

Competidor(a): _____

Número de inscrição: _____ – _____ (opcional)



OBI2021

Caderno de Tarefas

Modalidade Programação • Nível Sênior • Fase 3

25 de setembro de 2021

A PROVA TEM DURAÇÃO DE 5 HORAS

Promoção:



Sociedade Brasileira de Computação

Apoio:



Instruções

LEIA ATENTAMENTE ESTAS INSTRUÇÕES ANTES DE INICIAR A PROVA

- Este caderno de tarefas é composto por 12 páginas (não contando a folha de rosto), numeradas de 1 a 12. Verifique se o caderno está completo.
- A prova deve ser feita individualmente.
- É proibido consultar a Internet, livros, anotações ou qualquer outro material durante a prova. É permitida a consulta ao *help* do ambiente de programação se este estiver disponível.
- As tarefas têm o mesmo valor na correção.
- A correção é automatizada, portanto siga atentamente as exigências da tarefa quanto ao formato da entrada e saída de seu programa; em particular, seu programa não deve escrever frases como “Digite o dado de entrada:” ou similares.
- Não implemente nenhum recurso gráfico nas suas soluções (janelas, menus, etc.), nem utilize qualquer rotina para limpar a tela ou posicionar o cursor.
- As tarefas **não** estão necessariamente ordenadas, neste caderno, por ordem de dificuldade; procure resolver primeiro as questões mais fáceis.
- Preste muita atenção no nome dos arquivos fonte indicados nas tarefas. Soluções na linguagem C devem ser arquivos com sufixo *.c*; soluções na linguagem C++ devem ser arquivos com sufixo *.cc* ou *.cpp*; soluções na linguagem Pascal devem ser arquivos com sufixo *.pas*; soluções na linguagem Java devem ser arquivos com sufixo *.java* e a classe principal deve ter o mesmo nome do arquivo fonte; soluções na linguagem Python 3 devem ser arquivos com sufixo *.py*; e soluções na linguagem Javascript devem ter arquivos com sufixo *.js*.
- Na linguagem Java, **não** use o comando *package*, e note que o nome de sua classe principal deve usar somente letras minúsculas (o mesmo nome do arquivo indicado nas tarefas).
- Para tarefas diferentes você pode escolher trabalhar com linguagens diferentes, mas apenas uma solução, em uma única linguagem, deve ser submetida para cada tarefa.
- Ao final da prova, para cada solução que você queira submeter para correção, copie o arquivo fonte para o seu diretório de trabalho ou pen-drive, conforme especificado pelo seu professor.
- Não utilize arquivos para entrada ou saída. Todos os dados devem ser lidos da entrada padrão (normalmente é o teclado) e escritos na saída padrão (normalmente é a tela). Utilize as funções padrão para entrada e saída de dados:
 - em Pascal: *readln*, *read*, *writeln*, *write*;
 - em C: *scanf*, *getchar*, *printf*, *putchar*;
 - em C++: as mesmas de C ou os objetos *cout* e *cin*.
 - em Java: qualquer classe ou função padrão, como por exemplo *Scanner*, *BufferedReader*, *BufferedWriter* e *System.out.println*
 - em Python: *read*, *readline*, *readlines*, *input*, *print*, *write*
 - em Javascript: *scanf*, *printf*
- Procure resolver a tarefa de maneira eficiente. Na correção, eficiência também será levada em conta. As soluções serão testadas com outras entradas além das apresentadas como exemplo nas tarefas.

Ogro

Nome do arquivo: “ogro.x”, onde x deve ser c, cpp, pas, java, js ou py

O Ogro da Nlogônia está aprendendo a contar até dez usando os dedos das mãos (assim como os humanos, ele possui 2 mãos com 5 dedos cada)). Ele está treinando muito, mas gostaria de ter um aplicativo para ajudá-lo nessa empreitada.

O Ogro aprendeu a mostrar a representação de um número com as mãos da seguinte forma:

- se o número pode ser representado usando apenas uma das mãos, o Ogro usa os dedos na mão esquerda e mantém a mão direita fechada.
- caso contrário, o Ogro mostra todos os cinco dedos da mão esquerda, e na mão direita mostra os dedos que faltam para representar o número.

