ДИСКРЕТНА МАТЕМАТИКА И ОБУЧЕНИЕ ПО ИНФОРМАТИКА

Красимир Манев

ФМИ на СУ, бул. Дж. Баучер №5, 1164 София
manev@fmi.uni-sofia.bg

Резюме: *Съществува тенденция да се разглежда дискретната математика (ДМ) като екзотична и трудна част на математиката, и да се елиминира преподаването ѝ от учебните програми, включително и за паралелките с профил Информатика в българското училище. Тази тенденция е погрешна. В работата са посочени елементи от ДМ, които са важни за математическата култура на бъдещите програмисти и е обоснована необходимоста от познаването им. За целта е абсолютно необходимо включване на елементи от ДМ в учебните програми на българското училище.*

Ключови думи: *дискретна математика, обучение по информатика*

1. Въведение

В математиката наричаме *дискретни* обектите, които са съставени от отделни, ясно различими части, в противовес на *непрекъснатите* математически обекти. Това противопоставяне не е абсолютно, но е важно от методическа гледна точка. То подчертава наличието на съществени разлики между дискретните и непрекъснатите математически теории. Най-важната от тези разлики е, че дискретните математически теории са *конструктивни*. Огромна част от доказателствата на теореми за съществуване в тези теории съдържат *алгоритмични процедури* за построяване на обектите, чието съществуване се доказва. Затова, тази част на математиката е наричана още *конструктивна* или *алгоритмична математика*.

Дискретната математика е теоретична основа за изграждане на съвременните компютри, за разработване на езици за програмиране и транслатори, за построяване на бързи алгоритми. Тя предоставя формални механизми за моделиране на неформалните понятия *задача* и *алгоритъм,* за изследване на *алгоритмичната разрешимост* на задачите и *ефективността* на алгоритмите. И не на последно място, много задачи от живота могат да бъдат формулирани и решавани в термините на дискретните математически теории (например, теорията на графите, дискретната оптимизация, комбинаторната геометрия и т.н.). Затова, ДМ е математиката на Информатиката (Компютърните науки, Компютърното инженерство, Софтуерните технологии и Информационните системи) и нейните приложения (Информационните технологии).

Учените и преподавателите в областта на Информатиката и ИТ са приели този факт отдавна. Университетските програми по Информатика, традиционно, включват основни курсове по Дискретни структури, Езици, автомати и изчислимост, Дизайн и анализ на алгоритми, както и множество спец-курсове в областта. Много публикации от последните години на XX век очертават кръга от теми на ДМ, за които се смята, че трябва да бъдат изучавани систематично. Специално бихме искали да подчертаем изследванията на центъра DIMACS [1], съвместна инициатива на Rutgers University, Princeton University, Bell Labs, AT&T Labs, IBM, Microsoft и други американски институции в това направление и особено усилията на специалисти от този център за въвеждане на обучение по ДМ в средното училище [3].

За съжаление, нашият, повече от 20 годишен, опит от преподаване на ДМ показва, че повечето студенти постъпват в университетите без познания, дори и елементарни, по ДМ. Такива студенти завършват обучението си без да са овладели на необходимото ниво предвидения в университетските курсове материал и не са готови да решават, с помощта на компютърни програми, сериозни алгоритмични задачи. В тази статия ще се опитаме да обосновем колко важни са знанията по ДМ за оформянето на бъдещите специалисти, с цел да привлечем вниманието на колегията към **необходимостта от промяна на текущото състояние**. Ще демонстрираме важността на някои теории на ДМ за Информатиката и ще подчертаем вътрешната зависимост на съответните понятия и тяхната роля за повишаване на общата математическа култура на учениците.

В раздел 2 се дискутира мястото на ДМ в рамките на дисциплината математика. В раздел 3 са разгледани важни за Информатиката области на ДМ. За всяка от тези области са посочени фундаменталните понятия, които изграждат математическата култура на бъдещия програмист. В раздел 4 са направени някои изводи от изложеното в предния раздел и е оценен минималният хорариум, който ще бъде необходим за да се запълни съществуващата празнота. Поради ограничения обем на статията, не беше възможно да дефинираме всички дискутирани понятия. Съответните дефиниции могат да се намерят в [4].

