



# SEMANA DE INFORMÁTICA

## Maratona de Programação Seletiva Interna 2019 da UFV

Caderno de Problemas  
UFV, 23 de agosto de 2019

### Instruções:

- Este caderno contém 13 problemas: as páginas estão numeradas de 1 a 13, não contando esta página de rosto. Verifique se o caderno está completo.
- Em todos os problemas, a entrada de seu programa deve ser lida da *entrada padrão*. A saída deve ser escrita na *saída padrão*.

---

Universidade Federal de Viçosa

---

## Informações gerais

- A entrada deve ser lida da entrada padrão e a saída escrita na saída padrão
- Todas as linhas, inclusive a última, tanto da entrada quanto da saída, devem ter fim-de-linha
- Sempre que uma linha contém vários valores, eles são separados por um único espaço em branco e nenhum outro espaço deve aparecer (nem linha em branco), tanto na entrada quanto na saída
- O alfabeto inglês é sempre usado, não deve haver acentos, cedilhas, etc, nem na entrada nem na saída

## Tempos limites de execução

ATUALIZAR!

- O. Transportando Doce de Leite na Nlogonia – 8 s
- Nas demais – 1 s
- ATENÇÃO! Os tempos foram calibrados para C++ e são os mesmos para todas as linguagens; não há garantia que códigos em outras linguagens executem dentro destes tempos.

*“Esta é uma obra de ficção, qualquer semelhança com nomes, pessoas, fatos ou situações da vida real terá sido mera coincidência.”*

Fontes das figuras ilustrativas usadas nos problemas da prova:

ATUALIZAR!

A <https://twitter.com/Zanfa/status/1153189389868175360>

B <https://www.google.com.br/maps/search/2+irmaos/@-20.7455553,-42.8654407,14z>

C <http://gianthamster.com/2010/03/contest-putting-caplin-rous-in-my-life/>

D Adaptada de <https://tvefamosos.uol.com.br/noticias/redacao/2017/03/10/globo-divulga-estrela-de-senhora-do-destino-com-memes-de-nazare-confusa.htm>

E Adaptada de <https://texashillcountry.com/hungry-meet-and-eat-the-biggest-pizza-in-texas-at-big-lous/>

F <https://www.redbubble.com/people/chasekareng/works/33095702-the-golden-ratio-pizza?p=poster>

G <https://www.facebook.com/memphiszoo/posts/the-capybaras-had-an-egg-cellent-morning-they-were-egg-cited-for-a-special-enric/10156129241068815/>

H Adaptada de <https://www.worldatlas.com/articles/10-delightful-facts-about-capybaras.html>

I <https://www.instagram.com/p/BzWNSI7BnLb/>

J -

K <http://gianthamster.com/2013/03/my-3rd-birthday/>

L Adaptada de <http://wallyinuruguay.blogspot.com/2011/02/two-pesos.html>

M -

## Problema A. Maratona de Programação 2 Irmãos

Arquivo-fonte: "doisirmaos.x", onde x deve ser c, cpp, java ou py

Durante a viagem para a Maratona Mineira de Programação 2019, em Alfenas, o motorista da van postulou uma conjectura que deixou os competidores curiosos: em toda cidade há pelo menos um estabelecimento com o nome "2 irmãos".

Após passarem por diversas cidades no caminho até Alfenas, os passageiros observaram que a conjectura parecia realmente verdadeira! Perplexos, alguns alunos decidiram criar um software para extrair a lista de empresas em cada cidade do Brasil (utilizando o Google Maps) e, então, verificar se a conjectura é verdadeira.

A extração da lista, incluindo um pré-processamento para facilitar a verificação, foi um sucesso. Porém, ainda falta o módulo para processar o texto e verificar se a conjectura é válida (ou se há alguma cidade que funciona como contra-exemplo).

Dado um nome  $X$  de um comércio,  $X$  é considerado um comércio do tipo "2 irmãos" se, e somente se,  $X$  contiver o número 2 seguido (imediatamente) pela palavra *irmaos*.

### Entrada

Inicialmente há uma linha com um inteiro  $N$  indicando o número de cidades ( $N \leq 100$ ). A seguir, há  $N$  descrições de cidades.

A descrição de cada cidade começa com uma linha com seu nome (até 100 caracteres). A seguir, uma linha com um inteiro  $K$ , o número de comércios na cidade e, então,  $K$  linhas, cada uma com o nome de um comércio ( $K \leq 100$ ). O sistema de pré-processamento garante que os nomes contêm apenas letras minúsculas, números e espaço em branco (cada par de palavras/números consecutivos é separado por exatamente um espaço em branco e cada nome tem no máximo 100 caracteres).

### Saída

Escreva uma linha contendo a frase "A conjectura parece verdadeira!" se a conjectura for verdadeira para a entrada. Caso contrário, imprima uma linha contendo "Conjectura falsa:" seguida pela lista de cidades onde não há comércio do tipo "2 irmãos", uma por linha, na ordem em que aparecem na entrada.

