Universidade Federal do Espírito Santo
Departamento de Informática
Programa de Educação Tutorial – PET EngComp
E-mail: petengcomp@inf.ufes.br
Home-Page: www.inf.ufes.br/~pet
Tel. (27) 3335-2161

PROVA

Realização: Apoio: Patrocínio:

Suporte
Departamento de Informática

FACITEC

FCAA
FUNDAÇÃO CECILIANO ABEL DE ALMEIDA

Centro Tecnológico
UFES Departamento de Informática

PROBLEMA A
Máximo divisor comum

Problema elaborado pelo Prof. PhD. João Paulo Andrade Almeida DI – UFES

Faça um programa que calcule o máximo divisor comum entre dois números inteiros positivos (de até 20 dígitos cada).
Use a seguinte definição do máximo divisor comum da Wikipedia: "O máximo divisor comum ou MDC entre dois números inteiros a e b (frequentemente abreviada como mdc(a, b), mdc{a, b} ou (a, b)) é o maior número inteiro encontrado, que seja factor dos outros dois. Por exemplo, os divisores comuns de 12 e 18 são 1,2,3 e 6, logo mdc(12,18)=6."

A ENTRADA

A entrada consiste de linhas com dois números por linha. Os números são separados por um espaço em branco e cada número pode ter até 20 dígitos. Cada linha é terminada por um um caracter de retorno de linha. A entrada é terminada por uma linha que consiste em um único dígito zero (0), seguido por um caracter de retorno do linha.

A SAÍDA

A saída deve conter uma linha para cada linha da entrada. Cada linha da saída deve conter o máximo divisor comum relativo aos dois números de uma linha da entrada.

EXEMPLO DE ENTRADA

```
3 5
12 18
10 20
348 156
1200 1800
1200000000 1800000000
0
```

EXEMPLO DE SAÍDA

```
1
6
10
12
600
6000000
```

PROBLEMA B
Formigueiros

Problema elaborado pelo Prof. PhD. Patrícia Dockhorn Costa DI – UFES

Em uma fazenda há diversos formigueiros que podem ou não estar conectados uns aos outros através de túneis que possibilitam a passagem de formigas de um formigueiro para outro. Como a infestação de formigas atingiu um nível crítico, o fazendeiro resolveu tomar uma atitude e contra-atacar com veneno. Este veneno deixa as formigas frenéticas e faz com que elas se espalhem pelos túneis, carregando o veneno e eventualmente eliminado todas as formigas de todos formigueiros que estão conectados através dos túneis.

Por exemplo, considere os formigueiros identificados pelos números 1, 2, 3, 4 e 5. Os formigueiros 1 e 3 estão conectados diretamente por um túnel, assim como os formigueiros 3 e 4. Os formigueiros 2 e 5 não estão conectados entre si e a nenhum outro formigueiro. Se o veneno para formiga for colocado nos formigueiros 1 e 2, os formigueiros 1, 2, 3 e 4 serão contaminados pelo veneno; o formigueiro 5 não será contaminado.

Este problema recebe como entrada uma malha de formigueiros, os túneis que conectam os formigueiros e um conjunto de formigueiros contaminados com veneno. O objetivo do problema é identificar e retornar no arquivo de saída os formigueiros que **NÃO** serão contaminados pelo veneno.

A ENTRADA

A entrada consiste da malha de formigueiros (formigueiros e túneis) e de uma lista de formigueiros que estão contaminados. A primeira linha contém um número inteiro n (1<n<100) que determina o número de formigueiros da malha. Cada uma das n linhas seguintes contém um formigueiro e uma lista de formigueiros conectados. No exemplo de entrada a seguir, o formigueiro 1 está conectado ao formigueiro 3 e o formigueiro 3 está conectado aos formigueiros 1 e 4. O formigueiro 4 está conectado ao formigueiro 3. Os formigueiros 2 e 5 não estão conectados a outros formigueiros. A última linha da entrada consiste da lista de formigueiros contaminados (1 e 3). (Formigas trafegam nos túneis em ambas as direções.)

```
5
1 3
2
3 1 4
4 3
5
1 3
```

A SAÍDA
A saída consiste da lista de formigueiros que **NÃO** serão contaminados pelo veneno.
EXEMPLO DE ENTRADA
10 1 2 3 2 1 5 4 3 1 4 2 5 2 6 8 6 5 7 8 5 9 10 10 9 5
EXEMPLO DE SAÍDA
7 9 10

PROBLEMA C
Uma nova indústria em crescimento

Um biólogo experimentando com modificação de DNA das bactérias encontrou um caminho para tornar colônias bacterianas sensíveis à densidade de população circundantes. Alterando o DNA, ele é capaz de "programar" as bactérias para responderem às densidades variadas em sua vizinhança imediata.