2. Математика и дискретна математика

Основният проблем на обучението по ДМ е, че учебните програми по математика на българското училище, на практика, не включват преподаването на дискретни теории. В [2] дискутирахме в детайли тази негативна тенденция. Затова, тук ще се спрем само на няколко характерни примера.

Училищната математика използва понятието *множество* и алгебрата на операциите с множества (*обединение, сечение и допълнение* до някакво

универсално множество), както и понятието *съждение* и операциите на съждителната алгебра (*и, или* и *не*, съответно) интуитивно и, което е по-съществено, без дълбоко разбиране на ролята на тези основополагащи понятия и взаимната връзка между двете алгебри.

Операцията *декартово произведение* е непозната за учениците в средното училище. Без познаването на тази операция, обаче, е много трудно да се подчертае разликата между *наредените* (*вектори*, *редици*) и *ненаредените* (множества, *мултимножества*) математически обекти и да се обосноват комбинаторните принципи, използвани за намиране броя на такива обекти (комбинаториката също не се изучава в средното училище). Без познаване на понятието декартово произведение не може да се даде дефиниция на понятието *релация.* Училищната математика използва десетки релации (между числа или геометрични обекти) без да подчертава свойствата на тези релации (*рефлексивност*, *симетричност*, *антисиметричност*, *транзитивност*) или отсъствието на такива свойства. Заради това, в средното училище е невъзможно да бъдат въведени формално понятията *еквивалентност*, *наредба, граф* (в частност *дърво*), и най-вече *функция*, които са различни видове релации.

Математиката (не само училищната) не може да избегне употребата на понятията *функция* и *формула* (*израз*). В нито един училищен учебник, обаче, няма да намерим дефиниции на тези понятия. Дефиниция на понятието функция не може да бъде дадена, защото учениците не познават понятието релация, а на формула – защото не познават апарата на *формалните граматики,* понятието дърво или поне конструктивните техники *индукция* и *рекурсия*. Вместо да дефинира понятието функция в най-общ вид, училищната математика го замества на интуитивно ниво с понятието *функция на реална променлива* и абсолютно неправомерно отъждествява функция с някоя нейна формула. Така се създава превратна, и трудна за коригиране, представа за възможните видове функции. Не се осъзнава възможността една функция да има много или да няма нито една формула, както и същността на *еквивалентните преобразования* на формули.

В резултат, училищната математика лишава учениците от важни за тяхната математическа култура понятия и концепции, само защото са същинска част от ДМ. Обяснението, че ДМ не е необходима в училище, защото основите на математическото мислене и математическата култура се формират с изучаването на класическа (в огромната си част – непрекъсната) математика е несъстоятелно. Много от учениците с интерес в областта на математиката ще станат програмисти и класическата математика не е достатъчна за формирането на тяхната математическа култура. Ще покажем защо това е така.

3. Дискретна математика и информатика

3.1. Множества и булева алгебра

Без преувеличение може да се каже, че същност на работата на съвременните програмисти е да дефинират и имплементират алгебри в широкия смисъл на думата – множество от обекти и множество от операции над тях. Модерната парадигма – обектно-ориентираното програмиране – поощрява програмиста да дефинира *класове* (множества) от *обекти* и *методи* (*операции*) между обекти. Ето защо е необходимо бъдещият програмист да бъде запознат с *алгебричния* подход към обектите, с които работи – да може да дефинира обекти и операции с тях, да обосновава свойства на операциите, да строи правилни изрази (формули) в получената алгебра, да трансформира изрази в по-удобни еквивалентни изрази и т.н.

