



X Olimpíada Interna de Programação do IFSULDEMINAS

Caderno de Problemas – Competição Principal

Muzambinho, 20 de Junho de 2020

Realização:



Instruções:

- Este caderno contém 12 problemas: as páginas estão numeradas de 1 a 18, não contando esta página de rosto. Verifique se o caderno está completo.
- Atenção: alguns problemas possuem mais de uma página!
- Em todos os problemas, a entrada de seu programa deve ser lida da entrada padrão. A saída deve ser escrita na saída padrão.

Problema A

O Jardim

Nome do arquivo fonte: jardim.c, jardim.cpp, jardim.py ou jardim.java

O famigerado Jardim de Muzambinho... palco de apresentações de bandas como Rainhas da Era da Pedra e Sangue Real, e local de treino de corredores pois possui o formato retangular.

Em 2020, o mundo foi atingido pelo Coronga Virus, um vírus que ao infectar uma pessoa, causa risada infinita que pode até ser letal. Infelizmente, as pessoas tiveram que parar de frequentar o Jardim graças a pandemia do Coronga, que é muito contagioso e o Jardim seria um alvo fácil para pessoas serem contaminadas. Isso causou muita tristeza na cidade.

Porém, a prefeitura prometeu estar sempre dedetizando o Jardim para quando a pandemia acabar, as pessoas possam voltar às suas atividades no Jardim com segurança.

A prefeitura então criou uma competição para a população: criar um robô que consiga dedetizar ao máximo O Jardim. Obviamente ofereceu um bom prêmio em dinheiro para o criador do melhor robô.

Você, que não é bobo nem nada, decidiu utilizar seu talento em programação e ganhar a competição.

Basicamente, O Jardim contém vários coretos que são os focos de Coronga (onde as pessoas mais se aglomeravam). Para ajudar na competição, a prefeitura disponibilizou para cada participante um sensor que consegue detectar quantos vírus existem em determinado foco. O grande problema é que o Coronga é um vírus muito difícil de eliminar e seu robô deve ser capaz de ingerir alguns focos para que consiga produzir um bom dedetizador, com base na decodificação dos vírus ingeridos.

Você fez um mapeamento dos coretos armazenando em um vetor X a quantidade de vírus que cada coreto contém inicialmente. Detalhe importante: os focos podem ser ingeridos ou dedetizados em qualquer ordem.

Você então criou um sistema em que: inicialmente seu robô tem 1 de experiência (exp), pois é o pouco que sabe-se do Coronga. Quando um coreto i é dedetizado, elimina-se X_i para cada ponto de exp . Isso se deve pois o mapeamento inicial é apenas um indicativo de quantos vírus contém naquele coreto, possivelmente eles já se multiplicaram.

Para cada foco, você tem a opção de ingerir o foco, aumentando seu **exp** em 1, ou então, dedetizar o coreto, matando **exp** * X_i vírus, onde X_i é a quantidade de vírus do i -ésimo coreto.

Por exemplo, se o Jardim possui 3 coretos, o vetor X poderia ser: [3,2,5]. Uma solução possível seria a seguinte:

1. Primeiro foco: ingere. Aumenta o **exp** de 1 para 2;
2. Segundo foco: ingere. Aumenta o **exp** de 2 para 3;
3. Terceiro foco: dedetiza. Elimina $X[3] * exp = 5 * 3 = 15$ vírus.

Logo o total de vírus mortos seriam 15.

Porém, não é a melhor solução, a ótima seria:

1. Segundo foco: ingere. Aumenta o **exp** de 1 para 2.
2. Primeiro foco: dedetiza. Elimina $X[1] * \mathbf{exp} = 3 * 2 = 6$ vírus.
3. Terceiro foco: dedetiza. Elimina $X[3] * \mathbf{exp} = 5 * 2 = 10$.

Logo, total de vírus mortos seria 16.

