekwencyjny i twierdzenie o 5-kolorowaniu grafów planarnych.
22. Skojarzenia w grafach. Kolorowanie krawędzi grafów.
23. Metody generowania prostych obiektów kombinatorycznych.

Literatura

[1] Victor Bryant, *Aspekty kombinatoryki*, WNT, 1977.

[2] Witold Lipski, *Kombinatoryka dla programistów*, WNT, Warszawa 2004.

[3] Kenneth A. Ross, Charles R. B. Wright, *Matematyka dyskretna*, PWN, 1996.

[4] Robin J. Wilson, *Wprowadzenie do teorii grafów*, PWN, Warszawa 1985.

Literatura uzupełniająca

[5] Ronald Lewis Graham, Donald Ervin Knuth, Oren Patashnik, *Matematyka konkretna*, PWN, 1996.

[6] Witold Lipski, Wiktor Marek, *Analiza kombinatoryczna*, Państwowe Wydawnictwo Naukowe, Warszawa 1986.

Opracował Grzegorz Stachowiak

3.8 Elementy rachunku prawdopodobieństwa

Treści kształcenia: Prawdopodobieństwo dyskretne. Prawdopodobieństwo ciągłe. Wartości oczekiwane. Procesy stochastyczne. Próbkowanie. Estymacja. Testowanie hipotez statystycznych.

Umiejętności: obliczanie prawdopodobieństwa zdarzeń, wartości oczekiwanej, wariancji i odchylenia standardowego; przeprowadzenie prostego wnioskowania statystycznego.

3.9 Analiza numeryczna (L)

Wymagane przygotowanie studentów

Wiedza z zakresu wykładów *Analiza matematyczna* i *Algebra*.

Cel zajęć i wskazówki metodyczne

Celem zajęć jest przedstawienie podstawowych metod i algorytmów rozwiązywania typowych zadań obliczeniowych.

Program wykładu

1. *Analiza błędów.* Arytmetyka numeryczna. Uwarunkowanie zadania. Algorytmy numerycznie poprawne.
2. *Rozwiązywanie równań nieliniowych.* Ogólna teoria metod iteracyjnych. Metody: bisekcji, Newtona i siecznych. Równania algebraiczne – metoda Laguerre'a, obniżanie stopnia równania, metoda Bairstowa.
3. *Interpolacja.* Wzór interpolacyjny Lagrange'a. Reszta wzoru interpolacyjnego. Wzór interpolacyjny Newtona. Interpolacja Hermite'a. Interpolacja za pomocą funkcji sklejanych III stopnia.
4. *Aproksymacja.* Aproksymacja średniokwadratowa za pomocą wielomianów – wielomiany ortogonalne, twierdzenie o n-tym wielomianie optymalnym. Aproksymacja jednostajna – twierdzenie o alternansie, informacja o algorytmie Remeza konstrukcji wielomianu optymalnego, wielomiany prawieoptymalne.
5. *Kwadratury.* Kwadratura liniowa. Reszta i rząd kwadratury. Zbieżność ciągu kwadratur. Kwadratury interpolacyjne. Kwadratury Newtona-Cotesa. Wzory złożone: trapezów i Simpsona. Metoda Romberga. Kwadratury Gaussa.

6. *Rozwiązywanie układów równań liniowych.* Uwarunkowanie zadania. Rozkład macierzy kwadratowej na iloczyn macierzy trójkątnych. Metoda eliminacji Gaussa. Numeryczna poprawność eliminacji Gaussa z wyborem elementów głównych. Iteracyjne metody rozwiązywania układów równań liniowych.

Literatura

[**1**] A. Björck, G. Dahlquist, *Metody numeryczne*, PWN, 1987.

[**2**] M. Dryja, J. i M. Jankowscy, *Przegląd metod i algorytmów numerycznych*, cz. 1 i 2, WNT, 1988.

[**3**] D. Kincaid, W. Cheney, *Analiza numeryczna*, WNT, 2005.

[**4**] J. Stoer, R. Bulirsch, *Wstęp do analizy numerycznej*, PWN, 1987.