Класическата математика дава познания само за една алгебра – на реалните числа (и нейните подалгебри – на рационалните и целите числа, свойствата на които са подобни на свойствата на алгебрата на реалните числа). Много добра алтернатива, даже само за да се демонстрира наличието на алгебри с принципно различни свойства, е алгебрата на подмножествата на някакво универсално множество със споменатите операции обединение, сечение, допълнение (и други полезни операции – разлика, симетрична разлика и т.н.). Тази алгебра – наричана *булева* – е проста, дефинициите на операциите са лесни за разбиране и прилагане, а свойствата и се различават от тези на алгебрата на реалните числа. Имплементирането на клас Sets е чудесна възможност да се покаже, как може един и същ обект да се представи по различни начини в структури от данни и как може една и съща операция да се реализира с различни по качества алгоритми.

Но това не е единствената причина, поради която изучаването на алгебрата на подмножествата е важно. Съществуват други алгебри, които също са булеви, т.е. макар обектите с които оперират да са различни – има еднозначно съответсвие между операциите, при което се получава тотално съвпадение на свойствата. Доброто познаване на една булева алгебра позволява да се премине към коя да е друга булева алгебра без усилия. Една друга важна за програмистите булева алгебра, например, е споменатата вече алгебра на съжденията (в програмирането ги наричаме *логически изрази*) с операциите „и“ (and), „или“ (or), „не“ (not), „изключващо или“ (xor) и т.н. Тя е много важна за програмистите, защото езиците за програмиране използват логическите изрази за разклоняване на програмите и организиране на цикли. За още една важна булева алгебра – на *двоичните функции* – ще стане дума по-долу.

3.2. Декартово произведение

Декартовото произведение е една от най важните за информатиката операции с множества. Тя не е вътрешна операция на булевата алгебра на множествата. Декартовото произведение на две подмножества на някакво универсално множество не е подмножество на универсалното множество. René Descartes въвежда операцията за да постави геометричните разглеждания на алгебрична основа, създавайки по този начин *аналитичната геометрия.* Без декартовите координати не би било възможно не само създаването на съвременната компютърна визия – „лицето“ на Информационните технологии, но и управлението на компютърния монитор изобщо.

Ролята на декартовото произведение в Информатиката, обаче излиза далеч извън работата с декартови координати. Това е единствената операция, използвана в Информатиката, за структуриране на данни – в езиците за програмиране и в системите за управление на бази от данни. Например, всеки масив от тип `int` с дължина 5 е елемент на декартовото произведение

$$\texttt{int} \times \texttt{int} \times \texttt{int} \times \texttt{int} \times \texttt{int} = \texttt{int}^5,$$

двумерен масив от тип `int` с размери 5 на 5 – елемент на декартовото произведение

$$\texttt{int}^5 \times \texttt{int}^5 \times \texttt{int}^5 \times \texttt{int}^5 \times \texttt{int}^5 = (\texttt{int}^5)^5,$$

а *записът* на лице в релационна база от данни, съдържащ Идентификатор (число от тип `int`), Име (масив от 30 `char`) и Заплата (число от тип `double`) е елемент на декартовото произведение $\texttt{int} \times \texttt{char}^{30} \times \texttt{double}^5$.

3.3. Релации и графи

Релация е друго понятие на ДМ, което е важно за Информатиката. Всяка релация е подмножество на някакво декартово произведение. В най-общ вид понятието релация се използва главно в теорията на *релационните бази от данни* (РБД). Основните обекти на всяка РБД са релации, наричани още *таблици.* Всеки програмист, който работи с РБД трябва да познава добре операциите на релационната алгебра (още една важна, и доста различна от споменатите по-горе, алгебра) – *селекция*, *проекция*, *обединение* и т.н.

Много важни за Информатиката, както и за Математиката като цяло, са релациите над декартови квадрати и по специално *еквивалентностите* и *наредбите – пълни* или *непълни*. Всяка еквивалентност разбива множеството на релацията на *класове на еквивалентност*. За алгоритъм, изчерпващ пространството, в което е дефинирана еквивалентност, обикновено е достатъчно да разгледа само по един елемент от всеки клас на

еквивалентност. Така наличието на еквивалентност спомага за увеличаване на бързодействието на алгоритъма. Затова програмистът трябва да е в състояние да дефинира полезни еквивалентности.