Pois bem, finalize seu robô. Crie um algoritmo que dado as quantidades de vírus em cada coreto no mapeamento, consiga eliminar o máximo de vírus possíveis. Ah, e lembre-se de dedetizar seu robô depois :)

Entrada

A primeira linha da entrada contém um inteiro **t**, indicando o número de casos de teste.

A segunda linha contém um inteiro **n**, indicando o número de coretos.

A terceira linha contém n inteiros indicando o vetor **X**.

Limites:

$$1 \leq t \leq 10^5;$$


$$1 \leq n \leq 10^5;$$

$$1 \leq X_i \leq 10^7;$$

Saída

Um inteiro indicando o número máximo de vírus que podem ser eliminados.

Exemplo de entrada	Exemplo de saída
1 3 3 2 2	10
1 3 3 2 5	16

Por Samuel Eduardo da Silva, IFSULDEMINAS/UFF  Brasil

Problema B

A noiva do trevo

Nome do arquivo fonte: noiva.c, noiva.cpp, noiva.py ou noiva.java

Existe uma famosa história na cidade de Muzambinho de uma noiva que aparece no trevo da entrada da cidade sempre próximo a meia-noite.

Muitos moradores da cidade já relataram que viram a noiva, porém não existe um consenso de qual é o horário certo que ela aparece, principalmente porque todos dizem que viram "próximo a meia-noite".

Toguro, um grande estudioso de eventos sobrenaturais, está tentando organizar os relatos dos moradores e verificar se houve relatos verdadeiros.

Como um horário "próximo a meia-noite" pode ser tanto alguns minutos antes e alguns minutos depois, Toguro pediu a sua ajuda para criar um algoritmo que dado o valor M de minutos para ser considerado antes e depois da meia noite e o nome do morador e o horário do suposto avistamento, mostrasse de forma ordenada pelo horário os avistamentos que podem ser catalogados como relatos verdadeiros.

Entrada

A primeira linha da entrada contém um inteiro P representando o valor próximo da meia noite para mais e para menos e Q representando a quantidade de pessoas que relataram o avistamento.

Nas próximas Q linhas serão lidos duas strings H e N , indicando o horário do relato e o nome do morador.

Limites:

$$1 \leq P \leq 59;$$

$$1 \leq Q \leq 1000;$$

$$23:01 \leq H \leq 00:59$$


$$1 \leq |N| \leq 100;$$

Saída

A saída deverá ser o nome dos moradores que tiveram o relato como verdadeiro ordenados pelo horário que viram.

Exemplo de entrada	Exemplo de saída
15 5 23:44 Marcelo 00:11 Carlos 00:09 Ana 00:30 Nicolas 00:10 Bernardo	Ana Bernardo Carlos

2 4 23:44 Marcelo 00:11 Carlos 00:09 Ana 00:30 Nicolas	
10 5 23:44 Marcelo 00:10 Carlos 00:09 Bernardo 00:30 Nicolas 00:10 Ana	Bernardo Carlos Ana
10 5 23:50 Marcelo 00:11 Carlos 00:09 Bernardo 00:30 Nicolas 00:10 Ana	Marcelo Bernardo Ana

Por Gabriel Bianchin de Oliveira, IFSULDEMINAS/UNICAMP  Brasil

Problema C

O esconderijo do Rambo

Nome do arquivo fonte: rambo.c, rambo.cpp, rambo.py ou rambo.java

Rambo é um homem muito perspicaz que vive escondido no meio da mata na região do Sul de Minas Gerais. Devido ao seu estilo de vida, Rambo não gosta de ser visto por ninguém e não gosta de que ninguém veja o local em que se esconde.

Com a falta de alimentos em sua moradia, Rambo precisa ir em diversos pontos de da cidade mais próxima atrás de mantimentos e deve retornar ao seu lar.

