Увод

Понятието *алгоритъм* е основно за информатиката.То произлиза от името на персийския учен Мохамед ибн Муса **ал Хорезми** (от град Хорезми, днес Хива в Узбекистан). Роден около 780 г. и почина през 847 г., този древен мислител (математик, астроном и географ), един от основателите на Багдатската школа, е автор на три фундаментални математически съчинения. Едно от тях, трактатът „За индийското смятане“, е посветено на представянето на естествените числа в десетична позиционна бройна система и извършване на аритметични операции с такива представяния. Затова в средните векове с понятието *algorismus* или *algorithmus* са означавали дсетичната бройна система и правилата за извършване на операции в нея.

Постепенно смисълът на понятието се разширява и в работите на Кр. Рудолф (1525 г.) и Г. В. Лайбниц (1684 г.) добива съременния си смисъл: „систематичен изчислителен процес, който с краен брой стъпки решава определена задача“ (вж. *История на математиката*, том I, изд. „Наука и изкуство“, София, 1974 г.).

За разлика от обичайните понятия на математиката, за понятието алгоритъм няма формална дефиниция. Ето как определят това понятие някои тълковни речници:

Речник на българския език, Издателство на БАН, том I, 1977 г.: Система от правила, които ределят последователност от изчислителни операции, прилагането на които води до решение на дадена задача. Алгоритъм на Евклид.

Larousse de la langue Française, Lexis, 1979: Съвкупност от правила или предписания за получаване, с краен брой операции, на определен резултат.

Webster's New Collegiate Dictionary, G. & C. Merriam Company, 1973: Процедура за решаване на математически проблеми (например, намиране на най-голям общ делител) с краен брой стъпки, която често съдържа повтарящи се операции.

Да разгледаме вимателно дадените по-горе описания. Не е трудно да изброим някои важни характеристики на понятието алгоритъм, забелязани от езиковедите, които са съставили споменатите по-горе тълковни речници.

1. Безусловно, когато говорим за алгоритми, предполагаме наличието на някаква съвкупност от *обекти*, с които можем да изпълняваме някакви *операции*. Най-често това са съвкупности от математически дефинирани обекти и операциите с тях. Например, обичайно множество от обекти, за които строим алгоритми е множеството на целите числа с аритметичните операции (събиране, изваждане, умножение, деление) и релациите за сравняване на цели числа, за делимост и т.н. Когато става въпрос за нематематически обекти, често срещан подход в алгоритмиката е заместването им с математически, т.е. създаване на съответен *математически модел*. Затова при изучаването на алгоритми ще използваме само обекти с математически характер.

2. След като е зададено множеството от обекти и операциите с тях, интерес представляват *задачите* за тези обекти, които могат да бъдат решавани само с прилагане на зададените операции. В общ вид една задача изглежда така: „Дадени са обекти от избраното множество. С използване на зададените операции да се намери (построи) обект, имащ зададени характеристики и зависещ по определен начин от зададените обекти.“ В едно от цитираните по-горе описания се споменава такава задача – за *намиране на най-голям общ делител* (НОД). Точната формулировка на тази задача е следната: „Дадени са положителните естествени числа A и B. Да се намери най-голямото естествено число C, което дели без остатък както A, така и B. Допустими са аритметичните операции и сравняването на естествени числа.“ Задаваните обекти наричаме *входни данни* (или *вход*), а получаваните обекти – *резултат* (или *изход*). В случая входните данни са произволни естествени числа, различни от нула, а резултатът е също естествено число.

3. Алгоритъмът е процедура, предназначена да решава дадена задача, с прилагне на допустимите операции в строго определен ред. По зададените входни данни тя трябва да намира искания резлтат. Процедурата трябва да е описана така, че да не предизвиква никакви съмнения за това какви операции ще се прилагат и в какъв ред – казваме, че описанието трябва да е *формално*. Процедурата трябва да е *детерминирана* – който и да я изпълнява, при едни и същи входни данни, трябва да получи един и същ резултат. Споменатият, в едно от по-горе цитираните описания на понятието, *Алгоритъм на Евклид* е точно такава процедура за намиране на НОД на две положителни естествени числа.

