Състезателно програмиране

Николай Киров

21 май 2020 г.

Абстракт

Състезания по програмиране се провеждат от още времето на първите компютри, достъпни за ученици и студенти. Задачите, които се решават на тези състезания са основно алгоритмични. Често условията на задачите описват някаква псевдореалност, откъдето трябва да се извлече математическата (или информатичната) задача и да се намери подходящ алгоритъм за решаването ѝ. Преподаването на състезателно програмиране в университетите за студенти, които не са били състезатели като ученици, не е често срещано, но дава специфични знания и умения, които са важни за бъдещите програмисти.

В тази статия се разглежда преподавателския опит на автора в Нов български университет (НБУ), като са включени примерна тематична програма и задачи, давани на състезания с участници – студенти от курса по състезателно програмиране.

1 Въведение

Курсът „Състезателно програмиране" [37] подготвя студентите по Информатика за решаване на задачи, за които се изисква съставяне на по-сложни алгоритми. Изучава се целия път от поставяне на задачата до анализ на решенията – анализ на данните, преглед на съществуващи алгоритми за решаването ѝ, избиране на подходящ алгоритъм, модифициране на съществуващ или построяване на нов алгоритъм, избор на структури от данни, програмиране, тестване на програмата, анализ на ефективността на алгоритъма и програмата. Всички алгоритми се изучават заедно с най-ефективните им реализации като програми на C и C++.

Студентите са от втори курс, бакалавърски програми Информатика и Информационни технологии в Нов български университет (НБУ), като заедно с този курс изучават и основните курсове по структури от данни, алгоритми и графи. Изискването за курса по състезателно програмиране е студентите да могат да програмират на C или C++. От 2002/2003 учебна година курсът е включен в бакалавърските програми на НБУ с име „Програмиране за напреднали" [38] (едно- или двусеместриален), а от 2019/2020 учебна година е двусеместриален (2×30 учебни часа) с име „Състезателно програмиране" [37].

Редовните занятия включват преподаване на нови знания и обсъждане на задачите за самостоятелна работа. Всички разглеждани теми се представят с въвеждане или припомняне на понятията, обсъждане на конкретни алгоритми и понякога разглеждане на реализацията им като кодове на езика C++. Самостоятелната работа на студентите е организирана като домашни и състезания.

Домашните се състоят от две състезателни задачи и се дават регулярно по време на семестъра (по 16 задачи на семестър) със срок за предаване една седмица или малко повече. Алгоритмите за решаването им се обсъждат на занятията. Допустими са някои отклонения от правилата за състезателни задачи с цел даване на индивидуални задачи и генериране на по-големи входни данни (Раздел 5).

Състезанията се провеждат със системата Хакерранк (виж напр. [29]) за автоматична проверка на решенията. Разпределени са по 3 състезания на семестър с по 8-12 задачи [12]-[29]. На тези състезания студентът трябва да прояви умение да намери лесните задачи и тези, за решаването на които се използват обяснени на занятия алгоритми. Участието е индивидуално или в отбори по двама или трима студенти. При индивидуално участие решаването на 4 задачи е достатъчно за отлична оценка, а при отборното – малко повече (зависи от трудността на дадените задачи).

Курсове по състезателно програмиране се провеждат и в други университети по света [42]-[47], като учебните им програми са сходни. Държавният университет на Санкт Петербург (многократен победител на финалите на студентските състезания) предлага online курс Competitive Programmer's Core Skills [41]. Полезни дискусии по темата има напр. в Quora [48]. У нас се епизодично се провеждат курсове по състезателно програмиране за учители [50] и ученици [51], във връзка с ежегодните ученически състезания. Във Факултета по математика и информатика (ФМИ) на Софийски университет „Св. Климент Охридски" (СУ) имаше избираем курс „Проектиране и анализ на компютърни алгоритми" [49], който беше по същество курс по състезателно програмиране.

Системата за организиране на състезания по програмиране за ученици е добре развита и финансирана от много години [55], [52], [54], благодарение главно на Института по математика и информатика на БАН [56] и Съюза на математиците в България [57]. Добре известни са и успехите на учениците в международни състезания [58].

Републиканската студентска олимпиада по програмиране (РСОП, [4]-[6]) е най-голямото национално състезание за студенти. То е независимо състезание, организирано от участниците, като правилата за провеждането му се променят и утвърждават единствено от участващите в него. Те са максимално близки до тези на Международната студентска олимпиада по програмиране на ACM (ACM International Collegiate Programming Contest [1]). РСОП се провежда всяка година през месец май и се организира от някой български университет – домакин на олимпиадата. Състезанието е отборно. Участват студентски отбори от българските университети, където се изучава информатика и програмиране. Всеки участващ университет се представя от един или няколко отбора в състав: ръководител и трима състезатели. В официалното класиране на университетите участва най-добре представилият се отбор на съответния университет.

Историята и някои данни за РСОП през годините има на сайта на автора [2], на сайта на Мусала [3] и в публикациите [32], [31] и [30]. ФМИ на СУ е несъмнен фаворит в това състезание. Основна причина за тези успехи са студентите, които се били силни състезатели като ученици, а ученическите състезания по програмиране се провеждат у нас от много години на високо ниво (виж [52], [53]). В другите български университети ръководителите на студентски отбори за участие в РСОП трябва да организират подготовката им при условие, че студентите са без знания и опит по състезателно програмиране. С тази цел от 2009 година в НБУ функционира школа „Състезателно програмиране" [40]. На нея до голяма степен се дължат и успехите на отборите на НБУ за последните 10 години. На РСОП те завоюваха за НБУ (в класирането по университети) едно първо, пет втори и три трети места.

През последните години департамент „Информатика” на НБУ провежда междууниверситетско състезание през март-април, в НБУ и on-line с Хакерранк ([8]-[11]) по правилата на РСОП главно като подготовка на университетските отбори за Републиканската олимпиада през май.

2 Тематична програма за двусеместриален курс

1. Технология за състезателно програмиране
Стандарти на C и C++, компилатори. Стандартен вход и изход в C и C++. Пренасочване на входа и изхода. Примери за вход и изход за състезателни задачи.

2. Оценка и сложност на алгоритми
Размер на входните данни. Асимптотична нотация. Определяне на сложност на алгоритъм. Проблеми на асимптотичната нотация.

3. Структури от данни – STL
Контейнери, итератори, алгоритми.

4. Бройни системи. Прости и съвършени числа
Двоична, шестнадесетична и p-ична бройни системи. Проверка дали дадено число е просто. Мерсенови и съвършени числа.

5. Редица на Фибоначи. Най-голям общ делител, най-малко общо кратно. Рекурсия
Реализации на редицата на Фибоначи. Алгоритъм на Евклид за намиране на най-голям общ делител. Рекурсия и използване на променливите, глобални променливи.

6. Комбинаторни алгоритми. Разбиване на числа
Пермутации, вариации и комбинации (с повторения и без повторения) – брой и генериране. Триъгълник на Паскал. Разбиване на числа като сума от естествени числа и като сума от дадени числа.

7. Сортиране чрез сравнение и трансформация
Класификации на алгоритмите за сортиране. Сортиране чрез: броене, множество, пермутация. Бързо сортиране. Сортиране в STL.

8. Търсене. Търсене с връщане
Философия на търсенето. Линейно и двоично търсене. Класификация на задачите (сложност по време). NP задачи. Разходката на коня. Задача за осемте царици.

9. Разделяй и владей
Намиране на k-тия по големина елемент. Бързо умножение на дълги числа. Циклично преместване на елементите на масив.

10. Динамично оптимиране (4 занятия)
Класическа задача за раницата. Попълване на таблица. Най-дълга обща подредица. Хедонийски език. Формални системи. Задачи за монети. Домино подредица. Разстояние на Левенщайн.

11. Алчни алгоритми
Задачи за монети. Египетски дроби. Дробна задача за раница. Разходката на коня. Задания и крайни срокове.

12. Графи (3 занятия)
Понятия и дефиниции. Представяния на граф: списък на ребрата, матрица на съседство, списък на наследниците. Обхождане в ширина и обхождане в дълбочина. Компоненти

на свързаност. Минимален път. Най-дълъг път в ацикличен граф. Хамилтонов път и цикъл. Ойлеров цикъл. Максимален поток. Минимално покриващо дърво. Транзитивно затваряне и редукция. Топологично сортиране

13. Геометрични задачи
Основни понятия и формули. Изпъкнала обвивка.

14. Математически задачи
Полиноми. Производна и интеграл. Системи уравнения. Комплексни числа.

15. Компресиране
Общи понятия. Кодиране на редици с повтарящи се елементи. Кодиране на Хъфман. Код с разделители.

Всяка тема обикновено се разглежда на едно занятие от два учебни часа. Към тези теми се добавят по едно занятие за обсъждане на задачите, решенията и тестовите примери от състезанията (6 занятия).

3 Задачи за състезания

Възможни класификации на задачите за състезателно програмиране:
- по данните: низове, редици, геометрични фигури, графи, дървета, ...
- по най-ефективните алгоритми за решаването им: сортиране, търсене, динамично оптимиране, разделяй и владей, пълно изчерпване,...
- по трудност.

Всяка една класификация има своите недостатъци и при определяне коя задача къде да бъде класифицирана – има много задачи, които са еднакво отдалечени от няколко класа, или пък явно принадлежат на повече от един клас [53], [54]. Затова тук задачите са групирани (доста произволно) по начина им на описание, независимо от методите за решаването им и независимо от данните. Това групиране не е по-добро от всички останали, но не е и по-лошо, като дава възможност да се избират задачи с различни и интересни условия.

