Maratona de Programação

Caderno de Problemas

UFV, 6 de Novembro de 2021

Instruções:

- Este caderno contém 10 problemas: com páginas numeradas de 1 a 10, não contando esta página de rosto.
- Em todos os problemas, a entrada deve ser lida da entrada padrão e a saída deve ser escrita na saída padrão.

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

- Buscadecarro – 5s
- Distanciamento Capivaral 2.0 – 3s
- Maratona de Programacão Paralela – 1,5s
- Nas demais – 0,5 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:

- Promoções: https://twitter.com/capybara_man/status/1363577797730525184
- Buscadecarro: https://agenciaonzi.com/preco-por-inbox-e-crime/
- Capifrutas: https://www.anamariabrogui.com.br/receita/caipifruta-de-morango-113911
- Maratona de programação paralela: https://twitter.com/CapybaraCountry/status/1369360651202699267/photo/1
- Atrapalhando a foto: https://br.pinterest.com/pin/830984568742874392/
- Investindo em ações: https://www.reddit.com/user/CapivaraDaFariaLima/
- Coffee Break: https://revistapegn.globo.com/Banco-de-ideias/Alimentacao/noticia/2021/08/feirante-viraliza-com-pastel-e-coxinha-em-formato-de-capivara-e-triplica-vendas.html
- Já chegou o disco voador: https://www.theguardian.com/environment/gallery/2009/jun/04/crop-circles
- Já se foi o disco voado: https://www.reddit.com/r/capybara/comments/hoy7px/mama_im_in_love_with_a_criminal/

Problema A. Promoções

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

Capivaristo adora aproveitar promoções que encontra no comércio e em lojas online. Recentemente, tem aparecido vários cupons de desconto e a chuva de ofertas criou um desafio para que Capivaristo sempre faça a escolha mais vantajosa para ele.

Muitas vezes, uma mesma loja possui inúmeros cupons de desconto diferentes, (mas em uma compra apenas um cupom pode ser utilizado. Capivaristo precisa comprar um pacote de ração de capivara. Ajude-o a encontrar o menor preço em uma determinada loja.

Há dois tipos de cupons:

- Tipo 1: X% de desconto em compras a partir de Y reais. Pode ser aplicado apenas se o valor da compra for maior ou igual a Y. O valor da compra será reduzido em X%.
- Tipo 2: X reais de desconto em compras a partir de Y reais. Também pode ser aplicado apenas se o valor da compra for maior ou igual a Y. O valor da compra será reduzido em X reais.

Figura 1: Capivaristo procurando ofertas

Entrada

A entrada começa com uma linha contendo $P\ T1\ T2$, onde P é o preço da ração ($0 \leq P \leq 1000000$) e $T1$ e $T2$ são dois inteiros que indicam a quantidade de cupons de cada tipo ($0 \leq T1, T2 \leq 1000$).

A seguir, há $T1$ linhas, cada uma descrevendo um cupom do tipo 1 por dois inteiros X e Y ($0 < X \leq 100$, $0 \leq Y \leq 1000000$).

Por fim, há $T2$ linhas, cada uma descrevendo um cupom do tipo 2 por dois inteiros X e Y ($0 < X \leq 1000000$, $0 \leq Y \leq 1000000$, $X \leq Y$).

Saída

Escreva o valor mínimo que Capivaristo conseguirá pagar pela ração de capivara, sempre com duas casas decimais.

Exemplos

Entrada	Saída
80.00 1 2 5 20 6 50 90 200	74.00

Entrada	Saída
80.52 1 2 5 20 3 50 90 200	76.50

Obs: neste exemplo os 5% de desconto reduziriam o preço em R$ 4.026 – porém, décimos de centavos não são considerados nos cálculos e, portanto, o desconto será de R$ 4.02

Problema B. Buscadecarro

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

Devido à inflação, está cada vez mais difícil encontrar rações de capivara a preços acessíveis. Além dos preços altos, outro problema é que saber o preço exato de uma mercadoria às vezes é um desafio: muitas empresas anunciam produtos e oferecem cupons de desconto que, diferentemente do que ocorre na questão anterior, podem ser combinados! Pior, algumas aderiram à moda de alguns sites da internet e estão divulgando o preço apenas "inbox" (mas isso ficará para uma questão futura da Maratona SI 2022)

Para resolver esse problema, Capivaristo decidiu criar um site capaz de sempre encontrar o melhor preço: o Buscadecarro. Ao fazer uma pesquisa, o site exibirá o preço (após todos descontos possíveis) do produto em cada loja (considerando os descontos), ordenando-as em ordem crescente.

