Caderno de Problemas

Organização

Olá!

Prova preparada com muito empenho pela equipe do **PPCI** (UTFPR-Toledo) e da **Capimara** (UFPR-Curitiba) para 2025. Esperamos que gostem!

Good luck & Have Fun

Informações Gerais

Esta prova contém 29 páginas (excluindo a capa), numeradas de 1 a 29. Verifique se a prova está completa.

Entrada

- A entrada deve ser lida da entrada padrão.
- Todas as linhas da entrada, incluindo a última, terminam com o caractere de fim-de-linha (`\n`).
- Quando a entrada contém vários valores separados por espaço, existe exatamente um espaço entre dois valores consecutivos na mesma linha.

Saida

- A saida deve ser escrita na saida padrão.
- A saida deve ser impressa no formato especificado pelo problema. A saida não deve conter nenhum dado adicional.
- Todas as linhas da saida, incluindo a última, terminam com o caractere de fim-de-linha (`\n`).

Problema A. Amizade

Tempo limite: 1000 ms
Memória limite: 256 MiB

No Vale do Orvalho, a amizade com os moradores da Vila Pelicanos é medida em corações. Para encher um coração de amizade, você precisa acumular 250 pontos de amizade com um personagem.

Você recebe uma certa quantidade de interações N, onde cada interação representa os pontos de amizade que você ganhou em uma única interação com um morador.

Sua tarefa é calcular e retornar o número total de corações de amizade completos que você terá com esse morador após todas as interações.

O máximo de corações que se pode ter com cada morador é 8.

Os moradores da vila são, na seguinte ordem: Abigail, Sebastian, Penny, Sam, Leah, Harvey, Haley, Maru, Alex e Emily.

Entrada

A entrada consiste de um número N ($1 \leq N \leq 1000$) representando o número de interações com os moradores, seguida de N linhas: cada uma das N linhas terá uma string S_i (o nome de um morador) e um número M_i ($1 \leq M_i \leq 1000$), que representa os pontos de amizade ganhos daquele morador.

Saída

Para todo morador com pelo menos 1 coração completo, imprima o nome dele e o número de corações completos que ele obteve.

Os nomes dos moradores devem ser imprimidos na saída na mesma ordem apresentada anteriormente.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
4 Sebastian 1000 Leah 240 Sebastian 1000 Emily 250	Sebastian 8 Emily 1

Exemplo de entrada 2	**Exemplo de saída 2**
8 Abigail 500 Sebastian 100 Leah 750 Sam 250 Alex 200 Sebastian 170 Maru 250 Leah 1000	Abigail 2 Sebastian 1 Sam 1 Leah 7 Maru 1

Exemplo de entrada 3	**Exemplo de saída 3**
9 Alex 250 Alex 250 Alex 250 Alex 250 Alex 250 Alex 250 Alex 250 Alex 250 Alex 250	Alex 8

Problema B. Bruto

Tempo limite: 1000 ms
Memória limite: 256 MiB

Gabriel é o mestre de uma campanha de um RPG de mesa pós-apocalíptico chamado *A Era da Extinção*, e seu objetivo como mestre é tornar a campanha o mais hardcore possível.

Em uma das sessões, um grupo de N personagens está em uma boss fight épica contra o Terrível Bruto, uma enorme criatura que possui N pontos de vida.

Gabriel foi astuto: conseguiu deixar todos os personagens com poucos recursos e quase sem chances de vitória. A única forma do grupo evitar um **TPK** (a morte de todos os personagens do grupo) é derrotando o Terrível Bruto em apenas uma rodada de ataques.

Cada personagem realizará um teste de ataque contra o Bruto, rolando um dado de 20 faces. O valor X obtido, causa os seguintes efeitos:

Se $15 \leq X < 20$:	o personagem acerta o ataque e causa 1 de dano;
Se $X < 15$:	o personagem erra o ataque e não causa dano;
Se $X = 20$:	o personagem realiza um **acerto crítico** e causa 2 de dano.

Seguindo essas regras de ataque, determine se o Terrível Bruto foi derrotado ou se, felizmente para Gabriel, o grupo sofreu um **TPK**.

Entrada

A entrada é composta por um inteiro N ($1 \leq N \leq 10^5$), seguido de N linhas contendo o valor inteiro X ($1 \leq X \leq 20$) obtido no dado para cada personagem.

Saída

Caso a criatura tenha sido derrotada, imprima:

```
O BRUTO FOI DERROTADO
```

Caso contrário, imprima:

```
O BRUTO ESTA VIVO TPK PARA O GRUPO
```

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
5 20 15 17 6 19	O BRUTO FOI DERROTADO

Exemplo de entrada 2	**Exemplo de saída 2**
6 20 14 7 20 18 3	O BRUTO ESTA VIVO TPK PARA O GRUPO

Observações

No primeiro caso de teste, com $N = 5$ e rolagens `20, 15, 17, 6, 19`, o dano total causado pelos personagens é igual a 5, sendo suficiente para derrotar o Terrível Bruto. Portanto, a saída correta é `O BRUTO FOI DERROTADO`.

No segundo caso de teste, com $N = 6$ e rolagens `20, 14, 7, 20, 18, 3`, mesmo com dois acertos críticos e um acerto normal, o dano total é inferior aos 6 pontos de vida do Bruto. Assim, a saída correta é `O BRUTO ESTA VIVO TPK PARA O GRUPO`.