Cada um dos pontos pode ter estradas entre si, de modo que o Rambo possa andar de um ponto ao outro, não importando o sentido. Rambo DEVE utilizar cada uma das estradas e APENAS uma vez. Além disso, todos os pontos possuem pelo menos um caminho até outro ponto qualquer.

Rambo é seu amigo e sabe que não fará mal algum pra ele, então ele pediu para que escrevesse um programa que dado a quantidade **N** de lugares que precisa ir e as **M** estradas entre esses lugares, retornasse se ele conseguiu ou não chegar de volta para o abrigo.

Entrada

A primeira linha da entrada contém dois inteiros **N** e **M**, indicando o número de pontos e a quantidade de caminhos respectivamente. A casa do Rambo sempre é o primeiro ponto (ponto 0).

As próximas **M** linhas possuem dois valores, indicando a estrada entre os dois pontos.

Limites:

$$2 \leq N \leq 1000$$


$$M \geq N-1$$

Saída

Caso Rambo consiga voltar para a casa, a saída deverá ser “Rambo esta salvo”. Caso contrário, a saída deverá ser “Rambo esta perdido”.

Exemplo de entrada	Exemplo de saída
4 7 0 1 0 1 0 2 0 2 0 3 1 3 2 3	Rambo esta perdido

4 6 0 1 0 1 1 2 1 2 1 3 1 3	Rambo esta salvo
4 6 0 1 1 2 1 2 1 3 1 3 1 3	Rambo esta perdido

Por Gabriel Bianchin de Oliveira, IFSULDEMINAS/UNICAMP  Brasil

Problema D

O Iogurte Divino

Nome do arquivo fonte: iogurte.c, iogurte.cpp, iogurte.py ou iogurte.java

Aaah Muzambinho... além de seus belos cafezais, também é conhecido pelos seus deliciosos iogurtes. Milton, junto com seu amigo Neves, criaram um grupo de degustação de iogurtes chamado Encoders. No Encoders eles degustam os vários diversos tipos de iogurte da cidade. Milton e Neves perceberam que, misturando os iogurtes, conseguem sabores nunca antes provados! Então propuseram um desafio para divertimento do grupo: Juntos eles escolheram um "valor de gostosura" para cada iogurte. Eles também sabem o volume de cada um.

Milton quer encher uma garrafa para levar para sua faculdade com o melhor iogurte possível em termos de "gostosura" e esta garrafa tem um volume máximo **X**. O desafio então se resume a obter um iogurte com a "gostosura" máxima possível.

Sabendo o volume e o "valor de gostosura" de cada iogurte do grupo, determine o valor de "gostosura" máximo que Milton consegue formular em sua garrafa e fazer a alegria de seus amiguinhos na faculdade.

Obs: Eles nem sempre precisam colocar o volume total de um iogurte na garrafa, e assim o preço iogurte é proporcional ao volume colocado.

Entrada

A primeira linha da entrada indica o número **T** de casos de teste.

A segunda linha contém dois inteiros **N** e **X** indicando, respectivamente, o número de iogurtes e a capacidade da garrafa.

A seguir, seguem **N** linhas, cada uma contendo 2 inteiros, indicando o valor do iogurte e o seu volume, respectivamente.

Limites:


$$1 \leq T \leq 100;$$

$$1 \leq N, X \leq 100.$$

Saída

Um valor real com duas casas decimais indicando o valor máximo de "gostosura" a ser obtido.

Exemplo de entrada	Exemplo de saída
1 2 20 10 10 10 10	20.00

Por Samuel Eduardo da Silva, IFSULDEMINAS/UFF  Brasil

Problema E

Samuel, o cafeicultor

Nome do arquivo fonte: samuel.c, samuel.cpp, samuel.py ou samuel.java

“Você gosta de café? Pois é, a maioria das pessoas gostam de café. Já pensou em comprar um terreno e fazer o seu próprio café? Um sonho, não é mesmo? Hoje é seu dia de sorte. Venha negociar um terreno com a gente!!!”