Por exemplo, para o número 3, o Ogro mostra:

III *

onde cada letra I representa um dedo e a mão fechada é representada pelo símbolo ‘*’ (asterisco). Para o número 8 o Ogro mostra:

IIIII III

Sua tarefa é ajudar o Ogro em seu treinamento, escrevendo um programa para, dado um número entre 0 e 10, mostrar a configuração de dedos correspondente a esse número, de acordo com as regras acima.

Entrada

A primeira e única linha da entrada contém um inteiro N , o número que deve ser representado com os dedos das mãos.

Saída

Seu programa deve produzir duas linhas na saída. A primeira linha deve conter a representação dos dedos da mão esquerda, a segunda linha deve conter a representação dos dedos da mão direita. A letra ‘I’ deve ser usada para representar um dedo, e o caractere ‘*’ (asterisco) deve ser usado para representar a mão fechada (isto é, nenhum dedo mostrado).

Restrições

- $0 \leq N \leq 10$

| Exemplo de entrada 1 | Exemplo de saída 1 |
|----------------------|--------------------|
| 8 | IIIII III |

Explicação do exemplo 1: para representar o número 8 o Ogro mostra os cinco dedos da mão esquerda e três dedos da mão direita.

| Exemplo de entrada 2 | Exemplo de saída 2 |
|----------------------|--------------------|
| 3 | III * |

Explicação do exemplo 2: para representar o número 3 o Ogro mostra três dedos da mão esquerda e nenhum dedo na mão direita.

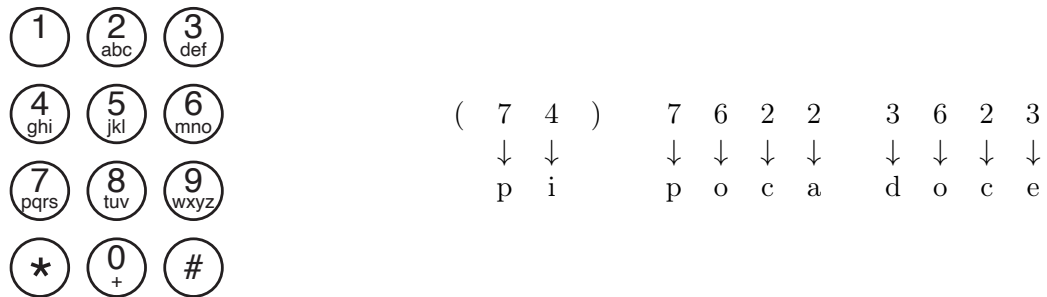
| Exemplo de entrada 3 | Exemplo de saída 3 |
|----------------------|--------------------|
| 0 | * * |

Explicação do exemplo 3: para representar o número zero o Ogro não mostra nenhum dedo da mão esquerda ou da mão direita.

Teclado

Nome do arquivo: “teclado.x”, onde x deve ser `c`, `cpp`, `pas`, `java`, `js` ou `py`

Os teclados de telefones mostram teclas com os dígitos de 0 a 9, para que possamos digitar o número do telefone que queremos contactar, como na figura abaixo. Mas as teclas também mostram letras, que podem ser usadas por exemplo para facilitar a memorização de um número de telefone em particular. Por exemplo, para memorizar o número (74) 7622 3623 podemos associar esse número à cadeia de caracteres `pipocadoce`:



Claramente, um número pode ser representado por diferentes cadeias de caracteres. Por exemplo, o número 3482 pode ser representado por `fita`, `diva`, `dita`, `egua`, e muitas outras cadeias de caracteres.

Dados um número e uma lista de cadeias de caracteres, sua tarefa é determinar quantas cadeias de caracteres da lista podem representar o número dado.

Entrada

A primeira linha da entrada contém uma cadeia de caracteres N , o número de telefone. A segunda linha contém um inteiro M , o número de cadeias de caracteres na lista. Cada uma das M linhas seguintes contém uma cadeia de caracteres C_i .

Saída

Seu programa deve produzir uma única linha na saída, contendo um único inteiro, o número de cadeias de caracteres da lista que podem representar o número dado.

