II Maratona SBC de Programação do Cerrado

Caderno de Problemas

25 de abril de 2026

(Este caderno contém 14 problemas)

Comissão Organizadora:
Adne Moretti Moreira
Alberto Tavares Duarte Neto
Daniel Porto
Daniel Saad Nogueira Nunes
Dayllon Vinícius Xavier Lemos
Edson Alves da Costa Júnior
Guilherme Novaes Ramos
Gustavo Machado Leal
Jeremias Moreira Gomes
José Marcos Leite
Maxwell Oliveira dos Reis
Pedro Gallo
Ruan Petrus Alves Leite

Patrocínio:
NEOSPACE

Apoio:
GigaCandanga

Universidade de Brasília, Faculdade de Ciências e Tecnologias em Engenharia, Gama

Orientações

- É permitido consultar livros, anotações ou qualquer outro material impresso durante a prova, entretanto, o mesmo não vale para materiais dispostos eletronicamente.
- A correção é automatizada, portanto, siga atentamente as exigências da tarefa quanto ao formato da entrada e saída conforme as amostras dos exemplos. Deve-se considerar entradas e saídas padrão;
- Para cada problema, além dos testes públicos, o juiz executará a sua submissão contra uma série de testes secretos para fornecer um parecer sobre a correção do programa.
- Procure resolver o problema de maneira eficiente. Se o tempo superar o limite pré-definido, a solução não é aceita. Lembre-se que as soluções são testadas com outras entradas além das apresentadas como exemplo dos problemas;
- Utilize a aba *clarification* para dúvidas da prova. Os juízes podem opcionalmente atendê-lo com respostas acessíveis a todos;

C/C++

- Seu programa deve retornar zero, executando, como último comando, `return 0` ou `exit(0)`.
- É sabido que em alguns casos de problemas com entradas muito grandes, os objetos `cin` podem ser lentos, por conta da sincronização de buffers da biblioteca `stdio`.
- Caso deseje-se utilizar `cin` e `cout`, um jeito de se contornar este problema é executando-se o comando `ios_base::sync_with_stdio(0)`, no início de sua função `main`.
- Note que, neste caso, o uso de `scanf` e `printf` no mesmo programa é contra-indicado, uma vez que, com buffers separados, comportamentos inadequados podem ocorrer.

Java

- Não declare 'package' no seu programa Java.
- Note que a convenção para o nome do arquivo fonte deve ser obedecida, o que significa que o nome de sua classe pública deve ser uma letra maiúscula (A, B, C, etc).
- Comando para executar uma solução Java:
  `java -Xms1024m -Xmx1024m -Xss100m codigo_do_problema`

Kotlin

- Não declare 'package' no seu programa Kotlin.
- Note que a convenção para o nome do arquivo de solução deve ser obedecida, o que significa que seu arquivo-fonte deve ser nomeado (A.kt, B.kt, C.kt, etc).
- Comando para executar uma solução Kotlin:
  `java -Xms1024m -Xmx1024m -Xss100m codigo_do_problemaKt`

Python

- Note que apenas Python 3 é suportado.
- As submissões em Python terão sua sintaxe verificada; programas que não passarem nessa verificação receberão o veredito "Compilation Error".
- Note que, este ano, as submissões em Python serão executadas com o interpretador PyPy, em vez do CPython tradicional. Isso geralmente resulta em melhor desempenho, mas esteja atento a eventuais diferenças sutis de comportamento.
- Comando para executar uma solução Python: `pypy3 {fonte}.py`

Problema A – Águas Claras

Limite de tempo: 1s
Limite de memória: 256MB

Autor: Adne Moretti Moreira

O bairro de Águas Claras, no Distrito Federal, é conhecido por seus prédios altos, diferentemente do Plano Piloto de Brasília, onde há restrições de altura. Com o crescimento da região, uma nova área foi planejada para expansão, permitindo a construção de novos prédios.

O planejamento da construtora responsável pela obra é de construir prédios lado a lado ao longo de uma linha horizontal e, a cada mês, terminar e entregar um deles para os moradores, cada um com uma altura inicial h. Os prédios estão dispostos em linha reta, numerados de 1 a N, em que o prédio i possui altura h_i.

Com a alta demanda esperada, a construtora pretende continuar a expansão dos prédios já entregues. Para isso, estabeleceu um cronograma com C construções, no qual, em alguns meses m, são acrescidos a novos andares sobre todos os prédios já existentes, incluindo o construído no próprio mês.

Thalisson está procurando um novo prédio para morar e, por ser admirador de cidades grandes, o critério de decisão é que, no momento da entrega, ele consiga ter o melhor campo de visão de prédios possível. O campo de visão a partir de um prédio j é definido como a quantidade de prédios à sua esquerda que podem ser vistos a partir dele.

Dizemos que um prédio j ($j < i$) é visível a partir do prédio i se não existe um prédio k tal que $j < k < i$ e $h_k \geq h_j$. Em outras palavras, o prédio j é visível se nenhum prédio entre j e i o bloqueia, sendo igual ou mais alto do que ele.

Seu objetivo é ajudar Thalisson a escolher o seu novo apartamento, indicando, para cada prédio j, quantos prédios à sua esquerda são visíveis a partir dele, no momento em que ele é entregue.

Assumindo que no primeiro prédio construído nunca será possível ver nenhum prédio, imprima $N - 1$ inteiros, correspondentes aos prédios de índices 2 até N, em que o i-ésimo valor representa o número de prédios visíveis à esquerda do prédio i.

Entrada

A primeira linha contém dois números inteiros, N ($2 \leq N \leq 10^5$), representando o número de prédios a serem construídos, e C ($1 \leq C \leq N$), representando o número de construções previstas, separados por um espaço em branco.

A segunda linha contém N números inteiros, separados por um espaço em branco, indicando a altura inicial h_i ($1 \leq h_i \leq 10^9$) do prédio entregue no mês i.