O prato de cultura é um quadrado dividido em 400 pequenos quadrados (20 x 20). A população em cada pequeno quadrado é medida numa escala de quatro pontos (de 0 a 3). A informação de DNA é representada como uma matriz D, indexada de 0 a 15, de valores inteiros e é interpretada do seguinte modo:

Em qualquer determinado quadrado do prato de cultura, seja K a soma da densidade desse quadrado e as densidades dos quatro quadrados imediatamente à esquerda, à direita, acima e abaixo desse quadrado (quadrados fora do prato são considerados tendo densidade 0). Então no dia seguinte, a densidade desse quadrado irá alterar de acordo com D[K] (que pode ser positiva, negativa ou zero). A densidade total não pode, no entanto, exceder 3 nem cair abaixo de 0.

Agora, claramente, alguns programas de DNA levam todas as bactérias a morrer (por exemplo, [-3, -3, …, -3]). Outros resultam em imediatas explosões de população (por exemplo, [3,3,3, …, 3]), e outros não fazem nenhuma modificação (por exemplo, [0, 0, … 0]). O biólogo está interessado em como alguns dos programas de DNA menos óbvios podem se comportar.

Faça um programa para simular o crescimento da cultura, lendo o número de dias para ser simulado, as regras de DNA e a densidade populacional inicial do prato.

A ENTRADA

A entrada desse programa consiste em três partes:
1. A primeira linha irá conter um único inteiro indicando o número de dias para ser simulado.
2. A segunda linha irá conter a regra de DNA (D) com o 16 valores inteiros, ordenados de D[0] a D[15], separados uns dos outros por um ou mais espaços em branco. Cada inteiro vai estar no intervalo - 3…3, inclusive.
3. As vinte linhas restantes da entrada irão descrever a densidade da população inicial no prato de cultura. Cada linha descreve uma linha de quadrados no prato de cultura e irá conter 20 inteiros no intervalo de 0…3, separados uns dos outros por 1 ou mais espaços em branco.

A SAÍDA

O programa irá produzir exatamente 20 linhas de saída, descrevendo as densidades

populacionais no prato de cultura ao final da simulação. Cada linha representa uma linha de quadrados no prato de cultura e será constituída por 20 caracteres, mais o terminador de fim-de-linha habitual.

Cada caractere representará a densidade de população em um único quadrado do prato, do seguinte modo:

Caractere de densidade:
0 .
1 !
2 X
3 #

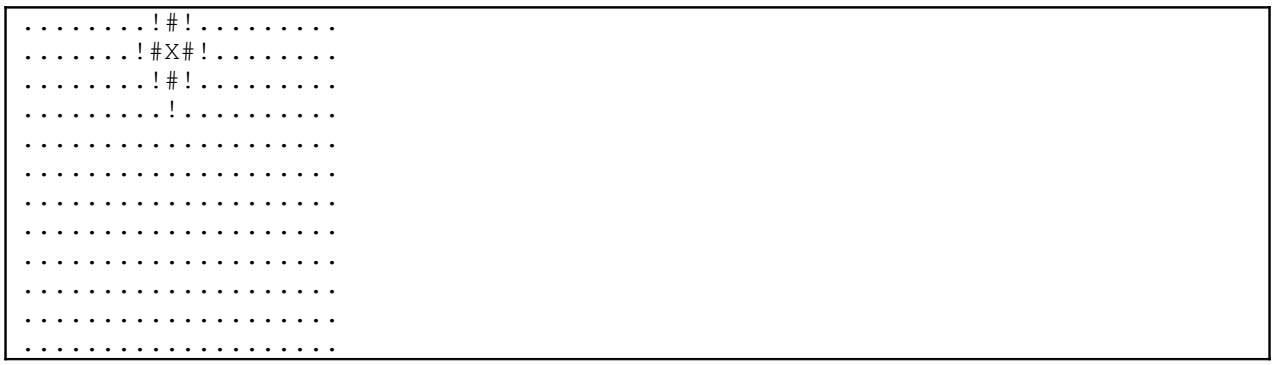
Outros caracteres não podem aparecer na saída.

EXEMPLO DE ENTRADA

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

EXEMPLO DE SAÍDA

```
##!.................
#!..................
!...................
....................
....................
....................
....................
.........!..........
```


.....!#!.....
.....!#X#!.....
.....!#!.....
.....!.....
.....
.....
.....
.....
.....
.....
.....
.....

PROBLEMA C
A New Growth Industry

A biologist experimenting with DNA modification of bacteria has found a way to make bacterial colonies sensitive to the surrounding population density. By changing the DNA, he is able to "program" the bacteria to respond to the varying densities in their immediate neighborhood.