Restrições

- $1 \leq \text{comprimento de } N \leq 1\,000$
- N contém apenas dígitos entre 2 e 9
- $1 \leq M \leq 1\,000$
- C_i contém apenas letras minúsculas não acentuadas, para $1 \leq i \leq M$
- $1 \leq \text{comprimento de } C_i \leq 1\,000$, para $1 \leq i \leq M$
- C_i são todas distintas para $1 \leq i \leq M$

Informações sobre a pontuação

- Para um conjunto de casos de testes valendo 13 pontos, comprimento de $N = 1$ e $M \leq 20$.
- Para um conjunto de casos de testes valendo outros 87 pontos, nenhuma restrição adicional.

| Exemplo de entrada 1 | Exemplo de saída 1 |
|---|--------------------|
| 3482 4 fita regua milho diva | 2 |

Explicação do exemplo 1: fita e diva são representações do número 3482.

| Exemplo de entrada 2 | Exemplo de saída 2 |
|---|--------------------|
| 7476223623 5 pipoca pipocadoce misobafobe doce docepipoca | 1 |

Explicação do exemplo 2: apenas pipocadoce representa o número 7476223623.

| Exemplo de entrada 3 | Exemplo de saída 3 |
|-----------------------------------|--------------------|
| 4444 3 mono tudo nada | 0 |

Explicação do exemplo 3: nenhuma das cadeias de caracteres representa o número 4444.

Falha de segurança

Nome do arquivo: “falha.x”, onde x deve ser `c`, `cpp`, `pas`, `java`, `js` ou `py`

Rafael foi contratado como programador por um grande banco que está atualizando todo o sistema computacional. O novo sistema vai ser instalado amanhã, mas Rafael acabou de descobrir uma falha grave na nova autenticação para acesso às contas do banco: se um usuário digitar como senha uma cadeia de caracteres que contenha, como sub-cadeia contígua, a senha correta para esse usuário, o sistema se confunde e permite o acesso.

Por exemplo, se a senha correta é ‘senhafraca’ e o usuário digitar

‘quesenhafracameu’ ou ‘senhafraca123’,

o sistema permite o acesso. Note que nesse caso o sistema não permite o acesso se o usuário digitar

‘senha’ ou ‘nhafra’ ou ‘senha123fraca’.

O chefe de Rafael chamou um programador mais experiente para alterar a autenticação do novo sistema, mas solicitou que Rafael determinasse, para o conjunto de senhas existentes, quantos pares ordenados (A, B) de usuários distintos existem tal que o usuário A , usando sua senha, consegue acesso à conta do usuário B . Você poderia por favor ajudar Rafael?

Entrada

A primeira linha da entrada contém um número inteiro N , o número de usuários no sistema. Cada uma das N linhas seguintes contém uma senha S_i , a senha do i -ésimo usuário.

Saída

Seu programa deve produzir uma única linha, contendo um único inteiro, o número de pares ordenados (A, B) de usuários distintos tal que o usuário A , usando sua senha, consegue acesso à conta do usuário B .

Restrições

- $1 \leq N \leq 20000$
- S_i inicia com letra minúscula sem acento e contém apenas letras minúsculas sem acento e dígitos de 0 a 9, para $1 \leq i \leq N$
- $1 \leq \text{comprimento de } S_i \leq 10$

Informações sobre a pontuação

- Para um conjunto de casos de testes valendo 12 pontos, comprimento de $S_i = 1$ e $N \leq 1000$.
- Para um conjunto de casos de testes valendo outros 28 pontos, $N \leq 2000$.
- Para um conjunto de casos de testes valendo outros 60 pontos, nenhuma restrição adicional.

| Exemplo de entrada 1 | Exemplo de saída 1 |
|-----------------------|--------------------|
| 3 xxx x23 xx | 1 |

Explicação do exemplo 1: o primeiro usuário consegue acesso à conta do terceiro usuário.

| Exemplo de entrada 2 | Exemplo de saída 2 |
|----------------------|--------------------|
| 3 a a a8 | 4 |

Explicação do exemplo 2: o primeiro usuário consegue acesso à conta do segundo usuário, o segundo usuário consegue acesso à conta do primeiro usuário, e o terceiro usuário consegue acesso à contas tanto do primeiro como do segundo usuário, totalizando quatro pares de usuários com falha de acesso.

| Exemplo de entrada 3 | Exemplo de saída 3 |
|------------------------------------|--------------------|
| 5 jus justa ta us t | 6 |

Explicação do exemplo 3: o primeiro usuário consegue acesso à conta do quarto usuário, o segundo usuário consegue acesso às contas dos outros quatro usuários, o terceiro usuário consegue acesso à conta do quinto usuário, totalizando seis pares de usuários com falha de acesso.