### Exemplos

Entrada	Saída
<pre>3 Vicoso 3 dois irmaos rodizio de pizza loja do ze das capivaras e sua irma curso de portugues 2 irmoes Governador Valadares 1 two brothers burger Ponte Nova 2 construtora de pontes 2 irmaos bar 2 filhos dos mesmos pais</pre>	<pre>Conjectura falsa: Vicoso Governador Valadares</pre>
<pre>3 Belo Oriente 2 pastelaria 2 irmaos do bruno agropecuaria dos outros 2 irmaos dele Braunas Rubim 2 loja de subsequencias 2 irmaos cinema 2 filhos de francisco Alvinopolis 1 loja 2 irmaos 4 irmas e 8 tios</pre>	<pre>A conjectura parece verdadeira!</pre>

### As franquias de maior sucesso no Brasil:



## Problema B. Pequenas empresas grandes negócios

Arquivo-fonte: "empresas.x", onde x deve ser c, cpp, java ou py

Para fomentar a abertura de novos negócios em Viçosa, o CENTEV contratou a sua empresa (a "Software 2 irmãos") para criar um sofisticado software de inteligência artificial que, dado o tipo de um estabelecimento comercial, gera automaticamente um nome para ele.

Após analisar vários exemplos de nomes de estabelecimento, você chegou à conclusão que a melhor forma de desenvolver seu software consiste em utilizar a estratégia mais simples e popular possível: basta utilizar o tipo do estabelecimento seguido por "2 irmãos" (com exatamente um espaço em branco entre o tipo do estabelecimento e o "2 irmãos").

Para que seu software tenha pelo menos um pouco de sofisticação, foi decidido que você utilizará duas variações dessa nomenclatura: tipos de estabelecimento com número par de letras utilizarão "2 irmãos" no nome, enquanto os com número ímpar utilizarão "dois irmãos".

Sua tarefa nesta questão consiste em implementar tal software.



### Entrada

A entrada é uma linha com no máximo 100 caracteres contendo apenas letras minúsculas e espaços em branco (cada par de palavras consecutivas é separado por exatamente um espaço em branco). Essa linha denota o tipo do estabelecimento comercial para o qual você deverá gerar um nome.

### Saída

A saída deve ser composta por uma linha contendo o nome gerado para a empresa.

### Exemplos

Entrada	Saída
rodizio de pizza	rodizio de pizza 2 irmãos
hotel para capivaras	hotel para capivaras 2 irmãos
loja de doce de leite	loja de doce de leite dois irmãos
ovos de capivara	ovos de capivara 2 irmãos

## Problema C. Rodízio de pizza 2 irmãos

Arquivo-fonte: "rodizio.x", onde x deve ser c, cpp, java ou py

Após os campeonatos da disciplina Programação Competitiva da UFV (INF333), alunos e professores costumam ir ao Rodízio de Pizza 2 irmãos (nomeado com o programa que você desenvolveu na questão anterior) para saborearem pizzas e discutirem sobre os problemas da competição. Porém, como as provas costumam terminar às 21 horas, os funcionários da pizzaria se irritaram por terem que trabalhar até mais tarde.

Graças a uma pesquisa do Departamento de Nutrição da UFV, um dos 2 donos do rodízio descobriu que os clientes sempre vão embora logo após estarem satisfeitos (leia-se: empanturrados de pizza).

Pensando nisso, ele teve uma ideia genial: quando o pessoal da INF333 chegar ao rodízio, eles deverão preencher um formulário informando quantos pedaços de pizza cada um come. Além disso, cada pessoa deverá informar os sabores que gosta.

Com base no formulário, os cozinheiros preparam uma quantidade de pizza suficiente para satisfazer a todos (assim que terminam de preparar as pizzas, já podem ir para casa descansar) e os garçons as servem o mais rápido possível. Devido a diferentes tempos de cozimento e a dificuldades de preparação, alguns sabores de pizza podem ser preparados mais rapidamente que outros. Além disso, a quantidade de ingredientes em estoque costuma limitar a quantidade de pizzas de determinados sabores que podem ser preparadas.

Sua empresa foi contratada para criar um software capaz de determinar qual o tempo mínimo durante o qual os cozinheiros da pizzaria deverão trabalhar de modo a prepararem pizza suficiente para empanturrar todo o pessoal da INF333.

Obs: assume-se que o pessoal da INF333 é muito guloso, eles comem sempre que lhes oferecem uma pizza da qual gostam.



### Entrada

A entrada começa com uma linha contendo um número  $P$  ( $1 \leq P \leq 500$ ) representando a quantidade de sabores de pizza que são servidos no Rodízio de Pizza 2 irmãos.

A seguir, há  $P$  linhas, uma para cada sabor de pizza. A  $i$ -ésima linha contém dois inteiros  $Ped_i$  e  $T_i$  ( $1 \leq Ped_i \leq 1000000$  e  $1 \leq T_i \leq 1000000000$ ) indicando, respectivamente, a quantidade de pedaços de pizza desse sabor que podem ser preparados (dadas as restrições de ingredientes) e o tempo (inteiro, em segundos) necessário para preparar as pizzas desse sabor (como possuem um forno muito grande, o tempo de um sabor independe da quantidade de pizzas). Após esses números há um espaço em branco e, então, o nome do sabor de pizza (string com até 50 caracteres). Não há sabores repetidos.

A seguir, há uma linha contendo um inteiro  $C$  ( $1 \leq C \leq 500$ ), o número de clientes.