В едно състезание за студенти трябва да има баланс между лесни и трудни задачи, разнообразие от методи за решаването им, както и използване на различни структури от данни. Важно умение за състезателя е да може бързо да се ориентира в трудността на задачите и да избере да решава онези, които са по-лесни или за които вижда идея за решаването им. Често това е основна трудност при начинаещите състезатели, въпреки, че проверяващата система (Хакерранк) показва временното класиране, откъдето се вижда кои задачи са решени вече от най-много отбори (състезатели).

Условията на всички дадени задачи са минали през редакторската работа на автора. Някои от задачите са оригинални, но повечето са преформулировки (в една или друга степен) на повече или по-малко известни задачи, описани в различни книги и/или давани главно на ученически състезания.

Събрани тук, тези задачи могат да подпомогнат университетския преподавател в курса по състезателно програмиране, а също така и в курсовете по класическо програмиране.

3.1 Задачи с математически формулировки

Най-точно и най-ясно една състезателна цадача се фармулира с помощта на математическото ‘и представяне. Студентите по информатика обаче, често смятат такизва задачи

за по-трудни, може би защата разбирането на условието зависи от някои математически понятия, формули или зависимости. Разбира се, няма такава връзка и в този раздел са поставени различни по трудност задачи.

Задача: Най-близо [27]

Дадена е редица $a_1, a_2, ..., a_n$ от цели положителни числа. За дадено число x се намери такова a_i, че $|a_i - x|$ да е най-малкото възможно число.

Вход. На стандартния вход за всеки пример се четат две числа: n и m – броя на x-овете. Следват два реда с n и m числа съответно.

Ограничения. $n \leq 10^5, m \leq 100$; $1 \leq a_i \leq 10^8$, $-10^5 \leq x \leq 10^9$

Изход. За всеки пример на един ред на стандартния изход се отпечатва редица с m числа – най-близките числа в редицата. Ако $|a_i - x| = |a_{i+1} - x|$, да се отпечати по-голямата стойност.

Пример:

Вход.	*Изход.*

`4 7`
`4 2 9 7`
`4 2 9 1 8 5 3`
	`4 2 9 2 9 4 4`

Задача: Полином [28]

Да се намери най-малката и най-голямата стойност на полинома
$p_n(x) = a_0 + a_1x + a_2x^2 + ... + a_nx^n$
в интервала $[a, b]$ за цели стойности на аргумента.

Вход. За всеки пример на стандартния вход на един ред са дадени числата $a, b, a_1, a_2, ..., a_n$.

Ограничения. Числата от входа са цели и в интервала [1, 10^6]. $2 \leq n \leq 10$

Изход. За всеки пример на отделен ред на стандартния изход се изведат двете търсени числа.

Пример:

Вход.	*Изход.*
`0 10 3 2 1` `-10 5 12 -2 3 -4 1`	`3 123` `4 14332`

Задача: Периодични редици [11]

В теорията на дискретните динамични системи се разглежда следната ситуация: Дадена е функция $f : R \to R$. Орбита на дадена точка x се нарича безкрайната редица

$$x, f(x), f(f(x)), f(f(f(x))), \ldots.$$

Понякога орбитата на някоя точка x е n-периодична редица, т.е. редица от вида

$$a_1, a_2, \ldots, a_n, a_1, a_2, \ldots, a_n, \ldots.$$

Тогава точката x се нарича n-периодична точка. Изучавайки с помощта на компютър орбитите на точките, се налага по зададена крайна част от орбитата да се определи дали

точката е периодична и да се намери периодът n, ако тя наистина е периодична. Предполагаме, че зададената част от орбитата е достатъчно дълга редица и ако точката е n-периодична, редицата съдържа поне $2n$ члена. Да се напише програма, която по дадена крайна редица от цели числа да намери периодът n, ако редицата е част от орбитата на периодична точка.

Вход. Първото число от всеки пример е дължината на крайната редица, най-много 1000. На следващите редове са дадени членовете на редицата – цели числа, по-малки от 100, разделени с интервал или нов ред. Входът съдържа до 10 примери и завършва с числото 0 на последния ред.

Изход. За всеки пример на отделен ред се извежда цяло число – периодът n или 0, ако редицата не е част от орбитата на периодична точка. Тъй като са възможни много периоди (ако n е период, то и $2n$, $3n$, и т.н. също са периоди), извежда се дължината на най-малкия.

Пример:

Вход.	*Изход.*
4 2 2 2 2 8 1 2 1 2 1 2 1 1 14 2 3 1 2 3 1 2 3 1 2 3 1 2 3 0	1 0 3

Задача: Дълга редица [11]

Дадена е числова редица:
$f_1 = f_2 = 1$; $f_n = (af_{n-2} + bf_{n-1}) \bmod n$, за $n = 3, 4, ..., m$.
(mod означава остатък от целочислено деление).
При зададени положителни цели числа a и b, да се намери броя на различните числа в редицата.

Вход. За всеки тестов пример на стандартния вход са зададени числата a, b и m.

Ограничения. $1 \leq a, b \leq 100$; $3 \leq m \leq 10^8$

Изход. За всеки тестов пример на стандартния изход на отделен ред се отпечатва търсения брой.

Пример:

Вход.	*Изход.*
1 2 5 2 8 100	3 48

Задача: Многозначно изображение [6]

Дадено е многозначно изображение, дефинирано върху естествените числа $E : N \to 2^N$, като $E(x)$ е множеството от прости числа, които се получават като се добави една цифра (по-голяма от 0) отляво на числото x. Например $E(2)$ е празното множество, а $E(3) = \{13, 23, 43, 53, 73, 83\}$. Ако X е множество от естествени числа, с $E(X)$ ще означаваме множеството $\{E(x), x \in X\}$. Сега можем да дефинираме $E^{(2)}(x) = E(E(x))$ и

$E^{(k)}(x) = E(E^{(k-1)}(x))$ за $k = 3, 4, ...$ Да се напише програма, която при зададени x и k намира елементите на множеството $E^{(k)}(x)$.

Вход. За всеки тестов пример на стандартния вход на един ред се задават двете числа x и k.

Ограничения. $2 \leq x \leq 1000$; $2 \leq k \leq 15$

Изход. За всеки тестов пример на стандартния вход на един ред се отпечатват елементите на търсеното множество, наредени по големина – от най-малкия към най-големия. Ако множеството е празно, да се отпечати 0.

Пример:

Вход.	*Изход.*
`13 2`	`1613 2113 3313 3613 5113 6113 9613`
`22 10`	`0`

Задача: Маршрути [10]

Самолет излита от летище в точка $(0, 0, 0)$ и трябва да кацне на летище в точка $(x_m, y_m, 0)$. При излитане самолетът достига точка $(1, 0, 1)$ и за да кацне, трябва да се намира в точка $(x_m - 1, y_m, 1)$ (което се дължи на посоките на пистите в двете летища). Когато е във въздуха, самолетът има възможност да прелети от точка (x, y, z)
– на същата височина в една от точките: $(x + 1, y, z)$, $(x + 1, y + 1, z)$, $(x + 1, y - 1, z)$;
– да се издигне до точка $(x+1, y, z+1)$, $(x+1, y+1, z+1)$ или $(x+1, y-1, z+1)$ в случай, че $z \leq h$ (таван за летене);
– да се спусне до точка $(x + 1, y, z - 1)$, $(x + 1, y + 1, z - 1)$ или $(x + 1, y - 1, z - 1)$ в случай, че $z \geq 2$ (за да не се разбие в земята).
Няма ограничения за летене по x и y. Да се намери броя на маршрутите, по които самолетът може да осъществи този полет.

Вход. За всеки тестов пример на стандартния вход се задават три числа на един ред – x_m, y_m и h.

Ограничения. $0 \leq y_m \leq x_m \leq 10$; $0 < h \leq 10$

Изход. За всеки тестов пример на стандартния изход на отделен ред се отпечатва намерения брой маршрути.

Пример:

Вход.	*Изход.*
`4 0 1`	`3`
`4 1 5`	`4`

Задача: Разлики [23]

Дадена е числова редица $a_1, a_2, ..., a_n$ от която можем да получим нова редица $b_1, b_2, ..., b_m$ по следния начин: $b_1 = |a_1 - a_2|, b_2 = |a_3 - a_4|, b_3 = |a_5 - a_6|$, и т.н. Ако n четно число, $m = n/2$ и $b_m = |a_{n-1} - a_n|$. Ако n е нечетно число, тогава $b_m = a_n$. Напишете програма, която прилага описания алгоритъм няколко пъти, докато редицата остане само с един елемент.

Вход. На стандартния вход за всеки тестов пример се задава числото n и елементите на редицата – n цели положителни числа.

Ограничения. $0 < n < 1000$; $0 < a_i < 1000, i = 1, 2, ..., n$

Изход. За всеки тестов пример на отделен ред да се изведе стойността на единствения елемент на последната получена редица.

Пример:

Вход.	*Изход.*

`4`
`1 6 20 11`
`3`
`8 7 5`
	`4`
	`4`

Обяснение.

За първия пример редиците са:

```
1 6 20 10
5 9
4
```

За втория пример са:

```
8 7 5
1 5
4
```

Задача: Минимуми и максимуми на функция [21], [15], [13]

Дадена е функция f, дефинирана върху множеството от целите числа. Дефинираме минимум на такава функция в точка n, ако $f(n) < f(n-1)$ и $f(n) < f(n+1)$. Аналогично се дефинира и максимум на функцията. Да се намери броят на минимумите и максимумите на зададена с редица такава функция.

