Задача А1. Стари карти

Картите, които не са в електронен вид вече се смятат за стари. Те съдържат вярна и може би пълна информация, но за какво може да се използва карта, която дори няма търсачка? А без автоматично намиране на пътища, картата е напълно неизползваема за съвременните потребители. За нещастие, старите карти представят данните в много неудобен вид на огромна картина, която е пълна с излишна и неизползваема информация. Единствената съществена информация са градовете и пътищата между тях. Известно е, че между всеки два града съществува или директен път, или път, минаващ през няколко други града. Всеки път има фиксирана дължина, като между два града може да има най-много един пряк път. Ако такъв съществува в едната посока, то съществува и пряк път със същата дължина в другата посока.

Единствената надеждна информация, която може да се вземе в електронен вид от стара карта, е таблица с минималните разстояния между всеки два града. Градовете са безкрайно малки и когато преминавате през град, се счита, че изминавате 0 км.

Входните данни започват с ред, съдържащ числото N – броя на градовете ($3 \leq N \leq 500$). На следващите N реда има по N числа. j-тото число на i-тия ред е разстоянието от град i до град j и нека го означим с Dij ($1 \leq i, j \leq$ N). Знаем, че $0 < Dij = Dji \leq 1000000$. Също така $Dii = 0$.

Вашата програма **MAPS** трябва да отпечата описание на всички преки пътища, съществуващи на картата, ако е известно, че техният брой е минимален. Описанието на всеки път трябва да бъде отпечатано на отделен ред и трябва да съдържа три числа, разделени с интервал – двата града, свързани с пътя и разстоянието между тях. Тъй като старата карта е надежден източник на информация – задачата винаги ще има решение.

Примерен вход 1:
```
3
0 10 20
10 0 30
20 30 0
```

Примерен изход 1:
```
1 2 10
1 3 20
```

Примерен вход 2:
```
3
0 10 20
10 0 25
20 25 0
```

Примерен изход 2:
```
1 2 10
1 3 20
2 3 25
```

Решение:

Ако разгледаме градовете като върхове, а пътищата като ребра, можем да решим задачата в термините на крайни неориентирани графи.

В задачата разполагаме с матрица на разстоянията между всяка двойка върхове в граф G, който обаче е неизвестен. Нека означим тази матрица с $D(G)$. От нея елементарно може да се построи пълен граф P, чиято матрица $D(P) = D(G)$, като между всеки два върха поставим ребро с дължина равна на разстоянието между тях в матрицата. Първо забелязваме, че G е подграф на P – т.е. те имат едни и същи върхове и ако едно ребро е от G, тогава то е и от P. Това е така, защото за ребро от G, (a, b), свързващо върха a с върха b е изпълнено, че разстоянието между двата върха е точно равно на дължината на това ребро и следователно това е ребро и от P (ако допуснем, че реброто е по-късо, то разстоянието би било по-малко, а ако допуснем, че реброто е по-дълго, тогава то би било излишно и това противоречи с минималния брой ребра в G).

Нека разгледаме алгоритъм на Дейкстра в пълния граф P с начален връх i. На всяка стъпка от алгоритъма избираме оня връх j, за който разстоянието от i до j е най-малко измежду разстоянията от i до върховете, които не са избирани досега. Заедно с такъв връх j, ние избираме и ребро, което е последното в най-късия път от i до j. Нека на всяка стъпка се стремим това ребро да е минимално, т.е. ако има няколко пътя до j, да избираме този, за който последното ребро е минимално. Съвкупностите от тези ребра образуват покриващо дърво на графа P, което ще означим с $D(i)$.

Твърдим, че графа G е обединение на дърветата $D(i)$ за всеки връх i.

Доказателство: Тъй като върховете на дърветата и G съвпадат, разглеждаме само ребрата. Нека реброто $r(a, b)$ е от $D(i)$ за някое i и дължината му е len. Тогава разстоянието между a и b е равно на len (дължината е по-малка или равна на len, но не може да е по-малка, тъй като пътят до b би бил по-къс, без да се включва $r(a, b)$). Тогава върховете a и b в G са свързани или с $r(a, b)$, или с някакъв друг път със същата дължина. Но ако са свързани с път, то последното ребро в този път е по-късо от $r(a, b)$, което е противоречие с минималността на последното ребро за пътя от i до b.

Нека $r(a, b)$ е ребро от G, с дължина $lenr$. Нека разгледаме $D(a)$. Реброто $r(a, b)$ или е ребро от $D(a)$, или в $D(a)$ има друг път между a и b. Нека допуснем, че има друг път a, $x1$, $x2$, …, b с дължина len. От това, че $D(a)$ е построено с алгоритъм за минимална дължина от a до всеки връх следва, че $len \leq lenr$. Но от първата част на доказателството всяко от ребрата $r(a, x1)$, $r(x1, x2)$, ... $r(xn, b)$ е от G. Тогава в G има път с дължина len ($len \leq lenr$), свързващ a и b. Тогава реброто $r(a, b)$ е излишно, защото не променя разстоянието между a и b в G, както и кое да е друго разстояние в G. Това е в противоречие с минималния брой ребра в G. Оттук следва, че $r(a, b)$ участва в $D(a)$.

С това доказахме, че че графът G е обединение на дърветата $D(i)$ за всеки връх i. В доказателството съществено използвахме и факта, че дължините на ребрата са положителни.