The culture dish is a square, divided into 400 smaller squares (20x20). Population in each small square is measured on a four point scale (from 0 to 3). The DNA information isrepresented as an array D, indexed from 0 to 15, of integer values and is interpreted as follows:

In any given culture dish square, let K be the sum of that square's density and the densities of the four squares immediately to the left, right, above and below that square (squares outside the dish are considered to have density 0). Then, by the next day, that dish square's density will change by D[K] (which may be a positive, negative, or zero value). The total density cannot, however, exceed 3 nor drop below 0.

Now, clearly, some DNA programs cause all the bacteria to die off (e.g., [-3, -3, ..., -3]). Others result in immediate population explosions (e.g., [3,3,3, ..., 3]), and others are just plain boring (e.g., [0, 0, ... 0]). The biologist is interested in how some of the less obvious DNA programs might behave.

Write a program to simulate the culture growth, reading in the number of days to be simulated, the DNA rules, and the initial population densities of the dish.

A ENTRADA

Input to this program consists of three parts:

1. The first line will contain a single integer denoting the number of days to be simulated.
2. The second line will contain the DNA rule D as 16 integer values, ordered from D[0] to D[15], separated from one another by one or more blanks. Each integer will be in the range -3...3, inclusive.
3. The remaining twenty lines of input will describe the initial population density in the culture dish. Each line describes one row of squares in the culture dish, and will contain 20 integers in the range 0...3, separated from one another by 1 or more blanks.

A SAÍDA

The program will produce exactly 20 lines of output, describing the population densities in the culture dish at the end of the simulation. Each line represents a row of squares in

the culture dish, and will consist of 20 characters, plus the usual end-of-line terminator.

Each character will represent the population density at a single dish square, as follows:

Density	Character
0	.
1	!
2	X
3	#

No other characters may appear in the output.

EXEMPLO DE ENTRADA

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

EXEMPLO DE SAÍDA

```
##!.................
#!..................
!...................
....................
....................
....................
....................
.........!..........
........!#!.........
.......!#X#!........
........!#!.........
.........!..........
```

PROBLEMA D
Jogo De Palavras

Algumas portas secretas contém um enigma de palavras muito interessante. Uma equipe de arqueólogos tem de resolvê-lo para abrir a porta. Não há outra maneira de abrir as portas, por isso o enigma é muito importante para eles.

Existe um grande número de placas magnéticas em cada porta. Cada placa tem uma palavra escrita sobre ela. As placas devem ser dispostas em uma sequência de tal forma que cada palavra começa com a mesma letra que a palavra anterior termina. Por exemplo, a palavra "pet" pode ser seguida pela palavra "topcom". Sua tarefa é escrever um programa de computador que irá ler a lista de palavras e determinar se é possível organizar todas as placas em uma sequência (de acordo com o enigma das palavras) e, consequentemente, para abrir a porta.

A ENTRADA

A entrada consiste de T casos de teste. O número de testes (T) é indicado na primeira linha do arquivo de entrada. Cada caso teste começa com uma linha contendo um único número inteiro N que indica o número de placas (1 <= N <= 100000). Então as N linhas a seguir conterá uma única palavra cada. As palavra contém, no mínimo dois e no máximo 1000 caracteres minúsculos, o que significa que só letras de "a" até 'z' irão aparecer na palavra. A mesma palavra pode aparecer várias vezes na lista.

A SAÍDA

Seu programa tem de determinar se é possível organizar todas as placas em uma sequência tal que a primeira letra de cada palavra é igual a ultima letra da palavra anterior. Todas as placas a partir da lista devem ser utilizadas uma única vez. As palavras mencionadas devem ser utilizadas de acordo com o numero de vezes que ela aparece na lista de palavras especificada para cada teste.

Se existe uma ordenação possível das placas de acordo com a regra do enigma, para cada teste, o seu programa deverá imprimir a frase "`Ordering is possible.`", ou caso contrario, "`The door cannot be opened.`"

EXEMPLO DE ENTRADA

```
3
2
acm
```

```
ibm
3
acm
malform
mouse
2
ok
ok
```

EXEMPLO DE SAÍDA

```
The door cannot be opened.
Ordering is possible.
The door cannot be opened.
```

PROBLEMA D
Play on Words

Some of the secret doors contain a very interesting word puzzle. The team of archaeologists has to solve it to open that doors. Because there is no other way to open the doors, the puzzle is very important for us.

There is a large number of magnetic plates on every door. Every plate has one word written on it. The plates must be arranged into a sequence in such a way that every word begins with the same letter as the previous word ends. For example, the word ``acm'' can be followed by the word ``motorola''. Your task is to write a computer program that will read the list of words and determine whether it is possible to arrange all of the plates in a sequence (according to the given rule) and consequently to open the door.

