

Maratona Júnior da UFFS 2018

Caderno de Problemas

Earliest Deadline First

Por Emilio Wuerges, UFFS  Brazil**Timelimit: 1**

Neste problema o seu trabalho é verificar se um conjunto de processos periódicos que possuem restrição de tempo-real pode ser escalonado.

Um processo de tempo real é caracterizado por dois números. O primeiro é o custo computacional do processo. Ou seja, o tempo que o processo gasta quando entrar em execução. O segundo número é o período em que o processo executa. Ou seja, a cada período de tempo, o processo reinicia.

O conjunto será escalonado usando o algoritmo EDF (Earliest Deadline First). Sabe-se que o algoritmo EDF é ótimo. Ou seja, se um conjunto de tarefas não poder ser escalonado pelo EDF, ele não poderá ser escalonado por nenhum outro algoritmo.

O sistema operacional que receberá estas tarefas está rodando em uma máquina single core. As tarefas são preemptíveis. Isto é uma tarefa pode tomar o lugar de outra durante a execução, se for necessário.

Considere que o custo de trocar entre tarefas é 0.

Entrada

A primeira linha da entrada possui um valor $1 \leq N \leq 10$, que é número de processos a ser avaliado.

Cada **N** linha seguinte representa um processo, e contém 2 valores $1 \leq C \leq 5$ e $C \leq P \leq 100$, que representam o custo computacional e o período de cada processo, respectivamente.

Saída

A saída consiste de uma única linha, contendo ou o string **OK** ou do string **FAIL**, caso o escalonamento seja possível ou não, respectivamente.

Exemplos de Entrada	Exemplos de Saída
2 3 5 2 5	OK
4 1 5 4 9 5 15 2 45	FAIL

Pudim

Por Emilio Wuerges, UFFS  Brazil**Timelimit: 1**

O famosíssimo portal pudim.com.br está criando um novo motor de busca na internet. O principal diferencial deste motor de busca é que ele fará as buscas a partir de um rank dos artigos, de acordo com as pesquisas feitas pelos usuários.

Está quase tudo pronto. Falta apenas o algoritmo de rankeamento.

Após muita discussão, se resolveu usar como nota para o rankeamento o tamanho da maior string comum entre os strings de busca e os strings contendo os artigos. Isto é, tenta-se calcular o tamanho da maior string que pode ser formada que contém apenas caracteres que estejam tanto no string de busca quanto no string do artigo, na mesma ordem.

Entrada

A entrada contém duas linhas, cada uma com um string. Na primeira linha está o string da busca, na segunda o do artigo. Ambos os strings contém apenas caracteres de **a** a **z**, sem espaços.

O tamanho das strings tanto de busca quanto do artigo são de no máximo 5×10^3 caracteres

Saída

A saída contém apenas uma linha com um inteiro: a nota do artigo.

Exemplos de Entrada	Exemplos de Saída
pudim phudim	5
Pudim pudim	4
axbxc zazbzc	3

L de Atreus!?

Por Gabriel Batista Galli, UFFS  Brazil

Timelimit: 3

No recentemente lançado e muito aclamado jogo God of War, Kratos está agora no mundo da mitologia Nórdica tentando esquecer seu passado. Lá ele conheceu Faye, uma mulher tão misteriosa quanto ele próprio, com quem teve um filho: Atreus. Mas com a chegada de um estranho, Kratos e seu filho embarcam em uma jornada de consequências inesperadas.

Ao longo de sua viagem, eles encontram vários escritos em paredes, painéis, pedras... Esses escritos contam interessantes histórias sobre o mundo nórdico e suas personalidades, mas também aconselham os viajantes desavisados. Infelizmente, Kratos não consegue lê-los, já que ele é de Esparta e não se interessou em aprender as várias línguas de seu novo lar. Então Atreus precisa traduzir toda mensagem escrita que eles encontram. E Atreus é muito bom em linguística, impressionando até Mimir, o homem mais sábio de todos!

Entretanto, não é incomum que eles encontrem mensagens com runas (letras nórdicas) faltando ou danificadas, impossibilitando que Atreus saiba o que elas estão tentando dizer. Então você, sendo um jogador esperto e entusiasmado, foi escolhido para ajudá-los. Sua tarefa é corrigir esses erros para que Atreus possa ler os dizeres e então eles estarão melhor preparados para os perigos dessa jornada.

Mas como alguém faria isso? Bem, Mimir sugere o seguinte: a Edda em verso e a Edda em prosa, dois trabalhos literários islandeses medievais compostos em sua maioria por poemas, são as duas principais fontes de conhecimento sobre a mitologia nórdica*. Sendo assim, procure nestas coletâneas de poemas as palavras que sejam mais similares às que o Atreus não consegue ler e use-as para corrigir os dizeres. Uma palavra p_1 é considerada mais similar a p_2 do que a p_3 se há menos inserções, deleções e substituições de letras a fazer para transformar p_1 em p_2 do que p_1 em p_3 .

Entrada

A entrada é composta por múltiplas linhas. A primeira contém um inteiro q ($1 \leq q \leq 10^3$) representando a quantia de palavras não-repetidas que você encontrou em um Edda. Cada uma das próximas q linhas contém uma dessas palavras. A próxima linha contém outro inteiro, c ($1 \leq c \leq 10^2$), que indica o comprimento em palavras do escrito que você precisa corrigir. A última linha é o próprio escrito. Cada palavra dele é separada por um único espaço.

Todas as palavras da entrada são compostas apenas por caracteres ASCII e têm comprimento $\leq 10^2$. Não há diacríticos, nem pontuação.

Saída

A saída é uma única linha contendo a mensagem corrigida para Atreus ler. Ela segue as mesmas regras que a entrada e precisa terminar com uma quebra de linha (`\n`). Além disso, no caso de um empate

entre duas palavras, escolha a que aparece primeiro dentre as da entrada.

