Fase Zero da Maratona SBC de Programação 2025

24 de maio de 2025

Caderno de Problemas

Informações gerais

Este caderno contém 13 problemas, as páginas estão numeradas de 1 a 25, contando esta folha de rosto. Verifique se o caderno está completo.

Nome dos programas

- Para soluções em C, C++ e Python, o nome do arquivo-fonte não é significativo, pode ser qualquer nome.
- Se a solução é em Java ou Kotlin, o arquivo-fonte deve ser chamado `A.java` ou `A.kt` respectivamente, onde `A` deve ser substituído pela letra maiúscula que identifica o problema. Lembre-se que em Java, o nome da classe principal deve ser igual ao nome do arquivo.

Entrada

- A entrada deve ser lida da *entrada padrão*.
- A entrada consiste em exatamente um único caso de teste, descrito em uma quantidade de linhas que depende do problema. O formato da entrada é **exatamente** como descrito em cada problema, sem nenhum conteúdo extra.
- Todas as linhas da entrada, incluindo a última, terminam com o caractere de fim de linha (\n).
- A entrada não contém linhas vazias.
- Quando a entrada contém múltiplos valores separados por espaços, existe exatamente um espaço em branco entre dois valores consecutivos na mesma linha.

Saída

- A saída deve ser escrita na *saída padrão*.
- A saída deve respeitar o formato especificado no enunciado sem nenhum dado extra.
- Espaços e quebras de linha são iguais para fins de correção, e espaços extras são ignorados.
- Quando a um valor da saída for um número real, use o número de casas decimais correspondente à precisão requisitada no enunciado.

Regras da competição

- É permitido o uso de até três computadores para acesso ao sistema, um para cada competidor.
- Não é necessário que os competidores estejam no mesmo espaço físico.
- É permitido consultar, e até mesmo copiar, qualquer material (impresso, digital, ou até mesmo online) que já estava disponível antes de a competição começar.
- O uso de ferramentas de IA generativa no geral como ChatGPT, DeepSeek, Gemini, Claude, Copilot, Cursor, entre outros **é proibido**.
- É permitido que os três competidores de um mesmo time comuniquem entre si através de um canal de comunicação restrito apenas ao time.
- Não é permitido que os competidores vazem qualquer informação sobre a prova para qualquer outra pessoa fora do seu próprio time.
- A comunicação entre times é proibida.
- Não é permitida a comunicação entre competidores e técnicos durante a prova. Os técnicos não podem participar da prova, não podem ver os problemas durante a prova, nem mesmo logar na conta especial criada no BOCA para os competidores.
- Toda submissão de código durante a prova deve ser uma tentativa legítima de resolver o problema. Tentativas de minerar os casos de teste e de burlar o sistema serão penalizadas.

Caso seja identificado comportamento que infrinja um dos itens acima ou em caso de denúncias a serem investigadas e comprovadas, os times infratores serão desclassificados.

Informações do ambiente de testes

- O tempo limite pode ser encontrado no enunciado de cada problema.
- O limite de memória para todos os problemas é de 1 GiB.
- A pilha de execução tem um limite de 100 MiB.
- O arquivo-fonte pode ter no máximo 100 KiB.
- A saída pode ter no máximo 1 MiB.

Comandos de compilação utilizados

- C: `gcc -g -O2 -std=gnu11 -static -lm`
- C++20: `g++ -g -O2 -std=gnu++20 -static`
- Java: `javac`
- Kotlin: `kotlinc -J-Xms1024m -J-Xmx1024m -J-Xss100m -include-runtime`

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 utilizar `cin` e `cout`, um jeito de se contornar este problema é executando 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 código-fonte.
- A convenção de nomear o arquivo de acordo com a letra do problema deve ser obedecida. Por exemplo, é necessário que para o problema A o seu arquivo seja nomeado `A.java` com uma classe pública `A`.
- Comando de execução: `java -Xms1024m -Xmx1024m -Xss20m <nomeDoArquivo>`.
- Atenção: não é garantido que soluções em Java executem dentro do tempo limite alocado.