Em seguida, a informação de cada cliente, começando com uma linha com um inteiro  $N$  ( $N \geq 0$ ), o número de pedaços que tal cliente comerá, e outro inteiro  $S$  ( $S > 0$ ), a quantidade de sabores de pizza que tal cliente gosta. E então há  $S$  linhas, cada uma contendo o nome de um dos sabores de pizza que agrada a tal cliente (clientes listam apenas sabores disponíveis no rodízio).

### Saída

Escreva o tempo mínimo necessário para preparar pizzas suficientes para satisfazer a todos (ou  $-1$  se for impossível).

### Exemplos

Entrada	Saída
<pre>5 4 300 pizza de capivara 30 400 pizza portuguesa 40 500 pizza de brigadeiro 30 400 pizza de frango com milho 4 2500 pizza de doce de leite 2 10 3 pizza de capivara pizza de doce de leite pizza de brigadeiro 15 2 pizza de capivara pizza portuguesa</pre>	<pre>500</pre>

## Problema D. Rodízio de pizza confuso

Arquivo-fonte: "confuso.x", onde x deve ser c, cpp, java ou py

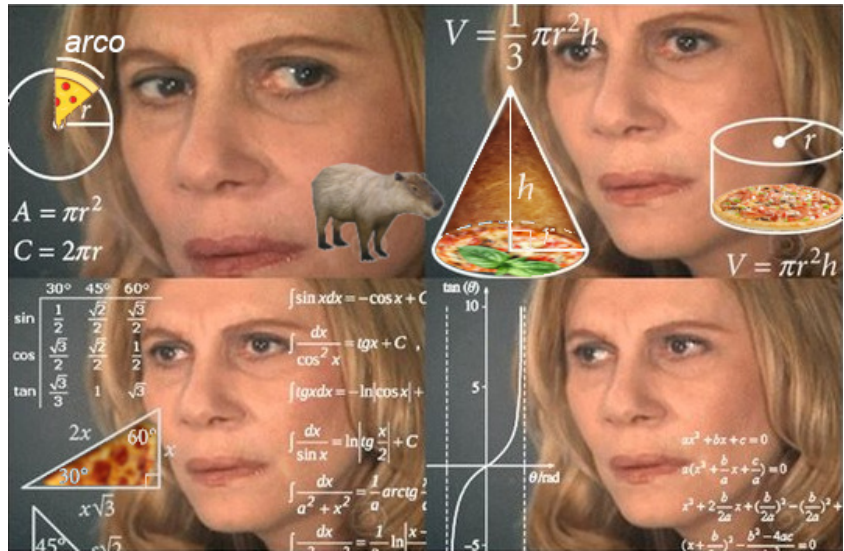
Conforme dito na questão anterior, após treinos para maratonas de programação, alunos e professores de INF333 da UFV (Programação Competitiva) sempre vão ao Rodízio de Pizza 2 irmãos para relaxarem.

Uma coisa que sempre intrigou tais pessoas é uma pequena burocracia que há nessa pizzeria: após comer uma pizza doce, a pessoa não pode mais comer pizza salgada.

Ao notar que os clientes vindos de INF333 são muito gulosos, um dos 2 donos da pizzeria decidiu criar várias novas regras para tentar fazer com que o pessoal coma menos pizzas quando voltarem ao rodízio futuramente (em Programação Competitiva 2).

A partir de agora, há um conjunto de regras indicando que um determinado tipo de pizza não pode ser comido após outro tipo. Além disso, cada pessoa pode comer apenas um pedaço de cada sabor. Essa burocracia tem complicado a vida dos clientes, que passam boa parte do tempo pensando na melhor estratégia para comer as pizzas.

Porém, o rodízio não contava com a astúcia dos alunos de INF333! Rapidamente eles decidiram criar um programa para determinar as possíveis ordens válidas para comer pizza, de modo a conseguir comer exatamente um pedaço de cada sabor. Sua tarefa é implementar tal programa.



### Entrada

A entrada contém apenas um caso de teste, que começa com uma linha contendo a lista de sabores de pizza disponíveis. Cada sabor é formado apenas por letras, contém até 20 letras e haverá até 30 sabores distintos. Cada par de sabores na lista é separado por um espaço em branco.

A seguir, há várias linhas, cada uma descrevendo uma restrição (a saída termina com o final do arquivo). Cada restrição é composta por um par de sabores de pizza (entre os descritos na primeira linha), indicando que o primeiro sabor não pode ser comido após o segundo.

### Saída

Escreva todas as ordens válidas (uma por linha) nas quais as pizzas podem ser consumidas. Cada par de sabores impressos deve ser separado por um espaço em branco. A saída terá até 1000000 linhas, que podem ser escritas em qualquer ordem.

### Exemplos

Entrada	Saída
<pre>lombo doceDeLeite modaDaCasa queijo lombo doceDeLeite lombo modaDaCasa</pre>	<pre>lombo doceDeLeite modaDaCasa queijo lombo doceDeLeite queijo modaDaCasa lombo modaDaCasa doceDeLeite queijo lombo modaDaCasa queijo doceDeLeite lombo queijo doceDeLeite modaDaCasa lombo queijo modaDaCasa doceDeLeite queijo lombo doceDeLeite modaDaCasa queijo lombo modaDaCasa doceDeLeite</pre>
<pre>medianaDaCasa capivara medianaDaCasa capivara capivara medianaDaCasa</pre>	<pre>Este caso não tem saída, não está em branco aqui porque estamos te informando isto! Não escreva isto na saída!!!!!!</pre>

## Problema E. Dividindo a pizza

Arquivo-fonte: "divisao.x", onde x deve ser c, cpp, java ou py

Como o problema anterior foi resolvido, um dos dois donos do Rodízio de Pizza 2 Irmãos decidiu criar mais uma complicação: antes de trazer uma nova pizza e distribuí-la para a mesa, cada cliente que ainda não está satisfeito deve informar o tamanho, representado pelo comprimento do arco que define a fatia de pizza preferido para os seus pedaços.