As próximas C linhas possuem 2 números inteiros m_i ($1 \leq m_i \leq 10^5$) e a_i ($1 \leq a_i \leq 10^9$), separados por um espaço em branco, indicando o mês que a construção será executada e a altura adicionada em cada prédio que já foi construído até o mês da construção, respectivamente.

Saída

Para cada prédio i ($2 \leq i \leq N$), imprima um número inteiro representando a quantidade de prédios visíveis à sua esquerda.

Exemplo

Entrada	Saída
5 1 12 17 18 2 1 1 6	1 2 1 2
6 2 6 11 120 200 20 7 1 6 3 81	1 2 1 2 3

Notas

No primeiro exemplo, a construtora planejou apenas uma construção, realizada no primeiro mês. Após a execução dessa expansão, as alturas dos prédios passam a ser:

18 17 18 2 1

A partir disso, temos que:

- Do prédio de índice 2, é possível ver apenas 1 prédio.
- Do prédio de índice 3, é possível ver 2 prédios.
- Do prédio de índice 4, é possível ver apenas 1 prédio, pois o prédio de índice 3 bloqueia a visão de todos os anteriores.
- Do prédio de índice 5, é possível ver 2 prédios, especificamente os de índices 3 e 4.

Problema B – Biscoitos Mineiros

Limite de tempo: 1s
Limite de memória: 256MB

Autor: Edson Alves

Após a realização bem-sucedida de mais um evento de programação competitiva, os professores Costa e Saad foram tomar um café na tradicional Casa Biscoitos Mineiros (boa parte dos *coffee-breaks* das maratonas do Distrito Federal são de lá). Como sempre há a discussão de quem vai pagar a conta, eles resolveram decidir a questão na sorte.

Porém, em vez de usar o tradicional "Par ou Ímpar", e inspirados pelo formato das duas casas do Congresso Nacional (as quais, juntas, remetem vagamente a uma seção de uma curva trigonométrica), eles optaram pelo "Seno ou Cosseno": nesta variante, os participantes optam por "Seno" ou "Cosseno" e depois sorteiam, por meio de um gerador de randômicos uniforme, um inteiro a entre 0 e 90, o qual representará um ângulo, em graus. Pagará a conta quem escolheu a função trigonométrica que tem o maior valor em a. Em caso de empate, ambos pagarão a conta, meio a meio.

Sabendo que o professor Costa optou por "Cosseno" e o professor Saad por "Seno", e que o ângulo sorteado foi a, determine quem pagará a conta (ou se ela será dividida entre ambos).

Entrada

A entrada contém um único inteiro a ($0 \leq a \leq 90$).

Saída

Imprima, em uma linha, o nome do professor que pagará a conta. Em caso de empate, imprima a mensagem "`Ambos`".

Exemplo

Entrada	Saída
30	Costa
60	Saad

Notas

No primeiro caso, temos que $\text{sen}(30) = \frac{1}{2} = 0.5 < \cos(30) = \frac{\sqrt{3}}{2} = 0.8660254037844386$, portanto o professor Costa pagará a conta.

No segundo caso, $\text{sen}(60) = \frac{\sqrt{3}}{2} > \cos(60) = \frac{1}{2}$, logo o professor Saad será o responsável pela conta.

Problema C – Consulta ao Oráculo

Limite de tempo: 1s
Limite de memória: 256MB

Autor: Daniel Saad Nogueira Nunes

Durante as obras de recuperação do Buraco do Tatu, localizado entre o Eixão Norte e Sul, foi encontrado o marco zero da capital do país, que teve importância histórica fundamental para a construção da capital do país. Além do marco zero, o Eng. Oswaldo Tatuska também encontrou um artefato antigo, em formato de uma máscara, capaz de responder perguntas sobre o futuro da capital federal: um verdadeiro oráculo.

Contudo, Tatuska verificou que, para acionar o oráculo, era necessário responder a um desafio, que consiste em, dadas as palavras $S = s_0 s_1 \dots s_{n-1}$ e $T = t_0 t_1 \dots t_{m-1}$, determinar se T^k ocorre em S^∞, em que T^k é a concatenação de k cópias de T e S^∞ é a concatenação de infinitas cópias de S, para algum $k \geq 0$. Como complicação adicional, o oráculo ainda possuía um sistema de pesos: cada posição i de S^∞ tem um peso $w_{i \mod n}$ associado. O peso de uma ocorrência de T^k em S^∞ é a soma dos pesos das posições iniciais, módulo n, das ocorrências de cada uma de suas cópias de T.

Assim, ajude Tatuska a determinar o maior peso possível de uma ocorrência de T^k em S^∞ para algum $k \geq 0$. Se o peso da ocorrência de T^k for infinito, Tatuska deve indicar ao oráculo que não existe uma resposta.

Entrada

A primeira linha da entrada possui uma palavra $S = s_0 s_1 \dots s_{n-1}$ $(1 \leq n \leq 2 \times 10^5)$.

A segunda linha da entrada possui uma palavra $T = t_0 t_1 \dots t_{m-1}$ $(1 \leq m \leq 2 \times 10^5)$.

A terceira linha da entrada possui n inteiros $(w_0, w_1, \dots, w_{n-1})$, onde w_i $(1 \leq w_i \leq 10^2)$ é o peso associado à posição i de S^∞.

As palavras S e T são formadas apenas por letras minúsculas de a a z.

Saída

Imprima o maior peso de uma ocorrência de T^k em S^∞ para algum $k \geq 0$. Se o peso de T^k for infinito, imprima `-1`.

Exemplo

Entrada	Saída
xyzxyxy xy 1 1 1 1 1 1 1	3
xx xxxxxxx 1 2	-1
xyx yxxxy 1 2 3	0
xyxyyxyy xy 1 2 3 4 5 6 7 8	6

Notas

Observe que T^0 é a palavra vazia, que ocorre em qualquer posição de S^∞ e tem peso 0.