A ENTRADA

The input consists of *T* test cases. The number of them (*T*) is given on the first line of the input file. Each test case begins with a line containing a single integer number *N*that indicates the number of plates (*1 <= N <= 100000*). Then exactly *N*lines follow, each containing a single word. Each word contains at least two and at most 1000 lowercase characters, that means only letters 'a' through 'z' will appear in the word. The same word may appear several times in the list.

A SAÍDA

Your program has to determine whether it is possible to arrange all the plates in a sequence such that the first letter of each word is equal to the last letter of the previous word. All the plates from the list must be used, each exactly once. The words mentioned several times must be used that number of times.

If there exists such an ordering of plates, your program should print the sentence "`Ordering is possible.`". Otherwise, output the sentence "`The door cannot be opened.`".

EXEMPLO DE ENTRADA

```
3
2
acm
```

```
ibm
3
acm
malform
mouse
2
ok
ok
```

EXEMPLO DE SAÍDA

```
The door cannot be opened.
Ordering is possible.
The door cannot be opened.
```

PROBLEMA E
SUS

Este problema foi elaborado pela Prof.ª Dr.ª Maria Claudia Silva Boeres (DI/UFES - ES) e pela Prof.ª Dr.ª Maria Cristina Rangel (DI/UFES – ES).

A Secretaria de Saúde de uma cidade necessita estabelecer dois pontos onde serão construídos dois Prontos Atendimentos (PA) com ambulâncias para o atendimento de chamados de emergência. Esses PA´s devem ser construídos apenas em oito localidades determinadas pela Secretária de Saúde. É fornecido um grafo onde os vértices representam as localidades candidatas a abrigarem PA´s e também os usuários desse serviço. As distâncias entre esses pontos são representadas pelas arestas do grafo. O objetivo do problema consiste em escolher os dois pontos mais adequados à construção dos PA´s de maneira que todos os usuários estejam o mais próximo possível de um e somente um dos PA´s.

Determine dois pontos p_1 e p_2 onde serão construídos os PA's e quais localidades (usuários) serão atendidas pelos seus respectivos PA's de maneira que todos os usuários estejam o mais próximo possível de um e somente um dos PA´s. Matematicamente, o problema pode ser representado da seguinte forma:

Considere o conjunto $I = \{1, \ldots, 8\}$ de localidades. Fixados $p_1, p_2 \in I$ e $p_1 \neq p_2$, construir os subconjuntos S_1 e S_2 de elementos de I associados respectivamente a p_1 e p_2, tal que $S_1 \cap S_2 = \emptyset$ e $S_1 \cup S_2 \cup \{p_1, p_2\} = I$ e obter o valor

$$s = \min \left(\sum_{i \in S_1} d_{p_1 i} + \sum_{j \in S_2} d_{p_2 j} \right)$$

- a matriz de distâncias com o valor 1000 associado as diagonais e quando não existir ligação direta entre os pontos.

1000	4	1000	5	1000	2	1000	1000
4	1000	7	3	8	1000	1000	1000
1000	7	1000	1000	10	1000	1000	8
5	3	1000	1000	2	7	1	1000
1000	8	10	2	1000	1000	3	2
2	1000	1000	7	1000	1000	1	1000
1000	1000	1000	1	3	1	1000	4
1000	1000	8	1000	2	1000	4	1000

- **ATENÇÃO:** escolhidos os pontos p_1 e p_2 onde serão construídos os PA's, não pode sobrar nenhum ponto sem estar associado a p_1 ou p_2. Além disso, nenhum ponto pode ser atendido pelos dois PA's.

A ENTRADA

A primeira linha do arquivo, indica a dimensão da matriz. Primeiramente o número de linhas, em seguida o número de colunas.
As linhas em seguida trazem a matriz. Cada elemento está separado do seguinte por um "\t".

A SAÍDA

Os pontos onde serão construídos os PA's, os conjuntos de pontos associados a cada p_1 e p_2, isto é, os pontos que serão atendidos pelos PA's construídos em p_1 e p_2, e o valor s da soma das distâncias que foi minimizada.

A saída deve ser no seguinte formato:
p1 = 1
p2 = 5
s1 = {2,6}
s2 = {3,4,7,8}

EXEMPLOS DE ENTRADA

8 8

1000	4	1000	5	1000	2	1000	1000
4	1000	7	3	8	1000	1000	1000
1000	7	1000	1000	10	1000	1000	8
5	3	1000	1000	2	7	1	1000
1000	8	10	2	1000	1000	3	2
2	1000	1000	7	1000	1000	1	1000
1000	1000	1000	1	3	1	1000	4
1000	1000	8	1000	2	1000	4	1000