Então, uma nova pizza é feita apenas se for possível parti-la em pedaços dos tamanhos informados pelos clientes, sem que haja sobra e de modo que todos clientes comam pelo menos um pedaço do tamanho preferido. Note que a pizzeria pode cortar mais de um pedaço do mesmo tamanho (a ideia é que o(s) cliente(s) que escolheu aquele tamanho será o felizardo para comer mais pedaços).



Os alunos de INF333 estão querendo saber se terão direito a uma nova rodada de pizza.

Crie um programa que, dados os tamanhos (em centímetros) escolhidos pelos clientes para seus pedaços de pizza e o raio da forma utilizada pela pizzeria, também em centímetros, determina se o rodízio assará mais uma pizza para tais pessoas.

### Entrada

A entrada começa com um inteiro  $N$ , o número de clientes ( $N \leq 300$ ). A seguir, há uma linha contendo  $N$  inteiros  $T_i$  representando o tamanho do arco da fatia de pizza que cada cliente deseja ( $1 \leq T_i \leq 300$ ).

Por fim, há uma linha informando o raio da pizza. O raio é representado por uma fração no formato  $K/\pi$ , onde  $K$  é inteiro ( $1 \leq K \leq 20000$ ).

### Saída

Escreva uma linha contendo "SIM", se for possível cortar a pizza nos tamanhos informados de modo a satisfazer todos os clientes, ou "NAO", caso contrário.

### Exemplos

Entrada	Saída
3 2 1 5 5/PI	SIM
2 3 6 5/PI	NAO

## Problema F. Pi.z.z.a

Arquivo-fonte: "pi.x", onde x deve ser c, cpp, java ou py

Em uma das vezes que os alunos e professores de INF333 foram ao rodízio de Pizza 2 Irmãos, um deles perguntou qual o volume de uma pizza de raio  $Z$  e altura  $A$ . A resposta?  $\pi \cdot Z \cdot Z \cdot A$ . Todos riram bastante!

A família Capivaristo estava na mesa ao lado, e também riu da resposta, exceto Capivarinho, que tem apenas 5 anos e não entende de constantes matemáticas. Seu pai Capivaro lhe explicou que o volume da pizza envolve uma conhecida constante,  $\pi$ , que vale 3.14. Sua mãe, engenheira de alimentos, disse que  $\pi$  na verdade é 3.1415. Seu irmão mais velho, estudante de computação da UFV, quis mostrar conhecimento e falou que vale 3.14159265. Sua irmã, estudante de matemática, explicou que na verdade é um número irracional e possui infinitas casas decimais; todos os valores falados estavam corretos, a precisão dependendo da aplicação.

Aproveitando a ocasião, explicou para Capivarinho que existem outras constantes matemáticas, como  $e$ , o número de Euler, usado como base dos logaritmos naturais. E algumas bem legais, como  $\phi$ , a proporção áurea, que supostamente teria sido usada em diversas construções arquitetônicas da antiguidade e ainda hoje usada nas artes para compor cenas esteticamente agradáveis. Todas elas são irracionais, e os primeiros dígitos são mostrados abaixo:

- $\pi = 3.14159265358979323846 \dots$
- $e = 2.71828182845904523536 \dots$
- $\phi = 1.61803398874989484820 \dots$

Capivarinho gostou disso tudo, mas ainda tem dificuldade em identificar as constantes, já que podem ser ditas de diversas formas diferentes. Faça um programa para ajudá-lo.

### Entrada

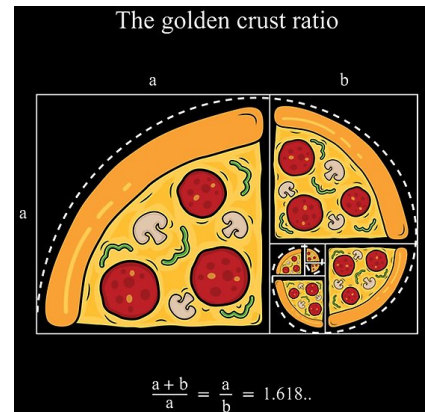
A entrada contém corretamente uma das constantes acima, contendo de 1 a 20 casas decimais, ou seja, o número de casas decimais pode variar, mas os dígitos usados corresponderão exatamente aos primeiros de uma das constantes listadas acima.

### Saída

Escreva '3' se a constante da entrada corresponder ao  $\pi$ , '2' se corresponder ao  $e$  e '1' se corresponder ao  $\phi$ .