Най-важното приложение на пълните наредби в информатиката е сортирането – на тази процедура можем да подлагаме само елементи на напълно подредени множества. Добре известно е, че сортирането е стъпка, която улеснява работата на други алгоритми и повишава бързодействието им. По-малко известен факт е, че елементите на непълна наредба също може да се подложат на сортиране (*топологическо сортиране*) и с това да улеснят следващи стъпки на алгоритми, работещи върху непълната наредба.

Графите са специален вид релации. На практика няма разлика между понятията релация и *ориентиран граф*, а *неориентираните графи* са симетрични и рефлексивни релации. Важни задачи от живота се моделират с графи, като теорията предоставя много алгоритми за решаването на такива задачи. Затова познаването на понятието граф и алгоритмите в графи е неизбежна част от културата на бъдещия програмист.

Дърветата, пък, са специален вид графи. Освен за решаване на задачи от живота, както беше при графите от общ тип, дърветата имат и друга важна за Информатиката роля. Представянето на данните в дървовидни структури (различни *пирамиди*, *балансирано дърво*, *индексно дърво* и т.н.) води до значително увеличаване на скоростта на много операции, в сравнение със случая, когато данните са представени в линейни структури (списък, опашка, стек). Компилаторите използват дървета за вътрешно представяне на програмните конструкции. Няма да преувеличим, ако кажем, че това е най-важното за Информатиката понятие на ДМ – с голямо теоретично, практическо и методологическо значение.

3.4. Функции

Понятието *функция* е фундаментално за Информатиката. Всъщност, всеки програмен модул пресмята стойностите на някаква функция. Училищната математика създава погрешната представа, че функция може да се дефинира само чрез формула (израз) от нейните аргументи. Ако за функцията, която трябва да се пресметне програмно, има такъв израз (независимо в каква алгебра), тогава създаването на съответна програма е елементарно. За алгебрата на реалните числа, например, програмните езици предлагат вграден инструмент – *аритметичния израз*.

Проблемът е, че за повечето функции няма израз, който да пресмята стойностите им. В дискретната математика дефинираме функцията f: $A \to B$, с *дефиниционна област* A и *ко-област* B, като релация $R_f \subseteq A \times B$ такава, че $\forall a \in A$ съществува не повече от едно $b \in B$ и a е в релация R_f с b. В общия

случай както *A,* така и *B*, може да не са числови множества, нито пък множества в известна алгебра и за формула, пресмятаща функцията не може да се говори. Нещо повече, възможно е *A* и *B* да са множества в известна алгебра, и въпреки това да няма формула, по която да се пресмята *f*.

Да разгледаме два примера на функции, които са необичайни за класическата математика. Вляво на Фиг. 1 е дефинирана *таблично* функцията

A	f_1
3	John
2	Mary
1	Peter
5	Johh
4	Mary
6	Peter

```
char y[10];
void f1(int x){ switch(x){
  case 1:
  case 6: strcpy(y,"Peter"); break;
  case 2:
  case 4: strcpy(y,"Mary"); break;
  case 3: case 5: strcpy(y,"John"); }
  }
```

Фиг. 1. Функция с нечислови стойности

f_1, с дефиниционна област A = {1, 2, 3, 4, 5, 6} и ко-област B = {Peter, Mary, John}. Всеки елемент x на A участва точно веднъж в лявата колона, а елементът в същия ред на дясната колона е стойността $f_1(x)$. Функцията f_1 не е с числови стойности и не може да бъде пресмятана с израз. Вдясно на Фиг. 1 е показан програмен модул, написан на езика C, който пресмята f_1. Няма нищо случайно в това, че в езика C такъв модул се нарича *функция*.