Festa olímpica

Nome do arquivo: “**festa.x**”, onde **x** deve ser **c**, **cpp**, **pas**, **java**, **js** ou **py**

Os atletas da Nlogônia obtiveram o melhor resultado do país em olimpíadas, e para comemorar o rei decidiu dar uma grande festa no Palácio Real. Todos os atletas foram convidados, mas o rei quer também convidar alguns de seus súditos.

Como não é possível convidar todos os súditos, o rei determinou que a seguinte Lei seja utilizada para calcular a lista de convidados:

LEI ESPECIAL SOBRE COMEMORAÇÃO DAS OLIMPÍADAS

Por ordem de Sua Majestade, fiquem todos sabendo que:

- Os N súditos de Nlogônia serão numerados $1, 2, 3, \dots, N$ e uma lista ordenada será criada com os números dos súditos. A primeira posição da lista será 1.
- Um número M de turnos serão então executados; em cada turno i , será sorteado um número T_i que será usado para remover súditos da lista, da seguinte forma: no turno i , devem ser removidos da lista todos os súditos que ainda continuam na lista e que ocupam posições que são múltiplas de T_i ; ou seja, devem ser removidos os súditos que estão nas posições $(T_i, 2T_i, 3T_i, \dots)$ da lista corrente. Ao final do turno, para não haver posições vazias na lista (cujos súditos foram removidos) a lista é reagrupada, mantendo-se a mesma ordem relativa, e contendo apenas os números dos súditos remanescentes.
- Os súditos que permanecerem na lista ao final dos M turnos serão convidados para a grande festa de comemoração do resultado das olimpíadas.

Dados o número de súditos e os números sorteados em cada turno, sua tarefa é determinar os súditos que serão convidados de acordo com a Lei Especial.

Entrada

A primeira linha da entrada contém um número inteiro N , o número de súditos de Nlogônia. A segunda linha contém um inteiro M , o número de turnos. Cada uma das M linhas seguintes contém um inteiro T_i , o número que foi sorteado para o turno i .

Saída

Seu programa deve produzir a lista de convidados de acordo com a Lei Especial, com uma linha para cada convidado, cada linha contendo somente o número de um convidado. Como a lista total dos convidados pode ser muito grande, o rei ordenou que, caso o número de convidados seja maior que 10.000, você deve listar apenas os 10.000 primeiros (ou seja, os com menores números) convidados.

Restrições

- $2 \leq N \leq 1\,000\,000\,000$
- $1 \leq M \leq 5\,000$
- $2 \leq T_i \leq 100\,000$ para $1 \leq i \leq M$

Informações sobre a pontuação

- Para um conjunto de casos de testes valendo 17 pontos, $N \leq 100$ e $M \leq 10$.
- Para um conjunto de casos de testes valendo outros 22 pontos, $N \leq 400\,000$ e $M \leq 5\,000$.
- Para um conjunto de casos de testes valendo outros 21 pontos, $T_i = 2$ para $1 \leq i \leq M$.

- Para um conjunto de casos de testes valendo outros 40 pontos, nenhuma restrição adicional.

| Exemplo de entrada 1 | Exemplo de saída 1 |
|----------------------|--------------------|
| 10 | 1 |
| 2 | 3 |
| 2 | 7 |
| 3 | 9 |

Explicação do exemplo 1: A lista inicial é

1 2 3 4 5 6 7 8 9 10

Após remover todos os que ocupam posições múltiplas de 2 a lista é

1 3 5 7 9

Após remover todos os que ocupam posições múltiplas de 3 a lista é

1 3 7 9

| Exemplo de entrada 2 | Exemplo de saída 2 |
|----------------------|--------------------|
| 6 | 1 |
| 3 | |
| 2 | |
| 2 | |
| 2 | |

Explicação do exemplo 2: A lista inicial é

1 2 3 4 5 6

Após remover todos os que ocupam posições múltiplas de 2 a lista é

1 3 5

Após remover todos os que ocupam posições múltiplas de 2 a lista é

1 5

Após remover todos os que ocupam posições múltiplas de 2 a lista é

1

| Exemplo de entrada 3 | Exemplo de saída 3 |
|----------------------|--------------------|
| 5 | 1 |
| 1 | 2 |
| 10 | 3 |
| | 4 |
| | 5 |

| Exemplo de entrada 4 | Exemplo de saída 4 |
|----------------------|--------------------|
| 1000000 | 1 |
| 3 | 3 |
| 2 | 7 |
| 15 | 9 |
| 3 | ... |
| | 32137 |
| | 32139 |