Opracował Stanisław Lewanowicz

3.10 Analiza numeryczna (M)

Wymagane przygotowanie studentów

Wiedza z zakresu wykładów *Analiza matematyczna* i *Algebra*.

Cel zajęć i wskazówki metodyczne

Celem zajęć jest wprowadzenie studentów w dziedzinę metod najczęściej przydatnych w obliczeniach naukowych, z uwzględnieniem zarówno matematycznych podstaw tych metod, jak i ich aspektów algorytmicznych.

Program wykładu

Program jest pogłębioną i bogatszą w realizacji wersją programu wykładu *Analiza numeryczna (L)*.

Literatura podstawowa

Pozycje [1]–[4] z wykazu literatury zalecanej do wykładu *Analiza numeryczna (L)*.

Literatura uzupełniająca

[**5**] W. Gautschi, *Numerical Analysis. An Introduction*, Birkhäuser, 1997.

[**6**] G. Hämmerlin, K.-H. Hoffman, *Numerical Mathematics*, Springer-Verlag, 1991.

[**7**] A. Quarteroni, R. Sacco, F. Saleri, *Numerical Mathematics*, Springer-Verlag, 2000.

Opracował Stanisław Lewanowicz

3.11 Algorytmy i struktury danych (L)

Wymagane przygotowanie studentów

Zakłada się, że słuchacze wykładu posiadają wiedzę z przedmiotów *Programowanie* i *Matematyka dyskretna* oraz umieją programować w języku *C/C++*.

Cel zajęć i wskazówki metodyczne

Celem wykładu jest zaprezentowanie studentom wielu różnorodnych zadań obliczeniowych oraz skutecznych i efektywnych metod ich rozwiązywania. Na wykładzie omawiane są podstawowe techniki konstruowania algorytmów i analizy ich złożoności obliczeniowej, a dla wybranych problemów przedstawione są ich dolne granice złożonościowe. Szczególny nacisk jest położony na sposób, w jaki dane są przechowywane w pamięci komputera, gdyż od organizacji danych bardzo często zależy czas działania programu rozwiązującego określone zadanie.

Zakłada się, że po zakończeniu wykładu studenci będą potrafili przeanalizować zadany problem, wybrać odpowiednią technikę jego rozwiązania, będą umieli zaprogramować wybrany lub wymyślony algorytm wykorzystując najbardziej odpowiednią strukturę danych dla danego zadania oraz będą potrafili oszacować jego czas działania i zapotrzebowanie na pamięć. W porównaniu z przedmiotem o tej samej nazwie wykładanym na studiach magisterskich, większy nacisk jest położony na praktyczne wykorzystanie omawianych algorytmów i struktur danych do rozwiązywania zadanych problemów, natomiast kwestia analizy kosztu, a w szczególności dolnych granic, traktowana jest bardziej faktograficznie.

Program wykładu

1. Złożoność obliczeniowa i jej szacowanie; pesymistyczna i oczekiwana złożoność algorytmu; szacowanie złożoności problemu; przykłady analizy kosztów. (3 godziny)

2. Sortowanie: elementarne metody, sortowanie Shella. Model drzew decyzyjnych i dolne ograniczenie na problem sortowania. Sortowanie w czasie liniowym: counting–sort, radix–sort, bucket–sort. (3 godziny)

3. Scalanie i sortowanie przez scalanie merge–sort. Sortowanie zewnętrzne. (3 godziny)

4. Podział i sortowanie szybkie quick–sort. Problem wyboru: algorytmy Hoare'a i „magicznych piątek". (3 godziny)

5. Analiza kosztu zamortyzowanego. Tablice dynamiczne. (3 godziny)

6. Haszowanie: rozwiązywanie kolizji metodą łańcuchową i za pomocą adresowania otwartego. (3 godziny)

7. Słowniki: drzewa BST, losowe BST, drzewa SPLAY, drzewa zrównoważone AVL i 2-3-4-drzewa, drzewa pozycyjne RST i TRIE. Słowniki zewnętrzne: B-drzewa. (3 godzin)