Вход. От стандартния вход трябва да се прочетат всички примери. Всеки пример започва с цяло положително число N – броя на точките, където е определена функцията. Следват стойностите на функцията в точките $n = 1, 2, 3, ..., N$.

Ограничения. $f : [1, 10000] \to [-1000, 1000]$

Изход. За всеки пример на стандартния изход на отделен ред да се изведат две числа – броя на минимумите и броя на максимумите на функцията, отделени с един интервал.

Пример:

Вход.	*Изход.*
`10` `3 2 3 2 2 7 8 2 9 1`	`2 3`

Задача: Еднакви елементи [21]

Дадена е редица от N цели неотрицателни числа. От текущата редица на всяка итерация получаваме нова, като $a'_k = |a_k - a_{k+1}|$, започвайки последователно от първия ѝ елемент и $a'_N = |a_N - a_1|$. Колко итерации са нужни, за да получим редица от еднакви елементи? За N = 4 и редицата `0 2 5 11` са нужни следните 8 итерации:

```
2 3 6 11
1 3 5 9
2 2 4 8
0 2 4 6
2 2 2 6
0 0 4 4
```

```
0 4 0 4
4 4 4 4
```

Вход. Всеки тестов пример се състои от един ред, на който с по един празен символ са разделени елементите на редицата.

Ограничения. $2 \leq N \leq 20$
Числата от входа са по-малки от 10000.

Изход. За всеки тестов пример, на отделен ред на стандартния изход изведете търсения брой итерации във формата, показан в примерите по-долу. Ако след 1000 итерации все още не се получава желаната редица от еднакви числа, изведете „not attained“ за съответния тест.

Пример:

Вход.	*Изход.*
`0 2 5 11`	`Case 1: 8 iterations`
`0 2 5 11 3`	`Case 2: not attained`
`0 4 0 4`	`Case 3: 1 iterations`
`300 8600 9000 4000`	`Case 4: 3 iterations`
`1 1 1`	`Case 5: 0 iterations`

Задача: 123456789 [16]

Да се намери броят на всички m-цифрени естествени числа, които са съставени от n_0 нули, n_1 единици, n_2 двойки, ... n_9 девятки в десетичния си запис, т.е. цифрата i се среща точно n_i пъти, $i = 0, 1, 2, \ldots, 9$.
Вход. За всеки тестов пример от стандартния вход се четат 10 цели неотрицателни числа – n_i, за които $0 < m = \sum_{i=0}^{9} n_i < 25$.
Изход. За всеки тестов пример на стандартния изход на нов ред се отпечатва резултата.

Пример:
Вход.

```
1 1 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 2 3
```

Изход.

```
4
10
```

3.2 Задачи без математика в условията

Условията на тези задачи са без математически означения – формули, променливи и числа. Тези данни често се задават когато се описва входа или изхода на програмата.

Задача: Квадратна честотна функция [16]

За даден низ, състоящ се от малки латински букви, дефинираме квадратна честотна функция като сума от квадратите на броя на срещанията на различните букви в низа. Дефинираме и операция смяна на буква – заместване на всички срещания на една буква с

друга буква. Да се намери максималната стойност на квадратната честотна функция за низа, получен при прилагане няколко пъти на тази операция.

Вход. За всеки тестов пример на стандартния вход на един ред се задава низ с дължина n и естествено число – броя k на прилагане на операцията смяна на буква.

Ограничения. $n, k < 100$
Изход. За всеки тестов пример на стандартния изход на отделен ред се отпечатва отговора.

Пример:

Вход.	*Изход.*
`alabala 1`	`37`

Задача: Робот [22]

Робот се движи в квадратчетата на квадратна мрежа, като може да прави ходове в съседно квадратче в две посоки — надясно и надолу. Във всяко квадратче на мрежата е зададено число – брой спечелени точки за робота, ако той премине през това квадратче. Задачата е да се намери маршрут на робота от най-горното ляво квадратче до най-долното дясно квадратче на мрежата с най-голям брой спечелени точки.

Вход. На стандартния вход са дадени: брой примери и за всеки пример – размер на квадратната матрица, определяща броя на спечелените точки във всяко квадратче и самата матрица.

Ограничения.
Размер на матрицата: $2 \leq n \leq 100$
Числа на матрицата: $0 \leq x \leq 20$

Изход. За всеки пример на отделен ред да се отпечати броя на спечелените от робота точки.

Пример:

Вход.	*Изход.*

`2`
`3`
`0 1 1`
`0 4 2`
`1 1 1`
`5`
`1 1 1 1 1`
`0 0 3 4 3`
`0 1 2 0 1`
`1 1 1 0 1`
`2 4 0 4 0`
	`8`
	`15`

Задача: Прозорец [22]

Плъзгащ се прозорец е подмасив с постоянен размер, който се движи отляво надясно през масив. За всяка позиция на прозореца искаме да изчислим някаква информация за елементите в прозореца. Задачата е поддържане на минимума на плъзгащия се прозорец, което означава, че трябва да се намира най-малката стойност във всеки прозорец.

Вход. На стандартния вход за всеки тестов пример се задават две числа – дължината на масива n и дължината на плъзгащия се прозорец m. Следват елементите на масива $a_1, a_2, ..., a_n$.

Ограничения.
$1 \leq m \leq n \leq 10^9$, $1 \leq a_i \leq 1000, i = 1, 2, ..., n$.

Изход. За всеки тестов пример на стандартния изход на отделен ред се извежда броя на различните най-малки стойности в плъзгащия се прозорец.

Пример:

Вход.	*Изход.*
`8 4` `2 1 4 5 3 4 1 2`	`2`

Обяснение:

```
2 1 4 5 -> 1
1 4 5 3 -> 1
4 5 3 4 -> 3
5 3 4 1 -> 1
3 4 1 2 -> 1
```

Задача: Пакети [11], [29]

По информационен канал се предават данни във вид на пакети, които трябва да бъдат обработени от вашата програма. Всеки пакет съдържа код и числови данни. Кодът с ключова дума, която определя каква операция трябва да бъде извършена с числовите данни – редица от цели числа.
Кодовете и операциите са:
`min` – най-малкото число;
`max` – най-голямото число;
`sum` – сумата на числата;
`num` – броя на числата.
От тези 4 кода се образуват още 8, като се добави суфикс `p` или `n`. Суфикс `p` в кода променя операцията, като от съответната операция се обработват само положителните числа в числовите данни, а суфикс `n` – само отрицателните.

Вход. На всеки отделен ред от стандартния вход е даден по един пакет: код и данни - редица от цели числа, разделени с интервали.

Ограничения. Числовите данни са цели числа в интервала $[-100, 100]$. Дължината на редицата, задаваща числовите данни в пакета, е число между 0 и 100.

Изход. За всеки пакет да се изведе на отделен ред на стандартния изход резултата от операцията. Резултатът от операцията върху празна редица е символът `*`.

Пример:

Вход. *Изход.*

```
sum 10 20                 30
min 3 1 7 -2              -2
max -9 -2 0 -2            0
num 0 0 0 0 0 0 0 0       8
sump 10 20 -1             30
minp 3 1 7 -2             1
maxp -9 -2 0 -2           *
nump 0 1 0 0 0 0 0 0      1
sumn 10 20                *
minn 3 1 7 -2             -2
maxn -9 -2 0 -2           -2
numn 0 0 1 0 0 0 0 0      *
sum                       *
min                       *
max                       *
num                       *
```

Задача: Просто сортиране [25]

Напишете програма, която сортира редица от цели положителни числа в нарастващ ред по броя на единиците в двоичното им представяне. Ако този брой е равен, то числата се сортират в намаляващ ред по тяхната стойност.

Вход. На всеки ред на стандартния вход е зададена редица от не повече от 10000 цели положителни числа, разделени с един или няколко интервала.

Изход. За всеки тест на отделен ред да се изведе редицата от числа, сортирана по искания начин. Числата трябва да са разделени с точно един интервал.

Пример:

Вход. *Изход.*

```
1 2 3 4 5 6 7             4 2 1 6 5 3 7
4 1 2 3 4 5 6 7           4 4 2 1 6 5 3 7
2 3 5 6 8                 8 2 6 5 3
2 4 5                     4 2 5
3 6 8 2                   8 2 6 3
1 2                       2 1
100                       100
```

Задача: Половинки [8]

Дадена е редица от N естествени числа. Операция half изважда от редицата най-големия елемент, дели го на две и връща в редицата двете получени числа. Задачата е да се намери най-голямото число, когато операцията half е приложена последователно n пъти.

Вход. На стандартния вход за всеки пример се задават: дължината на редицата N, числото n, елементите на редицата – цели положителни числа, не по-големи от 100.

Ограничения. $0 < N < 100$; $0 < n < 2000000$

Изход. Полученото най-голямо число като проста дроб (виж примера).

Пример:

Вход.	*Изход.*
3 2 1 2 3 2 1000000 2 3	3/2 2/262144

Задача: Три-ъгълници [11]

Ако се вземат 3 произволни естествени числа, те могат да бъдат или да не бъдат дължини на страни на правоъгълен триъгълник (напр. 3, 4, 5). Ако не могат, то тогава биха могли да са дължини на страни на остроъгълен триъгълник (напр. 1, 1, 1), или пък на тъпоъгълен триъгълник (напр. 5, 5, 9). Има още една възможност, тези числа да не могат да бъдат дължини на страни на триъгълник (напр. 1, 2, 4). Дадено е множество от естествени числа. Да се намери броя на правоъгълните, остроъгълните и тъпоъгълните триъгълници, чиито дължини на страни са различни елементи на множеството.