Samuel adora café. Vendo esse anúncio, logo correu para a imobiliária com seu rico dinheirinho para começar a plantar seu próprio café. Na imobiliária, foram oferecidos dois terrenos de mesmo preço para Samuel.

Como os terrenos foram adicionados ao sistema da imobiliária apenas a alguns dias atrás, o gerente sabe que os dois terrenos tinham quatro vértices e as localizações dos vértices, porém não sabe a área total do terreno.

Samuel quer sua ajuda para comprar o terreno de maior área. Quem sabe ele não te dê uma xícara de café?

Portanto, dado os quatro vértices do terreno A e quatro vértices do terreno B, informe para Samuel qual deles tem a maior área.


Entrada

A entrada consiste em 8 linhas com dois inteiros. As quatro primeiras linhas indicam as coordenadas (x, y) do terreno A na ordem horária. As quatro últimas linhas indicam as coordenadas (x, y) do terreno B na ordem horária.

Saída

A saída deve dizer qual dos dois terrenos Samuel deve comprar.

Exemplo de entrada	Exemplo de saída
4 10 9 7 11 2 2 2 1 3 2 2 3 1 1 1	terreno A
5 9 10 6 7 2 2 1 5 11 10 6 7 2 2 2	terreno B

Por Gabriel Bianchin de Oliveira, IFSULDEMINAS/UNICAMP  Brasil.

Problema F

Fake News

Nome do arquivo fonte: fake.c, fake.cpp, fake.py ou fake.java

Durante a pandemia do Coronga Virus, infelizmente o número de fakes news (notícias falsas, muitas vezes tendenciosas) cresceu exponencialmente. A redes sociais estão tentando buscar formas de controlar esse problema.

Após um estudo, concluiu-se que as fake news sempre começam com determinadas palavras. Você é estagiário da principal empresa que está lutando contra as fake news, e foi

pedido a você que criasse um algoritmo para isto.

A empresa coletou as informações para seu algoritmo e elas estão da seguinte forma: são N linhas, em cada linha contém duas palavras **A** e **B**. A palavra N sempre será:

- "add" e assim **B** será um texto todo concatenado a ser analisado.
- "find" e assim **B** será uma nova palavra que determina fake news.

Quando **A** for "add", seu algoritmo deverá retornar quantas vezes a palavra **B** apareceu no início dos textos com **A** igual a "find" anteriores.

Escreva este algoritmo e fuja das fake news!!

Entrada

A primeira linha da entrada contém um inteiro N indicando o número de conjuntos **A** e **B** que a empresa lhe forneceu.

As N linhas seguintes com os textos de **A** e **B**. Os textos contém apenas letras minúsculas.

Limites:


$$0 < N \leq 10^5;$$

$$0 < |A| \text{ e } |B| \leq 21.$$

Saída

Para cada linha em que $A = \text{"find"}$, um inteiro indicando quantas vezes **B** ocorreu no início dos textos em que **A** foi igual a "add", anteriores.

Exemplo de entrada	Exemplo de saída
4 add quiboaquinaehtop add quiboaquinacura find quiboa find terraplana	2 0

Por Samuel Eduardo da Silva, IFSULDEMINAS/UFF  Brasil

Problema G

A Grande Festa

Nome do arquivo fonte: festa.c, festa.cpp, festa.py ou festa.java

Famoso viajante interplanetário, ET Bilu quer convidar pessoas do planeta que está visitando para uma Grande Festa. Na cultura de Bilu, uma festa só pode ser considerada uma Grande Festa se pelo menos dois participantes tenham nascido no mesmo dia.

Diferente do planeta Terra, que cada ano tem 365 ou 366 dias, os planetas que Bilu visita possuem uma quantidade de dias diferentes.

Mesmo com toda a avançada tecnologia de viagem intergaláctica, Bilu não sabe calcular a chance de desse grande evento acontecer. Sendo assim, você, que estava em uma Grande Festa realizada no planeta Terra, irá ajudar Bilu a incorporar no computador de borda da nave um programa para estimar a chance de acontecer uma Grande Festa.