No primeiro exemplo, $T^3 =$ `xyxyxy` ocorre em S^∞ e as cópias de T começam nas posições 3, 5 e 7 de S^∞, que têm pesos 1, 1 e 1, respectivamente, totalizando um peso de 3. Existem outras possibilidades de

ocorrência de T^3 em S^∞, mas nenhuma delas tem peso maior do que 3. Já para $k \geq 4$, T^k não ocorre em S^∞.

No segundo exemplo, existem infinitas ocorrências de T^k em S^∞ para todo $k \geq 0$; assim, o peso total é infinito.

No terceiro exemplo, T^0, a palavra vazia é a única ocorrência de T^k em S^∞, e seu peso é 0.

No quarto e último exemplo, existem ocorrências de T^2 em S^∞ com peso 4, visto que as duas cópias de T casam a partir das posições 0 $\mod 8$ e 2 $\mod 8$ de S^∞. Contudo, há outra ocorrência de T^1 em S^∞ com peso 6, já que inicia na posição 5 $\mod 8$ de S^∞; logo a resposta é 6.

Problema D – Dossiês de Brasília

Limite de tempo: 1s
Limite de memória: 256MB

Autor: Ruan Petrus

No processo de digitalização dos documentos históricos de Brasília, os arquivistas do **Arquivo Público do Distrito Federal** trabalham com um documento principal representado por uma string S.

Com o passar dos anos, diversos manuscritos danificados foram encontrados em diferentes setores administrativos da capital. Cada manuscrito é representado por uma string T_i.

Para medir o grau de diferença entre um manuscrito e o documento principal, os especialistas analisam:

- qualquer **trecho contínuo** do documento principal S, e
- qualquer **trecho contínuo** do manuscrito T_i,

incluindo, em ambos casos, também o trecho vazio.

A **distância arquivística** entre dois trechos A e B é definida como o número mínimo de operações necessárias para transformar A em B, sendo permitidas apenas as seguintes operações:

- remover o último caractere de A;
- adicionar um caractere ao final de A.

Para cada manuscrito T_i, determine a **maior distância arquivística possível** entre um trecho de T_i e um trecho do documento principal S.

Entrada

A primeira linha contém dois inteiros N ($1 \leq N \leq 10^6$) e Q ($1 \leq Q \leq 10^6$), separados por um espaço em branco, representando o número de caracteres que compõem o documento principal e o número de manuscritos encontrados, respectivamente.

A segunda linha contém uma string S, composta de N caracteres alfabéticos minúsculos, representando o documento principal.

Cada uma das próximas Q linhas contém uma string T_i ($1 \leq i \leq Q, |T_i| \geq 1$), representando um manuscrito. Todas estas strings contêm apenas caracteres minúsculos do alfabeto e é garantido que $\sum |T_i| \leq 10^6$.

Saída

Para cada manuscrito, imprima um único inteiro representando a maior distância arquivística possível entre um trecho de T_i e um trecho de S.

Exemplo

Entrada	Saída
4 3 aabc abc aaa de	6 5 6
3 4 aaa a aa aaa aaaa	3 3 3 4

Problema E – Engarrafamento na Estrutural

Limite de tempo: 2s
Limite de memória: 256MB

Autor: Edson Alves

Alberto e Edson estão retornando do campus Darcy Ribeiro da UnB para as suas casas, em Ceilândia e Taguatinga, respectivamente, pela via Estrutural, e mais uma vez se depararam com um engarrafamento na pista.

Enquanto o carro se desloca a incríveis 2 km/h, eles decidiram se distrair com o seguinte jogo, que inicia com N inteiros positivos a_k. Os jogadores se alternam em rodadas, e no início de cada rodada o jogador tem que escolher um número primo p. Em seguida, ele deve realizar uma ou mais jogadas, sendo que cada jogada consiste nos seguintes passos:

1. Escolher um inteiro a_i, com $1 \leq i \leq N$, que seja divisível por p;
2. Somar p pontos à sua pontuação; e
3. Atualizar o valor de a_i de modo que $a_i \leftarrow q$, onde q é o quociente da divisão de a_i por p.

A rodada se encerra quando o jogador não puder fazer mais nenhuma jogada, e o jogo acaba se, em sua rodada, o participante não puder fazer nenhuma jogada. Como Edson está dirigindo, Alberto sempre joga a primeira rodada.

Você, que está de carona com eles, não acredita como os dois conseguem jogar este jogo de cabeça, sem papel, caneta nem calculadora! Para ter certeza de que eles não estão errando as contas, escreva um programa que determine a pontuação máxima de Alberto, caso ambos joguem de forma ótima.

Entrada

A primeira linha da entrada contém o valor do inteiro N ($1 \leq N \leq 2 \times 10^5$).

A segunda linha da entrada contém N inteiros a_i, ($1 \leq a_i \leq 10^7, 1 \leq i \leq N$), separados por um espaço em branco.

Saída

Imprima, em uma linha, a pontuação máxima que pode ser obtida por Alberto, caso ambos joguem de forma ótima.

Exemplo

Entrada	Saída
2 32 50	12
3 5 6 1	7
1 9699690	42

Notas

No primeiro caso, Alberto começa sua jogada com a escolha do número 2. Em seguida, ele procede com a seguinte sequência de jogadas, onde a primeira coluna mostra o número escolhido, a segunda o valor atualizado do número escolhido e a terceira a pontuação total de Alberto:

```
   nr          nr
escolhido   atualizado   pontuação
```

50	25	2
32	16	4
16	8	6
8	4	8
4	2	10
2	1	12

Após as jogadas de Alberto, restam os números 1 e 25, respectivamente. Edson escolhe o número 5 e seleciona os números 25 e 5, respectivamente, de modo que ele obtém 10 pontos e perde a partida para Alberto.