4. Важна характеристика на алгоритмичните процедури е, че те трябва да намират решението на задачата с *краен брой стъпки*, т.е. с краен брой прилагания на допустимите операции. Нещо повече, в общия случай, този брой може да бъде намерен предварително или поне да бъде

оценен сравнително точно. Този брой по-нататък ще наречем *сложност на алгоритъма (по време)*. Не е изключено за една и съща задача да можем да посочим алгоритми със значително различаваща се сложност. Сложността на алгоритъма по време е една от най-важните му характеристики и затова е добре да можем от няколко налични алгоритъма да избираме този, който има най-малка сложност. За контраст, нека да споменем, че в математиката се срещат процедури, които отговарят на посочените в 1., 2. и 3. условия, но броят на стъките, за които завършва процедурата не може да бъде определен предварително. Ще различаваме такива процедури (да ги наречем *числени методи*) от алгоритмите и няма да се занимаваме с тях в тази книга.

5. Не случайно, едно от речниковите описания на понятието алгоритъм подчертава, че в алгоритмичните процедури някои последователности от операции могат да се изпълняват многократно. Такива повтарящи се последователности от операции в програмирането наричаме *цикли*. Много често създаването на алгоритъм се състои в определянето на подходяща последователност от операции и многократното ѝ повтаряне (над евентуално променящи се данни). Може да се каже, че организирането на цикли е същностна черта на процеса на създаване на алгоритми.

Разбира се, съществуват процедури, които много приличат на алгоритми, но не отговарят на някои от поставените по-горе условия. Ето част от описанието на една такава процедура, целта на която е да се приготви яденето пържени патладжани: „*Посолете нарязаните на тънки кръгчета патладжани, потопете ги в галета или брашно и ги изпържете в силно нагрята мазнина, до златисто оцветяване.*“ Такава процедура не може да бъде наречена алгоритъм по много причини. Дебелината на кръгчетата, количеството на солта и видът на панировката (галета или брашно) са оставени на предпочитанията на изпълняващия проедурата. Видът на мазнината и степента, до която да бъде нагрята – също. Не е фиксиран и краят на процедурата, защото различните изпълнители могат да интерпретират по различен начин думите „златисто оцветяване“.

Като пример за алгоритъм нека разгледаме процедурата за сравняване на две естествени числа, представени в p-ична позиционна бройна система. Основни обекти в такава процедура са цифрите на p-ичната позиционна бройна система. Операцията, от която имаме нужда е сравняването на две p-ични цифри a и b, което винаги завършва с един от трите възможни резултата – a е по-малка от b, a е равна на b или a е по-голяма от b. Вход за процедурата са две последователности от p-ични цифри, представящи двете зададени числа. Ако едното от двете числа

е по-дълго от другото (това се вижда веднага, ако ги запишем едно над друго), процедурата го обявява за по-голямо и завършва. Ако двете числа са еднакво дълги, процедурата сравнява (в цикъл) поредните цифри на двете числа отляво надясно. При първото несъвпадение, процедурата дава като изход резултата, получен от сравняването на двете цифри. Ако равенството продължи до последните (най-десните) цифри, тогава двете числа са равни. Тъй като единствената прилагана операция е добре дефинирана от подредбата на p-ичните цифри, резултатът от изпълнението на процедурата е предопределен и детерминиран. Броят на стъпките, които процедурата ще извърши в най-лошия случай, е равен на общата (в този случай) дължина на двете зададени числа. Цикличният характер на процедурата е очевиден.