Dado a quantidade de dias por ano do planeta que Bilu está visitando e a quantidade de pessoas na festa, determine a probabilidade da festa ser considerada uma Grande Festa.

Entrada

A entrada consiste em dois inteiros **D** e **P** indicando a quantidade de dias e de pessoas, respectivamente.

Limites:


$$50 \leq D \leq 10^5;$$

$$2 \leq P \leq D - 1.$$

Saída

A saída deve ser a porcentagem de chance do evento ser uma Grande Festa com duas casas decimais.

Exemplo de entrada	Exemplo de saída
365 23	50.73
366 23	50.63
1200 42	51.61
100000 500	71.34

Por Gabriel Bianchin de Oliveira, IFSULDEMINAS/UNICAMP  Brasil.

Problema H

Sócrates e suas perguntas

Nome do arquivo fonte: socrates.c, socrates.cpp, socrates.py ou socrates.java

Sócrates era um homem dito por muitos como um verdadeiro chato. Todos diziam que ele era muito inteligente, mas sempre que alguém iria lhe perguntar algo, Sócrates só fazia mais perguntas e nunca respondia nada.

Após estudar sobre a vida e o universo, Sócrates se engajou em dominar a arte dos arrays. Um array é um conjunto de elementos arranjados em uma linha, um após o outro.

Sócrates se deparou com um problema no qual: dado um array de números inteiros, ele gostaria de saber, dado vários intervalos de índices do vetor, **L** e **R**, quantas vezes um número **x** ocorre exatamente **x** vezes no intervalo dado. Ele acabou por desistir de responder pois seu array era gigante e segundo ele, a razão humana ainda não havia chegado ao ponto de calcular com rapidez sua pergunta. Porém, ele lançou a seguinte hipótese: num futuro distante, pessoas conseguirão responder com o auxílio de ferramentas.

Será que Sócrates estava certo? Prove a hipótese de Sócrates criando um algoritmo que responda o problema.

Entrada

A primeira linha da entrada contém dois inteiros **N** e **Q** indicando o tamanho do array e o número de perguntas.

A segunda linha da entrada contém **N** inteiros, representando os números do array.

A seguir seguem **M** linhas, em que cada uma contém dois inteiros **L** e **R** representando o intervalo da pergunta.

Limites:

$$1 \leq N, Q \leq 10^5;$$


$$1 \leq \text{números do array} \leq 10^9;$$

$$1 \leq L, R \leq 10^5;$$

Saída

A saída contém **Q** linhas contendo em cada uma um inteiro representando a resposta da **Q**-ésima pergunta.

Exemplo de entrada	Exemplo de saída
7 2 3 1 2 2 3 3 7 1 7 3 4	3 1

Por Samuel Eduardo da Silva, IFSULDEMINAS/UFF  Brasil

Problema I

Xadrez Galático

Nome do arquivo fonte: xadrez.c, xadrez.cpp, xadrez.py ou xadrez.java

Cansados de perder seus preciosos peões e de ficar viúvos, os reis de todo universo se reuniram na ACM (Associação Comum dos Monarcas), para elaborar novas regras para o jogo de xadrez. Assim surgiu o xadrez galático.

Como os reis não queriam colocar seus preciosos lacaios em perigos, foi decidido que no xadrez galático o único participante seria o rei, no início da partida cada rei receberia uma pistola laser de alta potência e venceria a partida o rei que conseguisse atingir seu adversário com um potente disparo laser.

Para deixar o jogo ainda mais interessante foi decidido que o tabuleiro iria contar com diversos espelhos giratórios, capazes de refletir o raio laser disparado pelas pistolas. Em cada rodada do jogo o rei poderia decidir entre atirar em uma das quatro direções do tabuleiro, (norte, oeste, leste e sul) ou girar um dos espelhos, mudando assim a direção do raio laser.