Dado o preço de um produto em uma determinada loja e vários cupons de desconto, que reduzem o preço ou em forma percentual (tipo 1) ou em valores absolutos (tipo 2), tais cupons podem ser combinados desde que, ao aplicá-los, o valor mínimo do cupom seja maior ou igual ao valor atual da compra. Além disso, um mesmo cupom pode ser utilizado várias vezes, desde que o valor mínimo continue sendo respeitado.

Por exemplo, considere os dois cupons:

- 20% acima de R$100 (tipo 1)
- 5 reais de desconto acima de R$30 (tipo 2)

Em uma compra de 100 reais podemos aplicar o primeiro e, depois, o segundo. Note que não seria possível aplicá-los na ordem contrária, pois o segundo deixaria o valor do pedido abaixo de 100 reais e, assim, o primeiro cupom não poderia ser mais usado. Além disso, como após usar o segundo cupom a compra ainda estaria em R$75 reais (acima do limite de R$30), o segundo cupom poderia ser utilizado de novo. Na verdade, poderia ser usado várias vezes até reduzir o valor final para 25 reais (sim, nosso site pode gerar descontos insanos!).

Ajude Capivaristo a implementar tal sistema. Dado o preço de um produto e os cupons disponíveis, encontre o menor preço possível de se obter com tais descontos (a implementação da ordenação e outras funcionalidades do sistema já está sendo feita pela NoBugs e, portanto, você não precisa se preocupar com isso).

Caso o preço após descontos fique negativo, assumimos que o valor final será 0. Adicionalmente, o desconto gerado por cupons do tipo 1 é calculado de modo que décimos de centavos são ignorados. Por exemplo: 5% de desconto em R$11.99 geraria um desconto de R$0.59 (não R$0.5995) e esse truncamento é feito toda vez antes do cupom ser aplicado.

Entrada

A entrada começa com uma linha contendo P $T1$ $T2$, onde P é o preço da ração (número real sempre com 2 casas decimais, $0 \leq P \leq 50000$), e $T1$ e $T2$ são dois inteiros que indicam a quantidade de cupons de cada tipo ($0 \leq T1, T2 \leq 25$).

A seguir, há $T1$ linhas, cada uma descrevendo um cupom do tipo 1 por dois inteiros X e Y ($0 < X \leq 100$, $0 \leq Y \leq 50000$).

Por fim, há $T2$ linhas, cada uma descrevendo um cupom do tipo 2 por dois inteiros X e Y ($0 < X \leq 50000$, $0 \leq Y \leq 50000$, $X \leq Y$).

Saída

Escreva o valor mínimo que Capivaristo conseguirá pagar pela ração de capivara, sempre com duas casas decimais.

Exemplos

Entrada	Saída
80.00 1 2 5 20 6 50 90 200	19.00

Problema C. Capifrutas

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

Durante uma calourada (festa organizada por veteranos no início do período com o objetivo de se integrarem com os calouros fora das aulas de cálculo) envolvendo alunos da Computação e da Nutrição, os organizadores decidiram preparar Capifrutas, uma bebida que normalmente envolve refrigerante, leite condensado e várias frutas – fora da UFV, essa bebida é conhecida por Caipifrutas.

Capivaristo, um aluno da computação, ficou encarregado de desenvolver um software capaz de decidir quais ingredientes deverão ser usados na bebida. Sua ideia era economizar, usando o mínimo de frutas possíveis (das mais baratas) e gastando, para fabricar a bebida, um tempo linear em relação à quantidade. Porém, os alunos da nutrição ficaram aborrecidos com o baixo valor nutricional de algumas receitas e, portanto, pediram para Capivaristo desenvolver apenas receitas que contenham, no mínimo, uma quantidade determinada de nutrientes diferentes.