8. Kolejki priorytetowe: kopiec, kopiec minimaksowy. Złączalne kolejki priorytetowe: kopce dwumianowe, drzewa lewicowe. (3 godziny)

9. Zbiory rozłączne: reprezentacja listowa i drzewiasta. (3 godziny)

10. Projektowanie algorytmów metodą dziel i zwyciężaj. Przykłady: mnożenie długich liczb, mnożenie macierzy metodą Strassena. Uniwersalne twierdzenie o rekurencji. (3 godziny)

11. Projektowanie algorytmów za pomocą programowania dynamicznego. Przykłady: najdłuższy wspólny podciąg, optymalne mnożenie macierzy. Rekurencja ze spamiętywaniem. (3 godziny)

12. Projektowanie algorytmów przy pomocy strategii zachłannej. Przykłady: wydawanie reszty, kody Huffmana, ciągły problem plecakowy, problem wyboru zajęć. (3 godziny)

13. Algorytmy wielomianowe, pseudowielomianowe i ponadwielomianowe. Wielomianowe redukcje. Problemy optymalizacyjne i odpowiadające im problemy decyzyjne. Algorytmy niedeterministyczne i weryfikacja

w czasie wielomianowym. Klasy problemów P i NP. Problemy NP-zupełne: CIRCUIT-SAT, SAT, 3-CNF. (3 godziny)

14. Algorytmy aproksymacyjne: problem pokrycia wierzchołkowego i problem pokrycia zbioru. Algorytmy zrandomizowane: test pierwszości liczby. Algorytmy z nawrotami: kolorowanie grafu. Metoda podziału i ograniczeń: problem komiwojażera. (6 godzin)

Literatura podstawowa

[1] L. Banachowski, K. Diks, W. Rytter: *Algorytmy i struktury danych*. WNT, Warszawa 1996.

[2] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein: *Wprowadzenie do algorytmów*. WNT, Warszawa 2004.

Literatura uzupełniająca

[3] A. V. Aho, J. E. Hopcroft, J. D. Ullman: *Projektowanie i analiza algorytmów*. Wydawnictwo Helion, Gliwice 2003.

[4] L. Banachowski, A. Kreczmar, W. Rytter: *Analiza algorytmów i struktur danych*. WNT, Warszawa 1989.

[5] G. Brassard, P. Bratley: *Algorithmics — theory & practice*. Prentice Hall, 1993.

[6] J. Kleinberg, E. Tardos: *Algorithm design*. Addison–Wesley, 2005.

[7] D. E. Knuth: *Sztuka programowania (tom 1, 2, 3)*. WNT, Warszawa 2001.

[8] D. C. Kozen: *The Design and analysis of algorithms*. Springer–Verlag, 1992.

[9] R. Neapolitan, K. Naimipour: *Podstawy algorytmów z przykładami w C++*. Wydawnictwo Helion, Gliwice 2004.

[10] R. Sedgewick: *Algorytmy w C++*. Wydawnictwo RM, Warszawa 1999.

opracował Paweł Rzechonek

3.12 Algorytmy i struktury danych (M)

Wymagane przygotowanie studentów

Wiedza z zakresu wykładów *Programowanie* oraz *Matematyka dyskretna.*

Program wykładu

1. Przegląd metod projektowania efektywnych algorytmów: „dziel i zwyciężaj", programowanie dynamiczne, metoda zachłanna. (4 godz.)

2. Złożoność obliczeniowa algorytmu (pesymistyczna, oczekiwana, zamortyzowana). Przykłady analizy kosztu. (2 godz.)

3. Dolne granice: modele i metody. Drzewa decyzyjne, liniowe drzewa decyzyjne, gry z adwersarzem, redukcje. (2 godz.)

4. Sortowanie: Heapsort, Mergesort i Quicksort. Sortowanie w czasie liniowym: Countsort, Radixsort, Bucketsort. (6 godz.)

5. Selekcja: algorytmy Hoare'a i „magicznych piątek". (2 godz.)

6. Kolejki priorytetowe: kopce binarne, dwumianowe i Fibonacciego. Zastosowania w problemie najkrótszych ścieżek i minimalnego drzewa rozpinającego. (4 godz.)