Problema C. CS2 — A Faca de Fallen

Tempo limite: 1000 ms
Memória limite: 256 MiB

Counter-Strike 2 (CS2) é um jogo mundialmente conhecido, e parte do charme do jogo vem das skins — "roupas" para armas. Uma das formas mais emocionantes de obtê-las é abrindo caixas: a cada caixa, o jogador ganha uma skin aleatória.

As skins têm valores diferentes conforme sua raridade. As facas, por exemplo, costumam ser mais caras. Gabriel Toledo — ironicamente estudante da UTFPR, campus Toledo — mais conhecido como "Fallen", abriu algumas caixas e tirou uma faca rara. Sem saber seu valor, levou-a ao especialista Henrique, mais conhecido como "Gubi".

Gubi explica que cada faca possui um pattern, um número de série com dígitos de 0 a 9. Quanto mais testes de qualidade o pattern passa, mais rara e valiosa é a faca. Cada teste é descrito por três inteiros positivos l, r, d. A faca passa o teste se a substring do pattern entre as posições l e r (inclusive) tem período d — isto é, $s_i = s_{i+d}$ para todo i com $l \leq i \leq r - d$.

Às vezes, Gubi se distrai, e Fallen — que não é bobo — modifica o pattern para tentar valorizar a faca: ele escolhe um intervalo e troca todos os dígitos desse intervalo por um mesmo dígito c. Pra variar Gubi está com sono, por isso ele pede a você para que realize a avaliação. Sua tarefa é processar essas alterações e testes e dizer, para cada teste, se a faca passa ou não.

Entrada

A primeira linha contém três inteiros n, m, k ($1 \leq n \leq 10^5$, $1 \leq m + k \leq 10^5$): o tamanho do pattern, o número de operações de alteração e o número de operações de teste.

A segunda linha contém uma string s de tamanho n, composta apenas por dígitos de 0 a 9.

Em seguida, vêm $m + k$ linhas, cada uma descrevendo uma operação, em ordem arbitrária:

- 1 l r c — (alteração) substitua todos os caracteres de $s_l, s_{l+1}, \ldots, s_r$ pelo dígito c ($1 \leq l \leq r \leq n$, $0 \leq c \leq 9$).
- 2 l r d — (teste) verifique se a substring $s_l s_{l+1} \ldots s_r$ tem período d ($1 \leq l \leq r \leq n$, $1 \leq d \leq r-l+1$).

Os índices são 1-baseados.

Saída

Para cada operação de tipo 2 (teste), imprima uma linha contendo Sim se a substring possui período d e Nao caso contrário.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
3 1 2 112 2 2 3 1 1 1 3 8 2 1 2 1	Nao Sim

Exemplo de entrada 2	**Exemplo de saída 2**
6 2 3 334934 2 2 5 2 1 4 4 3 2 1 6 3 1 2 3 8 2 3 6 1	Nao Sim Nao

Observações

Uma substring $s_l \dots s_r$ tem período d se, para todo i com $l \leq i \leq r - d$, vale $s_i = s_{i+d}$. Em particular, se $d = r - l + 1$, a condição é satisfeita trivialmente.

Exemplos de operações:

- 1 3 7 5 faz $s_3, s_4, \dots, s_7$ se tornarem todos o dígito 5. - 2 2 9 3 testa se $s_2 \dots s_9$ repete a cada 3 posições.

Problema D. Deck do Erick

Tempo limite: 1000 ms
Memória limite: 256 MiB

Após perder muitas partidas seguidas jogando *Clash Royale*, Erick ficou completamente tiltado. Depois de várias derrotas consecutivas, ele chegou a uma conclusão absolutamente lógica:

> "Se eu estou perdendo é porque não sou igual o gigante do Clash Royale."

Desde então, Erick iniciou um treino insano baseado em cartas de *Clash Royale*.

Ele montou um deck com M tipos diferentes de carta, numerados de 1 a M. Cada tipo de carta x tem um valor f_x, que representa a fadiga base que o cérebro do Erick sente ao usar aquela carta uma vez.

Durante um dia de treino, Erick registrou uma sequência de N jogadas. Em cada jogada ele usou exatamente uma carta do deck, e a carta usada na i-ésima jogada é c_i.

Agora ele quer analisar um trecho contínuo de exatamente L jogadas dessa sequência. Para esse trecho, a fadiga é calculada da seguinte forma:

- cada tipo de carta possui uma fadiga base f_x;
- toda vez que uma carta do tipo x reaparece no trecho, a fadiga dessa carta dobra em relação à sua aparição anterior;
- a fadiga total causada por um tipo de carta é a soma dessas fadigas sucessivas;
- a fadiga total do trecho é a soma das fadigas de todos os tipos de carta.

Na prática, isso significa que:

- na primeira vez que uma carta aparece, ela custa f_x;
- na segunda vez, ela custa $2f_x$;
- na terceira, $4f_x$;
- na quarta, $8f_x$, e assim por diante.

Erick quer escolher o trecho de exatamente L jogadas com a menor fadiga total possível, para otimizar seu treino, parar de tiltar e finalmente alcançar o glorioso *shape do gigante*.

Ajude o Erick a encontrar esse trecho antes que ele tilte novamente.