Ajude Capivaristo a encontrar a quantidade mínima de ingredientes capaz de ter, pelo menos, o número de nutrientes desejado pelos alunos da nutrição.

Entrada

A entrada começa com dois inteiros N e K, respectivamente o número de ingredientes e nutrientes ($0 \leq N \leq 20$). A seguir, há N linhas, cada uma descrevendo um ingrediente.

Cada linha começa com o nome de um ingrediente, um número Ni ($0 \leq Ni \leq 50$) e, a seguir, o nome de Ni nutrientes. O nome dos nutrientes são separados por espaço, sendo cada um descrito por uma string de até 30 letras minusculas, dígitos e hífen ('-').

No total, os alunos da nutrição conhecem no máximo 50 nutrientes distintos.

Saída

Escreva a quantidade mínima de ingredientes que, combinados, possuem no mínimo K nutrientes distintos. Se não for possível atingir a quantidade mínima de nutrientes, escreva -1.

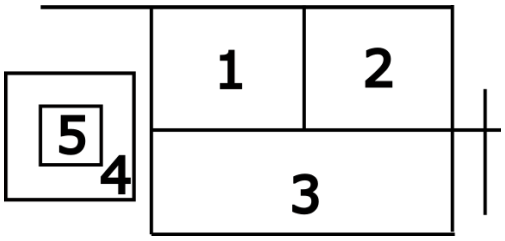
Exemplos

Entrada	Saída
5 5 morango 3 vitamina-c fosforo ferro grama 2 vitamina-c capivarina oroponobis 3 ferro fosforo proteina leite-condensado 2 acucar calcio jabuticaba 2 acucar vitamina-b2	2

Problema D. Distanciamento Capivaral 2.0

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

Como todos sabem, as capivaras da UFV adoram ficar na lagoa. Ano passado, devido à pandemia de CAPIVID20, foi necessário resolver um problema de distanciamento na lagoa (assunto da Maratona da SI 2020). Neste ano temos um novo problema.



Como todos também sabem, as capivaras da UFV adoram comer grama. Devido à pandemia, pesquisadores decidiram criar uma maneira de evitar que tais roedores fiquem muito próximos um dos outros durante a alimentação (ou, pior, que compartilhem grama).

Sendo assim, o gramado da UFV foi pintado com vários segmentos de reta (de espessura infinitesimal) e, assim, dividido em várias regiões fechadas. Cada capivara deverá se alimentar dentro de uma região.

A figura ao lado apresenta um exemplo de divisao onde 5 regiões fechadas foram formadas. Note que às vezes os pintores deixam algumas regiões incompletas (apenas as fechadas contam).

Para decidir quantas capivaras poderão se alimentar simultaneamente, um drone tirará fotos do gramado e um sistema de PDI fará o reconhecimento dos segmentos pintados. A No Bugs foi contratada para desenvolver um software que, dados os segmentos, retorna a quantidade máxima de capivaras que o gramado comporta em segurança (no exemplo acima ele deveria retornar 5). Sua tarefa será desenvolver tal sistema.

Sabe-se que no caso de entradas grandes (mais de 1000 segmentos) os pintores de gramado não conseguem pintar um segmento por muito tempo e, assim, eles são pequenos perto do tamanho da região onde são pintados (pelo menos 60 vezes menores). Além disso, devido a uma lei municipal, o número de cruzamentos de segmentos não é maior que 200000.

Entrada

A entrada começa com um número inteiro N ($0 \leq N \leq 100000$), o número de segmentos. A seguir, há N linhas, cada uma descrevendo um segmento. Cada segmento é composto por 4 números reais $X1, Y1, X2, Y2$ representando seus dois vértices ($X1$,$Y1$) e ($X2$,$Y2$) em coordenadas cartesianas ($0 \leq X1, X2, Y1, Y2 \leq 1000000$).