Problema F – Fifibonacci

Limite de tempo: 2s
Limite de memória: 256MB

Autor: Alberto Neto

Fifibonacci, filho de Fibonacci, visita Brasília. Após um passeio pela Praça dos Três Poderes, ele percebe que passam muitos pombos por ali. Ele decide anotar essa quantidade para cada momento k e observa que a quantidade $f(k)$ satisfaz a seguinte recorrência:

$$f(k) = a \times f(k-1) + b \times f(k-2)$$

para $k > 1$.

Ao voltar para seu país, Fifibonacci descobre que esqueceu suas anotações na sala de estudos da UnBalloon! Porém, além de lembrar dos coeficientes a e b, ele ainda se recorda de dois valores: $f(n) = v_n$ e $f(m) = v_m$ para momentos $n < m$. Sentindo-se desafiado, ele decide recuperar os valores iniciais $f(0)$ e $f(1)$ a partir das informações acima, já que, a partir dos valores iniciais, ele poderá calcular a recorrência f para qualquer valor.

Ajude Fifibonacci e calcule os valores iniciais $f(0)$ e $f(1)$. Como esses valores são grandes, calcule-os módulo 998244353.

Entrada

A primeira linha de entrada contém um inteiro t ($1 \leq t \leq 10^4$) — o número de casos de teste.

Cada uma das próximas t linhas contém os inteiros a, b, n, m, v_n, v_m ($0 \leq a, b, n, m, x, y < 998244353$; $0 < n < m$; $0 < a, b$), separados por um espaço em branco, representando os coeficientes da recorrência, os pontos conhecidos e os valores nos pontos, respectivamente.

É garantido que há uma resposta para os valores dados.

Saída

Para cada caso de teste, imprima uma linha com dois inteiros $f(0)$ e $f(1)$ — os valores da função f nos tempos 0 e 1.

Como os valores são grandes, calcule-os módulo 998244353.

Exemplo

Entrada	Saída
5	0 1
1 1 2 3 1 2	0 1
1 1 4 5 3 5	0 1
1 1 6 7 8 13	0 1
1 1 12 19 144 4181	6 7
1 2 3 6 33 279	
1	641313273 332333849
145407441 65246627 679743773 987515790 103343791 954296252	

Problema G – Gama F. C.

Limite de tempo: 1s
Limite de memória: 256MB

Autor: Guilherme Ramos

Quem ama, coda no Gama!

A *Gama Future Coders* é uma associação sem fins lucrativos para preparar os desenvolvedores do futuro. Além de muita prática de programação, os "gamars" também precisam se familiarizar com diversas linguagens de programação para ampliarem suas habilidades.

Um dos exercícios mais simples é transformar um programa em Python em instruções simples em Português. A ideia é ~~gerar um prompt para IA~~ reforçar como identificar os pontos relevantes de um programa de modo a replicar um comportamento em outra linguagem, afinal não se sabe qual a ferramenta adequada para um problema futuro...

O curso básico lida com as quatro instruções mais frequentes: entrada e saída de dados, condicional e repetição. Faça um programa que reconheça as estruturas fornecidas e gere um prompt de saída, conforme os exemplos.

Entrada

A entrada consiste uma estrutura dentre as quatro apresentadas, que são apresentadas em uma ou duas linhas com não mais de 100 caracteres conforme os exemplos.

Saída

Uma frase em Português descrevendo a instrução dada, conforme os exemplos.

Exemplo

Entrada	Saída
`x = input()`	`LEIA x`
`print(bar)`	`APRESENTE bar`
`if foo == bar:` `  print("if foo == bar:")`	`SE foo == bar ENTAO APRESENTE "if foo == bar:"`
`while ans != 'Hola!':` `  que_tal = input()`	`ENQUANTO ans != 'Hola!' LEIA que_tal`

Notas

É garantido que as instruções em Python são sintaticamente corretas e que só há um espaço entre os elementos. A entrada e saída são realizadas via funções `input` e `print`, respectivamente. É garantido que a entrada nunca recebe argumentos (sendo atribuída a uma nova variável) e a saída recebe exatamente um argumento (valor ou variável), e elas sempre são a última instrução fornecida. A condicional é definida pela instrução `if` e a repetição pela `while`, ambas sempre seguidas de uma expressão lógica terminada por dois

pontos. É garantido que não há estruturas aninhadas. Lembre-se de que Python aceita aspas simples ou duplas para definir uma string! O Gama F. C. não usa tabulação ou aspas triplas nessa atividade...

Mais especificamente, segue abaixo a gramática das entradas válidas em Python, em notação BNF:

```
  <P> ::= <IO> | <IF> | <WH>
 <IF> ::= if <E>: <IO>
 <WH> ::= while <E>: <IO>
 <IO> ::= <IN> | <OUT>
 <IN> ::= <id> = input()
<OUT> ::= print(<E>)
  <E> ::= <id> | <num> | <str> |
```

Para a versão em Português temos a seguinte gramática:

```
  <P> ::= <IO> | <IF> | <WH>
 <IF> ::= SE <E> ENTAO <IO>
 <WH> ::= ENQUANTO <E> <IO>
 <IO> ::= <IN> | <OUT>
 <IN> ::= LEIA <id>
<OUT> ::= APRESENTE <E>
  <E> ::= <id> | <num> | <str> | <E> <op> <E>
```

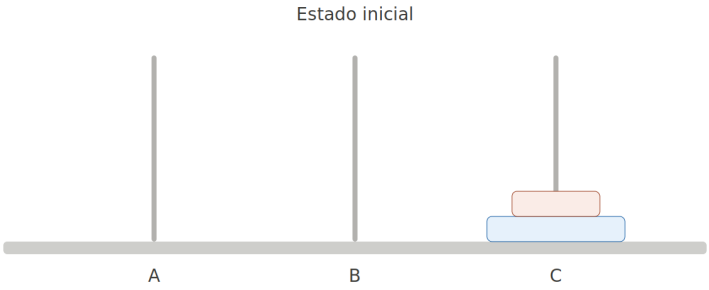
Problema H – Hanoi

Limite de tempo: 3s
Limite de memória: 512MB

Autor: José Leite

Oscar, arquiteto consagrado, sempre gostou de visitar o Vietnã. Durante suas viagens, se encantou pelo famoso problema da torre de Hanoi, descrito a seguir.