Abaixo estão as duas imagens que motivaram toda essa história:

Erick antes de sua sessão de treinos claramente abalado

Erick depois de sua sessão de treinos claramente mogando todo mundo

Entrada

A primeira linha contém três inteiros N, M e L ($1 \leq N \leq 2 \cdot 10^5$, $1 \leq M \leq 2 \cdot 10^5$, $1 \leq L \leq N$), onde:

- N é o número total de jogadas registradas;
- M é o número de tipos diferentes de carta no deck;
- L é o tamanho exato do trecho que deve ser analisado.

A segunda linha contém M inteiros $f_1, f_2, \ldots, f_M$ ($1 \leq f_x \leq 10^9$), onde f_x representa a fadiga base da carta de tipo x.

A terceira linha contém N inteiros $c_1, c_2, \ldots, c_N$ ($1 \leq c_i \leq M$), onde c_i é o tipo da carta usada na i-ésima jogada.

É garantido que, para qualquer trecho de tamanho L, a fadiga total será estritamente menor que $1 \cdot 10^{18}$.

Saída

Imprima uma única linha contendo um inteiro: a menor fadiga total possível entre todos os trechos contínuos de exatamente L jogadas.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
1 1 1 5 1	5

Exemplo de entrada 2	**Exemplo de saída 2**
5 3 2 10 3 7 1 2 2 3 1	9

Observações

No segundo caso teste temos $L = 2$, então cada trecho analisado possui duas jogadas. Os trechos possíveis e suas fadigas são:

- Trecho $[1, 2]$: $10 + 3 = 13$;
- Trecho $[2, 2]$: $3 + 6 = 9$ (segundo 2 custa o dobro);
- Trecho $[2, 3]$: $3 + 7 = 10$;
- Trecho $[3, 1]$: $7 + 10 = 17$.

O menor valor entre todos os trechos é 9.

Problema E. Esmeraldas do Caos

Tempo limite: 1000 ms
Memória limite: 256 MiB

Depois de muitas aventuras, Sonic já conseguiu reunir cinco das sete Esmeraldas do Caos que precisa para salvar o mundo da tirania do dr. Eggman! Entretanto, as últimas duas esmeraldas ainda estão sob posse do vilão, e ficou a cargo de Miles "Tails" Prower, seu fiel amigo, recuperá-las.

Tails descobriu que as esmeraldas estão no parque de diversões espacial do vilão[1]. Ao chegar no parque, dr. Eggman desafiou Tails para um jogo (que ele mesmo inventou) cujo vencedor irá ficar com as esmeraldas restantes.

O parque tem N estações e M conexões diretas, de mão única, entre elas. Inicialmente, as esmeraldas serão colocadas em duas estações distintas do parque, uma em cada. Então, Tails e dr. Eggman jogam em turnos alternados, onde Tails jogará primeiro. A cada turno, o jogador deverá escolher uma das esmeraldas que está em alguma estação e movê-la para outra estação, através de uma conexão direta.

Perde o jogador que não puder mais fazer uma jogada em seu turno. O parque não contém ciclos e, logo, o jogo terminará em um número finito de turnos.

Tails é uma raposa muito inteligente, e conseguiu convencer dr. Eggman a, antes do jogo começar, escolher ele mesmo o par de estações nas quais as esmeraldas irão iniciar. Ajude Tails a determinar quantos pares de estações distintas ele pode escolher de tal forma que, sabendo que ambos jogam de maneira ótima, ele vencerá o jogo se as esmeraldas começarem nessas estações.

Determine o resto da divisão da resposta por $10^9 + 7$.

Entrada

A primeira linha contém dois inteiros N e M ($1 \leq N \leq 10^5$, $0 \leq M \leq 2 \times 10^5$), o número de estações e de conexões diretas respectivamente. Cada uma das próximas M linhas contém dois inteiros A e B ($1 \leq A, B \leq N, A \neq B$) indicando uma conexão direta da estação A para a estação B. É garantido que o parque não contém ciclos.

Saída

Imprima uma linha com a quantidade de pares de estações distintas que Tails pode escolher, módulo $10^9 + 7$.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
3 3 1 2 1 3 2 3	3

[1] Não sabemos por que dr. Eggman construiu um parque de diversões, ainda mais no espaço!

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

Exemplo de entrada 3	**Exemplo de saída 3**
10 0	0

Observações

No primeiro exemplo, Tails consegue vencer o jogo se as esmeraldas começarem em qualquer dos 3 pares de estações possíveis (1 e 2; 1 e 3; 2 e 3). Se as esmeraldas começarem nas estações 1 e 2, por exemplo, Tails pode iniciar movendo uma da estação 1 para a 2. Na sequência, dr. Eggman é obrigado a mover uma da estação 2 para a 3. Tails então move a outra da estação 2 para a 3. Neste ponto dr. Eggman não pode mais fazer uma jogada, e portanto perde o jogo.

No segundo exemplo, Tails consegue vencer se as esmeraldas começarem em um dentre 6 pares possíveis, incluindo, por exemplo, o par 2 e 5.

Problema F. Factorio

Tempo limite: 4000 ms
Memória limite: 512 MiB

Eduardo estava jogando Factorio, um jogo onde ele precisa criar fábricas e gerenciar seus recursos. No momento ele precisa criar uma conexão entre suas N máquinas, de forma a transportar os items das máquinas geradoras, transformá-los nas máquinas transformadoras e por fim leva-los às máquinas coletoras.