Sabe-se que há apenos segmentos horizontais e verticais (ou seja, $X1 = X2$ ou $Y1 = Y2$) e nenhum deles possui os dois vértices extremos iguais.

Saída

Escreva uma linha contendo o número máximo de capivaras que o gramado comporta.

Exemplos

Entrada	Saída
5 0 0 10 0 10 0 10 10 0 10 10 10 0 0 0 10 0 5 10.3 5	2

Problema E. Maratona de Programação Paralela

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

Paralelamente à Maratona de Programação da SBC, há uma outra competição onde o objetivo é desenvolver programas o mais eficientes possíveis, mas utilizando programação paralela e programação de alto desempenho. A UFV tem sempre participado desse evento e obtido ótimos resultados.

Neste ano, decidiram disponibilizar um supercomputador composto por infinitos processadores. Um leigo pensaria que tal computador conseguiria resolver qualquer problema em um tempo próximo a zero, pois bastaria cada processador lidar com uma instrução do código. Porém, há dependências entre operações e, assim, em alguns casos, uma operação só pode ser realizada após outra ser concluída. Por exemplo, o cálculo da raiz de uma equação de segundo grau só pode ser realizado após obtido o resultado do delta.

Capivaristo lhe contratou para desenvolver um software capaz de estimar o tempo de execução de um programa paralelo que ele desenvolveu. Supondo que cada processador consiga realizar uma operação em um microssegundo, seu objetivo será calcular o tempo mínimo que infinitos processadores gastariam para processar uma determinada quantidade de operações. A entrada do seu problema será as dependências entre as operações do programa que ele implementou.

Figura 2: Capivaristo debugando seu código

Entrada

A entrada começa com dois números inteiros N e D, respectivamente o número de operações e o número de dependências ($0 \leq N, D \leq 10000000$). A seguir, há D linhas, cada uma composta por dois números inteiros Oi e Di indicando que a operação Oi só pode ser realizada após Di ter sido concluída ($0 \leq Oi, Di < N, Oi \neq Di$).

Saída

Escreva um número inteiro informando o número de microssegundos que o supercomputador gastará para processar todas operações, respeitando as dependências.

Se não for possível executar todas as tarefas, escreva -1.

Exemplos

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

Entrada	Saída
3 3 0 1 1 2 2 0	-1

Explicação do primeiro exemplo: Primeiro, as instruções 0 e 5 podem ser executadas. A seguir, o computador pode executar 2 e 1 (que dependem apenas de 0). Por fim, 3 e 4 podem ser executadas paralelamente (outras ordens gastando 3 microssegundos também são possíveis).

Problema F. Atrapalhando a foto

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

Na UFV há uma lagoa praticamente quadrada. Recentemente, um grupo de pesquisadores decidiu obter uma imagem de satélite para fazer análises nesse lago. Essa imagem foi representada por uma matriz armazenando 0 nos locais onde há água e 1 onde há terra. Esperava-se ter terra apenas próximo à borda da lagoa (borda da matriz). Porém, os pesquisadores viram vários pontos de terra no meio do lago, parecendo pequenas ilhas.

Ao visitarem a lagoa para tentar entender o que houve, os cientistas descobriram que havia muitas capivaras andando de bicicleta no lago e aram para desenvolver um programa capaz de remover essas capivaras da imagem.

Como elas ficam longe da terra (com medo do jacaré, que tem preguiça de nadar), seu objetivo será remover os números 1 (trocando-os por 0) que estão "ilhados" na lagoa, ou seja, pixels onde não é possível ir deles até a borda da lagoa sem passar por células contendo 0. Considere que cada pixel possui no máximo 4 vizinhos (ou seja, eles não são vizinhos na diagonal).

Figura 3: Algumas das capivaras que atrapalharam a foto

Entrada

A entrada começa com dois inteiros L, C que representam a dimensão da lagoa ($1 \leq L, C \leq 1000$). A seguir, há L linhas, cada uma descrevendo uma linha da matriz e contendo C caracteres (que poderão ser 0 ou 1).