Kotlin

- Não declare "`package`" no seu código-fonte.
- A convenção de nomear o arquivo de acordo com a letra do problema deve ser obedecida. Por exemplo, é necessário que para o problema A o seu arquivo seja nomeado `A.kt` sem declaração da classe principal (resultando numa classe implícita chamada `AKt`).
- Comando de execução: `java -Xms1024m -Xmx1024m -Xss20m <nomeDoArquivo>`.
- Atenção: não é garantido que soluções em Kotlin executem dentro do tempo limite alocado.

Python

- O interpretador Python utilizado é o PyPy que oferece mais velocidade na execução.
- O Python está na versão 3.8.13 e em erro de sintaxe, será retornado Runtime Error.
- Atenção: não é garantido que soluções em Python executem dentro do tempo limite alocado.

Instruções para uso do sistema de submissão BOCA

Tudo no BOCA é feito pela interface web, então antes de fazer qualquer ação, certifique-se que o navegador está aberto na página do juiz BOCA e que você está logado.

Utilizando o sistema

- O BOCA não possui atualizações dinâmicas. Para obter as últimas atualizações, certifique que o conteúdo da página está atualizado (por exemplo usando F5).
- É possível baixar os enunciados da prova e de cada um dos problemas na aba `Problems`.
- Para submeter soluções e ver o resultado das suas submissões, utilize a aba `Runs`. Para realizar uma submissão, escolha o problema apropriado, a linguagem utilizada e envie o arquivo-fonte.
- Para ver esclarecimentos globais e solicitar esclarecimentos sobre o enunciado de um problema, utilize a aba `Clarifications`. Escolha o problema apropriado, escreva sua dúvida e submeta e aguarde a resposta que vai aparecer nesta tela.
- Para ver o placar, utilize a aba `Score`.
- A aba `Tasks` serve para submeter tarefas de impressão e solicitar assistência em sedes presenciais. Em competições remotas, este recurso não é utilizado.
- A aba `Backups` provê um ambiente de compartilhamento de arquivos para o time. Se for necessário trocar de máquina durante a competição, utilize esta área para transferir arquivos para o novo computador.

Resultado da submissão

- O código será julgado por um juiz automático que avaliará o comportamento do programa para um conjunto de casos de teste secretos. Apesar de cada problema conter casos de teste de exemplo, com os quais os competidores podem testar seus códigos localmente, frisa-se que o conjunto de casos de teste secretos do juiz é geralmente muito maior. Note que a sua submissão também poderá ser revisada pelos juízes da competição de forma manual, estendendo o tempo de julgamento.
- Cada submissão pode resultar num dos seguintes vereditos:
  - YES: seu código foi aceito, a saída da sua solução bate com a dos juízes e sua equipe ganhou um balão! Parabéns!
  - NO - Wrong answer: seu código não foi aceito porque o programa não imprimiu a saída esperada para algum caso de teste.
  - NO - Time limit exceeded: seu código não foi aceito porque o programa demorou muito tempo para rodar para algum caso de teste.
  - NO - Runtime error: seu código não foi aceito porque o programa foi abortado pelo sistema operacional por executar alguma operação inválida, como acesso indevido à memória, alocação excessiva de memória ou exceção de ponto flutuante.
  - NO - Compilation error: seu código não foi aceito porque o compilador não conseguiu compilá-lo corretamente. Verifique as opções de compilação utilizadas para a sua linguagem de programação. Note que esse veredito não gera penalidade no placar.
  - NO - Name mismatch: apenas para submissões Java e Kotlin, veredito quando o time submete uma solução com nome da classe principal diferente do especificado.