Há três torres enfileiradas, nas quais é possível empilhar discos. A torre da esquerda começa com N discos de tamanhos distintos empilhados em ordem do menor em cima e o maior embaixo; as outras torres começam vazias. O objetivo é mover todos os discos para a torre central por uma série de movimentos. Em um movimento, apenas um disco é movido do topo de uma torre para o topo de outra. Entretanto, um disco maior nunca pode ser colocado em cima de um menor.



Oscar tem uma versão de bolso deste quebra-cabeça, a qual usa no seu tempo livre. Oscar anda sem tempo recentemente, e suas torres estão bagunçadas, então pediu sua ajuda para terminar de resolver o problema. Obviamente você vai continuar a resolver só com movimentos válidos, mas vai usar o menor número de movimentos para não perder muito tempo.

Você se lembra vagamente das suas primeiras aulas de programação, em que seu professor mostrou um algoritmo recursivo para resolver o problema original com todos os discos na torre da esquerda:

```
hanoi(disco, origem, destino, auxiliar)
  se(disco > 0)
    hanoi(disco - 1, origem, auxiliar, destino)
    mover(origem, destino)
    hanoi(disco - 1, auxiliar, destino, origem)
hanoi(n, A, B, C)
```

O problema é que as torres de Oscar estão bagunçadas com cada disco podendo estar em qualquer lugar! Você consegue ajudá-lo?

Entrada

A primeira linha da entrada contém um inteiro N ($1 \leq N \leq 20$), o número de discos a serem movidos.

Cada uma das próximas três linhas contém um inteiro T ($1 \leq T \leq N$), o número de discos na i-ésima torre ($1 \leq i \leq 3$), seguido de T inteiros t_i, separados por um espaço em branco, indicado os discos empilhados na torre i, do topo à base. O disco 1 é o menor e o disco N é o maior.

É garantido que cada disco aparece em exatamente uma pilha.

Saída

A primeira linha da saída contém um inteiro k ($0 \leq k \leq 10^7$), o número mínimo de movimentos necessários para resolver o problema.

Cada uma das próximas k linhas devem conter dois caracteres x e y ($x, y \in \{A, B, C\}$), separados po um espaço em branco, os quais representam um movimento da pilha x para a pilha y. O caractere A corresponde à torre da esquerda, o caractere B a torre central e o caractere C a torre da direita.

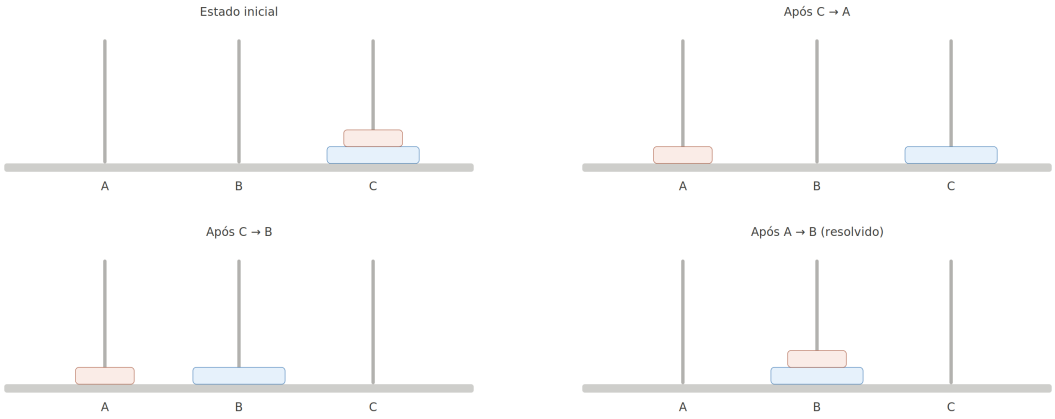
Cajo existam múltiplas soluções com o número mínimo de movimentos, imprima qualquer uma delas.

Exemplo

Entrada	Saída
2 0 0 2 1 2	3 C A C B A B
3 1 2 1 1 1 3	5 B A C B A C A B C B
10 0 10 1 2 3 4 5 6 7 8 9 10 0	0

Notas

O primeiro caso de teste corresponde a dois discos na terceira torre e pode ser resolvido por `hanoi(2, C, B, A)`.



Problema I – Índice Ecológico

Limite de tempo: 3s
Limite de memória: 256MB

Autor: Gustavo Leal

Em um projeto de monitoramento ambiental em Brasília, pesquisadores acompanham indicadores numéricos associados a áreas verdes da cidade. Cada posição de um vetor A representa uma região monitorada, e o valor A_i representa um índice ecológico inteiro daquela região.

Para estudar padrões locais, os pesquisadores analisam subsegmentos contíguos do vetor. O valor de uma faixa de regiões é definido como o *gcd* (máximo divisor comum) de todos os índices dentro dessa faixa.

Dado um inteiro fixo X, sua tarefa é processar atualizações no vetor e responder consultas sobre quantos subsegmentos contíguos, em uma determinada região do vetor, possuem *gcd* exatamente igual a X.

Mais precisamente, você deve processar Q operações de dois tipos:

- `1 i y`: substitua A_i por y;
- `2 l r`: determine quantos subsegmentos contíguos totalmente contidos em $[l, r]$ possuem *gcd* igual a X.