As N máquinas são númeradas de 1 à N, onde as primeiras G máquinas são geradoras, as últimas C máquinas são coletoras, e as demais são transformadoras. Existe pelo menos uma máquina geradora e uma máquina coletora, e $G + C \leq N$.

Para transportar os items entre entre suas máquinas para que elas produzam os items necessários, foram adicionadas M esteiras entre as máquinas. Cada esteira conecta duas máquinas, onde a esteira i tem velocidade v_i, isto é, consegue transportar até v_i items por segundo de uma máquina para a outra.

Acontece que após conectar todas as máquinas, ele queria configura-lás para transportar o máximo possível de items para as máquinas coletoras, porém ficou perdido com o tamanho do sistema que criou. Agora ele precisa da sua ajuda para configurar essas máquinas.

Para configurar uma máquina, ele deve definir quantos items irão para cada esteira que sai da máquina. O número total de items por segundo que saem de uma máquina transformadora deve ser igual ao número total de items por segundo que entram na máquina. Máquinas geradoras nunca possuem esteiras entrando e podem ter como saída qualquer número de items, enquanto máquinas coletoras não possuem esteiras saindo e podem ter como entrada qualquer número de items (sempre respeitando as velocidades das esteiras).

Entrada

A primeira linha da entrada consiste nos inteiros N, G, C e M ($2 \leq \mathrm{N} \leq 2000$, $1 \leq G, C \leq N$, $2 \leq G + C \leq N$, $M \leq N^2$).

Cada uma das próximas M linhas consiste em 3 inteiros x_i, y_i e v_i ($1 \leq x, y \leq N$, $1 \leq v_i \leq 10^9$), representando uma esteira que vai da máquina x_i para a máquina y_i com velocidade v_i. É garantido que as esteiras são dadas de forma a respeitar as restrições dadas no enunciado.

Saída

A saída deve conter um único inteiro, o número total máximo de items por segundo que chegam nas máquinas coletoras.

Exemplos

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

Exemplo de entrada 2	**Exemplo de saída 2**
9 5 2 7 4 7 5 1 6 3 2 6 1 1 7 6 2 8 6 5 9 3 6 9 4	13

Problema G. Grand Theft Auto

Tempo limite: 1000 ms
Memória limite: 256 MiB

Em Los Santos, um dos maiores golpes da história está sendo planejado. Você é o responsável por preparar o equipamento necessário para o assalto: armas, coletes, explosivos e ferramentas.

A loja de equipamentos Ammu-Nation oferece N produtos, cada um com um preço. Seu objetivo é comprar exatamente K produtos, gastando o mínimo possível, sem ultrapassar o orçamento total 'A' destinado à operação.

Caso não seja possível comprar todos os K produtos dentro do orçamento, o plano deve ser abortado.

Dado o número total de produtos disponíveis, o orçamento e o preço de cada produto, determine o custo mínimo total para adquirir exatamente K produtos. Se não for possível realizar a compra sem exceder o orçamento, imprima "Nao e possivel".

Entrada

A entrada contém três linhas:

A primeira linha contém dois inteiros N e K ($1 \leq K \leq N \leq 2 \times 10^5$); o número total de produtos disponíveis e o número de produtos que devem ser comprados.

A segunda linha contém um inteiro A ($1 \leq A \leq 10^{18}$); o orçamento total disponível para o assalto.

A terceira linha contém N inteiros p_i ($1 \leq p_i \leq 10^9$), representando os preços dos produtos disponíveis.

Saída

Imprima um único valor inteiro representando o custo mínimo total para comprar exatamente K produtos sem ultrapassar o orçamento. Caso não seja possível, imprima "Nao e possivel".

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
6 3 100 40 10 30 50 60 20	60

Exemplo de entrada 2	**Exemplo de saída 2**
5 4 50 10 15 20 25 30	Nao e possivel

Problema H. Hollow Knight

Tempo limite: 1000 ms
Memória limite: 256 MiB

Após longas batalhas pelos recantos esquecidos de Hallownest, o pequeno Cavaleiro chega novamente à loja de Salubra, a misteriosa vendedora de amuletos. Com novos desafios à frente, ele precisa montar a build perfeita, escolhendo com sabedoria os amuletos que irá equipar. O Cavaleiro possui M entalhes (notches) disponíveis em seu casco. Salubra oferece N amuletos diferentes, e cada amuleto i ($1 \leq i \leq N$) possui:

- Um custo em entalhes, C_i;
- Um poder de combate, P_i.

Porém, o Cavaleiro também domina uma técnica rara e perigosa: a Sobrecarga. Ao ativá-la, ele pode equipar um amuleto adicional, mesmo que o total de entalhes ultrapasse o limite natural de M. Contudo, essa sobrecarga só pode ser usada uma única vez — o Cavaleiro só suporta um amuleto em sobrecarga antes que o casco se quebre.

Sua tarefa é ajudar o Cavaleiro a escolher um conjunto de amuletos tal que a soma total dos custos C_i não ultrapasse os M entalhes disponíveis, e a soma total dos "Poderes de Combate" P_i seja a maior possível.