Saída

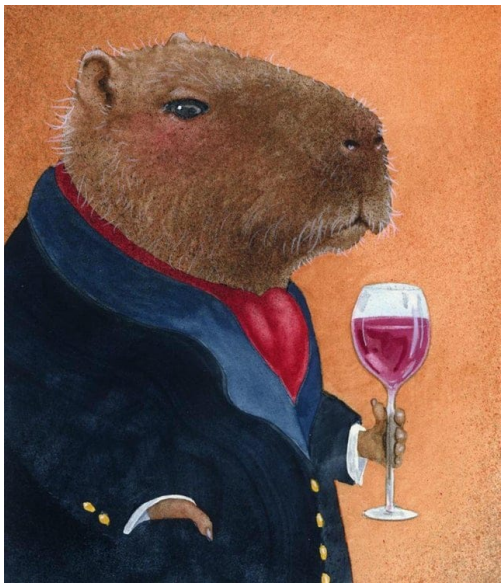
Escreva a matriz obtida a partir da remoção das ilhas da entrada.

Exemplos

Entrada	Saída
7 6 000100 110001 111100 100000 000010 011010 000000	000100 110001 111100 100000 000000 000000 000000

Problema G. Investindo em ações

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



Após aprender sobre ações em um trabalho de INF213, Capivaristo decidiu investir alguns milhões de capicoins nesse tipo de ativo e, com isso, receber todo ano seus dividendos. Os dividendos são participação no lucro, que as empresas distribuem aos acionistas e é paga proporcionalmente ao número de ações que cada investidor possui.

Como ele ainda não entende muito sobre o assunto, decidiu (talvez essa não seja uma decisão tão boa) simplesmente comprar ações de uma única empresa: a que atualmente paga mais dividendos.

Porém, há uma complicação: ações de diferentes empresas custam valores diferentes e, assim, uma ação que rende pouco em valores absolutos pode, na verdade, estar pagando mais em termos proporcionais ao seu preço. Por exemplo: uma ação que paga 1 capicoin por ano e custa 2 capicoins tem um rendimento melhor (50%) do que uma que paga 2 capicoins e custa 10 capicoins (20%). Note que o rendimento (conhecido como *dividend yield*) da primeira ação é similar ao de uma terceira que paga 2 e custa 4 capicoins.

Como há muitas ações na bolsa, Capivaristo lhe contratou para desenvolver um software capaz de descobrir qual delas possui o melhor rendimento.

Entrada

A entrada começa com um número N, o número de açoes ($N \leq 100000$).
A seguir, há N linhas, cada uma descrevendo uma ação.

Cada ação é descrita por dois inteiros D e P que representam respectivamente o valor que a ação paga de dividendos por ano, em capicoins (Capivaristo é muito rico e não está interessado em centavos) e o preço da ação, também em capicoins ($0 \leq D \leq 10000$, $0 < P \leq 10000$).

Saída

Escreva os valores D e P da ação que paga mais dividendos proporcionalmente ao preço. Em caso de empate, imprima a que, dentre as que empataram, aparecer primeiro na entrada.

Exemplos

Entrada	Saída
4 3 10 1 2 2 4 2 10	1 2

Problema H. Coffee Break

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

Em 2022 a organização da Semana de Informática (SI) decidiu capricchar ainda mais no evento. Além da tradicional maratona e das excelentes palestras, conseguiram fazer um Coffee Break muito caprichado. Em um dia, a quantidade de salgados foi tão grande que até sobrou! A organização decidiu, então, recolher os restantes e guardá-los para o próximo lanche.

Figura 4: Salgados do Coffee Break

Um aluno, que gosta muito de desafios, propôs o seguinte problema para o responsável pelo recolhimento dos salgados: como podemos agrupar as bandejas não vazias fazendo o mínimo possível de movimentos? Cada movimento consiste em: – Escolher um conjunto S de bandejas não vazias contíguas. – Deslocar todas essas bandejas de S simultaneamente um espaço para a esquerda ou direita. Isso só pode ser feito se não houver alguma bandeja cheia na próxima posição à esquerda da primeira bandeja de S (ou direita da última, respectivamente), ou seja, não queremos que uma bandeja cheia fique por cima de outra.