Следващият пример е с историческо зачение, тъй като той е един от алгоритмите, описан в споменатия по-горе трактат на ал Хорезми (преводът сме направили със съвсем малки съкращения от съвременното издание на руски език – М. ал Хорезми, „*Математические трактаты*“, изд. Фан, Ташкент, 1983): „*Ако искаш да прибавиш число към число ... постави двете числа в два реда, т.е. едно под друго и нека бъде разрядът на единиците под разряда на единиците и разрядът на десетиците под разряда на десетиците. Ако искаш да събереш двете числа, то ще съберeш вски разряд със съответния му, който е над него, т.е. единиците с единиците, десетиците с десетиците. Ако в някой от разрядите ... се събере десет, тогава постави единица в следващия разряд, например ако имаш в единиците десет, то сложи единица в десетиците и там тя ще означава десет. Ако от числото е останало нещо или самото то е било по-малко от десет, остави го в същия разряд. Ако нищо не остане, постави в разряда кръгче (нула, б.а.), за да не остане разрядът празен ... и да се приеме вторият за първи и да се излъжеш в числото си. Точно така ще направиш с всичките разряди ...*“ Оставяме на читателя сам да оцени алгоритмичния характер на тази процедура, изобретена от древните индийци и достигнала до нас благодарение на ал Хорезми.

Като последен пример да разгледаме едно необичайно литературно произведение. По-долу с удоволствие цитираме текста на късия разказ „Указание за изкачване на стълба“ на видния аржентински писател Хулио Кортасар (текстът е взет от изданието Х. Кортасар, „*Междуетажие*“, Народна култура, София, 1985, превод Румен Стоянов):

„*Едва ли някой е пропуснал да забележи, че нерядко земята се нагрява по такъв начин, щото една част се качва под прав ъгъл спрямо земната плоскост и после следващата част застава успоредно на тази*

плоскост, за да даде път на нова отвесна – поведение, което се повтаря в спирала или в начупена линия до крайно променливи височини. Прикляквайки и поставяйки лявата ръка върху една от отвесните части, а дясната върху съответната водоравна, вие сте в моментно владение на едно стъпало или степен. Всяко едно от тия стъпала, образувано, както се вижда, от две съставки, е разположено малко по-нагоре и малко по-напред, отколкото предхождащото, ръководно начало, което осмисля стълбата, понеже всяко друго съчетание би произвело форми може би по-красиви и живописни, ала неспособни да пренасят от приземен на първия етаж.

Стълбите се изкачват с лице напред, тъй като заднишком или странишком се оказват особено неудобни. Естественото положение е да се държите на крака, ръцете увиснали без усилие, главата повдигната, макар не толкова, че очите да не виждат стъпалата, непосредствено над онова, върху което стъпвате, и да се диша бавно и равномерно. Изкачването на стълба се започва, като повдигнете тая част на тялото, разположена вдясно долу, обвита почти винаги в кожа или велур, и която, овен при изключения, се побира точно върху стъпалото. След като е поставена върху първото стъпало, въпросната част, която за по-кратко ще наречем ходило, свива се частта, равностойна на лявата (също наричана ходило, но която не трябва да се бърка с гореупоменатото ходило), и издигайки я до височината на ходилото, трябва да продължи, докато бъде поставена върху второто стъпало, при което на последното ще почине ходилото, а на първото ще почине ходилото. (Първите стъпала са винаги най-трудните, докато се придобие необходимата съгласуваност. Съвпадението на име между ходилото и ходилото затруднява обяснението. Внимавайте да не повдигате едновременно ходилото и ходилото.)

Щом стигнете по тоя начин на второто стъпало, достатъчно е да повтаряте последователно движенията, докато се озовете на края на стълбата. От нея се излиза лесно с лек удар на пета, който я закрепва на мястото ѝ, от което не ще мръдне до момента на спускането.“

Всичко, което споменахме по-горе за същността на понятието алгоритъм, е почувствано и отразено в този текст. И необходимостта от точно определяне на обектите и операциите (включително подчертаната опасност от неточни дефиниции), и крайността на процедурата и нейният цикличен характер. И това не е единственият „алгоритмичен“ текст на Х. Кортасар.