Entrada

A primeira linha da entrada contém dois inteiros N ($1 \leq N \leq 1000$) e M ($1 \leq M \leq 1000$), que representam o número de amuletos diferentes que Salubra oferece e o número total de entalhes que o Cavaleiro possui.

As N linhas seguintes descrevem os amuletos. Cada linha i contém dois inteiros C_i e P_i:

- C_i ($1 \leq C_i \leq M$): o custo em entalhes do i-ésimo amuleto.
- P_i ($1 \leq P_i \leq 100$): o poder de combate do i-ésimo amuleto.

Saída

Imprima um único inteiro representando o máximo poder de combate total que o Cavaleiro pode obter, considerando que ele pode usar a sobrecarga uma vez.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
5 10 7 60 8 70 5 40 4 35 6 50	155

Exemplo de entrada 2	**Exemplo de saída 2**
8 20 5 40 4 30 3 30 2 20 2 20 1 10 1 10 12 80	240

Observações

Note que para o primeiro caso o amuleto de custo 8 que utiliza a sobrecarga, junto com ele os amuletos de custo 6 e 4 são utilizados pelo cavaleiro.

Problema I. Injustice

Tempo limite: 1000 ms
Memória limite: 256 MiB

Quando uma fenda dimensional instável se abriu sobre Metrópolis, os sensores do Batman registraram uma grandeza energética peculiar associada ao planeta, representada por um parâmetro discreto P. A situação se agravou quando o Coringa ativou um artefato kryptoniano capaz de intensificar essa energia segundo um fator multiplicativo M. A Liga da Justiça precisou avaliar o impacto dessa combinação antes que o planeta sofresse desequilíbrio irreversível.

Entrada

A entrada consiste em uma única linha contendo dois inteiros P e M ($-10^4 \leq P, M \leq 10^4$).

Saída

A saída deve conter apenas um inteiro representando o efeito máximo produzido pela combinação dos parâmetros P e M.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
`6 7`	`42`

Exemplo de entrada 2	**Exemplo de saída 2**
`31 27`	`837`

Exemplo de entrada 3	**Exemplo de saída 3**
`-532 414`	`-220248`

Problema J. Jornada do Ethan

Tempo limite: 3000 ms
Memória limite: 1024 MiB

Após os acontecimentos trágicos na mansão em Louisiana, Ethan e sua esposa Mia tentaram recomeçar a vida em um vilarejo isolado nas montanhas do leste europeu. Por algum tempo, Ethan, Mia e sua recém-nascida filha Rose viveram tranquilamente até que, numa noite fria, algo terrível aconteceu.

Um grupo de homens armados invadiu sua casa, abriu fogo contra Mia e levaram Ethan inconsciente. Algum tempo depois, acorda ferido e desorientado em meio à neve, cabanas e uivos em sua volta. Sem saber o que aconteceu com Mia e percebendo que sua filha desapareceu, Ethan parte em uma busca desesperada por Rose.

Para encontrá-la, ele terá de percorrer o vilarejo, recuperar as partes de uma antiga chave e abrir os portões que o separam de sua filha.

Mapa do Vilarejo

O vilarejo é representado por uma grade de caracteres com L linhas e C colunas. Cada célula do vilarejo pode conter um dos seguintes caracteres:

- `#`: Parede (caminho bloqueado);
- `.`: Caminho livre;
- `A`, `B`, `C`: portões de níveis 1, 2 e 3, respectivamente;
- `1`, `2`, `3`: partes da chave de níveis 1, 2 e 3, respectivamente;
- `E`: Posição inicial de Ethan fixada em (0,0) no mapa;
- `R`: Posição de Rose (destino final).

Movimento e Regras

- Ethan pode se mover nas quatro direções: cima, baixo, esquerda e direita.
- Cada movimento para uma célula adjacente conta como **1 passo**.
- Existe apenas um `E` e apenas um `R` no mapa.
- Ethan pode passar pela mesma célula quantas vezes precisar.
- Inicialmente, Ethan não possui nenhuma parte da chave (nível 0).
- As partes da chave devem ser coletadas na seguinte ordem obrigatória:

$$1 \rightarrow 2 \rightarrow 3$$

  Observação: Não é possível coletar a parte 2 antes da 1, nem a parte 3 antes da 2.
- As células que contêm partes da chave são consideradas caminhos livres (não bloqueiam a passagem).
- Sempre que Ethan estiver em uma célula contendo uma parte da chave, a coleta é automática desde que seja a parte correta da sequência.
- Para atravessar um portão de nível N, Ethan deve possuir uma chave de nível $\geq N$, ou seja uma chave de nível N também abre portões de níveis inferiores:
  - Chave nível 3 abre portões A, B e C;
  - Chave nível 2 abre portões A e B;
  - Chave nível 1 abre o portão A.

Entrada

A entrada será uma linha que contém dois inteiros L e C ($1 \leq L, C \leq 2500$), representando o número de linhas e colunas do vilarejo. Em seguida, seguem L linhas com C caracteres cada, descrevendo o vilarejo.

