Competidor(a): _____

Número de inscrição: _____–_____ *(opcional)*

OBI2021

Caderno de Tarefas

Modalidade Programação • Nível 2 • Fase 3

25 de setembro de 2021

A PROVA TEM DURAÇÃO DE 5 horas

Promoção:

Sociedade Brasileira de Computação

Apoio:

v1.0

Instruções

LEIA ATENTAMENTE ESTAS INSTRUÇÕES ANTES DE INICIAR A PROVA

- Este caderno de tarefas é composto por 11 páginas (não contando a folha de rosto), numeradas de 1 a 11. 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.

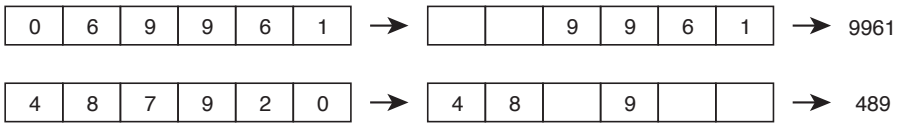
Casamento de inteiros

Nome do arquivo: "`casamento.x`", *onde* `x` *deve ser* `c`, `cpp`, `pas`, `java`, `js` *ou* `py`

Vamos definir a operação de *casamento* de dois números inteiros A e B da seguinte forma:

- inicialmente fazemos A e B terem o mesmo número de dígitos, adicionando zeros à esquerda conforme necessário;
- então cada dígito de A (do menos significativo ao mais significativo) é comparado com o dígito correspondente de B, e o dígito de menor valor é eliminado do número a que pertence (se os dígitos são iguais nenhum é eliminado).
- o resultado da operação de casamento é o par de números inteiros formados pelos dígitos remanescentes de A e B. No caso de não haver digito remanescente para um dos números, o resultado para esse número é -1.

Por exemplo, considere o casamento de 69961 com 487920:



O resultado do casamento é o par de números 489 e 9961.

Dados dois números inteiros, sua tarefa é determinar o resultado do casamento desses dois números.

Entrada

A primeira linha da entrada contém um número inteiro A, a segunda linha contém um número inteiro B.

Saída

Seu programa deve produzir uma única linha, contendo os dois números inteiros produzidos pelo casamento dos números dados, em ordem não decrescente.

Restrições

- $1 \leq A \leq 10^9$
- $1 \leq B \leq 10^9$

Informações sobre a pontuação

- Para um conjunto de casos de testes valendo 22 pontos, $100 \leq A \leq 999$ e $100 \leq B \leq 999$.
- Para um conjunto de casos de testes valendo outros 78 pontos, nenhuma restrição adicional.

Exemplo de entrada 1	**Exemplo de saída 1**
69961 487920	489 9961

Explicação do exemplo 1: este exemplo corresponde ao exemplo mostrado no enunciado.

Exemplo de entrada 2	**Exemplo de saída 2**
5678 1234	-1 5678

Explicação do exemplo 2: todos os dígitos são eliminados do segundo número.

Exemplo de entrada 3	**Exemplo de saída 3**
21 12	2 2

Explicação do exemplo 3: o dígito 1 é eliminado dos dois números.

Exemplo de entrada 4	**Exemplo de saída 4**
200 100	0 200

Explicação do exemplo 4: o dígito 1 é eliminado do segundo número.

Cubo e quadrado

Nome do arquivo: "`cubo.x`", *onde* `x` *deve ser* `c`, `cpp`, `pas`, `java`, `js` *ou* `py`

O número 729 tem uma particularidade interessante: é ao mesmo tempo o cubo e o quadrado de um número inteiro ($729 = 27^2$ e $729 = 9^3$). Outro número com essa particularidade é 4096 ($4096 = 64^2$ e $4096 = 16^3$).

Sua tarefa é, dados dois números inteiros A e B, determinar quantos números no intervalo entre A e B são ao mesmo tempo cubo e quadrado de um número inteiro.

Entrada

A primeira da entrada contém um inteiro A, o limite inferior do intervalo de interesse, a segunda linha contém um inteiro B, o limite superior do intervalo de interesse (A e B fazem parte do intervalo de interesse).

Saída

Seu programa deve produzir uma única linha na saída, contendo um único inteiro, a quantidade de números que são ao mesmo tempo cubo e quadrado de um número inteiro, para todos os números do intervalo de interesse.

Restrições

- $1 \leq A < B \leq 100\ 000\ 000$

Informações sobre a pontuação

- Para um conjunto de casos de testes valendo 30 pontos, $B \leq 100\ 000$.
- Para um conjunto de casos de testes valendo outros 70 pontos, nenhuma restrição adicional.

Exemplo de entrada 1	**Exemplo de saída 1**
64 729	2

Explicação do exemplo 1: os números que são cubo e quadrado de um outro número no intervalo entre 64 e 729 são somente 64 e 729, portanto a resposta é 2.

Exemplo de entrada 2	**Exemplo de saída 2**
3000 5000	1

Explicação do exemplo 2: 4096 é o único número no intervalo entre 3000 e 5000 que é cubo e quadrado de um outro número, portanto a resposta é 1.

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 $1, 2, 3, \ldots, N$ 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 $(T_i, 2T_i, 3T_i, \ldots)$ 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 Nlogôgina. 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 2 2 3	1 3 7 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 3 2 2 2	1

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 10	1 2 3 4 5

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

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 $1 \leq i \leq N$
- $1 \leq$ 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.

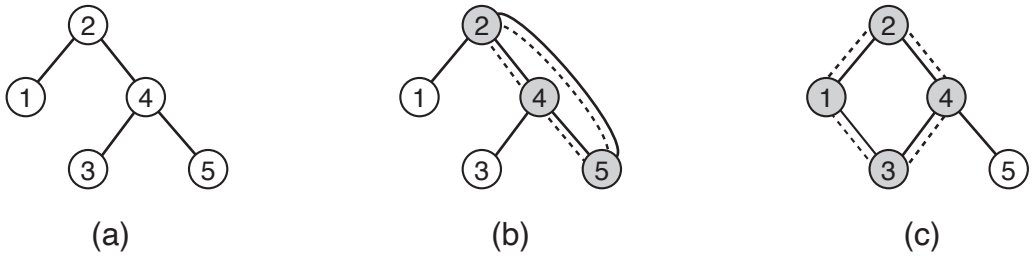
Dona Minhoca

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 $(6, 8)$.