As bandejas ainda cheias (e as vazias) estão agrupadas de forma linear em uma grande mesa no hall do CCE.

Considere o seguinte exemplo: VVCVVCCV, onde V e C representam, respectivamente, espaços com bandejas vazias e cheias.

Um movimento válido poderia ser mover as bandejas cheias das posições 6 e 7 uma unidade para a esquerda (VVCVCCVV) ou uma unidade para a direita (VVCVVVCC). As bandejas 3, 6 e 7 não podem ser movidas simultaneamente, pois não estão contíguas.

Uma forma de agrupar todas as bandejas cheias em 2 movimentos seria: VVCVVCCV $\rightarrow$ VVCVCCVV $\rightarrow$ VVCCCVVV (note que, agora, as bandejas das posições 3,4,5 podem ser movidas juntas. Porém, isso não é mais necessário pois todas bandejas cheias já ficaram agrupadas.

Entrada

A entrada começa com um inteiro N representando o número de espaços com bandejas ($1 \leq N \leq 1000$). A seguir, há uma linha contendo N caracteres, que podem ser '`V`' ou '`C`'. Há pelo menos um caractere '`C`' na entrada.

Saída

Escreva uma linha contendo um inteiro que representa a quantidade mínima de movimentos necessários para agrupar as bandejas cheias de forma que fiquem contíguas.

Exemplos

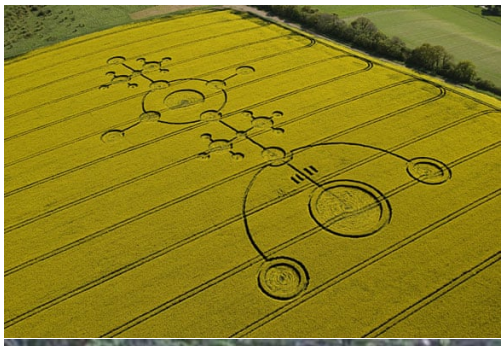
Entrada	Saída
8 VVCVVCCV	2

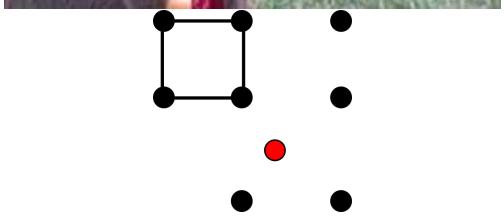
Entrada	Saída
8 CVCVVCCV	3

Problema I. Já chegou o disco voador

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

Como todos sabem, discos voadores frequentemente vêm ao planeta Terra com a importantíssima missão de desenhar padrões em plantações e, com isso, deixam os terráqueos intrigados. Ao chegar ao nosso planeta, as naves escolhem uma plantação e, com o uso de lasers, desenham algum padrão geométrico. Extraterrestres desenham padrões que representam o seu planeta (isso é uma forma de pichação intergaláctica, em que ficam marcando território na Terra).





Os ETs vindos de Vênus (segunda imagem ao lado), sempre desenham padrões retangulares. Um fato curioso é que habitantes de tal planeta possuem uma espécie de CPF (número sequencial começando de 0). Ao pichar a Terra, o ET desenha um padrão cujo número de retângulos distintos é igual ao seu CPF. Os UFólogos utilizam esse CPF para identificar o autor dos padrões e, assim, catalogá-los.

Em uma viagem recente, havia pouca tinta laser em uma das naves e, assim, ao chegar à Terra ela conseguiu desenhar apenas os vértices dos seus retângulos, que foram desenhados no gramado da UFV. Infelizmente, já se foi o disco voador e, assim, não será possível perguntar ao ET o seu CPF. A notícia boa é que, com base nos vértices desenhados, é possível contar quantos retângulos diferentes poderiam ser feitos. Você foi contratado para desenvolver um software capaz de realizar esse cálculo.

Assim, dado um conjunto de coordenadas x, y de vértices, seu objetivo será contar a quantidade de retângulos distintos que podem ser formados com quádruplas deles.