Вход. За всеки тестов пример на стандартния вход се задава броя на числата в множеството, а после и самите числа.

Ограничения. Всички числа от входа са по-малки от 1001.

Изход. За всеки тестов пример на отделен ред на стандартния изход се отпечатват три числа с разделител един интервал: броя на правоъгълните, остроъгълните и тъпоъгълните триъгълници, чиито дължини на страни са елементи на множеството.

Пример:

Вход.	*Изход.*
3 3 4 5 4 5 3 4 5	1 0 0 2 2 0

Обяснение:
За втория тестов пример триъгълниците са с дължини на страните: 5, 3, 4 и 3, 4, 5 – правоъгълни; 5, 3, 5 и 5, 4, 5 – остроъгълни.

3.3 Задачи с кратки условия

Предизвикателство за авторите на задачи е те да бъдат описани точно с колкото се може по-малко думи. Тези задачи изискват от състезателите внимателно да вникнат в съдържанието на всяка дума и дори препинателен знак. Тъй като те могат да бъдат от много лесни до много трудни, помагат на състезателите да преодолеят представата за зависимост между големината на условието и степента на трудност на една задача.

Задача: Правоъгълници [28]

Дадени са два правоъгълника със страни успоредни на координатните оси. Да се намерят: сумата от лицата им, лицето на обединението и сечението им.

Вход. На всеки от редовете на стандартния вход ще бъде зададена по една двойка правоъгълници с осем цели неотрицателни числа – координати на горния ляв и долния десен ъгъл на единия и другия правоъгълник.

Ограничения. Всички числа са по-малки или равни на 100.

Изход. За всяка двойка правоъгълници, на един ред на стандартния изход програмата трябва да изведе 3 числа — сумата от лицата на двата правоъгълника, лицето на обединението и лицето на сечението им. Пример:

Вход.	*Изход.*
0 1 1 0 0 2 2 0	5 4 1
0 1 1 0 2 1 3 0	2 2 0
0 2 3 1 1 3 2 0	6 5 1

Задача: Прости дроби [16]

Напишете програма, за събиране и изваждане на прости дроби, като представите резултата във вид на несъкратима дроб.

Стандартен вход: Всеки пример се задава със сума или разлика на няколко (най-малко 2 и не повече от 20) дроби и цели числа на отделен ред, като дробите са представени във вида m/n, където m и n са естествени числа. Всички числители, знаменатели и цели числа са по-малки от 10001. Входът съдържа няколко примери.

Стандартен изход: За всеки пример на изхода се записва резултата като несъкратимата дроб по същия начин, както зададените на входа дроби. Когато решението е цяло число, то се записва по нормалния начин.

Пример:

Вход.	*Изход.*
1/2 + 1/3	5/6
1/3 - 1/2 + 1	5/6
1/3 - 1/2 - 1/6	-1/3

Задача: Суми [16]

Дадено е мултимножество от цели числа. Напишете програма, която да брои колко различни суми може да се получат от събиране на двойки числа от множеството.
Вход. На стандартния вход се задават много тестови примери. Всеки пример се състои от число k – брой на елементите на множеството и още k числа за самите елементи.
Ограничения. $1 < k \leq 100$
Елементите на множеството са цели числа в интервала $[-1000, 1000]$
Изход. За всеки тестов пример на отделен ред се отпечатва отговора.

Пример:

Вход.	*Изход.*
3	1
1 1 1	5
4	
1 2 3 4	

Задача: Зигзаг [16]

Една крайна редица от цели числа се нарича зигзаг, ако всеки елемент на редицата (без първия и последния) е или по-голям от двата му съседа или по-малък от двата съседни елемента. Да се напише програма за определяне дали дадена редица е зигзаг.

Вход. На стандартния вход се задават числови редици – всяка на отделен ред с разделител един интервал между числата.

Ограничения. Всички числа се представят в типа `int` на C++.

Изход. За всяка редица се извежда на отделен ред `yes` за зигзаг и `no` – за редица, която не е зигзаг.

Пример:

Вход.	*Изход.*
`2 4 3 5 4 6`	`yes`
`10 2 9 4 2 5 4`	`no`

Задача: Прости числа [18]

Дадено е цяло число, по-голямо от две. Да се намери дължината на най-малкия затворен интервал с граници прости числа, който съдържа даденото число.

Вход. На стандартния вход е зададена редица от числа, по-големи от две и по-малки от един милиард.

Изход. На стандартния изход се записват последователно с разделител една шпация търсените дължини на интервалите.

Пример:

Вход.	*Изход.*
`10 5 34 18`	`5 1 7 3`

Задача: Търсене на сума [24]

Дадена е крайна числова редица. Да се намери дали дадено число може да се получи като сума от две члена на редицата.

Вход. Стандартния вход съдържа няколко теста по два реда: на първия ред е числовата редица, на втория са числата за проверка.

Ограничения. Дължината на редицата е от две до хиляда.
Числата в редицата са цели, неотрицателни и по-малки от хиляда.

Изход. За всеки тестов пример на отделен ред да се отпечати редица от yes/no в зависимост от отговора на поредния тест.

Пример:

Вход.	*Изход.*
`8 2 4 3 3`	`no no yes yes`
`8 3 10 7`	`yes`
`10 20`	
`30`	

Задача: Битове [12], [23]

Да се намери сумата от битовете на целите числа в даден затворен интервал. Битовете на числото са цифрите в двоичното му представяне.

Вход. За всеки пример на стандартния вход са дадени две естествени числа a и b, определящи интервала.

Ограничения. $0 < a \leq b < 18446744073709551615$, $b - a \leq 1000$

Изход. За всеки интервал на отделен ред се извежда търсената сума.

Пример:

Вход.	*Изход.*
`1 4`	`5`
`10 10`	`2`

Задача: Два цвята [22]

Даден е свързан неориентиран граф. Може ли да се оцветят върховете на графа в два цвята така, че да няма два съседни върха в един цвят?

Вход. На стандартния вход за всеки тестов пример е даден граф: брой ребра и двойки числа, задаващи ребрата на графа. Върховете са n на брой и са номерирани с числата от 1 до n.

Ограничения. $2 \leq n \leq 100$

Изход. За всеки тестов пример на стандартния изход да се отпечати YES или NO като отговор на поставения въпрос.

Пример:

Вход.	*Изход.*
`6`	`NO`
`1 2`	`YES`
`2 3`	
`1 4`	
`3 5`	
`2 5`	
`4 5`	
`2`	
`1 2`	

Задача: Низови интервали [11]

Дадени са два низа a и b, съдържащи малки букви от латинската азбука. Да се намери броят на низовете x със зададена дължина n, за които $a < x < b$.

Вход. За всеки тестов пример на един ред на стандартния вход се въвеждат двата низа a и b и числото n.

Ограничения. Дължината на низовете a и b е в интервала [1, 1000]. $1 \leq n \leq 1000$, $a < b$

Изход. На стандартния изход за всеки тестов пример на отделен ред да се изведе търсения брой по модул 26 (остатък от целочислено деление).

Пример:

Вход.	*Изход.*
`abc abcd 4`	`3`
`ab az 3`	`0`
`a b 1`	`0`

Задача: Общи цифри [10]

Дадено е множество от цели положителни числа. Да се намери броят на общите (различни) цифри на числата в множеството.

Вход. На стандартния вход за всеки тестов пример се задават броят на числата в множеството и самите числа.

Ограничения. Броят на числата в множеството е по-малък от 1000.
Числата са по-малки от 10^{100}.

Изход. За всеки тестов пример на отделен ред да се изведе търсения брой общи цифри.

Пример:

Вход.	*Изход.*

`3`
`112 111 9122221`
`4`
`444484144 1234567890 88111114 8421111134343434`
	`1`
	`3`

Задача: Максимална сума [10]

Да се намери максималната сума на m последователни члена на дадена редица от n цели числа.

Вход. На стандартния вход за всеки тестов пример се задават числата m и n и елементите на редицата $a_1, a_2, ..., a_n$ – цели числа.

Ограничения.
$0 < m \leq n \leq 1000000$; $-1000 \leq a_i \leq 1000, i = 1, 2, ..., n$

Изход. За всеки тестов пример на отделен ред да се изведе намерената максимална стойност.

Пример:

Вход.	*Изход.*

`2 4`
`1 6 20 11`
`3 3`
`8 7 5`
	`31`
	`20`

3.4 Задачи от политиката

Задача: Избори [21], [14]

На изборите за Народно събрание в парламента влизат партии, които са получили повече от 4% от действителните гласове на избирателите. Да се напише програма, която пресмята колко партии влизат в парламента при дадено разпределение на действителните гласове.

Вход. Няколко примера са зададени на стандартния вход. Всеки пример започва с числото n – брой на участващите в изборите партии и след него на нов ред n числа – брой на действителните гласове на партиите по реда на номерата на бюлетините.

Ограничения. Най-много 50 партии участват в изборите.

Има общо 6 милиона гласоподаватели.

Изход. За всеки пример на стандартния изход на отделен ред да се изведе броя на партиите, влизащи в парламента.

Пример:

Вход.	*Изход.*
4	3
100001 123456 2000000 12	1
2	2
100 4	
3	
100 50 0	

Задача: Коалиции [21], [14]

На изборите за Народно събрание n партии влизат в парламента. Да се напише програма, която определя всички възможни коалиции за образуване на правителство с изискването такава коалиция да има повече от половината депутати.