Saída

A saída será um inteiro, sendo menor número de passos necessários para Ethan alcançar Rose. Se não for possível chegar até ela, imprima -1.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
3 7 E....#R ##.#..B 12..C#.	16

Exemplo de entrada 2	**Exemplo de saída 2**
10 10 EC.B1#CC1. 1.....#.B. C#.#.1.... C...2.#.#3 A....###CC #...#.#... 3A..A2...B C...AB.AC. #2#2#B#.A. ##.#.3..CR	18

Observações

No primeiro exemplo, Ethan deverá primeiro pegar a parte 1 da chave (note que ele passa pela parte 2 para chegar à parte 1, mas a ignora em um primeiro momento). Após, Ethan poderá pegar a parte 2 da chave e abrir o portão B.

Problema K. Kurt, o Jogador de Golfe

Tempo limite: 1000 ms
Memória limite: 256 MiB

Kurt está jogando uma partida de minigolfe em um campo infinito com obstáculos.

Há N obstáculos em posições distintas (O_x, O_y) no plano. Cada obstáculo tem um tipo:

$<, >, \wedge, v$	portais que redirecionam a bola
$\#$	campo de areia, para a bola
o	buraco (único), objetivo

O restante do campo é grama.

Serão dadas M tacadas. Antes da primeira tacada a bola está em $(0, 0)$; cada nova tacada começa de onde a bola parou.

Em uma tacada com velocidade V_k e direção inicial T_k:

- a bola anda célula a célula na direção atual;
- a cada célula, a velocidade decrementa em 1;
- ao entrar em um portal $(<, >,' v)$, a direção muda imediatamente;
- ao entrar em areia $\#$, a tacada termina na hora (velocidade zera);
- ao entrar no buraco o, o jogo termina.

O movimento segue enquanto houver velocidade positiva.

Pergunta: em qual tacada (1-indexada) a bola atinge o buraco? Se nunca, imprima -1.

Entrada

A primeira linha contém N e M.

Cada uma das próximas N linhas tem O_x, O_y e id_i $(<, >, \wedge, v, \#, o)$.

Em seguida, M linhas com V_k e T_k $(<, >, \wedge, v)$.

$$1 \leq N, M \leq 1000, \quad -1000000 \leq O_x, O_y \leq 1000000, \quad 1 \leq V_k \leq 100$$

As posições são distintas e há exatamente um buraco.

Saída

Imprima o índice da tacada em que o buraco é atingido, ou -1 se isso nunca ocorrer.

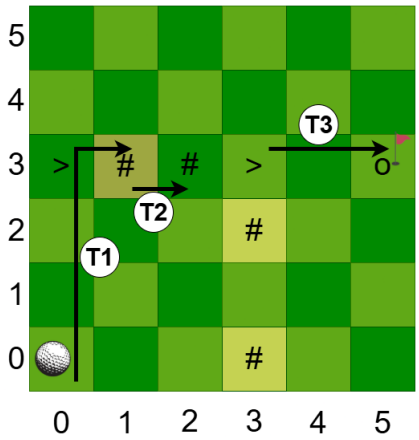
Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
7 3 3 0 # 3 2 # 0 3 > 1 3 # 2 3 # 3 3 > 5 3 o 7 ^ 10 > 8 >	3

Exemplo de entrada 2	**Exemplo de saída 2**
3 2 2 0 v 2 -3 > 5 -3 o 10 > 5 v	1

Observações

Exemplo (entrada 1):



- Obstáculos em $(3,0)$ #, $(3,2)$ #, $(0,3)$ >, $(1,3)$ #, $(2,3)$ #, $(3,3)$ >, $(5,3)$ *o*.
- Tacadas: $(7,\wedge)$, $(10,>)$, $(8,>)$.

Simulação:

- Tacada 1: sobe até portal $(0,3)$, vira para direita e para na areia em $(1,3)$.
- Tacada 2: começa em $(1,3)$, para na areia $(2,3)$.
- Tacada 3: anda pela direita, passa pelo portal $(3,3)$ e cai no buraco em $(5,3)$.

Resposta: 3.

Problema L. Lara Croft

Tempo limite: 1000 ms
Memória limite: 256 MiB

A destemida arqueóloga Lara Croft está no coração da Cordilheira dos Andes chilenos, perseguindo o Amuleto de Pillán, um artefato Mapuche que detém o poder vulcânico. O item está escondido no topo de uma formação rochosa traiçoeira, mapeada em uma grade $N \times N$.O Equipamento: Lara conta com seus Piolets Técnicos de última geração para a escalada.

O sucesso depende da identificação de pontos de ancoragem seguros: Ancoragem Segura (#): Rocha sólida ou gelo firme. O piolet se finca com segurança, permitindo a progressão. Superfície Ligeira (0): Gelo negro escorregadio ou granito polido. O piolet desliza, tornando o ponto impraticável para escalada.

Lara precisa otimizar sua rota para enfrentar a altitude e a ameaça de outros caçadores de tesouros. Seu desafio é programar um sistema que, dada a matriz da montanha, encontre as coordenadas (l, c) do ponto de ancoragem seguro (#) ideal, seguindo a prioridade de uma verdadeira Croft: Prioridade Máxima: O ponto deve ser o mais alto possível (ter o menor índice de linha, l). Critério de Desempate: Se houver múltiplos pontos na mesma linha mais alta, o ideal é o mais à direita (ter o maior índice de coluna, c). Encontre o ponto (l, c) onde Lara deve pousar seu último piolet!