Problema A

Ambíguo Gato de Schrödinger

Tempo limite: 0,5 s	Limite de memória: 1 GiB

No ano de 1935, o físico Erwin Schrödinger recebeu uma encomenda misteriosa rotulada apenas como "Caixa 42". Dentro dela, um experimento quântico bizarro foi montado por um grupo de cientistas excêntricos. A caixa contém um gato, um frasco de veneno ativado por um átomo radioativo instável e um bilhete ao lado com uma única instrução: "Não abra, a menos que esteja pronto para aceitar uma única verdade."

Segundo a física quântica, enquanto a caixa estiver fechada, não é possível determinar o estado do gato. O sistema inteiro está em uma superposição quântica: o gato está vivo e morto simultaneamente. Porém, se alguém tiver coragem de abrir a caixa, a superposição colapsa, revelando se o gato está de fato vivo ou morto.

Você, como aprendiz do velho Schrödinger, encontrou registros com o estado da caixa e a leitura interna do estado do gato, são informações privilegiadas que não são observáveis a quem não abriu a caixa. Sua missão é determinar o estado observável do gato no momento descrito.

Entrada

A única linha da entrada contém dois inteiros C e G ($0 \leq C, G \leq 1$). $C = 1$ indica que a caixa está fechada, e $C = 0$ indica que está aberta. $G = 1$ indica que o gato está vivo e $G = 0$ indica que o gato está morto.

Saída

Imprima uma única string, "`vivo e morto`" se não é possível saber o estado do gato, "`vivo`" se é possível determinar que o gato está vivo ou "`morto`" se é possível determinar que o gato está morto.

Exemplo de entrada 1	**Exemplo de saída 1**
`1 0`	`vivo e morto`

Exemplo de entrada 2	**Exemplo de saída 2**
`0 1`	`vivo`

Problema B

Busca Periódica

Tempo limite: 0,5 s	Limite de memória: 1 GiB

Foi feita uma análise da evolução dos estados quânticos possíveis de um sistema de partículas, resultando numa árvore enraizada de N estados. Cada estado, com exceção do estado raiz, é conectado a seu estado pai por uma aresta que possui um rótulo de uma letra latina minúscula de `a` a `z`. Esse rótulo descreve o tipo de interferência que levou o sistema a ter um colapso para outro estado. A sequência de interferências de um estado é definida como a concatenação dos rótulos das arestas no caminho da raiz até esse estado.

Para cada estado, definimos a *periodicidade mínima* de sua sequência como o menor inteiro $P \geq 1$ tal que a sequência pode ser obtida repetindo uma sequência menor de tamanho P múltiplas vezes (no mínimo duas vezes). Caso não exista nenhum inteiro $P \geq 1$ válido, a sequência é considerada de periodicidade 0. A sequência vazia da raiz é considerada de periodicidade 0.

Estamos interessados em estudar interferências periódicas dentro do sistema de partículas, então sua tarefa é determinar, dentre todos os estados a partir da raiz, o maior valor de periodicidade mínima das suas respectivas sequências de interferências.

Entrada

A primeira linha contém um inteiro N ($2 \leq N \leq 10^5$), o número de estados. A segunda linha contém $N-1$ inteiros $P_1, P_2, ..., P_{N-1}$ ($1 \leq P_i \leq i$), onde o estado $i+1$ está conectado ao estado P_i. A terceira linha contém uma sequência de $N-1$ letras latinas minúsculas, onde o caractere i representa o rótulo na aresta entre os estados P_i e $i+1$.

Saída

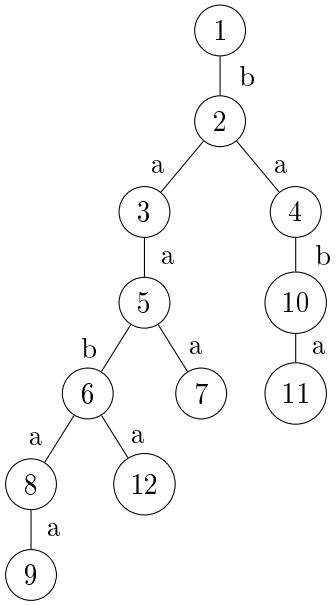
Imprima um único inteiro representando o maior período mínimo entre todas as sequências formadas pelos caminhos da raiz até cada estado.