### Exemplos

Entrada	Saída
3.14	3
2.71828	2
3.141592	3
1.6180	1





## Problema G. Ovos de Capivara 2 Irmãos

Arquivo-fonte: "oc2.x", onde x deve ser c, cpp, java ou py

Devido à dieta baseada em doce de leite<sup>1</sup>, observaram-se que os ovos de capivara da UFV são de excelente qualidade. Com isso, um grupo de estudantes da Computação decidiu incubar no CENTEV uma empresa para produção de ovos de capivara (a OC2 – *Ovos de Capivara 2 Irmãos*).

Como aprenderam nas disciplinas de Estruturas de Dados e Projeto e Análise de Algoritmos que algoritmos lineares são muito bons, tais alunos decidiram organizar os ninhos de capivara na granja de forma linear (ou seja, exceto para os da borda, cada ninho de capivara possui dois vizinhos). Após um ano de funcionamento dessa lucrativa empresa um dos sócios observou um fato curioso: como os roedores são muito competitivos, quando as capivaras vizinhas de uma determinada capivara  $C$  produzem mais ovos do que ela em um determinado dia, no próximo dia  $C$  tende a aumentar sua produção (e o contrário também ocorre!).

Após analisar o histórico de produção, descobriram que o número de ovos botados por uma capivara  $C$  em um determinado dia é igual à média (sempre truncada para um inteiro) do número de ovos botados por  $C$  e seus dois vizinhos no dia anterior (ou um vizinho, no caso das capivaras que estão nos dois extremos da granja).

Devido a essa surpreendente descoberta, a OC2 agora poderá planejar de forma muito melhor suas vendas, pois conseguirão prever a produção em qualquer dia do futuro!

Sua tarefa nesta questão é ajudar a OC2 a fazer essa predição.



### Entrada

A entrada começa com uma linha contendo dois inteiros  $N$  e  $T$  ( $1 \leq N \leq 1000$ ,  $0 \leq T \leq 1000$ ).  $N$  representa o número de capivaras na granja e  $T$  representa o número de dias no futuro para o qual a OC2 deseja prever a produção.

A seguir, há uma linha contendo  $N$  inteiros separados por um espaço em branco (cada número estará entre 0 e 1000000). Cada inteiro representa o número de ovos que a capivara correspondente botou em um determinado dia (a entrada está na mesma ordem em que as capivaras ficam na granja).

### Saída

Escreva uma linha contendo a produção de cada capivara  $T$  dias após o dia inicial. Cada par de inteiros na saída deve estar separado por um espaço em branco.

### Exemplos

Entrada	Saída
5 1 6 2 6 4 0	4 4 4 3 2
5 2 6 2 6 4 0	4 4 3 3 2
5 5 6 2 6 4 0	3 2 2 2 2

<sup>1</sup>o melhor doce de leite do Brasil, título anual conquistado pela décima vez no mês passado!

## Problema H. PD da Subsequência

Arquivo-fonte: "pds.x", onde x deve ser c, cpp, java ou py

Conforme falado na questão anterior, a empresa OC2 possui uma granja de produção de ovos de capivara, onde tais roedores são organizados de forma linear.

Um dos sócios observou que determinadas regiões da construção possuem uma produtividade melhor do que outras. Para estudar melhor esse fenômeno, ele criou uma métrica que avalia diariamente subsequências de capivaras consecutivas da granja: a Produtividade Diária da Subsequência (ou *PD da Subsequência*).

Seja  $x_n$  a produtividade da  $n$ -ésima capivara em um determinado dia. A PD da subsequência formada pelas capivaras entre as posições  $i$  e  $j$ ,  $pds(i, j)$ , é definida como:

$$pds(i, j) = \sum_{n=i}^j x_n$$



Dado o tamanho de uma janela  $K$ , a OC2 deseja encontrar o grupo de  $K$  capivaras consecutivas que produz mais ovos na granja. Por exemplo, se a produção de ovos das capivaras for 1, 2, 5, 9, 2, 4, 1 e  $K = 3$ , a subsequência máxima é formada pelas capivaras que estão entre as posições 3 e 5 (o valor da PD da subsequência máxima nesse caso é  $pds(3, 5) = 5 + 9 + 2 = 16$ ).

### Entrada

A entrada começa com uma linha contendo dois inteiros  $N$  e  $K$  ( $0 \leq N, K \leq 1000000, K \leq N$ ).  $N$  representa o número de capivaras na granja e  $K$  o tamanho das subsequências que a OC2 deseja avaliar.

A seguir, há uma linha contendo  $N$  inteiros separados por um espaço em branco (cada número estará entre 0 e 1000000). Cada inteiro representa o número de ovos que a capivara correspondente botou em um determinado dia (a entrada está na mesma ordem em que as capivaras ficam na granja).

### Saída

Escreva uma linha contendo o valor da PD da subsequência de tamanho  $K$  máxima da granja.

### Exemplos

Entrada	Saída
7 3 1 2 5 9 2 4 1	16

## Problema I. Capivaras querem doce!

Arquivo-fonte: "rotas.x", onde x deve ser c, cpp, java ou py

Na UFV as capivaras são super bem tratadas, inclusive recebendo uma porção de doce de leite por semana (mais do que é servido no RU!).

A entrega dos doces para os roedores é feita utilizando um conjunto de trilhas que conectam toda a universidade.

A rede de trilhas é composta por pontos de junção (onde trilhas começam, terminam ou se encontram) e pelas trilhas propriamente ditas (cada uma liga dois pontos de junção).