Като втори пример да разгледаме функцията на езика C от Фиг. 2, която трябва да пресмята квадратния корен на аргумента *x*. Тя работи коректно, ако *x* е точен квадрат. Ако *x* не е точен квадрат, тогава функцията попада в безкраен цикъл, никога не завършва работа и не дава резултат. Такава функция наричаме *частична*. Във функцията е реализирана *итеративна* процедура (цикъл) за пресмятане на стойността. *Итерацията* (както и *рекурсията*) са средствата на информатиката за пресмятане на функции, за които нямаме формули, а представянето им с таблици би довело до прекален разход на памет. При това функциите, дефинирани с итерационни процедури по принцип са частични – те може никога да не завършат изпълнението си, поради зацикляне или възникване на недопустимо действие (делене на нула, опит за промяна на ОП в забранен участък и т.н.).

```
int f2(int x){
   int s=0;
   while(x!=0){
      x=x-(2*s+1);
      s++;
   }
   return(s);
}
```

Фиг. 2. Частична функция

Дискретната математика се занимава основно с частични функции, дефиниционната област на които, ко-областта, или и двете, може да не са числови. Познанията, необходими за работа с такива функции се придобиват и при продължителни занимания с програмиране, но много по-добре е да се преподават систематично и целенасочено.

Специален клас функции, които са предмет на ДМ и са важни за информатиката са *двоичните* (б*улевите*) *функции*. Вече споменахме, че алгебрата на тези функции (операциите тук носят имената *дизюнкция*, *конюнкция* и *отрицание*) е булева алгебра и изучаването ѝ е много по-просто, когато обучаваните вече познават другите два примера за такива алгебри. Значението на двоичните функции за информатиката е огромно, защото с тяхна помощ се създават компютрите. Всяко отделно устройство на двоичния компютър е комплекс от няколко двоични функции на много (32, 64 и т.н, в зависимост от разрядността на компютъра) двоични променливи. Познаването на двоичните функции помага да се разбере в дълбочина устройството на компютъра и принципите на неговото функциониране.

3.5. Комбинаторика и сложност на алгоритми

Изброителната комбинаторика е измежду най-трудните дялове на ДМ. Нейната основна задача е намирането на броя на елементите на различни множества. Основното приложение на комбинаториката в работата на програмиста е при оценяването на *сложността* на използваните в програмите алгоритми – *по време* и *по памет*. Сложността по време на един алгоритъм, реализиран с компютърна програма, се измерва с броя на командите, които компютъра ще изпълни по време на работата на програмата. Оценката на сложността на алгоритмите по време е непроста комбинаторна задача и за нейното решаване е необходим опит. Пресмятайки сложността на алгоритъма, програмистът може да прецени дали програмата ще бъде в състояние да реши съответната задача за разумно време, от което пък до голяма степен се определя полезността ѝ. Когато за решаването на една и съща задача съществуват няколко различни алгоритъма, тогава основен критерий за избор на този от тях, който да бъде имплементиран, може да бъде по-добрата сложност.

Сложността на един алгоритъм по памет се измерва с обема (броя байтове) ОП, която алгоритъмът използва. Това също е не проста комбинаторна задача и нейното решаване, макар и не толкова критично, колкото оценката на сложността по време (предвид все по-големите размери на ОП), също е важно. В много случаи (при мобилни устройства, космически апарати и т.н.) всеки байт е изключително ценен и неразумното планиране на използването на паметта може да бъде фатално.

3.6. Формални езици, граматики и абстрактни машини

Теорията на формалните езици, граматиките и абстрактните математически машини също е от трудните дялове на ДМ. Нейният обект на изследване са *езиците за програмиране*, т.е. основният инструмент на програмиста. В този дял на ДМ се изследват възможностите за специфициране на формалните езици, с помощта на които програмираме компютрите и се разработват алгоритмите за транслиране (превод) от един формален език на друг. Най-подходящото средство за специфициране на *формален език* са *формалните граматики*. Теорията дефинира различни по сложност езици в зависимост от граматиките, които ги пораждат и изучава свойствата и възможностите на различните езици.