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

`12`
`1 2 2 3 5 5 6 8 4 10 6`
`baaabaaabaa`
	`3`

Explicação do exemplo 1:

Neste exemplo a árvore descrita é da seguinte forma:



Podemos verificar a periodicidade mínima em algumas sequências de estados da árvore:

- Na sequência que termina no estado 2, formando a palavra `b`, a periodicidade mínima é 0 pois não é possível formar esta palavra com duas ou mais repetições.
- Na sequência que termina no estado 11, formando a palavra `baba`, a periodicidade mínima é 2, pois é possível formar esta palavra repetindo duas vezes a palavra `ba`, que tem 2 caracteres.
- Na sequência que termina no estado 9, formando a palavra `baabaa`, a periodicidade mínima é 3, pois é possível formar esta palavra repetindo duas vezes a palavra `baa`, que tem 3 caracteres.

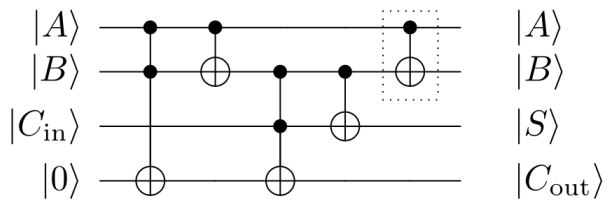
Problema C

Circuitos Lógicos Matriciais

Tempo limite: 0,5 s	Limite de memória: 1 GiB

Na computação quântica, as portas lógicas funcionam de um jeito um pouco diferente. Portas lógicas quânticas são reversíveis e o número de qubits de entrada é igual ao número de qubits de saída. Além disso, elas podem ser representadas por matrizes 2^N por 2^N, onde N é o número de qubits.

Um circuito quântico é um modelo para computação quântica onde a computação é realizada através de uma sequência de portas lógicas quânticas e dispositivos de medição. Uma sequência de portas lógicas pode ser representada por uma matriz resultante da multiplicação das matrizes das portas lógicas em ordem de aplicação, que é a ordem inversa de como elas são representadas graficamente. Por exemplo, o circuito de adição de dois bits em seu forma quântica é:



Temos nesse circuito duas variações de uma porta lógica que chamaremos de $\text{CNOT}(q_c, q_t)$ e $\text{CCNOT}(q_{c_1}, q_{c_2}, q_t)$. No desenho, o qubit q_t aparece marcado com um $\oplus$. A porta lógica $\text{CNOT}(q_c, q_t)$ pode ser vista como sendo igual a $\text{CCNOT}(q_c, q_c, q_t)$, ou seja, a aplicação da porta lógica CCNOT com $q_c = q_{c_1} = q_{c_2}$.

A porta lógica $\text{CCNOT}(q_{c_1}, q_{c_2}, q_t)$ tem o comportamento de inverter o qubit q_t da saída se os qubits de controle q_{c_1} e q_{c_2} ambos estiverem ligados. Matematicamente, $q'_t = q_t \oplus (q_{c_1} \wedge q_{c_2})$. Na sua forma de matriz,

$$\text{CCNOT}(q_{c_1}, q_{c_2}, q_t)_{ij} = \begin{cases} 1 & \text{se } i \text{ tem os bits } c_1 \text{ e } c_2 \text{ ligados e } i \oplus 2^t = j \\ 0 & \text{se } i \text{ tem os bits } c_1 \text{ e } c_2 \text{ ligados e } i \oplus 2^t \neq j \\ 1 & \text{se } i = j \\ 0 & \text{se } i \neq j \end{cases}$$

onde i é a linha e j a coluna com $0 \leq i, j < 2^N$, e i contém o bit k ($0 \leq k$) se $\left\lfloor \frac{x}{2^k} \right\rfloor \bmod 2 = 1$. A operação $\oplus$ é a operação bit a bit de ou exclusivo, comumente o $\wedge$ em linguagens de programação.