EXEMPLOS DE SAÍDA

p1 = 1
p2 = 5
s1 = {2,6}
s2 = {3,4,7,8}

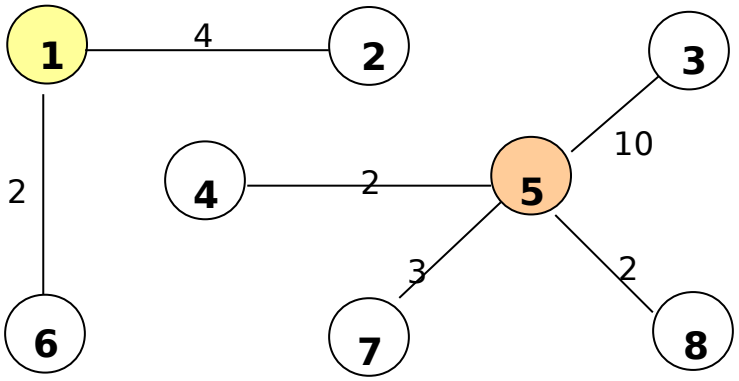
EXEMPLOS:

Uma solução válida:

a) $p_1 = 1$ e $p_2 = 5$ (soma = $d_{12}+d_{16}+d_{53}+d_{54}+d_{57}+d_{58} = 23$)

Pontos associados a $p_1 \rightarrow S_1 = \{2,6\}$

Pontos associados a $p_2 \rightarrow S_2 = \{3,4,7,8\}$

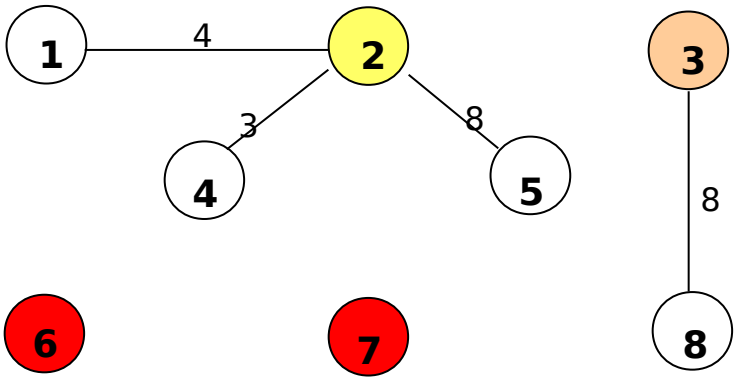


Uma solução inválida (faltam associar os pontos 6 e 7 (em vermelho)):

b) $p_1 = 2$ e $p_2 = 3$

Pontos associados a $p_1 \rightarrow S_1 = \{1,4,5\}$

Pontos associados a $p_2 \rightarrow S_2 = \{8\}$



PROBLEMA F
BR-101

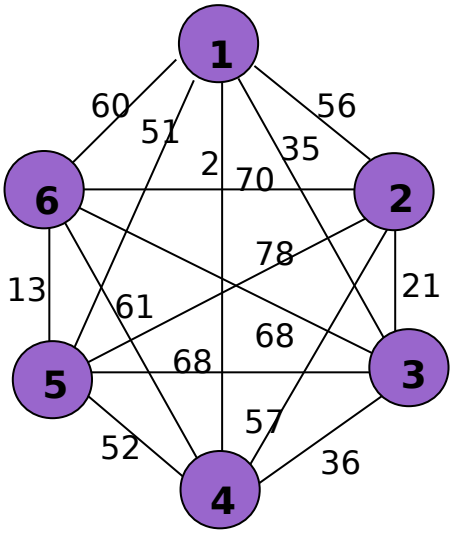
Este problema foi elaborado pela Prof.ª Dr.ª Maria Claudia Silva Boeres (DI/UFES - ES) e pela Prof.ª Dr.ª Maria Cristina Rangel (DI/UFES – ES).

O governo de um determinado estado deseja construir uma malha rodoviária de mão dupla para facilitar o transporte de carregamentos entre seis cidades que se beneficiarão das mercadorias produzidas em duas delas. É fornecido o custo de construir uma estrada entre cada par de cidades. Deseja-se projetar uma malha rodoviária de maneira a minimizar o custo total de projeto. Para obter o projeto que atenda a esse objetivo, o governo estadual irá lançar um edital público para escolher aquele que propõe uma malha rodoviária com custo mínimo de construção. Os custos de construção de uma estrada entre cada par de cidades é representado em um grafo. Os vértices representam as cidades que pertencerão à malha e as arestas representam as estradas com o custo de sua construção.

Dado o grafo, determine o conjunto de arestas que compõem a árvore geradora mínima que representará a malha rodoviária que liga todas as cidades com o menor custo de construção. Matematicamente, uma árvore geradora mínima é um subgrafo gerador (que compreende todos os vértices do grafo) cuja soma dos pesos de suas arestas é minima.