7. Scalanie. Drzewa turniejowe. Sortowanie zewnętrzne. (2 godz.)

8. Słowniki. Drzewa BST, zrównoważone drzewa BST (AVL, 2-3-4-drzewa, drzewa czerwono-czarne). Optymalne drzewa wyszukiwań binarnych. Drzewa samoorganizujące się. Haszowanie. Słowniki statyczne. (8 godz.)

9. Wyszukiwanie zewnętrzne — B-drzewa. (2 godz.)

10. Problem sumowania zbiorów rozłącznych i jego zastosowania. (4 godz.)

11. Algorytmy grafowe: DFS i jego zastosowania, przepływy w sieciach, skojarzenia. (4 godz.)

12. Algorytmy tekstowe: Wyszukiwanie wzorca (algorytm Karpa-Rabina, Algorytm Knutha-Morrisa-Pratta). Drzewa sufiksowe (4 godz.)

13. Geometria obliczeniowa. Lokalizacja punktu. Otoczka wypukła. Technika zamiatania. (4 godz.)

14. Algorytmy algebraiczne i teorioliczbowe. FFT. Szybkie mnożenie liczb i wielomianów. (4 godz.)

15. NP-zupełność. Algorytmy aproksymacyjne dla problemów obliczeniowo trudnych. Heurystyki dla problemów trudnych (algorytmy genetyczne, simulated annealing). (4 godz.)

16. Modele obliczeń równoległych: PRAM, tablica procesorów, hiperkostka. Algorytmy równoległe. Klasa NC i problemy P-zupełne. (2 godz.)

17. Specjalne modele obliczeń: sieci komparatorów, obwody logiczne. (2 godz.)

18. Algorytmy randomizacyjne.Przykłady zastosowań randomizacji w geometrii obliczeniowej, algorytmach grafowych, algorytmach równoległych, konstrukcji struktur danych. (2 godz.)

Literatura podstawowa

[1] Alfred V. Aho, John E. Hopcroft, Jeffrey D. Ullman, *Projektowanie i analiza algorytmów komputerowych*, PWN, Warszawa 1983.

[2] G. Brassard, P. Bratley, *Algorithmics. Theory & practice*, Prentice Hall, 1993.

[3] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, *Introduction to algorithms*, MIT Press, 1992.

[4] J. Kleinberg, E. Tardos, *Algorithm Design*, Addison Wesley, 2006.

Literatura uzupełniająca

[5] S. Baase„ A. von Gelder *Computer Algorithms: Introduction to Design and Analysis*, Addison-Wesley, 2000.

[6] L. Banachowski, K. Diks, W. Rytter, *Algorytmy i struktury danych*, WNT, Warszawa 1996.

[7] L. Banachowski, A. Kreczmar, W. Rytter, *Analiza algorytmów i struktur danych*, WNT, Warszawa 1989.

[8] S. E. Goodman, S. T. Hedetniemi, *Introduction to the Design and Analysis of Algorithms*, McGraw-Hill, 1977.

[9] D. H. Greene, D. E. Knuth, *Mathematics for the Analysis of Algorithms*, Birkhäuser, 1982.

[10] D. E. Knuth, *Sztuka programowania*, T. 1–3, WNT, 2003.

[11] Dexter C. Kozen, *The Design and Analysis of Algorithms*, Springer, 1992.

[12] M. Mitzenmacher, E. Upfal, *Probability and Computing. Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005.

[13] R. Motwani, P. Raghavan, *Randomized Algorithms*, Cambridge University Press, 1995.

[14] T. Ottmann, P. Widmayer, *Algorithmen und Datenstrukturen*, Wissenschaftsverlag, 1993.

[15] E. M. Reingold, J. Deo, N. Nievergelt *Algorytmy kombinatoryczne*, PWN, Warszawa 1985.

[16] V. Vazirani, *Approximation Algorithms*, Springer, 2001.

Opracował Krzysztof Loryś

3.13 Języki formalne i złożoność obliczeniowa

Wymagane przygotowanie studentów

Wiedza z zakresu wykładów *Logika dla informatyków* i *Matematyka dyskretna*.