Desta forma, a matriz do circuito quântico de adição de dois bits é dada por

$$\text{CNOT}(q_0, q_1)\,\text{CNOT}(q_1, q_2)\,\text{CCNOT}(q_1, q_2, q_3)\,\text{CNOT}(q_0, q_1)\,\text{CCNOT}(q_0, q_1, q_3),$$

onde os qubits q_0, q_1, q_2, q_3 são utilizados com entrada $|A\rangle, |B\rangle, |C_{\text{in}}\rangle, |0\rangle$ respectivamente e resultam em $|A\rangle, |B\rangle, |S\rangle, |C_{\text{out}}\rangle$ respectivamente.

Sua missão é dada a descrição de um circuito com portas lógicas CNOT e CCNOT na ordem de aplicação, imprimir a matriz resultante.

Entrada

A primeira linha da entrada contém os inteiros N ($2 \leq N \leq 8$), o número de qubits do circuito e M ($1 \leq M \leq 10^5$), a quantidade de portas lógicas do circuito.

Seguem M linhas, cada uma com a descrição de uma porta lógica. O primeiro inteiro T ($1 \leq T \leq 2$) define o tipo da porta lógica. Se $T = 1$, a descrição é da porta lógica CNOT(q_C, q_T) e seguem os inteiros distintos C e T ($0 \leq C, T < N$). Se $T = 2$, a descrição é da porta lógica CCNOT(q_{C_1}, q_{C_2}, q_T) e seguem os inteiros distintos C_1, C_2 e T ($0 \leq C_1, C_2, T < N$). Note que as portas lógicas são dadas na ordem de aplicação.

Saída

Imprima 2^N linhas, cada uma com exatamente 2^N caracteres '0' ou '1', correspondendo a matriz do circuito quântico completo.

Exemplo de entrada 1	**Exemplo de saída 1**
2 1 1 0 1	1000 0001 0010 0100

Explicação do exemplo 1:

Este circuito representa apenas a porta lógica CNOT(c, t). Se você já leu sobre essa porta lógica, a matriz parece estar errada pois é diferente da que está na literatura. Porém, é uma questão de convenção. Ao compormos o qubit c com o qubit t, aqui estamos usando a convenção de que o primeiro bit é menos significante que o segundo.

$$\begin{array}{l} i\text{=0 representando } |00\rangle \text{ com } c\text{=0 e } t\text{=0} \\ i\text{=1 representando } |01\rangle \text{ com } c\text{=1 e } t\text{=0} \\ i\text{=2 representando } |10\rangle \text{ com } c\text{=0 e } t\text{=1} \\ i\text{=3 representando } |11\rangle \text{ com } c\text{=1 e } t\text{=1} \end{array} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

Então se entrada for $|00\rangle$ ou $|10\rangle$, ambos onde $c = 0$ e t varia, a porta lógica não atua. Quando a entrada é $|01\rangle$ ou $|11\rangle$, então a porta lógica atua e inverte o valor de t. Essa convenção é utilizada por exemplo pela biblioteca Qiskit.

Exemplo de entrada 2	**Exemplo de saída 2**
4 5 1 0 1 1 1 2 2 1 2 3 1 0 1 2 0 1 3	1000000000000000 0000000000000100 0000000000000010 0000000000010000 0000100000000000 0100000000000000 0010000000000000 0000000000000001 0000000010000000 0000010000000000 0000001000000000 0001000000000000 0000000000001000 0000000001000000 0000000000100000 0000000100000000

Exemplo de entrada 3	**Exemplo de saída 3**
3 1 2 0 1 2	10000000 01000000 00100000 00000001 00001000 00000100 00000010 00010000

Exemplo de entrada 4	**Exemplo de saída 4**
3 1 1 0 1	10000000 00010000 00100000 01000000 00001000 00000001 00000010 00000100

Problema D

Decoerência Quântica

Tempo limite: 0,5 s	Limite de memória: 1 GiB

A SBC (Sociedade Brasileira de Computação) está desenvolvendo diversos modelos de arquiteturas para computadores quânticos, com o objetivo de torná-los acessíveis a todas as pessoas no futuro. Um dos principais desafios enfrentados pelas equipes de desenvolvimento é a decoerência quântica, que ocorre quando um qubit em superposição (representando simultaneamente os estados 0 e 1) colapsa para 0 ou 1 devido à interferência do ambiente.