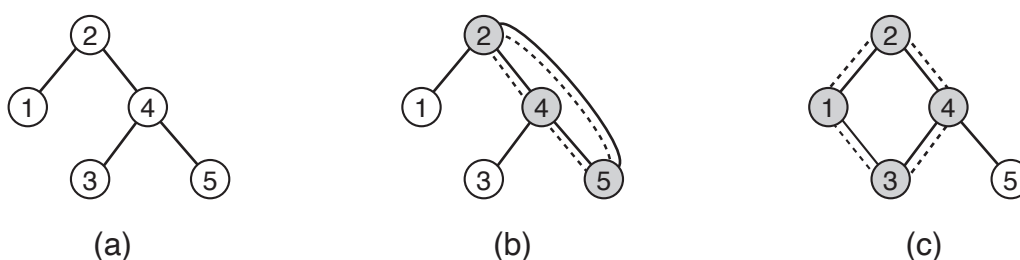
Dona Minhoca

Nome do arquivo: “minhoca.x”, onde x deve ser `c`, `cpp`, `pas`, `java`, `js` ou `py`

Dona Minhoca construiu uma bela casa, composta de N salas conectadas por $N - 1$ túneis. Cada túnel conecta exatamente duas salas distintas, e pode ser percorrido em qualquer direção. A casa de dona Minhoca foi construída de modo que, percorrendo os túneis, é possível partir de qualquer sala e chegar a qualquer outra sala da casa.

Dona Minhoca quer se exercitar, e para isso planeja construir um túnel adicional, de modo a criar um “ciclo” de salas e túneis. Vamos chamar de *comprimento* do ciclo o número de salas do ciclo.

A figura (a) abaixo mostra um exemplo de casa. É possível obter um ciclo de comprimento três construindo um túnel entre as salas 2 e 5, ou um ciclo de comprimento quatro construindo um túnel entre as salas 1 e 3.



Dada a descrição da casa de dona Minhoca, escreva um programa para determinar o número de salas do ciclo de maior comprimento que é possível construir, e de quantas maneiras é possível construir um ciclo com esse comprimento.

Entrada

A primeira linha da entrada contém um inteiro N , o número de salas da casa de dona Minhoca. As salas são identificadas por números de 1 a N . Cada uma das $N - 1$ linhas seguintes contém dois inteiros X e Y , indicando que há um túnel entre a sala X e a sala Y .

Saída

Seu programa deve produzir duas linhas. A primeira linha deve conter somente um inteiro, o número de salas do ciclo de maior comprimento que é possível construir. A segunda linha deve conter somente um inteiro, o número de ciclos distintos que é possível contruir com esse comprimento.

Restrições

- $3 \leq N \leq 50\,000$
- $1 \leq X \leq N$; $1 \leq Y \leq N$; $X \neq Y$
- nos testes, o número de possíveis ciclos distintos é menor do que 100 000 000

Informações sobre a pontuação

- Para um conjunto de casos de testes valendo 40 pontos, $N \leq 5\,000$
- Para um conjunto de casos de testes valendo outros 60 pontos, nenhuma restrição adicional.

| Exemplo de entrada 1 | Exemplo de saída 1 |
|-------------------------------|--------------------|
| 5 1 2 2 4 4 5 4 3 | 4 2 |

Explicação do exemplo 1: este exemplo corresponde à figura do enunciado. O comprimento do maior ciclo possível é quatro, e há duas maneiras de conseguir um ciclo desse comprimento: criando um túnel entre as salas 1 e 3 ou entre as salas 1 e 5.

| Exemplo de entrada 2 | Exemplo de saída 2 |
|--|--------------------|
| 8 1 2 2 3 3 4 3 6 5 3 1 8 1 7 | 5 6 |

Explicação do exemplo 2: o comprimento do maior ciclo possível é cinco, e há seis maneiras de conseguir isso: criando um túnel entre os pares de salas (4, 7) (4, 8), (5, 7), (5, 8), (6, 7) ou (6, 8).