Program wykładu

A1. Deterministyczny automat skończony. Języki regularne. Lemat o pompowaniu. Twierdzenie o indeksie.

A2. Niedeterminizm. Niedeterministyczny automat skończony. Determinizacja automatu.

A3. Wyrażenia regularne. Równoważność automatów skończonych i wyrażeń regularnych. Aspekty algorytmiczne: rozstrzyganie równoważności wyrażeń regularnych jest możliwe ale zagadkowo czasochłonne.

A4. Uwagi o automatach na obiektach innych niż słowa skończone: automaty na drzewach skończonych i na słowach nieskończonych.

A5. Gramatyki bezkontekstowe. Przykłady.

A6. Postać Chomsky'ego i lemat o pompowaniu dla języków bezkontekstowych.

A7. Automaty ze stosem. Postać Greibach gramatyk bezkontekstowych. Równoważność gramatyk i automatów ze stosem.

A8. Własności zamkniętości klasy języków bezkontekstowych. Niemożliwość determinizacji.

R1. Zbiory rekurencyjne i rekurencyjnie przeliczalne. Funkcje rekurencyjne — częściowe i całkowite. Numeracja funkcji rekurencyjnych. Nierozstrzygalność problemu stopu.

R2. Pojęcie redukcji. Twierdzenie Rice'a. Uwagi o implikacjach tw. Rice'a dla możliwości automatycznej weryfikacji programów.

R3. Maszyna Turinga. Teza Churcha.

R4. Nierozstrzygalność problemu słów dla semiprocesów Thuego.

R5. Nierozstrzygalność problemu słów dla procesów Thuego (problemu słów w półgrupie).

R6. Nierozstrzygalność problemu odpowiedniości Posta, i przykłady nierozstrzygalnych problemów dotyczących gramatyk.

R7. Nierozstrzygalność teorii pierwszego rzędu liczb naturalnych z dodawaniem i mnożeniem (ewentualnie: z dodawaniem, mnożeniem i potęgowaniem). Uwagi o niemożliwości zaksjomatyzowania arytmetyki.

R8. Nierozstrzygalność rachunku predykatów pierwszego rzędu.

C1. Klasa PTIME i PSPACE. Redukcje wielomianowe. Wielomianowa równoważność 3SAT i 3COL.

C2. Niedeterminizm i klasa NP. Przykłady. Przykłady języków z klasy co-NP, oraz z przecięcia NP i co-NP. Charakteryzacja NP jako klasy języków będących projekcjami języków z P.

C3. NP zupełność. Twierdzenie Cook'a. Więcej przykładów problemów NP-zupełnych. Uwagi o SAT-solverach.

C4. PSPACE. Tw. Savitcha.

C5. Problemy PSPACE-zupełne: QBF i totalność wyrażeń regularnych. Wyjaśnienie zagadki z wykładu A3.

C6. Funkcje jednostronne. Uwagi o teoriozłożonościowych założeniach kryptografii z kluczem publicznym.

C7. Słaba wersja twierdzenia o hierarchii czasowej: różność EXPTIME i PTIME. Uwagi o sposobach uogólnienia tej słabej wersji.

C8. Problemy wymagające dowodliwie czasu wykładniczego. Pebble games. Totalność wyrażeń regularnych

C9. Inne niż PTIME formalizacje intuicji "łatwej obliczalności". Uwagi o klasie FPT. Zrandomizowane algorytmy testowania pierwszości. Uwagi o komputerach kwantowych.

C10. Przykład problemu rozstrzygalnego ale nieelementarnego: totalność wyrażeń regularnych z dopełnieniem.

Literatura podstawowa

[1] Alfred V. Aho, John E. Hopcroft, Jeffrey D. Ullman, *Projektowanie i analiza algorytmów komputerowych*, PWN, Warszawa 1983.

[2] John E. Hopcroft, Jeffrey D. Ullman, *Wprowadzenie do teorii automatów, języków i obliczeń*, WNT, Warszawa 1994.