Тази книга е замислена като учебно пособие по (машинно зависима) теория на алгоритмите за студентите от информатичните специалности

на ФМИ на СУ и като такава се нуждае от известна историческа рамка. В цялата (повече от 40-годишна) история на преподаването на информатични дисциплини в Софийския университет, едва от учебната 2005/2006 година, в специалността „Компютърни науки“ на ФМИ се чете основен курс по алгоритми в началното (бакалавърско) ниво на обучение. През това време конкретни алгоритми са изучавани в различни дисциплни (най-вече *Увод в програмирането* и *Структури от данни*), но само в контекста на съответната дисциплина.

Затова не е учудващо, че в края на миналия и началото на настоящия век, в България няма сериозно издание в областта на алгоритмите. Не е учудващо също, че автори на първите по-сериозни текстове в областта бяха млади хора (в късна ученическа и студентска възраст), решили да поставят на хартия своя опит от участие в ученическите олимпиади по информатика. При цялото ни уважение към тези млади хора и към техния ентусиазъм, бихме искали да подчертаем, че, според нашето разбиране, нито ранното издание на Пр. Наков *Основи на компютърните алгоритми*, нито силно отличаващата се по стил *Програмиане = ++Алгоритми* на Пр. Наков и П. Добриков са учебници по теория на алгоритмите, без това да омаловажава ролята на тези книги.

В този дух бихме искали да споменем, че дори много популярната у нас в края на миналия век книга *Алгоритми + структури от данни = програми* на N. Wirth, според нас, не е учебник по алгоритми (близостта на едно от българските заглавия с това на книга на Wirth не е случайно). Ако трябва да се търси съвременен световен образец в областта, то това е книгата на T. Cormen, Ch. Leiserson и R. Rivest *Introduction to algorithms*. Голямото предизвикателство пред автора на този учебник беше, да се опита да преодолее влиянието на този фундаментален труд.

Книгата на T. Cormen, Ch. Leiserson и R. Rivest, обаче, не е единствената, оказала влияние върху разбиранията на автора. Преди нея си струва да бъдат споменати четири други книги, от които сме се повлияли сериозно. В началото беше руският превод на E. Reingold, J. Nievergelt и N. Deo *Combinatorial Algorithms: Theory and Practice*, после велколепните *The Design and Analisis of Computer Algorithms* и *Data Structures and Algorithms* (тук близостта на заглавието с това на някои от споменатите по-горе книги е подвеждаща) на A. Aho, J. Hopcroft и J. Ulmann и накрая – балансраната и изящна *Algoritmics: Theory and Practice* на G. Brassard и P. Bratley.

Що се отнася до книгите посветени специално на сложността на алгоритми, в световната литература също има безспорен образец и това е книгата на M. Garey и D. Johnson *Computers and Intractabillity: A Guide*

to the Theory of $\mathcal{NP}$-Completeness. Почти всичко, което съвременният информатик трябва да знае за предизвикателствата, решените и нерешени проблеми на теорията, може да бъде научено от тази „неувяхваща“ книга.

Съдържанието и структурата на този учебник следват, до голяма степен, една установена вече традиция. В началото се разглеждат необходимите за теорията и практиката ***формални модели*** на понятието алгоритъм. Очевидно е, че не може да се прилага математически апарат за изследване на такова сложно понятие, като неформалното „алгоритъм“, без да се използва формализация. Освен двете крайности – *машините на Тюринг*, които са най-удобни за теоретични разглеждания, заради простотата на модела и *езиците за програмиране*, които нямат алтернатива в практически аспект, е разгледан и един междинен модел – *машини с прозволен достъп до паметта*. Той е не само свързващо звено между двете крайности, но и сериозна методологическа основа, за по-доброто разбиране на понятията сложност по време и сложност по памет.