Вход. Няколко примера са зададени на стандартния вход. Всеки пример започва с числото n и след него на n реда са дадени име на партия (низ без интервали) и брой на депутатите на тази партия.

Ограничения. Най-много 10 партии влизат в парламента.
Общият брой на депутатите е най-много 500.

Изход. На стандартния изход да се изведат списъци от имена на партии, влизащи в коалицията за съставяне на правителство, всеки списък на отделен ред с по един интервал между имената на партиите. Редът на партиите в коалицията е по броя на депутатите в намаляващ ред. Ако в коалицията две партии имат еднакъв брой депутати, наредбата на тези две партии е лексикографска по имената на партиите.
Най-напред се отпечатват коалиции с най-малко партии. При еднакъв брой партии в две коалиции по-напред се отпечатва коалицията с повече депутати. Ако и броят на партиите и броят на депутатите в две коалиции са равни, наредбата е лексикографска на списъците на имената на партиите, както са подредени в коалицията.

Пример:

Вход.	*Изход.*
4	A D
A 10	A B
B 6	D B
C 3	A D B
D 10	A D C
2	A B C
par1 100	D B C
par2 400	A D B C
	par2
	par2 par1

Задача: Политическа сила [7], [27], [28]

Във всеки съюз или обединение от политически субекти (напр. държави) се налага да се приеме система за вземане на решения. Една такава система е да се гласува с “да” или “не”, като всяка държава да има определен брой гласове. Решение се взема когато броят на гласовете “да” е по-голям или равен на определена граница. Коалиция се нарича група държави, която гласува с “да” за дадено предложение. Ако сумата от гласовете на държавите в коалицията е по-голяма или равна на определената граница, то предложението се приема и тази коалиция се нарича печеливша. Например през 1958 г. се създава Европейския съюз с точно такава система за вземане на решения. Участващите в съюза държави и гласовете им са: Франция, Германия, Италия – по 4 гласа, Белгия, Холандия – по 2 гласа, Люксембург – 1 глас. Предложение се приема, ако за него са гласували с "да"12 от общо 17 гласа. Две печеливши коалиции в съюза са например Франция, Германия и Италия или Франция, Германия, Белгия, Холандия и Люксембург.
Една от няколкото известни мерки за политическата сила на дадена държава в един съюз е индексът на Шапли-Шубик. Ето как се дефинира този индекс: Нека съюзът се състои от държавите $p_1, p_2, \ldots, p_n$. Разглеждаме всички възможни наредби на тези n държави. Нека индексите $i_1, i_2, \ldots, i_n,\ 1 \leq i_j \leq n,\ j = 1, 2, \ldots, n$ задават една конкретна наредба. Държавата $p_{i_k},\ 1 \leq k \leq n$ се нарича централна за тази наредба, ако коалицията, състояща се от $p_{i_1}, p_{i_2}, \ldots, p_{i_{k-1}}$ не е печеливша, а коалицията $p_{i_1}, p_{i_2}, \ldots, p_{i_k}$ е печеливша. Индекс на Шапли-Шубик за държавата p се нарича отношението на броя на наредбите, в които p е централна към броя на всички възможни наредби. Да се напише програма за пресмятане на индекса на Шапли-Шубик.

Вход. Стандартният вход съдържа няколко тестови примера. Данните за всеки от примерите са записани на два последователни реда. Първият ред съдържа две цели числа, разделени с един интервал - броят n на държавите в съюза и необходимият брой гласове v за вземане на решение. На следващия ред има n цели положителни числа x_i по-малки от 100 (разделени с по един интервал), които са гласовете на участниците в съюза. Входът завършва с ред, съдържащ числото 0.

Ограничения. $3 \leq n \leq 20$; $x_i < v \leq \sum_{i=1}^{n} x_i$

Изход. За всеки пример на стандартния изход трябва да се изведат n числа, на един ред (с разделител един интервал), всяко равно на индекса на Шапли-Шубик, изразен в проценти за поредния участник в съюза. Числата да са с точност точно 1 цифра след десетичната точка.

Пример:

Вход.	*Изход.*

`3 51`
`50 49 1`
`6 12`
`4 4 4 2 2 1`
`0`
	`66.7 16.7 16.7`
	`23.3 23.3 23.3 15.0 15.0 0.0`

3.5 Задачи за Пешко, Тошко, Гошко и другите

В условието на тези задачи е развит някакъв сценарии и е дефиниран главен герой, често с име, който трябва да направи нещо или да реши дадена конкретна задача (виж напр. [53]).

Задача: Големи двоични числа [25]

Пешко от малък обича големите числа. Веднага, след като изучи двоичната бройна система забеляза, че числата изглеждат още по-големи в двоична система. Но пък има много нули, които Пешко мрази. Числата, съдържащи само единици в двоична система се намират лесно – те се получават по формулата $2^n - 1$. Предизвикателството за Пешко е да напише програма, която да брои колко единици има в двоичния запис на голямо число. Помогнете на Пешко.

Вход. За всеки пример на един ред на стандартния вход се задава числото. Входът съдържа много примери.

Ограничения. $10^{10} \leq n \leq 10^{100}$

Изход. За всеки пример от входа да се отпечати на стандартния изход на отделен ред търсения брой.

Пример:

Вход.	*Изход.*
`11111111111111111111111111111`	`48`
`12345678900`	`19`

Задача: Боядисване [5]

В града на децата всичко е шарено. И най-дългата улица трябва да се оцвети в 5 цвята. Поставили тази задача на изкуствения интелект (ИИ) Ганчо. Ето какво произвел Ганчо като алгоритъм за оцветяване:
1. На всеки метър на улицата записваме случайно число в интервала $[0, 9]$ (Ганчо участва във всички стъпки на оцветяването!).
2. Слагаме номера 1, 3, 5, 7, и 9 на кофите с различни на цвят бои.
3. Вземаме число $k = 1$.
4. Вземаме кофата с боя номер k и тръгваме по улицата, докато
– срещнем числото k и изпълняваме т. 5, или
– края на улицата и изпълняваме т. 7.
5. Започваме боядисване с боя номер k и боядисваме до достигане на
– числото $(k + 1) mod 10$ където спираме да боядисваме и изпълняваме т. 6, или
– края на улицата и изпълняваме т. 7.
6. Продължаваме по улицата до достигане на
– числото k и изпълняваме т. 5 или
– края на улицата и изпълняваме т. 7.
7. Когато е достигнат края на улицата, увеличаваме числото k с две. Ако
– $k > 9$, обявяваме край на боядисването,
– $k \leq 9$ връщаме се в началото на улицата и изпълняваме т. 8.
8. Изпълняваме отново т. 4, 5, и 6. Ако срещнем боядисан участък, пропускаме го (не го боядисваме отново) и продължаваме с т.6.
Сложен алгоритъм, но ИИ е това, не е като естествения! Сега програмистката Ганка трябва да пресметне по колко боя от всеки цвят е необходима за боядисването на улицата. Естественият интелект на Ганка ѝ подсказва, че алгоритъмът на Ганчо не е най-добрият, защото с тези случайни числа улицата може да остане съвсем не боядисана. Помогнете на Ганка, като напишете програма за пресмятане на количествата боя, ако за един метър боядисване се изразходва един кг боя.

Вход. За всеки тестов пример на един ред на стандартния вход е зададена редицата случайни числа, записана на улицата за боядисване.

Ограничения. Дължината на улицата е цяло число в интервала $[2, 10^3]$

Изход. За всеки тестов пример на един ред на стандартния изход се отпечатват 5 числа с разделител един интервал – количествата боя от кофи с номера 1, 3, 5, 7 и 9, необходими за боядисване на улицата по алгоритъма на Ганчо.

Пример:

Вход.	*Изход.*
7 2 9 3 7 6 7 4 8 1 5 9 7 5 3 1	1 4 0 3 0 0 1 1 1 1

Задача: Пак низове [25]

Тошко обича низове. От каквото и да е. Любими са му тези от малки латински букви. А повторения не обича. Затова пише програма, която премахва всички повтарящи се букви от един низ, като оставя само първото появяване на буквата.

Вход. На стандартния вход са зададени много низове – всеки на отделен ред.

Ограничения. Дължините на низовете на надвишават 1000.

Изход. За всеки низ от входа да се отпечати конструирания по зададените правила низ на отделен ред.

Пример:

Вход.	*Изход.*
alabala	alb
abcdefghij	abcdefghij
heellooo	helo

Задача: Разузнавачи [25]

Гошко ще става войник. Цифровизацията на казармените дела е в пълен ход. Всеки войник има лични оценки за различните военни дейности. Един ден старшината строява взвода войници в една редица с цел да избере трима от тях, съседи в редицата, за разузнавателна тройка. Гошко трябва да напише програма за този избор, като старшината му дава разузнавателната оценка на всеки войник от редицата. И разбира се, иска най-силната тройка.

Вход. За всеки пример на един ред от стандартния вход се задава число n – дължината на редицата от войници. На следващия ред са записани n цели числа с разделител интервал – личните оценки по разузнаване на войниците в редицата. Входът съдържа много примери.

Ограничения. $3 \leq n \leq 10^{10}$
Личните оценки на войниците са в интервала $[-1000, 1000]$.

Изход. За всеки пример от входа на стандартния изход да се отпечати сумата от личните оценки по разузнаване на тримата избрани войници.