[3] Jose Luis Balcazar, Josep Diaz, Joaquim Gabarro, *Structural complexity I*, Springer, 1988.

[4] Christos H. Papadimitriou, *Computational complexity*, Vol. 1. Addison-Wesley, 1994.

Opracował Jerzy Marcinkowski

3.14 Przedmioty gwarantowane

Wymienione w tym rozdziale przedmioty zawierają w swoim programie wymienione treści pokrywające, obok przedmiotów obowiązkowych, obowiązkowe treści kierunkowe ze standardów.

3.14.1 Wstęp do informatyki

Treści kształcenia: Pojęcie algorytmu. Podstawowe konstrukcje programistyczne. Implementacje algorytmów w językach programowania. Podstawowe struktury danych. Rekurencja i jej implementacja w językach wysokiego poziomu.

Umiejętności: pisanie i uruchamianie prostych programów; specyfikacja i implementacja prostych problemów algorytmicznych.

3.14.2 Architektury systemów komputerowych

Treści kształcenia: Technika cyfrowa i systemy cyfrowe. Maszynowa reprezentacja danych i realizacja operacji arytmetycznych. Organizacja komputera na poziomie asemblera. Organizacja i architektura systemów pamięci. Interfejsy i komunikacja. Organizacja jednostki centralnej. Wieloprocesorowość i architektury alternatywne.

Umiejętności: Obliczanie reprezentacji liczb całkowitych i rzeczywistych oraz wykonywanie podstawowych operacji na tych reprezentacjach; pisanie prostych programów na poziomie asemblera.

3.14.3 Bazy danych

Treści kształcenia: Systemy baz danych. Modelowanie danych. Relacyjne bazy danych. Języki zapytań do baz danych. Projektowanie relacyjnych baz danych. Przetwarzanie transakcji.

Umiejętności: formułowanie zapytać w SQL; przygotowywanie schematu relacyjnej bazy danych na podstawie modelu E-R; tworzenie aplikacji z odwołaniami do baz danych; ocena efektywności strategii wykonania zapytania do bazy danych.

3.14.4 Systemy operacyjne

Treści kształcenia: Przegląd systemów operacyjnych. Zasady działania systemów operacyjnych. Procesy i wątki. Współbieżność. Szeregowanie zadań. Zarządzanie pamięcią.

Umiejętności: Rozwiązywanie klasycznych problemów synchronizacji, w tym problemu producent-konsument i czytelnicy pisarze oraz problemu pięciu filozofów; dobieranie algorytmu szeregowania zadań do specyfiki aplikacji.

3.14.5 Sieci komputerowe

Treści kształcenia: Wprowadzenie do sieci komputerowych. Komunikacja i sieci komputerowe. Bezpieczeństwo w sieciach i kryptografia. Technologie udostępniania informacji w sieciach komputerowych. Budowa aplikacji sieciowych.

Umiejętności: instalowanie prostej sieci z dwoma klientami i pojedynczym serwerem z wykorzystaniem narzędzi typu DHCP; korzystanie z kluczy i pakietów kryptograficznych PGP.

3.14.6 Inżynieria oprogramowania

Treści kształcenia: Projektowanie oprogramowania. Korzystanie z API. Narzędzia i środowiska wytwarzania oprogramowania. Procesy wytwarzania oprogramowania. Wymagania i specyfika. Walidacja i testowanie oprogramowania. Ewolucja oprogramowania. Zarządzanie przedsięwzięciem programistycznym.

Umiejętności: posługiwanie się wzorcami projektowymi; projektowanie oprogramowania zgodnie z metodyką strukturalną lub obiektową; dokonywanie przeglądu projektu oprogramowania; wybór narzędzi wspomagających budowę oprogramowania; dobór modelu procesu wytwarzania oprogramowania do specyfiki przedsięwzięcia; specyfikowanie wymagań dotyczących oprogramowania i przeprowadzania ich przeglądu; tworzenia oceny i realizacji

planu testowania; uczestniczenia w inspekcji kodu; zarządzania konfiguracją oprogramowania; opracowywania planu przedsięwzięcia dotyczącego budowy oprogramowania.