Утвърдил: ……..
/ проф. дмн Иван Сосков /
Утвърден от Факултетен съвет
с протокол № 3 / 25.03.2013

СОФИЙСКИ УНИВЕРСИТЕТ “СВ. КЛИМЕНТ ОХРИДСКИ”

Факултет по Математика и Информатика

Специалност: Информатика

М	И	И	0	1	0	1	1	1

Курс: втори
Учебна година: 2020/2021
Семестър: 4 (летен)

УЧЕБНА ПРОГРАМА

Дисциплина:

Е	2	1	6

Дизайн и анализ на алгоритми
Design and analysis of algorithms

Тип: Задължителна дисциплината

Преподавател: доц. Минко Марков

Асистенти: хон. Димо Чанев, хон. Андрей Дренски, хон. Цветелин Костадинов, хон. Светослав Богданов

Учебна заетост	**Форма**	**Хорариум**
Аудиторна заетост	Лекции	45
	Семинарни упражнения	30
	Практически упражнения (хоспетиране)	
Обща аудиторна заетост		**75**
Извънаудиторна заетост	Подготовка на домашни работи	
	Контролни работи и подготовка за тях	25
	Учебен проект	
	Самостоятелна работа в библиотека или с интернет ресурси	50
	Доклад/Презентация	
	Подготовка за изпит	30
Обща извънаудиторна заетост		**105**
ОБЩА ЗАЕТОСТ		**180**
Кредити аудиторна заетост		**2.5**
Кредити извънаудиторна заетост		**3.5**
ОБЩО ЕСТК		**6**

№	Формиране на оценката по дисциплината[1]	% от оценката
1.	Контролни работи	35%
2.	Участие в час	
3.	Домашни работи	3 x 5% = 15%
4.	Учебен проект	
5.	Тестова проверка	
6.	Текуша самостоятелна работа /контролно	
7.	Workshops {информационно търсене и колективно обсъждане на доклади и реферати)	
8.		
9.		
10.		
11.	Изпит – практика (решаване на задачи)	50%
12.	Изпит - теория	

Анотация на учебната дисциплина:

Частта анализ въвежда понятията големина на входа и сложност по време в най-лошия случай. Въвеждат се петте стандартни нотации: тита, о-голямо, омега-голямо, о-малко и омега-малко, които се използват при изследването на асимптотичното нарастване на функции.

Освен сложността в най-лошия случай се

засяга и анализирането на средната сложност. Демонстрират се формални доказателства за коректност на някои базови алгоритми, като за итеративните алгоритми се въвежда инварианта на цикъла като средство за прецизна верификация.

Илюстрира се как използването на ефикасни структури данни подобрява сложността по време. Разглеждат се техники за оценка на сложностите на алгоритми по време и се демонстрира използването на тези техники върху класически образци. Демонстрират се техники за доказване на асимптотична долна граница на сложността по време чрез дървета на решението.

Въвеждат се понятията NP-пълнота, полиномиални редукции и апроксимиране. Обяснява се значението на феномена NP-пълнота и значението на нерешената задача „P = NP ?”. Демонстрират се някои основни полиномиални редукции. Анализират се няколко апроксимиращи алгоритъма като сложност по време и като отдалеченост на решението от оптималното.

В частта дизайн се въвеждат редица алгоритмични схеми, предназначени да улеснят създаването на ефикасни алгоритми: разделяй и владей, динамично програмиране, greedy и обхождане на графи. Прилагат се схемите за решаване на типични задачи, като се демонстрират техниките за оценяване на сложност на получените алгоритми. Демонстрират се схеми за съставяне на апроксимиращи алгоритми.

[1] В зависимост от спецификата на учебната дисциплина и изискванията на преподавателя е възможно да се добавят необходимите форми, или да се премахнат ненужните.

Предварителни изисквания:

Студентите трябва да са запознати с материала от курсовете:

Е106 Дискретни структури

И246 Езици, автомати и изчислимост

К136 Диференциално и интегрално смятане-1

К146 Диференциално и интегрално смятане-2

3106 Увод в програмирането

Очаквани резултати:

Студентите да придобият реален опит и умения в решаването на изчислителни задачи:

- Да създават дискретен модел за задачата и да го кодират в компактна структура данни.
- Да реализират ефективен алгоритъм чрез използване на алгоритмични схеми, свеждане до известни алгоритми, или използване на комбинация от подобни техники.
- Да доказват коректността и да правят анализ на сложността на решението. Да търсят по-добри решения чрез използване на ефикасни структури от данни.
- Когато задачата е NP-трудна, да разпознават този факт и да търсят подходящ апроксимиращ алгоритъм или да отделят частни случаи, в които задачата може да бъде решена бързо.