Пример:

Вход. *Изход.*

```
4                       44
12 10 22 11             -30
3
-20 20 -30
```

Задача: Ветроходец [24]

Ветроходец попаднал близо до остров в област на слаби, но постоянни ветрове, които сменяли посоката си с настъпване на деня и нощта. През деня духали към острова, а през нощта – в обратна посока. Всяка сутрин ветроходецът вдигал платната, а през нощта ги свалял. Така всеки ден успявал да се доближи с a мили до острова. За жалост през нощта обратно течение го връщало с b мили навътре в морето. Една сутрин GPS системата му отчела c мили разстоянието до острова. След колко дни ветроходецът ще достигне острова?

Вход. На стандартния вход за всеки тестов пример на един ред се задават числата a, b и c.

Ограничения. $0 \leq a \leq 100$; $0 \leq b \leq 100$; $0 \leq c \leq 100$

Изход. За всеки тестов пример на отделен ред да се изведе 0, ако ветроходецът ще достигне острова същия ден; 1, ако ще достигне острова на следващия ден и т.н. Да се изведе -1, ако при дадените входни данни ветроходецът никога няма да достигне острова.

Пример:

Вход. *Изход.*

```
10 5 8                  0
10 5 14                 1
10 11 12                -1
```

Задача: Иван-юнака и змея [26]

Иван-юнака е героят на нашата компютърна игра и трябва да се срещне със змея и да го убие. Но преди това Иван-юнака трябва да премине през много препятствия. Змеят също гори от желание да се срещне с Иван-юнака, но и по неговия път трябва да има достатъчно препятствия. На Иван-програмиста е поставена задача да подреди препятствията в компютърната игра така, че Иван-юнака да срещне змея възможно най-късно, като се знаят времената за преминаване през всяко препятствие от Иван-юнака и змея. Срещата ще стане на някое от препятствията.

Вход. За всеки пример от стандартния вход се въвежда цяло число N – броя на препятствията. Всеки от следващите N реда съдържа по две положителни (цели или дробни) числа a_i и b_i – времената на Иван-юнака и змея за преминаване на поредното препятствие, разделени с по един интервал.

Ограничения. $1 \leq N \leq 256$; $a_i \leq 10^6$; $b_i \leq 10^6$
Времената a_i и b_i са дадени като цели числа или като дробни числа с десетична точка, като дробната част съдържа до шест цифри.

Изход. За всеки пример на стандартния изход да се изведе едно число (с точност точно 3 цифри след десетичната точка), равно на максималното време, за което Иван-юнака и змея ще се срещнат.

Пример:

Вход.	*Изход.*
4 1 2 1 2 0.5 1.5 7 3.5	6.000

Задача: Автоматична проверка [10]

Студент по информатика на стаж в едно начално училище решил да направи нововъведение. Учениците по математика да си пишат домашните на текстов файл, който да му изпращат по е-мейла. И разбира се, той проверява изпратеното домашно с програма. Напишете такава програма за проверка на домашните на ученици, които могат за смятат с числата от 0 до 100 като прилагат четирите аритметични действия.

Вход. От стандартния вход се четат редовете на домашното, като е неизвестно колко реда има домашното. На всеки ред има по една задача, която се състои от число, аритметично действие, второ число, знак за равенство и трето число – резултатът от аритметичното действие с аргументи първите две числа. Означенията на аритметичните действия са стандартните: събиране `+`, изваждане `-`, умножение `x` (малка латинска буква) и деление `:` (двуеточие).

Изход. За всеки ред от домашното на отделен ред се извежда Correct или Incorrect в зависимост дали пресмятането е вярно или не.

Пример:

Вход.	*Изход.*
2+3=5	Correct
100-2=99	Incorrect
9x9=99	Incorrect
25:5=5	Correct

4 Задачи с тематични условия

През 2009 година актуална тема в обществото беше световната икономическа криза [39]. По този повод на поредното състезание по програмиране на НБУ бяха дадени задачи с тематични условия за кризата и за ИТ сектора, включващи реални данни, лек хумор и голяма доза иронична хиперболизация.

Задача А. Ипотеки на правоъгълни парцели

Преди финансовата криза много американци вземали кредити срещу ипотека на земя, която притежават по силата на закона за правоъгълното разпределение на фиксирани квадратни участъци. По този закон всеки притежава правоъгълен парцел със страни успоредни на направленията север-юг и изток-запад. Вярвайки в джентълменската дума на клиента и пазарната икономика, банките давали кредити само срещу декларация на клиента за притежаваната земя. В паниката на финансовата криза банките започнали да пресмятат загуби като за всеки двама клиенти искали да изчислят сумата от лицата на декларираните участъци, лицето на обединението и лицето на сечението им.

Необходимият софтуер за това трябвало да създаде великата фирма от ИТ сектора МПМ — МикроПравоъгълникоМек (MRS -– MicroRectangleSoft). И разбира се, шефът на фирмата Б.Г. (B.G.) възложил тази работа на Вас, като водещ програмист в БГ филиала на компанията.

Вход. На входа е дадена редица от двойки правоъгълници — осем цели неотрицателни числа (≤ 100), координати на горния ляв и долния десен ъгъл на единия и другия правоъгълник. Всяка двойка правоъгълници е на отделен ред.

Изход. За всяка двойка правоъгълници на един ред се извеждат 3 числа — сумата от лицата, лицето на обединението и лицето на сечението им.

Пример:

Вход.	*Изход.*
`0 1 1 0 0 2 2 0`	`5 4 1`
`0 1 1 0 2 1 3 0`	`2 2 0`
`0 2 3 1 1 3 2 0`	`6 5 1`

Задача B. Финансова пирамида

"70-годишният финансист Бърнард Мадоф, който до скоро беше един от най-уважаваните икономисти, както и председател на съвета на директорите на американската фондова борса Насдак е отмъкнал 50 милиарда долара от различни банки, съобщи Би Би Си. За целта той използвал финансова пирамидална структура, като успял да обере водещи световни банки в Европа, Азия и САЩ." Възможно ли е това да стане с проста финансова пирамида?

Отговор трябвало да даде великата фирма от ИТ сектора МРМ — МикроПирамидоМек (MPS -– MicroPyramidSoft). И разбира се, шефът на фирмата Б.Г. (B.G.) възложил тази работа на Вас, като водещ програмист в БГ филиала на компанията.

Математическият модел на проста финансова пирамида е следният: Първата година се събират A долара, а всяка следваща $k^n A$ долара, където $k > 1$, защото в края на всяка година се изплаща лихва $p\%$, $p \geq 20$ върху вложената сума. Лихвата е висока и е лесно да се убедят хората с пари да инвестират тук, като се твърди, че парите им се влагат в супер изгодни сделки. Всъщност само се събират пари и се изплащат редовно годишните лихви по тях. За колко години Бърнард Мадоф по тази схема е спечелил (отмъкнал) 50 милиарда долара?

Подсказка: В края на първата година има $a_1 = A - pA/100$ долара, в края на втората $a_2 = a_1 + kA - pa_1/100$, в края на третата $a_3 = a_2 + k^2 A - pa_2/100$ и т.н.

Вход. На входа е дадена редица от тройки числа A, k и p, където $0 < A < 1010$, $1 < k \leq 2$ и $20 \leq p < 30$ са числа с не повече от 5 цифри след десетичната точка.

Изход. За всяка тройка входни данни на отделен ред се извежда като цяло число броят на годините, за които се достига сумата 50 милиарда долара.

Пример:

Вход.	*Изход.*
`1000 1.5 20`	`44`
`10000 1.6 20`	`33`

Задача C. Азбука

След измамата на Бърнард Мадоф по време на финансовата криза от 2008 година, световните лидери засилили работата по следене (подслушване) на кодирани съобщения между частни лица. Като малък, но важен елемент от тази работа се оказва следната задача: Даден е низ с елементи букви от латинската азбука. Да се намери най-къс подниз, съдържащ всички букви от тази азбука.
Необходимият софтуер за това трябва да създаде водещата фирма от ИТ сектора МБМ — МикроБуквоМек (MLS — MicroLetterSoft). И разбира се, шефът на фирмата Б.Г. (B.G.) възложил тази работа на Вас, като водещ програмист в БГ филиала на компанията.

Вход. На системния вход се задават низове, съдържащи само малки латински букви. Всеки низ започва на отделен ред.

Изход. На системния изход се отпечатва най-късия подниз, съдържащ всички букви от латинската азбука. Ако има два такива низа с еднаква дължина, извежда се онзи, който е по-напред във входния низ. Ако няма се извежда числото 0.

Пример:
Вход.

```
aabcdefghijklmnaaaaaaaaaaaaaaaapqrstaaaaaaaaaaaaaaaauvwxyzo
abcdefghiiiiiiiiiabcjjjjjjjjdefghijkiiilmnopqrstuvwxyzabjhadsbcahds
```

Изход.

```
bcdefghijklmnaaaaaaaaaaaaaaaapqrstaaaaaaaaaaaaaaaauvwxyzo
efghijkiiilmnopqrstuvwxyzabjhadsbc
```

Задача D. Археологическо изследване

Археолози от 2409 година решили да направят анализ на финансовата криза, обхванала планетата Земя през 2008 година и по-специално състоянието на малката древна държава България по това време. За тази цел те искали да знаят колко пъти се срещат думите „финансова” и „криза” във файла `Всички публикации.2008.България`. Според системни изследвания на водещи научни институти това била трудна работа и решаването ѝ изисквало създаване на поне 2 суперкомютъра, 4 грида, 8 нови операционни системи, 16 нови езика за програмиране и т.н., включително и построяване на нова водно-термоядрена-квантово-позитронна-колайдерна електроцентрала. Обаче шефът на най-мощния и велик (единствен!) ИТ световен холдинг ММMТ – МакроМикроМекТвърд (MMSH -– MacroMicroSoftHard) Б.Г. (B.G.) измислил гениално решение -– да възложил тази работа на БГ филиала на компанията от началото на 21 век (тогава пътуването във времето не било проблем) и разбира се на Вас, като водещ програмист в този филиал.