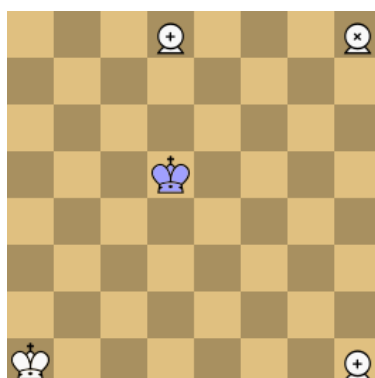
A reflexão dos espelhos funciona da seguinte forma:

Tip /Direção do raio	Sul	Norte	Leste	Oeste
Espelho Padrão	Esquerda	Direita	Cima	Baixo
Espelho Invertido	Direita	Esquerda	Baixo	Cima

Então podemos deduzir que se um raio disparado para baixo encontrar um espelho padrão (representado pelo sinal de +) o raio será refletido para esquerda, caso o mesmo o raio encontrasse um espelho invertido (representado pelo sinal de x) ele seria refletido para direita.

Além disso os reis definiram que pelo fato do preço dos espelhos ser bem elevado, cada tabuleiro pode conter no máximo vinte espelhos.

Apesar de ter gostado bastante do fato de não precisar mais arriscar a vida de seus companheiros o Rei Branco, está bem preocupado ele acha as regras do xadrez galático muito complicadas, então ele ordenou a você, que desenvolva um programa que dado o estado atual do tabuleiro, calcule se existe alguma combinação de espelhos que fará com que o raio laser chegue até o rei adversário e caso exista informe essa qual é essa combinação. A imagem a seguir representa o primeiro caso de teste.



Entrada

A primeira linha da entrada contém dois inteiros **N** e **M** indicando respectivamente a altura e largura do tabuleiro de xadrez galático.

As próximas **N** linhas contém **M** caracteres descrevendo o tabuleiro.

O Rei Branco é representado pelo caractere 'B' e o Rei Preto, pelo caractere 'P,' um espelho padrão é representado pelo caractere '+', um espelho invertido é representado pelo caractere 'x', o caractere '.' descreve um espaço em branco no tabuleiro.


Limites:

$$1 \leq N, M \leq 100;$$

Saída

A saída contém a string 'NO' se não for possível girar os espelhos de forma que o raio atinja o rei adversário, ou 'YES' seguido por uma string representando a combinação de espelhos que faz o raio atingir o rei adversário, se houver mais de uma combinação possível, imprima a que formar a menor string lexicograficamente. Os espelhos são ordenados de cima para baixo da esquerda para direita.

Exemplo de entrada	Exemplo de saída
8 8 ...+...XP.... B.....+	YES +x+
8 8 ...+...XP.... B.....+	NO
2 20 B.+..+...+..... ..+..+...+.....P	YES x+xx+x
2 20 B.+..+...+..... ..+..+...+...+.....P	NO

Por Abner Samuel Pinto Palmeira, IFSULDEMINAS  Brasil

Problema J

Viva a Ciência!

Nome do arquivo fonte: viva.c, viva.cpp, viva.py ou viva.java

Oronob é um garoto muito peculiar. Para ele, apenas a sua opinião importa e nada mais. Durante a pandemia do Coronga Virus, ele usa sua opinião para retaliar o que a ciência diz. Ele então, para impulsionar suas opiniões, criou um bot (robô que gera mensagens automáticas) para disparar seus textos para muitas pessoas.

Você conhece Oronob, porém, ao contrário dele, você é sensato e sabe que ciência não se refuta com opinião, além também de entender de programação, então decidiu hackear o bot de Oronob para evitar suas falácias.

Pois bem, crie então um algoritmo que muda o texto das opiniões de Oronob para a real verdade.

Entrada


Um texto **S** com caracteres alfanuméricos e possíveis espaços entre as palavras.