Entrada

Um inteiro N ($1 \leq N \leq 20$) que é a dimensão da matriz com N linhas subsequentes, cada uma com N caracteres (# ou 0).

A matriz sempre terá um ou mais #.

Saída

Um único par de inteiros (l, c), representando as coordenadas (linha, coluna) do ponto de ancoragem ideal.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
3 00# 0#0 #00	(1, 3)

Exemplo de entrada 2	**Exemplo de saída 2**
5 00000 0#### 00000 00000 00000	(2, 5)

Exemplo de entrada 3	**Exemplo de saída 3**
1 #	(1, 1)

Exemplo de entrada 4	**Exemplo de saída 4**
5 00000 00000 00000 00000 0000#	(5, 5)

Exemplo de entrada 5	**Exemplo de saída 5**
4 #0#0 0000 0000 0000	(1, 3)

Problema M. Monstros do RicardAsh

Tempo limite: 1000 ms
Memória limite: 256 MiB

Ricardo, mais conhecido no mundo Pokémon como RicardAsh, é um treinador extremamente dedicado. Quando não está batalhando em ginásios ou participando de torneios, ele atua como professor da UTFPR nas horas vagas.

Certo dia, RicardAsh foi convidado a visitar um antigo laboratório Pokémon que possui um total de n Pokémons disponíveis para adoção. Como treinador experiente, ele pode escolher quantos Pokémons quiser para levar consigo.

Cada Pokémon tem uma força inteira s_i. RicardAsh sabe, porém, que grupos de Pokémons com certas combinações de forças tendem a brigar entre si.

Mais precisamente, qualquer grupo com $k > 1$ Pokémons, cujas forças sejam

$$\{s_{i_1}, s_{i_2}, \ldots, s_{i_k}\},$$

vai brigar se

$$\text{mdc}(s_{i_1}, s_{i_2}, \ldots, s_{i_k}) = 1.$$

RicardAsh não quer que os seus Pokémons briguem, mas também quer levar o maior número possível de Pokémons do laboratório.

Ajude RicardAsh a descobrir o tamanho máximo de um conjunto de Pokémons que ele pode escolher de forma que nenhum subconjunto com dois ou mais Pokémons tenha máximo divisor comum igual a 1.

Obs: um Pokémon sozinho nunca briga consigo mesmo.

Entrada

A entrada consiste em duas linhas.

A primeira linha contém um inteiro n ($1 \leq n \leq 10^5$), o número de Pokémons disponíveis no laboratório.

A segunda linha contém n inteiros $s_1, s_2, \ldots, s_n$ ($1 \leq s_i \leq 10^5$), onde s_i é a força do i-ésimo Pokémon.

Saída

Imprima um único inteiro: o máximo número de Pokémons que RicardAsh pode escolher de forma que nenhum grupo com dois ou mais Pokémons tenha máximo divisor comum igual a 1.

Exemplos

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

Exemplo de entrada 2	**Exemplo de saída 2**
5 2 3 4 6 7	3

Observações

O máximo divisor comum (mdc) de um conjunto de inteiros positivos $\{a_1, a_2, \ldots, a_k\}$ é o maior inteiro positivo que divide todos eles.

Problema N. Naval

Tempo limite: 1000 ms
Memória limite: 256 MiB

Este é um problema interativo. Você deve usar a operação de flush após cada linha impressa. Por exemplo, em C++ você deve usar `fflush(stdout)`*, em C você deve usar* `fflush(stdout)`*, em Java você deve usar* `System.out.flush()`*, e em Python você deve usar* `sys.stdout.flush()`*.*

Henrique e Tanaka um dia estavam jogando juntos *Batalha Naval*, um jogo estratégico onde cada jogador possui uma grade 14×14. Cada jogador deve posicionar em sua grade uma série de navios, e em seguida ambos se revezam tentando adivinhar as posições dos navios do oponente. A cada tentativa, o adversário deve informar se a coordenada escolhida resultou em um erro, acerto ou se afundou um dos navios.

Após várias partidas, Henrique saiu vitorioso em todas elas. Tanaka afirmava que isso era apenas sorte, mas Henrique discordava, dizendo que escolhia suas coordenadas estrategicamente a cada turno.

Diante desse conflito, foi proposto um desafio: Henrique afirmou que, se Tanaka posicionar um navio porta-aviões (um navio de tamanho 5×1, na horizontal ou na vertical) em qualquer lugar de sua grade, ele seria capaz de afundá-lo em no máximo 44 jogadas.

Sua tarefa é ajudar Henrique a escrever um programa que jogue *Batalha Naval* de forma ótima, garantindo que o navio seja afundado em no máximo 44 tentativas.

