Competidor(a): _____

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

Este Caderno de Tarefas não pode ser levado para casa após a prova. Após a prova entregue este Caderno de Tarefas para seu professor guardar. Os professores poderão devolver os Cadernos de Tarefas aos competidores após o término do período de aplicação das provas (1 de outubro de 2022).

Olimpíada Brasileira de Informática

OBI2022

Caderno de Tarefas

Modalidade Programação • Nível Sênior • Fase 3

1 de outubro de 2022

A PROVA TEM DURAÇÃO DE 5 HORAS

Promoção:

Sociedade Brasileira de Computação

Apoio:

FUNDAÇÃO BEHRING

Coordenação:

INSTITUTO DE COMPUTAÇÃO UNICAMP

Instruções

LEIA ATENTAMENTE ESTAS INSTRUÇÕES ANTES DE INICIAR A PROVA

- Este caderno de tarefas é composto por 10 páginas (não contando a folha de rosto), numeradas de 1 a 10. 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.

Caravana

Nome do arquivo: `caravana.c`, `caravana.cpp`, `caravana.pas`, `caravana.java`, `caravana.js` *ou* `caravana.py`

No deserto da Nlogônia, uma longa caravana de camelos carregados de especiarias está parada num oásis para descansar. O chefe da caravana notou que alguns camelos pareciam mais cansados do que os outros, e descobriu que cada camelo estava carregando um peso diferente, de forma que alguns camelos carregam um peso muito maior do que outros e portanto se cansam mais.

Aproveitando a parada para descanso, o chefe da caravana quer redistribuir as especiarias entre os camelos, de forma que todos os camelos carreguem exatamente o mesmo peso.

Dados os pesos carregados por cada camelo antes da parada, escreva um programa que determine, para cada camelo, qual o peso que deve ser retirado ou adicionado, para que todos carreguem exatamente o mesmo peso.

Entrada

A primeira linha contém um inteiro N, o número de camelos na caravana. Os camelos são numerados de 1 a N. Cada uma das linhas seguintes contém um inteiro P_i, o peso que o camelo de número i carregava antes da parada. Os camelos são dados em ordem crescente de numeração.

Saída

Para cada camelo da caravana, seu programa deve produzir uma linha, o valor que deve ser adicionado ou retirado desse camelo para que todos os camelos carreguem o mesmo peso. A ordem dos camelos na saída deve ser a mesma ordem dada na entrada. Para todos os casos de teste o peso que cada camelo deve carregar é um número inteiro.

Restrições

- $1 \leq N \leq 1\ 000$
- $1 \leq P_i \leq 10\ 000$ para $1 \leq i \leq N$

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
3 100 104 108	4 0 -4

Exemplo de entrada 2	**Exemplo de saída 2**
5 30 40 23 5 32	-4 -14 3 21 -6

Exemplo de entrada 3	**Exemplo de saída 3**
3 10000 10000 10000	0 0 0

Dígitos

Nome do arquivo: `dígitos.c`, `dígitos.cpp`, `dígitos.pas`, `dígitos.java`, `dígitos.js` *ou* `dígitos.py`

Joãozinho te propôs o seguinte desafio: ele escolheu dois inteiros A e B, com $1 \leq A \leq B \leq 10^{1000}$, e escreveu na lousa todos os inteiros entre A e B, em sequência, porém colocando um espaço após cada dígito, de forma a não ser possível ver quando um número termina ou começa. Por exemplo, se Joãozinho escolher $A = 98$ e $B = 102$, ele escreveria a sequência "9 8 9 9 1 0 0 1 0 1 1 0 2".

Seu desafio é: dada a lista de dígitos escritos na lousa, encontrar os valores de A e B. Caso exista mais de uma possibilidade para os valores que geraria a lista, você deve encontrar uma em que o valor de A é o menor possível.

É garantido que a lista de dígitos da lousa tem no máximo tamanho 1000.

Entrada

A primeira linha da entrada contém um único inteiro N, indicando o número de dígitos. A segunda linha contém N inteiros d_i, indicando os dígitos escritos.

Saída

Imprima o menor valor possível de A.

Restrições

- $1 \leq N \leq 1000$
- $0 \leq d_i \leq 9$

Informações sobre a pontuação

- Para um conjunto de casos de testes valendo 21 pontos, $1000 \leq A \leq B \leq 9999$.
- Para outro conjunto de casos de testes valendo 23 pontos, $B = A + 1$.
- Para outro conjunto de casos de testes valendo 40 pontos, $A, B < 10^6$.
- Para outro conjunto de casos de testes valendo 16 pontos, nenhuma restrição adicional.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
6 1 2 3 1 2 4	123

Exemplo de entrada 2	**Exemplo de saída 2**
6 8 9 1 0 1 1	8

Quadrado

Nome do arquivo: `quadrado.c`, `quadrado.cpp`, `quadrado.pas`, `quadrado.java`, `quadrado.js` *ou* `quadrado.py`

Um *quadrado fantástico* é um conjunto de números inteiros positivos dispostos em N linhas por N colunas tal que:

- Não há números repetidos no quadrado.
- A média dos números em cada linha é um número inteiro que está presente na linha.
- A média dos números em cada coluna é um número inteiro que está presente na coluna.

Entrada

A primeira e única linha da entrada contém um número inteiro N, indicando a dimensão do quadrado.

Saída

Seu programa deve produzir N linhas, cada uma contendo N números inteiros X_i, representando um quadrado fantástico.