Задача А2. Цифри

Дадено е число N, записано в K-ична бройна система, $2 \leq K \leq 36$. Главните латински букви A, B, C, D, E, F,… означават цифри, съответно със стойности 10, 11, 12, 13, 14, 15, ... Дадена е и една цифра M (число между 0 и 9, или главна латинска буква, такава че цифрата която тя означава е по-малка от K). Да се напише програма **DIGITS**, която намира колко пъти цифрата M се използва за записването на всички числа от 1 до N в K-ична бройна система.

На първия ред от входния файл са записани числото R ($1 \leq R \leq 1000000$) (което задава дължината на числото N), цифрата M (записана с десетично число от 0 до $K - 1$) и числото K, всичките разделени с интервал. На следващия ред се намира числото N, записано в K-ична бройна система. То се състои от последователни цифри, без интервал между тях.

На единствения ред в изходния файл програмата трябва да изведе търсения брой, записан в K-ична бройна система.

В около 40% от тестовете M ще е равно на 0.

В около 20% от тестовете $R \leq 6$.

В около 50% от тестовете $R \leq 5000$.

Примерен вход:
```
6 11 36
GWFERZ
```
Съответният изход:
```
3CMATS
```

Примерен вход:
```
6 0 10
107351
```
Съответният изход:
```
49517
```

Задача А3. Американска рулетка

Студентите Тошо и Гошо дълго се чудили как да се присмеят на приятеля си Пешо хакера, който не знаел че 36, а не 49, е най-голямото число, на което може да спре топчето в Американската Рулетка. Накрая им дошла гениалната идея да играят следната игра. За целта нашите герои имат нужда от тесте от карти, в което да има по 4 карти с номера 1, 2, 3, 4, 5 и 6, т. е. тесте с 24 карти. Играта протича по следния начин. Разпръскват картите на една маса, така че да могат да виждат стойностите им и, редувайки се, избират по една карта от масата и я хвърлят на земята. Ако даден играч не може да избере карта, така че като я хвърли на земята, сумата от стойностите на всички карти на пода да е не повече от 49, то той губи.

Главната цел на Тошо и Гошо била те да играят играта докато Пешо влиза в стаята, той да ги попита за правилата и те да се пошегуват с него. Така и станало, но Пешо, понеже е хакер, има един скрит коз. Той би могъл с ваша помощ да покаже на Тошо и Гошо, че всъщност е много умен, като им каже кой може да спечели вече започната игра и кой е печелившият ход. Ако помогнете на Пешо да отговори бързо на горните въпроси, той ще възвърне уважението на своите приятели. За целта напишете програма **ROULETTE**, която да има следния вход и изход:

Вход. Влизайки в стаята, Пешо заварва една вече започната игра. Той ви подава на първия и единствен ред на стандартния вход картите на земята в реда, в който са били хвърлени. Например 335 значи, че първо Гошо е хвърлил 3, после Тошо е хвърлил 3 и накрая пак Гошо е хвърлил 5. Сега на ред е Тошо. Винаги първи играе Гошо.

Изход. На първия ред на стандартния изход се очаква вие да кажете кой ще победи, като изведете една от латинските букви ‘T’ или ‘G’, съответно за Тошо и Гошо. Също така трябва да изведете и каква карта трябва да изиграе този, който е наред, за да спечели, или 0, ако той няма никакъв шанс да спечели при оптимална игра на противника. Ако съществуват няколко печеливши хода, изведете този, при който се играе най-голяма карта.

Примери

Вход
```
355
```

Изход
```
T 6
```

Вход
```
44223553556
```

Изход
```
G 0
```

Вход
```
1221
```

Изход
```
G 6
```

Решение:

В задачата се иска да се реши една проста комбинаторна игра. Позиция в играта ще наричаме броя от останалите карти от всеки вид и това кой е наред. Тъй като за всеки вид карти може да имаме останали на масата карти цяло число между 0 и 4 в ключително, това прави 5 ситуации за всеки вид карти. Понеже имаме 6 вида карти, то значи можем да съпоставим на всяка ситуация от останали карти на масата число между 0 и $5^6-1 = 15624$, вкючително. Трябва да имаме и в предвид кой е на ход в момента и, след като имаме двама играчи, то всяка Позиция може да се кодира с число между 0 и 31249. Когато позициите на една игра се кодират с числа в достатъчно малкък интервал и можем да заделим масив с такава дължина, казваме, че играта е обхватна.

Играта, която играем, също така е ациклична. Тоест, от дадена Позиция след извършването на x ($x > 0$) хода, не можем да се върнем пак в същата Позиция, тъй като всеки ход намалява сумата от стойностите на всички карти на масата.

Печеливша Позиция в играта е такава Позиция, от която съществува ход, с който отиваме в губеща Позиция (тоест пращаме противника в тази губеща Позиция).

Губеща Позиция е такава от която или всички ходове ни водят до печеливша Позиция, или нямаме възможни ходове.

От направените дотук расъждения забелязваме, че можем да напишем рекурсивна процедура, която за дадена Позиция да сметне дали тя е печеливша или губеща. За целта трябва да знаем за всички позиции, до които се стига с едни ход, дали са губещи или печеливши, което можем да направим, като извикаме рекурсивната процедура за тях.

За да изведем кой е печелившият ход, трябва просто в масив да помним за всяка печеливша Позиция кой ход я превръща в губеща.