Limites: $0 < |S| \leq 280$. ;)

Saída

A verdade.

Exemplo de entrada	Exemplo de saída
Li no face que tal remédio cura o Coronga a ciencia coronguista diz que nao	Nao se refuta ciencia com opiniao
Coronga eh criacao narniana para vender vacina	Nao se refuta ciencia com opiniao
Idai que coronga mata importante eh a economia	Nao se refuta ciencia com opiniao

Por Samuel Eduardo da Silva, IFSULDEMINAS/UFF  Brasil

Problema K

O bom presidente

Nome do arquivo fonte: bom.c, bom.cpp, bom.py ou bom.java

Livrolândia é um país que, como o nome já diz, preza pela leitura. Nesta cidade há uma regra universal: toda cidade do país deve ter acesso a bibliotecas. Todos os presidentes que passaram por Livrolândia conseguiram manter esta regra.

Roci é o atual presidente, e fez questão de dar manutenção a todas as bibliotecas do país, além de manter a boa qualidade das estradas entre as cidades, para que cidades que não tem biblioteca, consigam acesso a cidades vizinhas que tenham.

Infelizmente, Roci é muito azarado e, logo em seu mandato, um tornado destruiu todas as bibliotecas e obstruiu todas as estradas de Livrolândia. Agora, o presidente terá que bolar um plano para reconstruir o país, seguindo sua regra universal e com o menor custo possível para as obras.

Livrolândia tem **N** cidades numeradas de **1** a **N**. As cidades são conectadas por **M** estradas bidirecionadas. Uma cidade tem acesso a uma biblioteca se:

- Esta cidade tem uma biblioteca;
- É possível, a partir desta cidade, viajar para uma cidade que contém uma biblioteca.

O custo para reparar uma estrada é **E** tolkiens (tolkiens é a moeda de Livrolândia) e o custo para construir uma biblioteca é **B** tolkiens.

Dado o mapa de Livrolândia e os custos de reparo e construção, escreva um programa que retorne o custo mínimo para reconstruir o país, seguindo a regra universal, e assim, salve Roci.

Entrada

A primeira linha da entrada contém um inteiro **T** indicando o número de possíveis mapas.

A segunda linha da entrada contém 4 inteiros, **N**, **M**, **B** e **E**, número de cidades, número de estradas, custo para construir uma biblioteca e o custo para construir uma estrada, respectivamente.

Depois há **M** linhas indicando as estradas obstruídas, em que cada uma há dois inteiros **X** e **Y**, indicando que há uma estrada que liga a cidade **X** à cidade **Y**.

Limites:

$$1 \leq T \leq 10;$$

$$1 \leq N \leq 10^5;$$

$$0 \leq M \leq \min(10^5, (N*(N-1))/2);$$


$$1 \leq B, E \leq 10^5;$$

$$1 \leq X, Y \leq N.$$

Saída

Para cada possível mapa, indique o custo mínimo para reconstruir o país.

Exemplo de entrada	Exemplo de saída
2 3 3 2 1 1 2 3 1 2 3 6 6 2 5 1 3 3 4 2 4 1 2 2 3 5 6	4 12

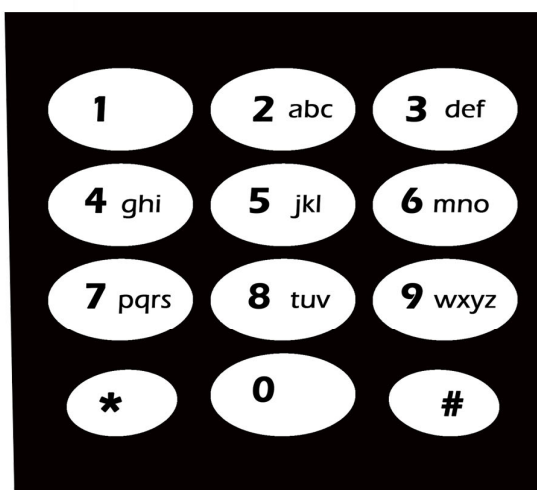
Por Samuel Eduardo da Silva, IFSULDEMINAS/UFF  Brasil