Exemplos de Entrada	Exemplos de Saída
7 Tyr Baldur Freya Loki Mimir Odin Thor 1 Thyr	Tyr
13 aware beware Freya God if Jormungandr Kratos Loki Mimir Odin of War war 4 Frwus bAWARE of Odnn	Freya beware of Odin
9 Midgard Niflheim Muspelheim Helheim Jotunheim Alfheim Vanaheim Svartalfheim Asgard 1	Muspelheim

Midgelheim

* fonte: <https://pt.wikipedia.org/wiki/Edda>

Léxico

Por Emilio Wuerges, UFFS  Brazil**Timelimit: 1**

Como se sabe, léxico é o conjunto de palavras que existe em uma língua. Nas línguas ocidentais, é comum escrever usando o alfabeto latino, com 26 letras que vão de **a** até **z**.

É comum enumerar as letras na seguinte ordem: a, b, c, d, e f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z.

Se uma lista de palavras está organizadas de acordo com esta ordem, fica muito mais rápido pesquisá-las. Seu trabalho neste problema é ordenar duas palavras de acordo com esta ordem.

Sejam duas palavras **A** e **B**. Caso o primeiro caractere de A venha antes do primeiro de B, coloca-se A antes de B. Se o primeiro caractere for igual, usa-se o seguinte para desempate. E se o segundo empatar, usa-se o terceiro, etc. Quando todos os caracteres de A forem iguais ao começo de B, ou todos os de B forem iguais ao começo de A, coloca-se a menor palavra primeiro.

Entrada

A entrada contém 2 palavras com caracteres minúsculos de **a** até **z**, O comprimento das palavras não ultrapassa 20 caracteres.

Saída

A saída contém as mesmas 2 palavras, só que na ordem lexicográfica.

Exemplos de Entrada	Exemplos de Saída
abc def	abc def
bcd abc	abc bcd
abcd abc	abc abcd

Contador de tokens

Por Guilherme Dal Bianco, UFFS  Brazil**Timelimit: 1**

Um hospital pretende migrar seus registros de pacientes para um novo sistema. No entanto, foi observado que a tabela clientes possui o problema registros duplicados. Ou seja, um mesmo paciente que ingressou duas vezes no hospital pode estar registrado duas vezes na base de dados. Tal problema ocorre devido a presença de erros de tipografia (por exemplo: "ketlyn da Silva" ou "ketlin da Silva").

Uma alternativa para contornar esse problema é quebrar o texto ("string") em unidades menores ("tokens") de um determinado tamanho e computar o número de tokens em comum. Por exemplo, os tokens de tamanho 3 para a string "ketlyn" são: "ket", "etl", "tly" e "lyn"; já para a string "ketlin" são gerados as substrings: "ket", "etl", "tli", "lin". Note que a geração de substrings é feita a partir de uma "janela deslizando" variando em 1 caractere. Por fim, é necessário contabilizar quantos tokens as strings tem em comum.

A tarefa a ser realizada é encontrar qual o token (de tamanho 2) que mais se repete e a sua frequência em uma dada string. No caso de empate, um token em ordem alfabética deve ser recuperado. A string deve ser transformada para caixa baixa.

Entrada

Somente uma linha da entrada contém uma string S representando o texto a ser processado. O comprimento máximo desta linha é de 10^5 caracteres.


Saída

Imprima o token mais frequente utilizando um token de tamanho dois (2). Também deve ser impresso a frequência do token.

Tokens em caixa alta devem ser convertidos para caixa baixa. Caso exista um empate na frequência, deve ser impresso somente o primeiro token (segundo a ordem alfabética).

Exemplos de Entrada	Exemplos de Saída
casa carro casa	ca:3
casa	as:1
ewrwsffdes	de:1

Anagramas

Por Emilio Wuerges, UFFS  Brazil**Timelimit: 1**

Um anagrama é uma permutação das letras de uma palavra.

Por exemplo, os strings **ananab** e **anbana** são anagramas da palavra **banana**.

Neste problema, sua tarefa é computar quantos anagramas distintos existem para uma palavra dada.

Como podem existir muitos anagramas, calcule o **resto** da divisão do número de anagramas pelo nosso primo favorito: $10^9 + 7$.

Entrada

Uma linha, com uma única palavra, com o comprimento máximo de 10^5 caracteres. A palavra só contém caracteres minúsculos de **a** a **z**.

Saída

A saída contém apenas uma linha contendo um inteiro, o resultado.

Exemplos de Entrada	Exemplos de Saída
aaa	1
abc	6
aab	3

Maiúsculas, Porque?

Por Emilio Wuerges, UFFS  Brazil**Timelimit: 1**

O problema Léxico nos ensinou como ordenar duas palavras. Entretanto, quando estávamos construindo nossa lista de palavras, o estagiário encarregado confundiu algumas letras maiúsculas com as minúsculas!

Agora temos uma lista enorme de palavras, que podem ter letras minúsculas, de **a** a **z**, e maiúsculas, de **A** a **Z**!

Precisamos ordená-las, obedecendo a mesma ordem de antes, só que quando compararmos uma letra maiúsculas com uma minúscula, a palavra com a letra maiúsculas deve vir antes.

Entrada

A primeira linha contém um inteiro $1 \leq N \leq 5 \times 10^4$.

As **N** linhas seguintes contém uma palavra da lista, com comprimento máximo de 20 caracteres.

Saída

A saída deve conter a lista de palavras ordenadas.

Exemplos de Entrada	Exemplos de Saída
4 Abcd zbcd zBce abzd	Abcd abzd zbcd zBce
5 Abc abc xyz xYz ZZZ	Abc abc xYz xyz ZZZ