Вход. На системния вход се задава файла с текст на български език.

Изход. На системния изход се отпечатват две числа — броят на думата финансова и броят на думата криза. Поради особеностите на българския език, да се броят и членуваните думи финансовата и кризата. Също така да се броят и думите, написани с главни букви или с първа главна буква.

Пример:

Вход. *Изход.*

```
Част от отделените 5 милиарда евро от ЕС за борба с финансовата криза ще бъде
предоставена на България за изграждането на т. нар. интеркънекшън - свързване
на българската газопреносна мрежа с мрежите на Гърция и Румъния.
```

Задача E. Дау Джонс

По време на финансова криза индустриалният индекс на нюйоркската борса Дау Джонс се променя интензивно в рамките на един ден, като на всеки две минути се прави запис на стойността му. В края на всеки ден се пресмятат 4 числа, свързани с промяната му през деня — минимална и максимална стойности, средна стойност и медиана. Медианата е средния елемент в сортирания масив от стойности. Борсата в Ню Йорк отваря в 9:30 часа и затваря в 16:00 часа.
Задачата за тези пресмятания е дадена на великата фирма от ИТ сектора МММСММ — МикроМинимумМаксимумСреденМедианенМек (MMMAMS – MicroMaximumMinimum-AverageMedianSoft). И разбира се, шефът на фирмата Б.Г. (B.G.) възложил тази работа на Вас, като водещ програмист в БГ филиала на компанията.

Вход. На входа са дадени n редици от дневни записи на индекса, като числото n е най-напред във входа. Всички стойности са с два знака след десетичната точка.

Изход. За всяка редица на отделен ред се извеждат четири числа – минимална и максимална стойности, средна стойност и медиана. Средната стойност да се закръглява с два знака след десетичната точка.

Пример:
Вход.

```
1
7041.67 7334.00 8169.24 8478.58 7962.64 8705.45 8281.27 8961.91 8902.53 7292.82
8421.16 8718.95 8447.26 7771.38 8869.12 8667.99 8673.64 8141.11 7253.68 8547.44
7662.57 7037.59 7723.41 8529.78 8040.42 8264.48 8446.05 8890.29 7370.50 8006.01
7393.48 8629.23 8931.08 7944.39 7626.23 8537.38 7118.82 7929.41 7833.15 7639.58
8021.45 7924.72 7270.29 7777.73 8097.12 8986.90 8161.36 7355.67 7941.24 8966.30
8107.91 7007.37 8457.87 8753.83 7945.09 7209.58 7413.68 7900.91 7762.55 8410.59
8624.37 8548.83 8595.41 8602.50 8348.99 8668.84 7281.34 7053.99 7418.38 7900.88
7127.67 8728.93 7813.14 7309.16 7935.51 7600.49 7519.56 7798.03 7224.08 8844.09
7523.87 8314.03 8448.00 8458.18 7580.96 7798.81 8589.98 7009.57 7190.57 8958.91
8815.88 8156.11 7202.34 7272.55 7328.46 7362.86 8881.98 7322.51 7021.99 8557.76
8892.89 8075.12 7600.10 8003.69 8002.85 7182.85 8088.26 7617.57 8832.32 7169.54
8721.89 8976.29 8441.12 7145.60 8718.53 7139.23 7279.96 7687.29 7549.37 8866.49
7488.82 7455.34 7114.01 8316.71 8786.63 7313.55 8185.53 7912.08 8646.82 7481.44
8196.22 8129.61 8535.50 8173.66 7044.59 7292.39 8186.13 7474.22 7168.18 7787.05
8958.91 7202.25 7477.14 8314.24
```

Изход.

```
7007.36 8986.89 7956.67 8807.20
```

Задача F. Петролоносач

Заради финансовата криза и рецесията, цената на петрола и продуктите, произвеждани от него падна значително. Относителният дял на превоза се увеличи и петролоносачите се налага да сменят обичайните си маршрути през световния океан с цел намаляване на транспортните разходи.
Петролоносач трябва да се придвижи от точка A в точка B, като A е горен ляв ъгъл, а B долен десен ъгъл на правоъгълник. В правоъгълника е зададена квадратна мрежа, като петролоносачът може да се премества от дадено квадратче (i, j) в съседно надясно $(i, j+1)$, надолу в $(i+1, j)$ или по диагонала в $(i+1, j+1)$. Всяко преместване струва 1000 долара. Тъй като в океана има острови и плитчини, през някои квадратчета петролоносачът не може да премине.
Задачата за най-евтин превоз е поставена на най-великата фирма от ИТ сектора МППМ – МикроПетролПревозМек (MPSS – MicroPetrolShippingSoft). И разбира се, шефът на фирмата Б.Г. (B.G.) възлага тази работа на Вас, като водещ програмист в БГ филиала на компанията.

Вход. Дадена матрица $m \times n$, като всеки елемент представя едно квадратче от маршрута $(m, n \leq 1000)$. Стойността на елемента на матрицата е 0, ако петролоносачът може да мине през това квадратче и 1, ако не може да мине. На входа са зададени k матрици, като най-напред е даден размера на матрицата, а после елементите по редове. Първото число на стандартния вход е k.

Изход. За всяка матрица на отделен ред се извежда число – минимална стойност на превоза. Ако няма път през островите и плитчините се извежда 0.

Пример:

Вход.	*Изход.*
2	3000
3 3	4000
0 0 0	
0 1 0	
0 0 0	
3 4	
0 0 1 0	
0 1 1 0	
0 0 0 0	

Задача G. Газовата криза

В добавка към световната финансова криза на някои европейски държави възникна и газова криза. Тя е свързана с доставки на газ за Европа от руската газова компания Газпром. Компанията е държавна собственост, но малък процент се притежава от частни лица. Интересно е да се пресметне тяхната печалба за последните 6 месеца, което разбира се ще направи известната ИТ фирма МГМ – МикроГазМек (MGM – MicroGasSoft) и разбира се с помощта на най-великия програмист, който държи да остане в анонимност. Ето публичните данни за задачата:

```
Tuesday, December 30, 2008 Gazprom Profit Surges 83\%
Simon Kennedy
MarketWatch Pulse
LONDON - Russia natural gas giant OAO Gazprom said Tuesday that its net profit for the s
```

```
months ended June 30 rose 83\% to 573.8 billion roubles ($19.4 billion) from 313.2 billi
Copyright 2008 MarketWatch, Inc.
```

Вход. На входа се задават имена на хора и проценти, както е дадено в примера и на сайта `http://en.wikipedia.org/wiki/Gazprom`.

Изход. На изхода се извеждат данните в същия формат, като вместо проценти има цели числа — печалбата в долари.

Пример:
Вход.

```
* Alexander Ananenkov - 0.00709654%
```

Изход.

```
* Alexander Ananenkov - 1376728
```

5 Задачи за домашно с персонализация и използване на генератор за случайни числа

Задача:

Напишете програма за обхождане на граф с n върха в ширина (BFS), като началният връх има номер $1 + f \bmod n$ където f е факултетният номер на студента.

Вход. На входа се задава най-напред броя на дъгите на свързан неориентиран граф, а после списък от тези дъги. Списъкът се състои от не повече от 1000 дъги, зададени с двойки номера на върхове. Върховете на графа са номерирани с последователни цели положителни числа, започвайки от 1.

Изход. Отпечатва се списък (редица) на един ред (с разделител един интервал) от номерата на върховете на графа, получени при обхождането му в ширина.

Примерен вход:

```
3
1 2
2 3
1 4
```

Задача:

Да се напише програма, която намира броя на простите числа в случайно генерирана редица.

Вход. На стандартния вход са зададени не повече от 100 примери. Всеки пример се определя с две положителни цели числа s и N на един ред. s определя числова редица (чрез `srand(s)`) с дължина N, която се генерира с `rand()%1000`.

Ограничения. s $\leq 10^3$, $N \leq 10^9$

Изход. Извежда се намерения брой – за всеки пример на отделен ред.

Пример:

Вход.	*Изход.*
`25 10`	`3`
`102 10000`	`1616`

Задача:

Да се напише програма, която намира n-тия член в сортирана (в нарастващ ред!) редица, чийто членове са елементите на случайно генерирана редица от m числа. Редицата се генерира с `rand()%k` с начална стойност `srand(FN%100)`, където `FN` е факултетният номер на студента.

Вход. Всеки тестов пример е зададен с 3 числа на отделен ред – m, n и `k`.

Ограничения. $0 < m \leq 10^7$, $0 < n \leq 10^2$, $n \leq m$, $0 < \texttt{k} \leq 10^7$.

Изход. За всеки тестов пример на отделен ред се извежда отговора.

Примерен вход:

```
100 100 100
1000 1 99
```

6 Заключение

Включването на избирателен курс по състезателно програмиране в бакалавърските програми дава допълнителни възможности за изява на добрите студенти и тези, за които програмирането е интересна, а не само задължителна за усвояване технология. Изборът на теми за този курс ще зависи само от преподавателя, а даденият тук учебен план може да бъде една отправна точка за този избор.