A fábrica de doce de leite e os ninhos de capivaras sempre ficam nas junções de trilhas. Atualmente, sempre é possível sair da fábrica de doce e levar a magnífica sobremesa para qualquer ninho de capivara.

Devido a falta de recursos, o reitor te pediu para desenvolver um sofisticado software capaz de analisar informações sobre as trilhas e determinar se é possível destruir algumas delas (pelo menos uma) para deixar de dar manutenção, mas ainda assim garantindo que seja possível levar doce de leite da fábrica para qualquer outra junção pelas trilhas não destruídas. Afinal, como dito em questões anteriores, doce de leite é essencial para manter a boa produtividade de ovos.



### Entrada

A entrada começa com uma linha contendo um inteiro  $N$  indicando o número de junções e um inteiro  $T$  indicando o número de trilhas ( $1 \leq N \leq 1000$ ,  $0 \leq T \leq 1000000$ ). A fábrica está na junção 0 e os ninhos de capivaras nas junções de 1 a  $N - 1$ .

A seguir, há  $T$  linhas, representando as trilhas, cada uma contendo um par de inteiros  $A, B$  ( $0 \leq A, B < N$ ,  $A \neq B$ ), indicando que há uma trilha conectando a junção  $A$  à junção  $B$ . Há no máximo uma trilha conectando um mesmo par de junções.

### Saída

Escreva uma linha contendo "sim", se for possível destruir pelo menos uma trilha sem desabastecer as capivaras, ou "nao", caso contrário.

### Exemplos

Entrada	Saída
6 7 0 1 1 2 2 3 3 4 4 0 4 5 1 4	sim

## Problema J. Vai chover doce de leite!

Arquivo-fonte: "chuva.x", onde x deve ser c, cpp, java ou py

Após uma explosão no reator nuclear número 4 da fábrica de doce de leite, provavelmente causada por um programador que não se preocupou com a eficiência do software do reator, toneladas de doce foram jogadas para a atmosfera.

Pesquisadores do DPI (Departamento de Pesquisas "Ídricas") analisaram correntes atmosféricas e estimaram que em uma semana todo esse doce de leite cairá em forma de chuva na UFV.

Vendo uma oportunidade para renovar seus estoques, um grupo de capivaras calouras de Eng. Civil decidiu construir um sofisticado sistema de telhados para recuperar tal guloseima.

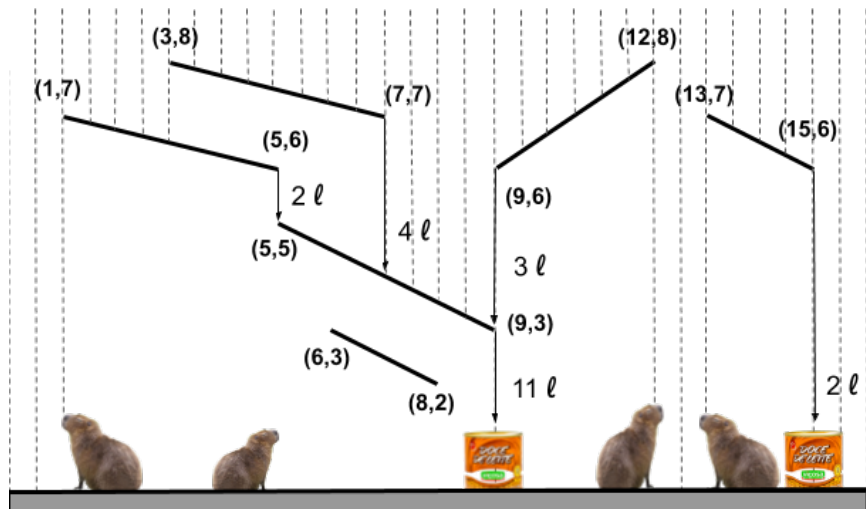
Infelizmente elas ainda não entendem tanto de construção e, com isso, alguns telhados foram construídos encobrimdo outros (assim, elas não conseguirão capturar tanto doce).

As capivaras te contrataram para estimar a quantidade de doce que cada telhado capturar por minuto (o objetivo será ver se eles suportarão o peso). Por simplicidade, elas te passaram um modelo 2D dos telhados (as informações sobre os telhados originais poderão ser calculadas posteriormente a partir dos resultados que você obtiver em 2D).

A quantidade de doce é igual à quantidade vinda do céu (sempre verticalmente) somada com a quantidade que cai de outros telhados. O DPI estimou que em cada metro (na horizontal) choverá 1 litro de doce de leite por minuto.

Nenhum dos telhados é horizontal e nenhum par de telhados se intercepta.

Veja o exemplo acima: o telhado que vai de (1,7) a (5,6) seria capaz de capturar 4 litros de doce. Porém, como 2 metros dele estão encobertos por outro telhado, ele captura apenas 2 litros - que posteriormente cairão no telhado que vai de (5,5) a (9,3).



### Entrada

A entrada começa com um número  $T$  indicando a quantidade de casos de teste. A seguir, há  $T$  casos de teste.