Informações necessárias para a resolução do problema:

- o grafo representativo das cidades e os custos de construção das rodovias entre elas



- A matriz de custos referente ao grafo acima (grafo completo)

0 56 35 2 51 60
56 0 21 57 78 70
35 21 0 36 68 68
2 57 36 0 52 61
51 78 68 52 0 13
60 70 68 61 13 0

A ENTRADA

A matriz de custos relativa ao grafo apresentado. Cada elemento está separado do seguinte por um espaço em branco.
A primeira linha é composta pelo número de linhas e de colunas respectivamente.

A SAÍDA

O conjunto de arestas que compõem a árvore geradora mínima do grafo apresentado e o custo associado.

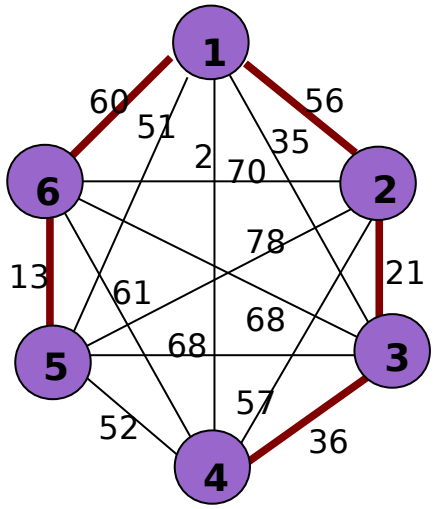
EXEMPLOS DE ENTRADA

6 6
0 56 35 2 51 60
56 0 21 57 78 70
35 21 0 36 68 68
2 57 36 0 52 61
51 78 68 52 0 13
60 70 68 61 13 0

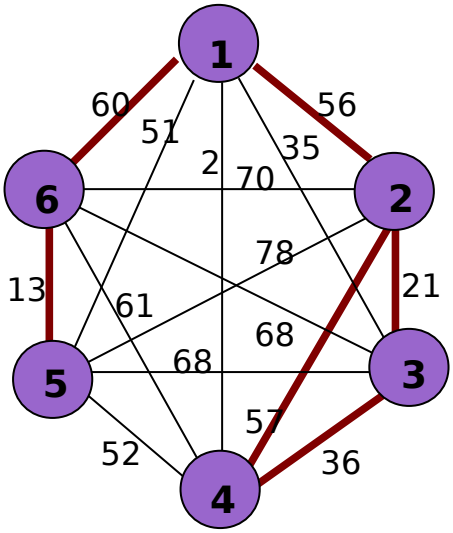
EXEMPLOS DE SAÍDA

conjunto de arestas:
1 2
2 3
3 4
5 6
6 1
custo: 186

Uma solução válida: custo = 186



uma solução inválida: custo = 243



PROBLEMA G
O caso do porquinho branco

A idéia é simples. Sempre que alguém tem qualquer pequena quantidade de dinheiro, ela toma todas as moedas e joga em um porquinho-banco (cofrinho). Você sabe que este processo é irreversível, as moedas não podem ser removidas sem quebrar o porquinho.

Não é possível determinar o quanto de dinheiro está lá dentro. Então nós podemos quebrar o porquinho em pedaços só para descobrir o quanto de dinheiro ele guarda.

Obviamente, nós queremos evitar esta situação desagradável. A única possibilidade é a de pesar o porquinho-banco e tentar adivinhar quantas moedas estão no seu interior, assumindo que somos capazes de determinar o peso do porquinho, e que sabemos exatamente o peso de todas as moedas de uma determinada moeda. Depois, há algum montante mínimo de dinheiro no porquinho-banco que podemos garantir que exista. Sua tarefa é descobrir esta quantidade de moeda no pior caso e determinar o montante mínimo de dinheiro dentro do porquinho-banco. Precisamos da sua ajuda. Não queremos mais porquinhos quebrados desnecessariamente!

A ENTRADA

A entrada consiste de T casos testes. O número T é indicado na primeira linha do arquivo de entrada. Cada caso teste começa com uma linha contendo dois inteiros E e F. Elas indicam o peso do porquinho vazio e do porquinho recheado com moedas, respectivamente. Ambos os pesos são indicados em gramas. Nenhum porquinho vai pesar mais de 10 kg, isso significa que $1 <= E <= F <= 10000$. Na segunda linha de cada caso teste, existe um número inteiro N ($1 <= N <= 500$) que dá o número de diferentes moedas utilizadas em determinada moeda. Após isto são exatamente N linhas, especificando cada moeda . Essas linhas contem dois inteiros cada, P e W ($1 <= P <= 50000$, $1 <= W <= 10000$). P é o valor da moeda em unidades monetárias, W é o peso em gramas.

A SAÍDA

Imprimir exatamente uma linha de resultado para cada caso teste. A linha deve conter a frase "The minimum amount of money in the piggy-bank is *X*." onde X é a quantidade mínima de dinheiro que pode ser alcançada utilizando moedas dadas com o peso total. Se o peso não pode ser alcançado exatamente, imprima uma linha "This is impossible.".