No exemplo ao lato (terceira imagem), é possível formar 3 retângulos pequenos (um deles está desenhado) e dois maiores e, portanto, o CPF do ET que o desenhou é 5. Observe que o vértice vermelho não estará em nenhum retângulo. Obs: os ETs conhecem apenas retângulos alinhados com os eixos x/y (ou seja, que usam apenas segmentos verticais e horizontais) e, portanto, outros devem ser ignorados.

Entrada

A entrada começa com um número N, o número de pontos desenhados no gramado ($N \leq 10000$). A seguir, há N linhas, cada uma descrevendo um ponto.

Cada ponto é composto por duas coordenadas x, y (inteiras) separadas por um espaço em branco. Não há pontos repetidos na entrada.

Saída

A saída deve ser uma linha contendo um inteiro, que representa a quantidade de retângulos distintos que podem ser formados.

Exemplos

Entrada	Saída
9 0 2 0 3 1 0 1 2 1 3 2 1 3 0 3 2 3 3	5

Problema J. Já se foi o disco voador

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

O disco voador da questão anterior acaba de sair de Viçosa. Apesar das últimas visitas terem sido pacíficas, com apenas inocentes pixações em plantações e gramados, dessa vez ele cometeu um terrível crime: abduziu 5 capivaras da lagoa da UFV e, também, o jacaré.

Felizmente, um aluno do DPI percebeu o que estava acontecendo e anotou a placa da nave pouco antes do momento em que se foi o disco voador.

Capivaristo, um importante Ufólogo da UFV (ou melhor, um UFVólogo), foi encarregado de identificar a cidade Venuziana de onde tal OVNI veio. Porém, há um problema: a testemunha que anotou a placa tem medo de alguma ilusão ótima ter afetado sua visão e, assim, pode ser que a placa anotada esteja errada.

Felizmente, o Detran de Vênus recentemente adotou uma estratégia (similar a um *hashing*) para dar mais confiabilidade às suas placas: todas placas vindas desse planeta são representadas por um número inteiro (com até 1 milhão de dígitos) onde o produtório dos dígitos de todas suas substrings gera apenas valores distintos.

Por exemplo, a placa 234 pode ser de Vênus, pois $2 \times 3 \times 4$, 2×3, 3×4, 2, 3 e 4 são números distintos. A placa 123, por outro lado, não pode ser de Vênus pois os produtórios dos dígitos nas substrings 123 e 23 são iguais.

Desenvolva um programa para ajudar Capivaristo a verificar se uma placa pode ser de Vênus.

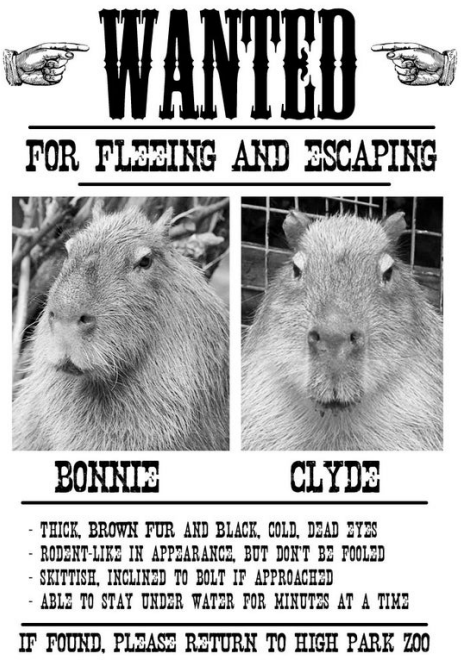


Figura 5: Foto das Venuzianas suspeitas da abdução

Entrada

A entrada é composta por um inteiro N não-negativo com até 1 milhão de dígitos que representa uma possível placa do disco voador.

Saída

Escreva uma linha contendo "`SIM`" caso a placa possa ser de Vênus e "`NAO`", caso contrário.

Exemplos

Entrada	Saída
123	NAO

Entrada	Saída
234	SIM