Проверяващата система Хакерранк дава възможност студентите да се потопят в атмосферата на истинско спортно състезание, вкъщи или в компютърната зала и да покажат на практика своите знания и умения в алгоритмите и програмирането.

Повечето дадени тук задачи са включени в едно или повече състезания, организирани от автора в Хакерранк както за студентите от курса по състезателно програмиране, така и за подготовка на университетските отбори за участие в РСОП. Те могат свободно да бъдат използвани за съставяне на нови състезания от всеки с регистрация в Хакерранк.

Надявам се да бъдат полезни на преподаватели и организатори на състезания по програмиране.

Литература

[1] ACM International Collegiate Programming Contest
`https://icpc.global/`

[2] Републиканска студентска олимпиада по програмиране (история)
`http://nikolay.kirov.be/rsop/`

[3] InfoMan – Състезания – Републиканска Студентска Олимпиада по Програмиране
https://infoman.musala.com/contests/?page=2

[4] XXV Републиканска студентска олимпиада по програмиране
http://bcpc.eu/XXV/

[5] XXXI Републиканска студентска олимпиада по програмиране
http://bcpc.eu/XXXI/index.html

[6] XXXII Републиканска студентска олимпиада по програмиране
https://www.hackerrank.com/openbcpc

[7] Пети междууниверситетски турнир по програмиране, БСУ-2002
http://nikolay.kirov.be/2002/5cp/cp.html

[8] 25 Years NBU Contest
https://www.hackerrank.com/25-years-nbu

[9] Third Interuniversity NBU Programming Contest
https://www.hackerrank.com/third-interuniversity-nbu-programming-contest

[10] Fourth Interuniversity NBU Programming Contest
https://www.hackerrank.com/fourth-interuniversity-nbu-programming-contest

[11] Fifth Interuniversity Programming Contest
https://www.hackerrank.com/fifth-interuniversity-nbu-programming-contest

[12] NBU December 2015 Programming Contest
https://www.hackerrank.com/nbu-december-contest

[13] NBU Christmas Programming Contest 2016
https://www.hackerrank.com/nbu-christmas-programming-contest-2016

[14] NBU March 2017 Programming Contest
https://www.hackerrank.com/nbu-march-2017-programming-contest

[15] NBU November 2017 Programming Contest
https://www.hackerrank.com/nbu-november-2017-programming-contest

[16] NBU December 2017 Programming Contest
https://www.hackerrank.com/contests/nbu-december-programming-contest

[17] NBU January 2018 Programming Contest
https://www.hackerrank.com/nbu-january-2018-programming-contest

[18] NBU March 2018 Programming Contest
https://www.hackerrank.com/nbu-march-2018-programming-contest

[19] NBU April 2018 Programming Contest
https://www.hackerrank.com/nbu-april-2018-programming-contest

[20] NBU April 2018 Programming Contest II
https://www.hackerrank.com/nbu-april-2018-programming-contest-ii

[21] NBU November 2018 Programming Contest
https://www.hackerrank.com/nbu-november-2018-programming-contest

[22] NBU December 2018 Programming Contest
https://www.hackerrank.com/nbu-december-2018-programming-contest

[23] NBU January 2019 Programming Contest
https://www.hackerrank.com/nbu-january-2019-programming-contest

[24] NBU March 2019 Programming Contest
https://www.hackerrank.com/nbu-march-2019-programming-contest

[25] NBU October 2019 Programming Contest
https://www.hackerrank.com/nbu-october-2019-programming-contest

[26] NBU November 2019 Programming Contest
https://www.hackerrank.com/nbu-november-2019-programming-contest

[27] NBU January 2020 Programming Contest
https://www.hackerrank.com/nbu-january-2020-programming-contest

[28] NBU March 2020 Programming Contest
https://www.hackerrank.com/nbu-march-2020-programming-contest

[29] NBU May 2020 Programming Contest
https://www.hackerrank.com/nbu-may-2020-programming-contest

[30] N. Kirov, The Bulgarian Collegiate Programming Contest, Proceedings of the 12th Annual International Conference on Computer Science and Education in Computer Science, July 1-2 in Fulda, and July 3-4 in Nuremberg, Germany, ISSN 1313-8624 (2016), 325-333.

[31] Н. Киров, Републиканска студентска олимпиада по програмиране, Сборник доклади на Национална конференция по информатика посветена на 80-годишнината от рождението на проф. Петър Бърнев, 12-13 ноември, 2015, София, 87-92.

[32] Г. Атанасова, М. Димитров, П. Христова, Характерни особености на състезателното програмиране, НАУЧНИ ТРУДОВЕ НА РУСЕНСКИЯ УНИВЕРСИТЕТ – 2015, т. 54, 156-160.

[33] П. Наков, П. Добриков, Програмиране = ++Алгоритми; TopTeam Co., 2003, 696 стр.
https://programirane.org/

[34] Antti Laaksonen, Competitive Programmer's Handbook, 2017
https://cses.fi/book.html

[35] F. Ernst, J. Moelands, S. Pieterse. Programming contest strategies. (Teamwork in Programming Contests: 3 * 1 = 4) XRDS: Crossroads, The ACM Magazine for Students, Vol. 3, November 1996.
https://dl.acm.org/doi/10.1145/332132.332139

[36] A. Bloomfield, B. Sotomayor. A Programming Contest Strategy Guide. SIGCSE'16: Proceedings of the 47th ACM Technical Symposium on Computing Science Education, February 2016, Pages 609–614.
https://dl.acm.org/doi/10.1145/2839509.2844632

[37] Сайт на курса „Състезателно програмиране" за 2019/2020 учебна година.
http://nikolay.kirov.be/2020/CSCB324/index.html – първа част
http://nikolay.kirov.be/2020/CSCB325/index.html – втора част

[38] Сайт на курса „Програмиране за напреднали"
http://nikolay.kirov.be/2003/PASA/pn.html – 2002/2003 учебна година
http://nikolay.kirov.be/2019/CSCB300/index.html – 2018/2019 учебна година

[39] Financial crisis of 2007–2008
https://en.wikipedia.org/wiki/Financial_crisis_of_2007%E2%80%932008

[40] Школа "Състезателно програмиране"
http://nikolay.kirov.be/2010/WCP/index.html – 2009/2010 учебна година
http://nikolay.kirov.be/2020/WCP/index.html – 2019/2020 учебна година

[41] Competitive Programmer's Core Skills
https://www.coursera.org/learn/competitive-programming-core-skills

[42] CC3036: Competitive Programming (Universidade do Porto)
https://www.dcc.fc.up.pt/~pribeiro/aulas/pc1920/

[43] 15-295: Competition Programming and Problem Solving (CMU, USA)
https://contest.cs.cmu.edu/295/

[44] CS104c: Competitive Programming (UTAustin, USA)
https://www.cs.utexas.edu/users/downing/cs104c/

[45] CS3233: Competitive Programming (NUS, Singapure)
https://www.comp.nus.edu.sg/~stevenha/cs3233.html

[46] CS 97SI: Introduction to Programming Contests (Stanford, USA)
http://web.stanford.edu/class/cs97si/

[47] T-414-ÁFLV: A Competitive Programming Course (Reykjavik University, Iceland)
https://algo.is/t-414-aflv-competitive-programming-course-2016/

[48] What is the best strategy to improve my skills in competitive programming in C++ in 2-3 months?
https://www.quora.com/What-is-the-best-strategy-to-improve-my-skills-in-competitive-programming-in-C++-in-2-3-months

[49] Проектиране и анализ на компютърни алгоритми
https://www.nakov.com/pranka/

[50] Е. Келеведжиев, Състезателно програмиране за преподаватели
http://vivacognita.org/ocs/course/view.php?id=4
http://vivacognita.org/ocs/course/view.php?id=7

[51] Е. Келеведжиев, З. Дженкова, АЛГОРИТМИ, ПРОГРАМИ И ЗАДАЧИ, „РЕГАЛИЯ 6”, 2005. http://www.math.bas.bg/~keleved/books/D/

[52] Сайт за националните състезания по информатика за ученици
http://www.math.bas.bg/infos/

[53] А. Георгиев, Сайт за алгоритми, състезателна информатика и програмиране ($infO(n)$).
http://www.informatika.bg/

[54] П. Петров, М. Шаламанов, Е. Келеведжиев, Т. Брънзов, Арена (Жив архив със задачи)
https://arena.infosbg.com/

[55] План-календар (за ученическите състезания по програмиране)
http://keleved.com/calendar.html

[56] Институт по математика и информакика на БАН
https://math.bas.bg/

[57] Съюз на математиците в България
http://www.math.bas.bg/smb/

[58] International Olympiad in Informatics
https://ioinformatics.org/

Доц. д-р Николай Киров
Департамент "Информатика"
Нов български университет
Бул. "Монтевидео"21
1618 София
E-mail: nkirov@nbu.bg

Competitive Programming

Abstract. Programming competitions have been held since the time of the first computers available to pupils and students. The tasks to be solved in these competitions are mainly algorithmic. Often the conditions of the problems describe some pseudo-reality, from which the mathematical (or computer) problem must be derived and a suitable algorithm must be found for solving the problem. Teaching Competitive Programming in universities for students who have not been competitors as students is not common, but provides specific knowledge and skills that are important for future programmers.
This article examines the teaching experience of the author at the New Bulgarian University, including a sample thematic program and tasks given in competitions with participants – students of the course in competitive programming.

Dr. Nikolay Kirov, Assoc. Prof.
Department of Computer Science
New Bulgarian University
21, Montevideo Blvd.
1618 Sofia, Bulgaria
E-mail: nkirov@nbu.bg