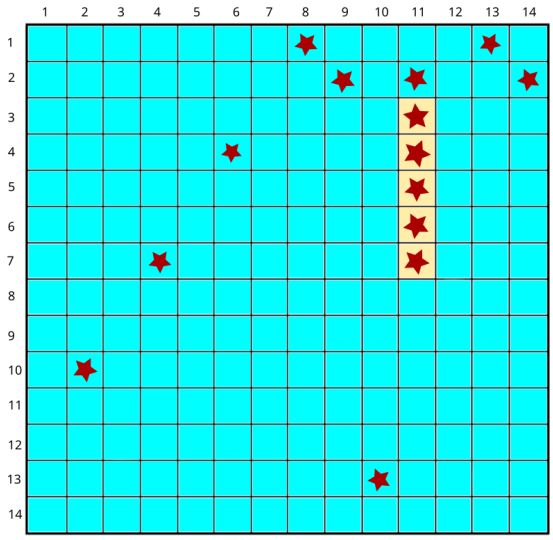


Figura representando os chutes do exemplo 1

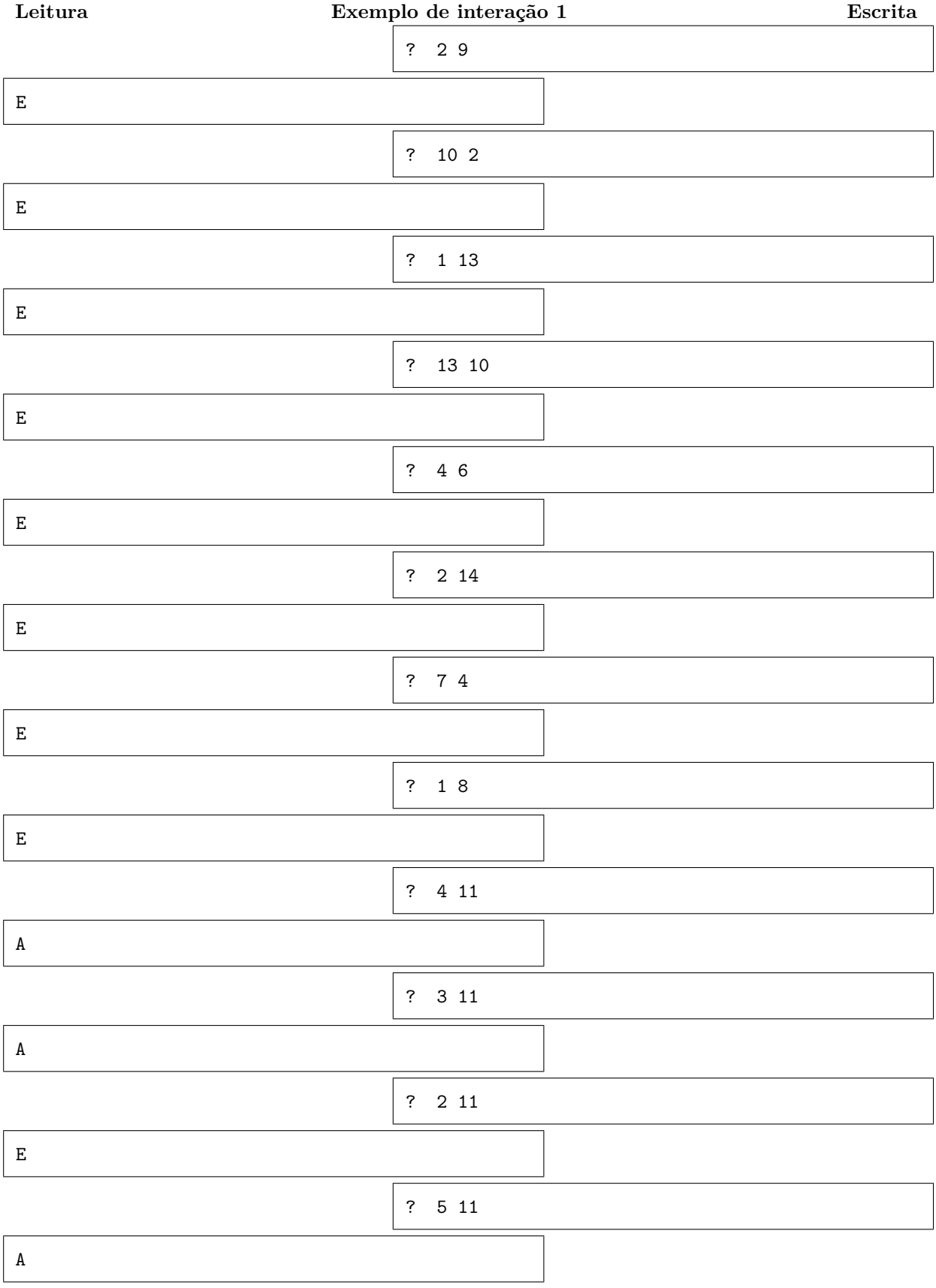
Interação

Em cada turno, seu programa deve imprimir um chute, sendo ele uma interrogação (?) seguida de uma coordenada (X, Y), com $(1 \leq X, Y \leq 14)$. Em seguida seu programa deve ler a resposta do juiz, que será uma das seguintes:

- `E`: Caso o tiro tenha errado o navio;
- `A`: Caso o tiro tenha acertado parte do navio;
- `BOOM`: Caso o tiro tenha afundado completamente o navio.

Na sequência, o próximo turno começa com o seu programa imprimindo outro chute, e assim por diante. Seu programa deve encerrar imediatamente após receber a resposta `BOOM` do juiz. É garantido que, para todos os casos de teste, é possível afundar o navio em no máximo 44 jogadas.

Exemplos



```
? 6 11
```

```
A
```

```
? 7 11
```

```
BOOM
```

Observações

Um grande agradecimento ao Roberto Sales pela ajuda na complicada tarefa de fazer o package do problema interativo rodar em nosso BOCA!