Em outras palavras, para uma operação do tipo 2, você deve contar o número de pares (i, j) tais que $l \leq i \leq j \leq r$ e $gcd(A_i, A_{i+1}, \ldots, A_j) = X$.

Entrada

A primeira linha contém três inteiros N, Q e X ($1 \leq N, Q \leq 10^5$, $1 \leq X \leq 10^5$), separados por um espaço em branco.

A segunda linha contém N inteiros $A_1, A_2, \ldots, A_N$ ($1 \leq A_i \leq 10^5$), separados por um espaço em branco.

Cada uma das próximas Q linhas descreve uma operação, em um dos seguintes formatos, em que $i \in [1, N]$, $y \in [1, 10^5]$ e $1 \leq l \leq r \leq N$:

- `1 i y`
- `2 l r`

Saída

Para cada operação do tipo `2`, imprima uma linha contendo a resposta para a pergunta do enunciado.

Exemplo

Entrada	Saída
6 7 1 6 15 35 77 143 221 2 1 6 1 3 21 2 2 4 1 6 33 2 4 6 1 1 10 2 1 3	10 1 0 1
6 7 2 4 8 16 32 64 2 2 1 6 1 2 4 2 2 5 1 6 8 2 6 6 1 1 16 2 1 3	6 0 0 0

Problema J – Jacaré F. C.

Limite de tempo: 1s
Limite de memória: 256MB

Autor: Edson Alves

O treinador de goleiros do Brasiliense (conhecido regionalmente também pelo nome do seu mascote, o Jacaré) está participando de um projeto de pesquisa que envolvem as faculdades de Educação Física e de Tecnologia da UnB. Eles desenvolveram um robô que simula cobranças de pênaltis para testar os reflexos dos goleiros. A cada instante, o robô "chuta" uma bola em uma dentre três posicões pré-definidas: centro do gol ('`C`'), lado direito ('`D`') ou lado esquerdo ('`E`').

O goleiro inicia no centro do gol e, a cada instante, ele pode tomar uma dentre três ações:

1. permancecer na posição em que se encontra;
2. se mover uma posição para a esquerda; ou
3. se mover uma posição para a direita.

Por exemplo, se o goleiro estiver à direita e se mover uma posição à esquerda, ele retornará ao centro do gol; por outro lado, se ele já estiver à esquerda e tentar se mover novamente para a esquerda, ele permanecerá onde está (para evitar de deixar a região do gol).

Uma vez que o teste visa primariamente o desenvolvimento do reflexo, o goleiro saberá, de antemão, onde o robô realizará seus N chutes. Assumindo que o goleiro é capaz de defender a cobrança se estiver na posição onde a bola foi chutada, auxilie a equipe do treinador a determinar qual seria o número máximo de defesas que o goleiro pode realizar para cada série de cobranças.

Entrada

A primeira linha da entrada contém um inteiro N ($1 \leq N \leq 2 \times 10^5$), que indica o número de cobranças a serem realizadas pelo robô.

A segunda linha contém uma string composta por N caracteres dentre '`C`', '`D`' e '`E`'.

Saída

Imprima, em uma linha, o número máximo de defesas que o goleiro pode realizar para a série de cobranças dada.

Exemplo

Entrada	Saída
3 CDE	2
1 D	1
10 EDDECEEDDE	6

Notas

No primeiro caso, em uma das soluções válidas, o goleiro inicialmente permanece no centro do gol, defendendo a primeira cobrança. Em seguida, ele se move para a direita, defendendo o segundo chute. Por fim, ele se move uma posição para à esquerda, retornando ao centro do gol, mas falhando em defender a última cobrança.

No segundo caso, o goleiro se move uma posição para a direita e defende a única cobrança.

Problema K – Kubitschek Monumentos

Limite de tempo: 6s
Limite de memória: 512MB

Autor: Ruan Petrus

Os monumentos de Brasília são conectados por pontes suspensas sobre o Lago Paranoá. Cada ponte conecta dois monumentos e possui um comprimento.

Devido ao desgaste natural e às frequentes reformas urbanas, algumas pontes podem desabar, enquanto novas pontes podem ser construídas para ligar diferentes monumentos. Um caminho é uma sequencia de pontes que conecta dois monumentos, seu custo é a soma dos pesos dessas pontes. Como os custos para a construção de pontes são altos, é garantido que em qualquer momento (no instante inicial, após um desabamento e após uma construção), existe no máximo um caminho entre dois monumentos quaisquer.

Para cada conjunto de monumentos conexos (todos os pares de monumentos do conjunto possuem um caminho que os conecta), define-se o **percurso máximo** como o caminho de maior custo entre dois monumentos desse conjunto, sem repetição de pontes.

São dadas Q eventos de um dos seguintes tipos:

- `1 u v w` — uma nova ponte de comprimento w é construída entre os monumentos u e v;
- `2 u v` — a ponte existente entre os monumentos u e v desaba.

O **valor total da cidade** é a soma dos percursos máximos de todos os conjuntos de monumentos conexos. Seu trabalho é determinar o valor total da cidade antes dos eventos e também após cada evento. É garantido que os eventos são consistentes com as definições apresentadas.

Entrada

A primeira linha contém três inteiros N, M e Q ($2 \leq N \leq 10^5$, $0 \leq M \leq N-1$, $1 \leq Q \leq 10^5$), separados por um espaço em branco, representando o número de monumentos, o número de pontes iniciais e o número de consultas, respectivamente.

Cada uma das próximas M linhas contém três inteiros u, v e w ($1 \leq u, v \leq N$, $1 \leq w \leq 10^9$), separados por um espaço em branco, descrevendo uma ponte da configuração inicial.

Cada uma das próximas Q linhas descreve um evento em um dos dois formatos a seguir:

- `1 u v w` ($1 \leq u, v \leq N$, $1 \leq w \leq 10^9$);
- `2 u v` ($1 \leq u, v \leq N$).

Saída

Imprima $Q + 1$ linhas.