EXEMPLO DE ENTRADA

```
3
10 110
2
1 1
30 50
10 110
2
1 1
50 30
1 6
2
10 3
20 4
```

EXEMPLO DE SAÍDA

```
The minimum amount of money in the piggy-bank is 60.
The minimum amount of money in the piggy-bank is 100.
```
This is impossible.

PROBLEMA G

Piggy-Bank

Before ACM can do anything, a budget must be prepared and the necessary financial support obtained. The main income for this action comes from Irreversibly Bound Money (IBM). The idea behind is simple. Whenever some ACM member has any small money, he takes all the coins and throws them into a piggy-bank. You know that this process is irreversible, the coins cannot be removed without breaking the pig. After a sufficiently long time, there should be enough cash in the piggy-bank to pay everything that needs to be paid.

But there is a big problem with piggy-banks. It is not possible to determine how much money is inside. So we might break the pig into pieces only to find out that there is not enough money. Clearly, we want to avoid this unpleasant situation. The only possibility is to weigh the piggy-bank and try to guess how many coins are inside. Assume that we are able to determine the weight of the pig exactly and that we know the weights of all coins of a given currency. Then there is some minimum amount of money in the piggy-bank that we can guarantee. Your task is to find out this worst case and determine the minimum amount of cash inside the piggy-bank. We need your help. No more prematurely broken pigs!

A ENTRADA

The input consists of *T* test cases. The number of them (*T*) is given on the first line of the input file. Each test case begins with a line containing two integers *E* and *F*. They indicate the weight of an empty pig and of the pig filled with coins. Both weights are given in grams. No pig will weigh more than 10 kg, that means $1 <= E <= F <= 10000$. On the second line of each test case, there is an integer number *N* ($1 <= N <= 500$) that gives the number of various coins used in the given currency. Following this are exactly *N* lines, each specifying one coin type. These lines contain two integers each, *P*and *W* ($1 <= P <= 50000$, $1 <= W <= 10000$). *P* is the value of the coin in monetary units, *W* is it's weight in grams.

A SAÍDA

Print exactly one line of output for each test case. The line must contain the sentence "The minimum amount of money in the piggy-bank is *X*." where *X* is the minimum amount of money that can be achieved using coins with the given total weight. If the weight cannot be reached exactly, print a line "This is impossible.".

EXEMPLO DE ENTRADA

```
3
10 110
2
1 1
30 50
10 110
2
1 1
50 30
1 6
2
10 3
20 4
```

EXEMPLO DE SAÍDA

```
The minimum amount of money in the piggy-bank is 60.
The minimum amount of money in the piggy-bank is 100.
This is impossible.
```

PROBLEMA H
Oh, Esses Pés Que Doem!

Nos dias atuais, várias pessoas têm se machucado após serem empurradas das calçadas por multidões. A Prefeitura está interessada em acompanhar o tráfico de pedestres nas suas calçadas diariamente. Os resultados desse estudo serão usados para determinar se é necessário que a cidade forneça mais calçadas. A Prefeitura examinou (inspecionou, acompanhou) vários prédios de diferentes quarteirões para determinar os padrões de tráfico que eles geram. Seu trabalho é receber esses dados e converter em informação utilização de calçadas.

Seu programa vai ler o tamanho de um mapa e um mapa de vários quarteirões da cidade. Prédios, ruas e entradas/saídas de prédios serão marcados no mapa. Você também receberá uma lista de carga de pedestre de todos os caminhos usando cada rua, e fornecerá como saída uma tabela de cargas totais de pedestre em cada quadrado.

Notas:

- O mapa está dividido em quadrados. Cada quadrado do mapa pode ser um quadrado de rua, de prédio ou de entrada/saída. Um quadrado de entrada/saída serve como entrada ou saída para aquele prédio. Não haverá mais que 90 quadrados de rua no mapa.
- Pessoas sempre seguirão o caminho mais curto entre a origem e o destino. Nenhum caminho mais curto excederá 75 quadrados.
- Se existirem múltiplos caminhos mais curtos de igual comprimento, a carga será dividida igualmente entre esses caminhos. Para os caminhos mais curtos, haverá menos de 50000 combinações de igual comprimento.
- Se uma entrada/saída tem múltiplos lados virados para a rua (por exemplo, uma esquina de um prédio), os pedestres podem entrar ou sair por qualquer lado que esteja virado para a rua.
- Todos os movimentos serão estritamente nas direções N(norte), S(sul), L(leste), O(oeste). Nenhum movimento em diagonal será permitido.
- Pedestres não podem se mover através de prédios ou além dos limites do mapa.
- Por conveniência, você pode ignorar o fato de que cada rua deve ter duas calçadas.
- A carga de tráfico não é aplicada aos quadrados de entrada/saída.
- Se uma origem e um destino são adjacentes no mapa, pedestres podem se mover diretamente entre eles. Nesse caso, não há uma carga resultante situada em nenhuma porção do mapa, já que nenhuma rua foi usada.