Cada caso começa com uma linha contendo um inteiro  $N$  ( $1 \leq N \leq 40000$ ) representando o número de telhados. A seguir, há  $N$  linhas, cada uma descrevendo um telhado. Cada telhado é representado por 4 inteiros  $x_1, y_1, x_2, y_2$  ( $0 \leq x_1, y_1, x_2, y_2 \leq 1000000, x_1 < x_2, y_1 \neq y_2$ ) separados por espaços em branco. Eles representam as coordenadas cartesianas da parte mais à esquerda do telhado,  $(x_1, y_1)$ , e da parte mais à direita,  $(x_2, y_2)$ . Os telhados não possuem pontos em comum e há no máximo 25 telhados acima de qualquer ponto no solo.

### Saída

Para cada caso de teste, escreva  $N$  linhas na saída. A  $i$ -ésima linha deve conter a quantidade de doce (em litros) que cairá do  $i$ -ésimo telhado por minuto.

### Exemplos

Entrada	Saída
1	2
6	4
13 7 15 6	2
3 8 7 7	11
1 7 5 6	0
5 5 9 3	3
6 3 8 2	
9 6 12 8	

Este exemplo possui apenas um caso de teste, que é o representado na figura do enunciado.

## Problema K. Supercapy

Arquivo-fonte: "supercapy.x", onde x deve ser c, cpp, java ou py

A cidade de Viçosa tem enfrentado uma onda de assaltos. "Oh! e agora, quem poderá nos defender?"



Felizmente agora temos duas supercapivaras atuando no combate ao crime, que surgiram após terem comido doce de leite radioativo da questão anterior,

Um dos poderes mais incríveis desses animais é voar rapidamente para o local de cada crime. Porém, infelizmente, elas têm um ponto fraco: como são radioativas, se estiverem voando e uma chegar muito perto da outra, os nêutrons emitidos por uma delas podem iniciar uma reação de fissão nuclear na outra e, com isso, Viçosa seria destruída! Felizmente este problema só pode ocorrer enquanto ambas capivaras estiverem em voo.

O Departamento de Processamento de Imagens (DPI) da UFV foi contratado para analisar rotas de pares de capivaras e, com isso, determinar qual seria a distância mínima que uma chegaria da outra enquanto ambas estiverem em velocidade de voo. O objetivo final será verificar o risco delas percorrerem essas trajetórias (mas você foi encarregado de apenas calcular a distância).

Por sorte, as supercapivaras possuem o poder de atravessar qualquer objeto, e assim, quando um crime ocorre, elas sempre voam em linha reta e a uma velocidade constante até o local do crime. Além disso, por algum motivo misterioso, os crimes em Viçosa sempre ocorrem simultaneamente em pares. Com isso, duas capivaras saem simultaneamente das suas bases secretas.

Obs: favor manter sigilo sobre as coordenadas das bases secretas presentes nos casos de teste!

### Entrada

Cada entrada começa com uma linha contendo um inteiro  $N$  ( $N \leq 1000000$ ) representando o número de casos de teste.

Cada caso de teste é representado por 2 linhas e cada uma possui 7 números reais  $x_1 y_1 z_1 x_2 y_2 z_2 t$  representando a posição inicial  $(x_1, y_1, z_1)$  de uma capivara, sua posição  $(x_2, y_2, z_2)$  após 1 segundo de voo e o tempo  $t$  de voo da capivara até seu destino (dados em segundos, sendo  $t > 0$ ).

Assuma que tanto no tempo inicial (0) quanto no final ( $t$ ) as capivaras estão em voo.

### Saída

Escreva uma linha para cada caso de teste, com a menor distância (com 2 casas de precisão) que uma capivara ficará em relação a outra enquanto ambas estiverem voando.

### Exemplos

Entrada	Saída
1 0.0 0.0 0.0 1.0 0.0 0.0 5.0 1.0 1.0 1.0 1.0 1.0 0.0 4.0	1.00

Neste exemplo, uma das capivaras sai da sua base secreta na posição  $(0, 0, 0)$  e passa pelo ponto  $(1, 0, 0)$  após 1 segundo de voo. A outra capivara sai de  $(1, 1, 1)$  e passa pelo ponto  $(1, 1, 0)$  após 1 segundo de voo. A distância mínima entre essas duas trajetórias é 1.00 e ocorre 1 segundo após elas saírem das bases.

## Problema L. McFly e a cotação do Capicoín

Arquivo-fonte: "capicoín.x", onde  $x$  deve ser c, cpp, java ou py

Após ganharem muito dinheiro vendendo o doce de leite coletado durante a chuva mencionada no problema J, as capivaras da UFV decidiram investir tal recurso criando uma moeda própria, a Capicoín (esta moeda tem seu peso lastreado em doce de leite).

Após trabalhar em um projeto de IC relacionado a plutônio e capacitores de fluxo do capivaro Dr. Brown, o capivaro McFly teve a oportunidade de testar uma máquina do tempo!

Em uma de suas viagens para o futuro, McFly comprou um almanaque que continha o histórico da cotação diária das principais moedas do mundo. Com essa informação privilegiada, ele decidiu comprar e vender Capicoíns para se enriquecer.

Para não chamar a atenção de outras capivaras (o que poderia influenciar a cotação), ele decidiu fazer apenas uma compra e uma venda da moeda. Além disso, McFly não quer ficar muito tempo guardando os Capicoíns e, assim, ele deseja efetuar a venda até um determinado número de dias após ter comprado a moeda.