Познаването на апарата на формалните граматики е абсолютно задължително за всеки информатик, защото описанието на *синтаксиса* (правилата за написване на програма) и *семантиката* (смисълът вложен в отделните езикови конструкции) на езиците за програмиране се извършва в термините на формални граматики. Докато граматиките са идеалното средство за специфициране на език, те не са подходящи за решаването на другата задача – анализа на правилността на програмата и превода ѝ на друг език (обикновено на езика на компютъра). За решаването на тази задача се използват *абстрактните математически машини* (*автомати*). Теорията дефинира различни по сложност автомати, съответни на различните по сложност езици и предлага алгоритми, с помощтта на които от формалните граматики, специфициращи езиците, да бъдат построени автоматите, които да извършват анализа на програмите. Всъщност, се оказва, че ролята на тази теория на ДМ за Информатиката много по-голяма от решаването само на практическите задачи на специфицирането и транслацията на езици за програмиране.

Най-сложните възможни автомати – *машините на Тюринг*, съответни на формалните езици от общ тип, са най-подходящият математически модел на неформалното понятие алгоритъм. Така теорията на формалните езици се явява теоретичен инструмент за изследване на най-трудните от философска гледна точка въпроси на Информатиката – какво е алгоритъм, какви задачи можем да решаваме алгоритмически и можем ли да решим с алгоритъм (а следователно и с компютър) всяка смислено поставена задача.

4. Заключение

В този кратък текст се опитахме да посочим някои от най-важните за информатиката области на ДМ. Целта ни беше да покажем, че макар по произхода си и използваните инструменти ДМ да е част от математическата наука, от друга страна, тя е неразделно свързана, както с теорията, така и с

приложенията на Информатиката. Надяваме се, че с посочените аргументи сме убедили читателя, че е невъзможно професионалното изграждане на един информатик без солидна и задълбочена подготовка в областта на ДМ.

В редица дискусии за ролята на ДМ в образованието по информатика могат да се чуят възражения, че това е една сложна материя, която трудно се разбира от учениците и е по-добре изучаването ѝ да се отложи за по-късен етап, визирайки обучението в университет. Факт е обаче, че подобни възражения може да се чуят и в Университета, този път в смисъл дали не е по-добре да се отложи изучаването на ДМ за последните семестри.

Опитът показва друго. Учениците с интереси в областта на Информатиката започват отрано да се занимават с ДМ и без проблеми се справят с материала. По-късно, в Университета, те показват, средно-статистически разбира се, по-добри резултати от състудентите си, които за пръв път се срещат с ДМ едва на студентската скамейка. От тук и твърдата убеденост, че ДМ трябва час по скоро да намери мястото си в училищната програма. За профилираните паралелки по Информатика това е задължително, а за тези по Информационни технологии би било много полезно.

Прости пресмятания показват, че ако в четирите класа на гимназиалната степен се отделят по 5 учебни часа годишно и се обвърже преподаването на ДМ с подходящи теми от съществуващата учебна програма, това ще е достатъчно да се въведат необходимите знания за множества, релации и функции, както и да се направи кратко въведение в комбинаториката и теорията на графите. Ако подобни промени се направят и в прогимназиалния курс, завършващите средно образование ученици, особено тези от профил Информатика, ще имат напълно задоволителна подготовка в областта на ДМ.

Тази работа е частично финансирана от Фонд Научни изследвания на СУ „Св. Климент Охридски“ по Договор ???/2011 г.

Литература

1. Center for Discrete Mathematics & Theorethical Computer Science, http://dimax.rutgers.edu/.

2. Manev, Kr., Mathematics and Discrete Mathematics, in *Proceedings of Seventh International Conference on Discrete Mathematics and Applications*, June 2004, Bansko, Bulgaria, pp. 131-138.

3. Rosenstein, J.G., D.S. Franzblau and F.S. Roberts (Eds.), *Discrete Mathematics in the Schools*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, v. 37, 1997.

4. Манев, Кр., *Увод в дискретната математика*, Четвърто издание, КЛМН, София, 2006.