A ENTRADA

Linha 1: X Y

X é o número de colunas no mapa, Y é o número de linhas. Cada um deles é um número inteiro positivo inferior a 20.

Linha 2-(Y + 1):
Cada linha contém exatamente X símbolos indicando o conteúdo desse quadrado no mapa. Os símbolos são:
X: edifício, não é entrada/saída
.: (ponto) Rua
{A-O}: Letra indicando entrada/saída. Cada letra pode ocorrer no máximo uma vez.

Linhas (Y + 2)-?:
Cada linha indica uma rota pedestre e especifica a origem, o destino e a carga pedestre. Origem e destino serão, cada um, uma letra {A-O} sem espaços entre eles. O fator de carga será um inteiro não negativo, separado do destino por espaço em branco. Origem e destino nunca serão iguais. Serão dadas no máximo 25 rotas. Haverá um caminho válido no mapa para cada rota solicitada.

O arquivo terminará com a linha:
XX 0

A SAÍDA

A saída consiste em Y linhas, cada uma com X campos separados por espaço indicando o fator de carga. Cada fator de carga é impresso com duas casas decimais com três espaços para dígitos inteiros.

EXEMPLO DE ENTRADA

```
4 4
. . . .
A.X.
XXX.
B . . .
AB 2
BA 1
XX 0
```

EXEMPLO DE SAÍDA

```
1.50     3.00     3.00     3.00
```

0.00	1.50	0.00	3.00
0.00	0.00	0.00	3.00
0.00	3.00	3.00	3.00

PROBLEMA H
Oh, Those Achin' Feet

In recent days, a number of people have been injured after being pushed off the sidewalks due to overcrowding. City Hall is interested in figuring out how much pedestrian traffic its sidewalks receive every day. The results of this study will be used to determine whether the city needs to fund more sidewalks. The city has surveyed various buildings in several blocks to determine the traffic patterns they generate. Your job is to take this survey data and convert it into sidewalk utilization information.

Your program will read in the size of the map and a map of several city blocks. Buildings, streets, and building entrance/exits will be marked on the map. You will also be given a list of pedestrian load between several pairs of exits and entrances. Your program will determine the paths used by pedestrians between each source and destination, add up the total pedestrian load from all paths using each street, and output a table of the total pedestrian load on each square.

Notes:

- The map is divided into squares. Each square of the map can be a street square, a building square, or an entrance/exit square. An entrance/exit square serves as both entrance and exit for that building. There will be no more than 90 street squares in the map.
- People will always follow the shortest path between their origin and destination. No shortest path will exceed 75 squares.
- If there are multiple equal-length shortest paths, the load will be divided equally amongst the paths. For shortest paths, there will be fewer than 50000 equal-length path combinations.
- If a building entrance/exit has multiple sides facing a street (for example, a corner of a building), the pedestrians may enter or exit through any street-facing side.
- All movement will be strictly N, E, S, or W. No diagonal movement is permitted.
- Pedestrians cannot move through buildings or off the edge of the map.
- For convenience, you may ignore the fact that each street section may have two sidewalks.
- Traffic load is not applied to the actual exit/entrance squares themselves.
- If an origin and destination are adjacent on the map, pedestrians may move directly between them. In this case, there is no resulting load placed on any portion of the map because no streets are used.

A ENTRADA

Line 1: X Y

X is the number of columns in the map, Y is the number of rows. Each is a positive integer less than 20.

Line 2-(Y+1):
Each line contains exactly X symbols indicating the contents of that square on the map. The symbols are:
X: building, non-entrance/exit
.: (period) street
{A-O}: letter indicating exit/entrance. Each letter may occur at most once.

Lines (Y+2)-?:
Each line indicates a pedestrian route and specifies a source, destination, and pedestrian load. Source and destination will each be a letter {A-O} with no spaces in between. The load factor will be a nonnegative integer, separated from the destination by whitespace. Source and destination will never be equal. At most 25 routes will be given. There will be a valid path in the map for each requested route.

The file will terminate with the line:
XX 0

A SAÍDA

The output consists of Y lines, each with X space-separated fields indicating the load factor. Each load factor is printed to two decimal places with 3 spaces for integer digits.

EXEMPLO DE ENTRADA

```
4 4
. . . .
A.X.
XXX.
B . . .
AB 2
BA 1
XX 0
```

EXEMPLO DE SAÍDA

```
1.50    3.00    3.00    3.00
0.00    1.50    0.00    3.00
```

0.00	0.00	0.00	3.00
0.00	3.00	3.00	3.00