Втората глава е посветена на необходимия ***математически апарат*** – техниките за определяне на асимптотическото поведение на целочислени функции, решаване на рекурентни отношения, сумиране и т.н. От разглеждане са изключени важни за изложението теми, които са обект на основния курс по *Дискретна математика* (или алтернативната двойка курсове *Дискретни структури* и *Езици, автомати и изчислимост*) – дискретни множества, релации и функции (и по-специално булеви функции); елементарна комбинаторика; графи и дървета; формални езици и разпознаване. На читателите, които недостатъчно добре познават тази тематика, бихме препоръчали да се запознаят първо с един университетски учебник в областта (разбира се, че от терминологична и концепционална гледна точка, най-лесно това може да стане от учебника Кр. Манев, *Увод в дискретната математика*, КЛМН, София, 2005).

Ролята на ***абстрактните типове*** и ефективното им имплементиране в ***структури от данни*** е основен проблем на теорията на алгоритмите. Затова трета глава е посветена на някои от абстрактните типове данни, които по традиция не се разглеждат в основните програмистки курсове – пирамиди, разбиване, индексни дървета и т.н. Предполага се, че читателите добре познават класическите абстрактни типове - списъци, стекове, опашки, хеш-таблици и т.н. Затова в този учебник съвсем накратко се разглеждат някои статични и динамични имплементации на тези абстрактни типове и то от гледна точка на сложността на

алгоритми.

Четвърта глава е посветена на основни ***техники за разработване на алгоритми*** - „разделяй и владей“, динамично програмиране, „greedy“ и „bactrackig“. Владеенето на тези техники и умелото им използване при разработка на алгоритми за решаване на нови, несрещани още задачи, е ключов момент от алгоритмичната култура и професоналната подготовка на бъдещите програмисти. Пета, шеста и седма глава са посветени на ***алгоритмиката на графите, низовете и геометричните обекти***, области със своя специфика и голямо практическо приложение.

Последната глава е посветена на едно от най-големите предизвикателства на съвремнната дискретна математика – ***алгоритмичната сложност на задачи*** и по-специално на $\mathcal{NP}$-пълнотата на задачи и свързаните с нея проблеми.

Материалът в този учебник надхвърля по обем необходимото за обичайния бакалавърски курс и по-скоро е достатъчен за един основен и един допълнителен избираем курс по Алгоритми. За четенето на основен курс, бихме препоръчали следните раздели: .... Този списък може да се разглежда и като препоръчителен за читателите, които биха искали да се запознаят с материала самостоятелно. При наличие на интерес, главите пета, шеста и седма могат да бъдат основа за спец-курсове в съответните направления, но материалът от този учебник не е достатъчен за да обезпечи такива курсове. За курс по алгоритми в графи бихме препоръчали като допълнителна книга N. Christofides, *Algorithms in Graphs* (за съжаление книгата не е издавана отдавна и трудно се намира), за курс по алгоритми в низове – Dan Gusfield, *Algorithms in Strings, Trees and Sequences*, а за курс по алгоритмична (комбинаторна) геометрия – M. van Berg, O. Cheong, M. van Krevdel and M. Overmars, *Computational Geometry: Algorithms and Applications*. Тези книги могат да бъдат интересни и за слушателите в основния курс, които биха искали да разширят познанията си в съответната област или да разработват курсов проект.

Добавените в края на всяка глава задачи не са предназначени, а и не са достатъчни, за да осигурят упражненията на основен курс по Алгоритми и съответния практикум. Много задачи за математическите основи на курса могат да се вземат от книгата I. Parberry, *Problems on Algorithms*. Колкото до задачи за осигуряване на практикум по алгоритми, такива могат да бъдат намерени на многобройните сайтове в Интернет, където се съхраняват архиви на различни ученически и студентски състезания по програмиране. На сайта `infoman.musala.com`, например, могат да се намерят различни по трудност задачи, давани на ученичес-

ките състезания по програмиране в България през полседните години, както и връзки към други подобни сайтове.

Като всеки новосъздаден текст, и този не е съвършен. Затова с благодарност ще приемем (на адрес `manev@fmi.uni-sofia.bg`) всяка забележка или препоръка, целта на която е да се подобри съдържанието на книгата.

София, 2008 г.

Автора