Учебно съдържание

№	Тема:	Хорариум
1	Математически инструменти за анализ на алгоритми – асимптотични нотации и техните свойства, методи за решаване на рекурентни отношения.	5+8
2	Алгоритмични схеми.	5+8
3	Сортировки.	12+5
4	Бързи алгоритми върху графи.	12+5
5	Класове на сложност и труднорешими задачи.	11+4

Конспект за изпит

№	Въпрос
1	Алгоритми – същност и неформално определение. Изчислителни задачи. Големина на входа на алгоритми. Сложност по време в най-лошия случай, средна и в най-добрия случай. Други аспекти на анализирането на алгоритмите – сложност по памет и коректност.
2	Асимптотичен анализ на сложността – важност, предимства и недостатъци. Нотации, използвани при асимптотичния анализ: O, o, Θ, Ω, ω. Свойства на нотациите.
3	Сортиране – приложения, важност, примери за използване на сортирането като първа фаза от решението на други изчислителни проблеми. Стабилни и нестабилни сортиращи алгоритми. Важност на стабилността.
4	Елементарни сортиращи алгоритми: Selection Sort и Insertion Sort. Въвеждане на понятието инвариантa на цикъла. Анализ на коректността на двата алгоритъма чрез инвариантa на цикъла. Анализ на сложността по им по време и по памет. Анализ на стабилността им.
5	Двоична пирамида. Двоичните пирамиди като попълнени двоични дървета и като масиви. Формули за индексите на родителя и на децата на даден връх, ако пирамидата е масив. Построяване на пирамида: наивен начин чрез HeapInsert.
6	Построяване на пирамида чрез функция Heapify. Анализ на сложността по време на двата начина за построяване на пирамида. Процедура Build Heap. Анализ на сложността: наивна имплементация ($\Theta(n.\lg(n))$) и бърза имплементация ($\Theta(n)$). Пирамидална сортировка Heapsort.
7	Приоритетни опашки като абстрактен тип данни (АТД). Имплементация на приоритетни опашки чрез обикновени масиви и чрез двоични пирамиди – сравнителен анализ на тези две имплементации.
8	Рекурсивни алгоритми и рекурентни отношения. Методи за решаване на рекурентни отношения: развиване, дърво на рекурсията, индукция, Master Theorem, методът с характеристичното уравнение. Примери.
9	Алгоритмична схема Разделяй и владей: същност, приложение и ограничения върху възможностите за прилагането ѝ. Сортиращ алгоритъм Mergesort. Анализ на сложността по време и на стабилността на Mergesort. Анализ на коректността на Mergesort чрез инвариантa на цикъла.

10	Сортиращ алгоритъм Quicksort . Анализ: сложност по време и памет, стабилност, анализ на средната сложност по време, коректност.
12	Долни граници върху асимптотичната сложност по време на алгоритми. Дървета на решението (decision trees). Метод за доказване на долни граници, базиран дървета на решението. Долна граница върху асимптотичната сложност по време на сортиращи алгоритми, базирани на директно сравнение.
13	Долна граница, доказана чрез анализ на дървото на решението, върху асимптотичната сложност на изчислителната задача Element Uniqueness. Доказване на асимптотични долни граници чрез редукции между изчислителни задачи – пример с Closest Pair.
14	Сортиране, което не е базирано на директно сравнение. Сортиране в линейно време – алгоритми Counting Sort и Radix Sort. Анализ на коректността им и на сложността им по време.
15	Ориентирани и неориентирани графи от алгоритмична гледна точка. Примери за задачи, които се моделират чрез графи. Представяния на графи чрез матрици на съседства и чрез списъци на съседства. Сравнителен анализ на предимствата и недостатъците на тези представяния. Анализ на сложността по време на алгоритми върху графи – въвеждане на асимптотични нотации на функции на две променливи. Линейна сложност по време при графи.
16	Обхождане на графи. Основна схема за обхождането – маркиране на върховете с цветове. Коректност и ефикасност на основната схема. Обхождането в ширина и в дълбочина като различни имплементации на основната схема.
17	Обхождане в ширина (BFS) на графи. Дърво на обхождането в ширина. Видове ребра в обхождането в ширина. Обхождане в дълбочина (DFS) на графи. Дърво на обхождането в дълбочина. Видове ребра в обхождането в дълбочина при неориентираните и ориентираните графи.
18	Ориентирани ациклични графи (dags). Алгоритми за топологично сортиране на ориентирани ациклични графи. Анализ на сложността по време на тези алгоритми.
19	Силно свързани компоненти на ориентирани ациклични графи. Алгоритъм с линейна сложност по време за откриването на силно свързаните компоненти на ориентирани графи.
20	Срязващи върхове (cut vertices) и срязващи ребра (bridges) в неориентираните графи. Двусвързани компоненти в неориентирани графи. Алгоритми с линейна сложност по време за откриване на срязващите върхове на граф и на срязващите ребра на граф.
21	Минимални покриващи дървета на неориентирани графи. Примери за приложения на минимални покриващи дървета. Обща алгоритмична схема за алгоритми за откриване на минимални покриващи дървета. Greedy стратегии. Алгоритъм на Прим – псевдокод и анализ на сложността по време.
22	Алгоритъм на Крускал за откриване на минимално покриващо дърво. Сложност по време на наивната имплементация. Усъвършенстване на наивната имплементация – Union-Find операции върху множества, имплементирани чрез дървета. Евристики Union by rank и Path compression. Анализ на сложността по време на алгоритъма на Крускал при използването на двете евристики.
23	Най-къси пътища в ориентираните графи. Примери за приложения. Потенциални трудности при наличието на отрицателни тегла. Анализ на