Problema L

Tijolão

Nome do arquivo fonte: tijolao.c, tijolao.cpp, tijolao.py ou tijolao.java

Durante essa quarentena Amélia estava lembrando do que gostava de fazer na infância. De maneira nostálgica ela se lembrou do quanto gostava de jogar Snake no celular da sua mãe. Era o famoso 'tijolar', da marca Aikon. Ela até conseguiu encontrar um aplicativo para Android que simulava o joguinho, mas ela começou a se perguntar: "Como eu conseguia escrever mensagens com aquele teclado mesmo?". Na imagem abaixo tem-se uma foto do celular à esquerda e o detalhe de seu teclado à direita.



Ela lembrou como o teclado era utilizado para escrever mensagens. Nesse problema, para simplificar, iremos considerar um celular no idioma inglês(não há acentos).

As teclas de 2 a 9 são usadas para digitar as letras de 'a' a 'z', e funcionam assim: se quisermos obter uma das letras associadas a uma das teclas, precisamos pressioná-la um número de vezes igual a posição da letra desejada. Por exemplo, pressionando a tecla 6 uma vez obtemos 'm'. Se pressionarmos novamente, obteremos 'n' e depois 'o'. Se continuarmos pressionando-a obteremos o número '6' e depois reiniciamos em 'm', assim sucessivamente. A tecla 0 é utilizada para inserir espaços na mensagem; a tecla 1 era apenas o

dígito '1' e algumas pontuações (que não serão usadas nesse problema) e a tecla # ela é usada para colocar a próxima letra em maiúsculo, por exemplo, se apertar #2 obteremos 'A', se apertar #27 obteremos 'Ap'.

No caso de termos duas letras consecutivas na mensagem que são formadas pela mesma tecla era necessário esperar um tempinho para continuar digitando, nesse problema vamos considerar que tecla * represente esse intervalo de tempo, ou seja, a função da tecla * no nosso caso é separar as sequências de pressionamentos de duas letras na mesma tecla. Por exemplo, para digitar a palavra “cafe”, a sequência de teclas pressionadas seria a seguinte: 222*2333*33 . Mas se a segunda letra na mesma tecla for maiúscula não é necessário pressionar * pois você terá de pressionar # entre elas. Por exemplo, a palavra 'cAfe' corresponde à sequência 222#2333*33.

Como o teclado era bem duro, Amélia odiava cometer erros, por isso ela quer saber a sequência mínima de teclas que deve apertar para digitar cada mensagem.

Entrada


A primeira linha de cada caso de teste contém um inteiro N ($1 \leq N \leq 666$) que representa a quantidade de mensagens que Amélia quer escrever. Cada uma das N linhas seguintes possuem uma frase de 1 a 140 caracteres. As frases serão compostas por letras minúsculas e maiúsculas sem acentos.

Saída

Para cada frase informada por Amélia você deve mostrar em uma linha a sequência mínima de teclas a serem pressionadas, todas juntas.

OBS: não terão espaços duplos, nem espaços no início ou final da frase. Ou seja, apenas espaços simples e entre as palavras.

Exemplo de entrada	Exemplo de saída
4 cAfe cafe CAFE Cafe	222#2333*33 222*2333*33 #222#2#333#33 #222*2333*33
1 Pior que ta nao fica Tiririca	#7444666777077883308206626660333444222*20#84 44777444777444222*2
1 Qual a ave mais verdadeira do mundo Avestruces ba dum tssss	#778825550202888330624447777088833777323*334 4477720366606886636660*0#2888337777877788337 777022*203886087777*7777*7777*7777

Por Aline Regina de Oliveira, IFSULDEMINAS/UFSCAR  Brasil