Restrições

- $1 \leq N \leq 40$
- $1 \leq X_i \leq 1000000$

Informações sobre a pontuação

- Para um conjunto de casos de testes valendo 44 pontos, $1 \leq N$ é ímpar.
- Para outro conjunto de casos de testes valendo 56 pontos, nenhuma restrição adicional.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
`1`	`1`

Exemplo de entrada 2	**Exemplo de saída 2**
`2`	`-1`

Exemplo de entrada 3	**Exemplo de saída 3**

`3`	`1 2 3`
	`4 5 6`
	`7 8 9`

Exemplo de entrada 4	**Exemplo de saída 4**
4	1 2 3 6 7 8 9 12 13 14 15 18 31 32 33 36

Dona Minhoca

Nome do arquivo: `minhoca.c`, `minhoca.cpp`, `minhoca.pas`, `minhoca.java`, `minhoca.js` *ou* `minhoca.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.

Para deixar sua casa mais segura, Dona Minhoca decidiu instalar radares anti-furto em algumas das salas. Ela comprou K radares, e deve agora decidir em quais salas colocará um radar. Além disso, todos radares terão um *raio de alcance*, cujo valor R também deve ser decidido. Quando um radar com raio de alcance R é instalado na sala s, todas as salas com distância menor ou igual a R da sala s (incluindo a própria s) ficam sob o alcance do radar, e estarão protegidas.

Devido à política estranha de cobrança da empresa de radares, todos os K radares devem ter o mesmo raio de alcance. Dona Minhoca então se pergunta: qual seria o menor valor possível para R, tal que, se o raio de alcance dos radares for R, é possível escolher K salas para instalar os radares de forma que todas as N salas estejam protegidas?

Entrada

A primeira linha da entrada contém dois inteiros N e K, indicando o número de salas, e de radares que Dona Minhoca possui. As $N-1$ linhas seguintes contém dois inteiros a_i e b_i cada, indicando que existe um túnel conectando essas duas salas.

Saída

Seu programa deve produzir uma única linha, contendo um único inteiro, o menor valor possível para R.

Restrições

- $1 \leq N \leq 300000$
- $1 \leq K < N$
- $a_i \neq b_i$

Informações sobre a pontuação

- Para um conjunto de casos de testes valendo 25 pontos, $K = 1$
- Para outro conjunto de casos de testes valendo 17 pontos, o túnel i conecta as salas i e $i+1$ ($1 \leq i \leq N-1$). Ou seja, a casa possui o formato de uma linha reta.
- Para outro conjunto de casos de testes valendo 17 pontos, $N, K \leq 100$
- Para outro conjunto de casos de testes valendo 41 pontos, nenhuma restrição adicional.

Exemplos

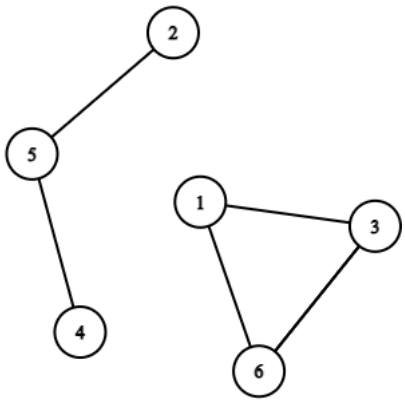
Exemplo de entrada 1	**Exemplo de saída 1**
6 1 1 2 2 3 3 4 4 5 4 6	2

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

Construção de Rodovia

Nome do arquivo: `rodovia.c`, `rodovia.cpp`, `rodovia.pas`, `rodovia.java`, `rodovia.js` *ou* `rodovia.py`

O reino de Nlogonia é composto por N cidades, numeradas de 1 a N, e M rodovias bidirecionais, ou seja, é possível usar uma rodovia (x, y) para ir tanto da cidade x à cidade y, quanto de y a x.



Vamos definir o valor da *conectividade* do reino como o número de pares não ordenados (x, y), com $x \neq y$, tais que é possível viajar de x a y (talvez indiretamente, passando por outras cidades intermediárias pelo caminho). Na figura acima, por exemplo, o valor da conectividade é 6, sendo que os pares em questão são: $(1, 3), (1, 6), (3, 6), (2, 4), (2, 5)$ e $(4, 5)$.

O governo de Nlogonia está planejando construir uma única nova rodovia (A, B), também bidirecional. Muitas discussões estão sendo feitas para escolher a rodovia ideal, porém no momento, o maior receio é se há alguma possibilidade de ser feita uma escolha que seja considerada redundante pelos habitantes do reino. Em particular, foi dada a você a tarefa de descobrir se existe algum par (A, B) de cidades tal que:

- $A \neq B$
- Não existe nenhuma rodovia originalmente no reino que conecta as cidades A e B.
- Caso adicionarmos a rodovia (A, B), o valor da conectividade do reino permanecerá o mesmo.

Também foi pedido que, caso existam pares que cumpram todas as condições, você deve informar algum deles. Caso tenha mais de um par válido, você pode escolher qualquer um deles.

Entrada

A primeira linha da entrada contém dois inteiros N e M, indicando o número de cidades e rodovias. Seguem M linhas contendo dois inteiros x_i e y_i cada, indicando que existe uma rodovia que pode ser usada para viajar entre as cidades x e y.

Saída

Caso exista algum par que satisfaça todas as condições, seu programa deve imprimir qualquer um desses pares, em uma única linha. Caso contrário, imprima -1.

Restrições

- $1 \leq N \leq 200000$
- $1 \leq M \leq 400000$
- $x_i \neq y_i$
- Não existem duas rodovias que conectam o mesmo par de cidades.

Informações sobre a pontuação

- Para um conjunto de casos de testes valendo 43 pontos, $N, M \leq 100$.
- Para outro conjunto de casos de testes valendo 57 pontos, nenhuma restrição adicional.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**

`4 3`
`1 2`
`2 4`
`4 1`
	`-1`

Exemplo de entrada 2	**Exemplo de saída 2**

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