	задачата и сравнението и със задачата за най-дълъг път в граф. Алгоритъм на Дийкстра за намиране на най-късите пътища от даден връх до всички останали върхове. Анализ на коректността и сложността по време на този алгоритъм.
24	Алгоритми за намиране на най-къси пътища между всички двойки върхове – алгоритъм, използващ алгоритъма на Дийкстра, и два алгоритъма, използващи идеята на динамичното програмиране: по броя на ребрата в пътя и по най-големия номер на връх в пътя (алгоритъм на Флойд-Уоршъл). Анализ на сложността по време.
25	Динамично програмиране – същност, предимства и приложения. Примери: числа на Фибоначи, оптимално верижно умножение на матрици, изчислителен проблем Independent Set при неориентираните графи, ограничен до дърветата. Сравнение между динамичното програмиране и схемата „Разделяй и владей“.
26	Практически решими изчислителни задачи – клас на сложност P. Практически нерешими (intractable) задачи. Примери върху неориентирани графи: VERTEX COVER, INDEPENDENT SET, DOMINATING SET. Практическата нерешимост като фундаментално, независещо от технологичния прогрес, ограничение – принцип на Landauer. Практически нерешими задачи с кратък сертификат – клас на сложност NP.
27	Недетерминирани алгоритми за решаване в полиномиално време на задачи с кратки сертификати. Въпросът дали P = NP като фундаментална нерешена задача на теоретичната компютърна наука.
28	Полиномиални редукции между изчислителни задачи. NP-пълнота. Изчислителна задача SATISFIABILITY. Доказване на NP-пълнотата на SATISFIABILITY – теорема на Кук.
29	Изчислителна задача 3-SATISFIABILITY. Доказване на NP-пълнотата на 3-SATISFIABILITY. Съпоставка на 3-SATISFIABILITY с 2-SATISFIABILITY. Доказване на NP-пълнотата на VERTEX COVER, CLIQUE, HAMILTON CYCLE и LONGEST PATH. Други NP-пълни изчислителни задачи.
30	Апроксимацията като средство за „справяне“ с практическата нерешимост. Апроксимиращи алгоритми с точност до мултипликативна константа. Примери с VERTEX COVER и задачата за търговския пътник в „планарен вариант“. Условна невъзможност за добро апроксимиране на някои изчислителни задачи.

Библиография

Основна:

1. T. Cormen, Ch. Leiserson, R. Rivest, *Introduction to Algorithms*, MIT Press, 1990.
2. M. Garey, D. Johnson, *Computers and Intractability. A Guide to the Theory of NP-c Completeness*, W.H. Freeman & Co., San Francisco, 1979.
3. S. Dasguta, C. H. Papadimitriou, U. Vazirani, *Algorithms*, McGraw-Hill Science/Engineering/Math, first edition, 2006. Книгата е достъпна безплатно и свободно на адрес http://www.cs.berkeley.edu/~vazirani/algorithms/all.pdf .
4. D. Knuth, *The Art of Computer Programming, vol.1 Fundamental Algorithms*, second edition, Addison-Wesley, 1973.
5. D. Knuth, The Art of Computer Programming, vol.3 Sorting and Searching, second edition, Addison-Wesley, 1998.
6. I. Parberry, *Problems on Algorithms*, Prentice Hall, 1995. Книгата е предоставена за безплатен достъп от автора, но само след приемането на условията от лиценза, на адрес http://www.eng.unt.edu/ian/books/free/ .
7. I. Parberry, *Lecture Notes on Algorithm Analysis and Complexity Theory*, fourth edition, 2001. Книгата е предоставена за безплатен достъп от автора, но само след приемането на условията от лиценза, на адрес http://www.eng.unt.edu/ian/books/free/ .
8. H. Wilf, Algorithms and Complexity, Prentice-Hall, 1986. Книгата е достъпна безплатно и свободно на адрес http://www.math.upenn.edu/~wilf/AlgoComp.pdf .

Допълнителна:

9. G. Brassard, P. Bratley, *Fundamentals of Algorithmics*, Prentice Hall, Englewood Clifs, 1996.
10. R. Graham, D. Knuth, O. Patashnik, *Concrete Mathematics. A Foundation for Computer Science,* Second Edition, Addison-Wesley, Reading, 1998.
11. Dan Gusfield, *Algorithms on Strings, Trees and Sequences, Cambridge Univ. Press, 1997.*
12. R. Sedgewick, *Algorithms in C*, Addison-Wesley, Reading, 1990.
13. S. Skiena, *The Algorithm Design Manual*, Springer-Verlag, 1998.
14. S. Skiena, лекционни записи в аудио и видеоформат, както и слайдовете на презентациите, достъпни безплатно и свободно на адрес http://www.cs.sunysb.edu/~algorith/video-lectures/ .

Дата: **Съставил:**

25.02.2013 **гл. ас. д-р Георги Георгиев**

Прието на заседание на катедра „Изчислителни системи” – протокол № 30 от 27.02.2013 г.