A primeira linha deve conter o valor total da cidade na configuração inicial.

Cada uma das próximas Q linhas deve conter o valor total da cidade após o respectivo evento.

Exemplo

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

Problema L – Leis Contraditórias

Limite de tempo: 1s
Limite de memória: 256MB

Autor: Gustavo Leal

Este é um problema interativo!

Em Brasília, arquitetos construíram uma estrutura experimental inspirada nas formas geométricas da teoria modernista de Oscar Niemeyer. A estrutura é composta por módulos dispostos em camadas, formando um grande triângulo, nomeado de Triângulo de Niemeyer.

Cada módulo pode assumir um valor binário: 0 ou 1.

O triângulo possui $N + 2$ níveis:

- o nível 1 contém apenas um módulo (o topo);
- o nível 2 contém dois módulos;
- ...
- o nível $N + 2$ contém $N + 2$ módulos (a base).

Os módulos são numerados sequencialmente por nível, de cima para baixo e da esquerda para a direita:

- nível 1: índice 1;
- nível 2: índices $2, 3$;
- nível 3: índices $4, 5, 6$;
- e assim por diante.

Cada módulo (exceto os da base) possui dois filhos, que são os dois módulos imediatamente abaixo dele.

Os arquitetos definiram algumas leis para que um triângulo pudesse ser denominado de Triângulo de Niemeyer:

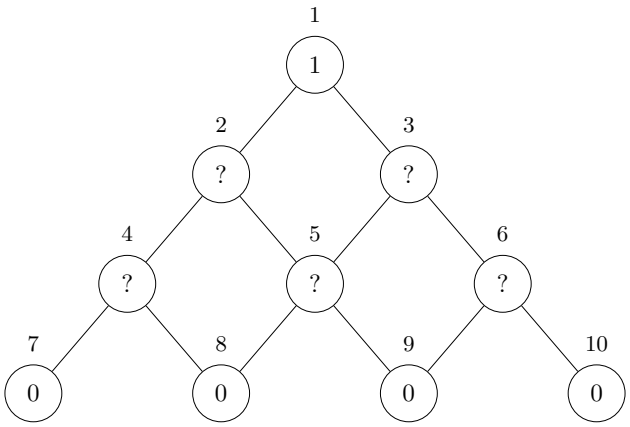
1. o módulo do topo possui valor 1;
2. todos os módulos da base possuem valor 0;
3. se um módulo possui valor 1, então pelo menos um de seus dois filhos também possui valor 1.

Só depois os arquitetos perceberam que a terceira lei gera contradições e nenhum triângulo se enquadra nela (considerando que as duas anteriores são sempre verdadeiras).

Assim, o que começou como uma estrutura arquitetônica se tornou um ponto turístico. Os arquitetos prepararam um jogo para os turistas de Brasília, em que um falso triângulo de Niemeyer é apresentado, com os valores de seus modulos escondidos.

E os turistas devem encontrar um modulo que quebre a terceira regra, perguntando o valor deles. Para deixar o jogo mais interessante, os turistas devem tentar encontrar o tal módulo com poucas perguntas.

Segue abaixo um exemplo de um triângulo de Niemeyer usado no jogo:



Encontre um módulo contraditório usando no máximo $N + 10$ consultas!

Entrada

No início da interação, você recebe um único inteiro N ($1 \leq N \leq 1000$), representando a quantidade de níveis do triângulo (desconsiderando a base e o topo).

Saída

A interação termina quando você imprime uma linha no formato:
`! i`
indicando um módulo contraditório.

Interação

Você pode realizar consultas do tipo:

- `? i` — retorna o valor (0 ou 1) do módulo i.

Quando encontrar um módulo que quebre a terceira lei, imprima a resposta:

- `! i`

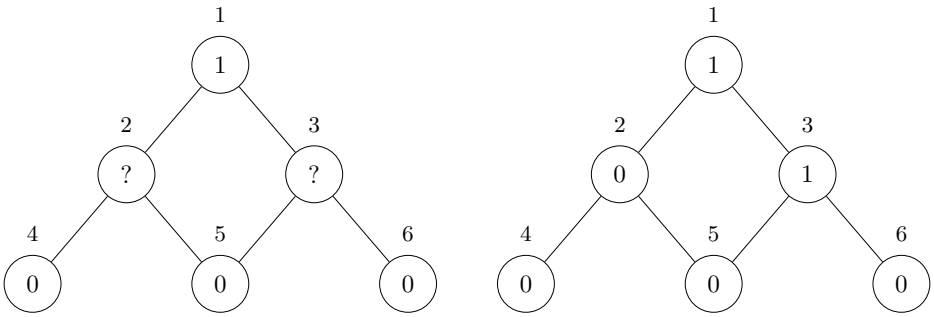
e termine o programa.
Em ambos os casos, i deve ser um índice de um módulo que pertence ao triângulo. Seu programa não pode realizar mais do que $N + 10$ consultas. A resposta não é uma consulta. É garantido que os triângulos da entrada respeitam as regras 1 e 2.

Exemplo

Entrada	Saída
1	
	? 2
0	
	? 3
1	
	! 3

Notas

Segue abaixo o triangulo do caso de exemplo antes e depois das consultas:



O modulo 3 é uma contradição, pois os seus dois filhos possuem o valor 0.

- Lembre-se de limpar o buffer de saída após cada consulta e após a resposta. Em C++, use `cout << endl;` ou `cout.flush();`. Em Python, use `print(..., flush=True)`.
- Se o seu programa fizer uma consulta inválida ou exceder o número máximo de consultas, o veredito será `Wrong Answer`.

Problema M – Metrô Maluco

Limite de tempo: 2s
Limite de memória: 256MB

Autor: Alberto Neto

Você é um dos organizadores da prestigiada II Maratona do Cerrado! Mais especificamente, você está trabalhando em análises para a escolha do hotel do evento e do local da competição. Como o transporte público de Brasília é maravilhoso, os competidores deverão utilizá-lo para se deslocar do hotel para a competição.