Como McFly tem dificuldades em fazer cálculos, sua empresa (a 2 Irmãos Financial Analysis) foi contratada para calcular o valor do lucro máximo que McFly poderá obter por Capicoín comprado. Observe que McFly está interessado em maximizar a quantidade de reais (não o lucro percentual) que ganha comprando um Capicoín e o vendendo.

O lucro deverá ser 0 caso não seja vantajoso fazer esse tipo de transação.

Obs: quando ficou sabendo do almanaque, Dr. Brown proibiu McFly de usar a máquina do tempo novamente; assim, ele não poderá comprar a moeda em um dia e vendê-la no passado.

### Entrada

A entrada começa com uma linha contendo um inteiro  $N$  ( $1 \leq N \leq 1000000$ ), indicando o número de dias para os quais McFly possui a cotação do Capicoín, e um inteiro  $K$  ( $1 \leq K \leq N$ ), indicando o número máximo de dias após os quais a moeda deverá ser vendida.

Na próxima linha há  $N$  números inteiros  $x_i$  ( $0 \leq x_i \leq 2000000000$ ) indicando a cotação do Capicoín (em reais) no  $i$ -ésimo dia. Cada par de números é separado por um espaço em branco.

### Saída

Escreva um número inteiro informando o lucro máximo (em reais) que poderá ser obtido por Capicoín comprado.

### Exemplos

Entrada	Saída
7 3 5 2 6 7 10 9 1	8
7 2 5 2 6 7 10 9 1	5
4 1 5 3 4 10	6

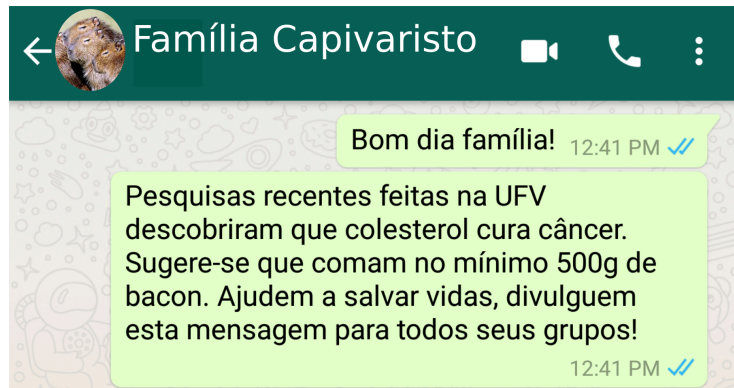


## Problema M. Tio do zapi zapi

Arquivo-fonte: "tio.x", onde x deve ser c, cpp, java ou py

Após assistir a palestra de abertura da semana do software da professora Jussara sobre desinformação na internet, um estudante do DPI - Departamento de Pesquisas da Internet, estudou o comportamento de pessoas em relação à distribuição de *fake news* em grupos de família do WhatsApp e observou que as pessoas podem ser classificadas em três tipos únicos:

- trolador: quem cria *fake news* e as envia para outras pessoas – um trolador detesta receber *fake news* e, portanto, nunca está em grupos de WhatsApp;
- tio: pessoa que, ao receber uma *fake news*, a espalha rapidamente para outras pessoas; e
- normal: pessoa que nem inventa nem espalha *fake news* para outras pessoas.



Esse estudante decidiu fazer uma pesquisa científica sobre o comportamento de pessoas no aplicativo de mensagens e, para isso, te contratou para criar um programa que, dadas informações sobre transmissões de uma notícia falsa, classifica as pessoas envolvidas em cada uma das três categorias acima.

### Entrada

A entrada começa com uma linha contendo um inteiro  $N$  ( $N \leq 1000000$ ) representando o número de interações em que a *fake news* passou e um inteiro  $P$  ( $P \leq 1000000$ ) indicando o número de pessoas estudadas.

A seguir, há  $N$  linhas, cada uma descrevendo uma interação, não necessariamente na ordem em que ocorreram. Cada interação  $i$  começa com um inteiro  $o_i$ , seguido por um valor  $q_i$  e, então,  $q_i$  inteiros  $d_{i1}, d_{i2}, \dots, d_{iq_i}$ . Isso indica que a pessoa com identificador  $o_i$  enviou a mensagem para um grupo de WhatsApp contendo as pessoas  $d_{i1}, d_{i2}, \dots, d_{iq_i}$ .

### Saída

Escreva uma linha para cada pessoa contendo sua classificação ("trolador", "normal" ou "tio"). A linha 1 deverá ter a classificação da pessoa com identificador 1 e assim por diante.

Obs: a notícia espalhada por cada tio sempre terá se originado a partir de pelo menos um trolador.

### Exemplos

Entrada	Saída
3 5	trolador
1 1 2	normal
4 2 2 3	tio
3 3 2 5 3	trolador
	normal

Nesta entrada, a pessoa 1 inventou a notícia e a espalhou para um grupo de WhatsApp contendo apenas a pessoa 2. Esta mesma notícia também foi inventada pela pessoa 4 e enviada para um grupo com as pessoas 2 e 3. A pessoa 3 recebeu a notícia e a enviou para um grupo com 2, 5 e 3. Assim, os troladores são 1 e 4, apenas 3 é tio, e as pessoas 2 e 5 são normais.