Os pontos de parada da cidade são representados por n vértices indexados de 1 até n, onde 1 corresponde ao Plano Piloto. Existem $n-1$ rotas de ônibus disponíveis, de forma que é possível viajar entre qualquer par de pontos de parada, formando uma estrutura de árvore enraizada em 1. O custo de utilizar um ônibus é 1 real, e note que as linhas de ônibus são **bidirecionais**.

Já as linhas do avançadíssimo metrô de Brasília funcionam de maneira pouco convencional. Existe uma estação de metrô em cada ponto de parada u, com destino sendo o ponto de parada diferente de u com maior índice na subárvore[†] de u. O custo de utilizar o metrô também é igual a 1 real, e note que as linhas de metrô são **unidirecionais**.

São analisadas q possibilidades. Na i-ésima possibilidade, é considerado que o hotel da competição está no ponto a_i e o local do evento em b_i, e você deve calcular o menor custo de ir do hotel para a competição.

Ajude a organização para que o evento seja um sucesso!

Entrada

A primeira linha contém um único inteiro n ($2 \leq n \leq 10^5$), indicando o número de pontos de parada.

Cada uma das próximas $n-1$ linhas contém dois inteiros u_i and v_i ($1 \leq u_i, v_i \leq n$, $u_i \neq v_i$), separados por um espaço em branco, indicando uma rota de ônibus entre u_i e v_i. É garantido que as linhas de ônibus formam uma árvore.

A próxima linha contém um único inteiro q ($1 \leq q \leq 10^5$), representando o número de possibilidades a serem analisadas.

Cada uma das próximas q linhas contém dois inteiros a_i e b_i ($1 \leq a_i, b_i \leq n$), separdos por um espaço em branco, referentes aos locais do hotel e competição da i-ésima possibilidade, respectivamente.

Saída

Imprima uma linha com q inteiros, em que o i-ésimo inteiro é o menor custo de ir de a_i para b_i.

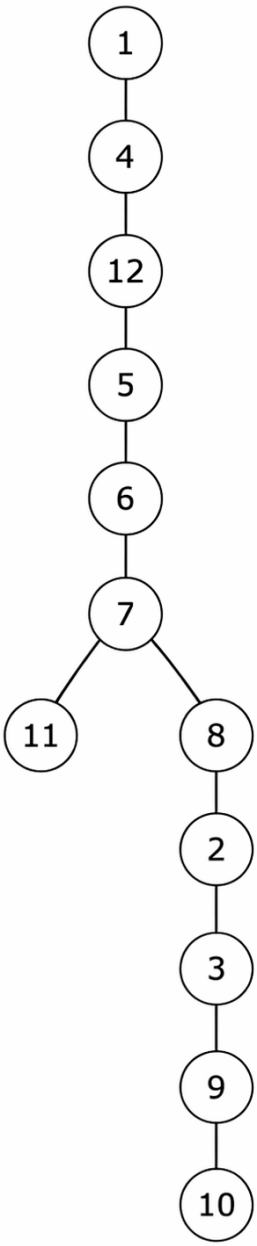
Exemplo

Entrada	Saída
12 1 4 4 12 12 5 5 6 6 7 7 11 7 8 8 2 2 3 3 9 9 10 7 1 11 1 8 1 10 1 9 1 2 12 10 4 7	2 4 5 6 5 4 3
7 1 2 2 3 3 4 4 5 5 6 6 7 4 1 7 1 6 1 5 1 4	1 2 3 3

Notas

[†]Seja T uma árvore enraizada em 1. A subárvore de u é definida como o conjunto de vértices de v tais que o único caminho de 1 até v passa por u.

A árvore (sem as rotas de metrô) do primeiro teste é:


1
4
12
5
6
7
11
8
2
3
9
10

Problema N – Nibas e o contador

Limite de tempo: 1s
Limite de memória: 256MB

Autor: Adne Moretti Moreira

BOCA é um juiz eletrônico amplamente utilizado no Brasil para a realização de competições de programação. Dentre suas funcionalidades, destaca-se um contador de tempo: antes do início da prova, ele exibe uma contagem regressiva até o começo da competição; já após o início, passa a mostrar o tempo restante até o seu término.

Esse contador é essencial para os competidores, pois permite que eles organizem suas estratégias de resolução de problemas de acordo com o tempo disponível. Entretanto, durante a II Maratona do Cerrado, uma instabilidade no sistema fez com que o contador regressivo para o final da prova ficasse indisponível logo após o início da competição.

O professor Nibas, ocupado com outras demandas acadêmicas, pediu a sua ajuda para informar aos participantes quanto tempo ainda resta até o fim da prova.

Dado o horário de finalização F e o horário atual A, determine quanto tempo falta para o término da prova.

Considere que a maratona ocorre inteiramente em um único dia, dentro do intervalo entre 08:00 e 20:00.

Entrada

A primeira linha contém o horário de finalização da prova F, no formato `hh:mm`, tal que ($08 \leq hh \leq 20, 0 \leq mm < 60$).

A segunda linha contém o horário atual A, também no formato `hh:mm`, tal que ($08 \leq hh \leq 20, 0 \leq mm < 60$) e $A \leq F$.

Saída

A saída esperada é um horário, também no formato `hh:mm` indicando o tempo que falta para finalização da prova, `hh` representando as horas restantes e `mm`, os minutos que restam.

Exemplo

Entrada	Saída
20:00 08:00	12:00
08:00 08:00	00:00

Notas

No primeiro caso, o evento terminará às 20 horas, e como o relógio marca, atualmente, 8 horas, restam ainda 12 horas de evento.

No segundo caso, o evento terminou, de modo que não resta mais tempo para os participantes enviarem suas soluções.