Para cada modelo desenvolvido, será analisada a taxa de decoerência quântica. Para isso, os qubits serão observados em estado isolado e sob condições normais de temperatura e pressão. A taxa de decoerência quântica é a razão entre a quantidade de qubits que colapsaram em condições normais de temperatura e pressão e a quantidade de qubits que estavam em superposição no estado isolado.

Como existem vários modelos, foi solicitado a você o desenvolvimento de um programa que calcule essa taxa. Afinal, você está precisando de horas complementares para se formar, não é mesmo?!

Entrada

A primeira linha contém um inteiro N ($10 \leq N \leq 10^5$) indicando o número de qubits do computador. As duas próximas linhas contêm as strings S em estado isolado e T sob condições normais de temperatura e pressão, respectivamente, de tamanho N, compostas pelos caracteres $\{0, 1, *\}$, onde '*' indica um qubit em superposição.

É garantido que pelo menos um qubit esteja em superposição em estado isolado e que todo qubit que não está em superposição na string S permanece idêntico na string T.

Saída

A saída deve conter a taxa de decoerência quântica em forma decimal, com exatamente duas casas decimais.

Exemplo de entrada 1	**Exemplo de saída 1**
10 0*1**100*1 0110*100*1	0.50

Exemplo de entrada 2	**Exemplo de saída 2**
13 *1*01*100*01* 01*0101001011	0.80

Exemplo de entrada 3	**Exemplo de saída 3**
25 *10*1*110*01*011100*110*0 *1011*110001*011100*110*0	0.29

Problema E

Energização de Partículas

Tempo limite: 0,5 s	Limite de memória: 1 GiB

Há uma partícula no ponto $X = 1$ de uma reta numérica infinita com um valor de carga Y. Ao interagir com a reta, ela ganha propriedades incomuns: ao absorver energia essa partícula libera energia cinética suficiente para avançar $\gcd(X, Y)$ passos na reta numérica, em que $\gcd(X, Y)$ é o máximo divisor comum de X e Y. Ou seja, a cada procedimento, a partícula sai da posição X para a posição $X + \gcd(X, Y)$.

Cientistas precisam energizar a particula K vezes a fim de descobrir propriedades novas à respeito dessa, no entanto, eles precisam prever em qual ponto a partícula irá parar após esses procedimentos para reutilizá-la em estudos futuros.

Portanto, ajude a determinar qual a posição X final que a partícula irá parar após os K processos.

Entrada

A entrada consiste de uma linha com dois números Y ($1 \leq Y \leq 10^9$) e K ($1 \leq K \leq 10^9$).

Saída

Imprima um inteiro contendo a posição X que a partícula irá parar seguindo os procedimentos acima.

Exemplo de entrada 1	**Exemplo de saída 1**
`4 3`	`8`

Exemplo de entrada 2	**Exemplo de saída 2**
`7 15`	`70`

Exemplo de entrada 3	**Exemplo de saída 3**
`123 123`	`10086`

Problema F

Feynman Decorando Números

Tempo limite: 2 s	Limite de memória: 1 GiB

Richard Feynman foi o primeiro a propor a utilização de um fenômeno quântico para executar rotinas computacionais. Foi numa palestra apresentada na Primeira Conferência de Computação Física no MIT. Ele mostrou que um computador clássico levaria muito tempo para simular um simples experimento de física quântica. Reza a lenda que ele conseguia memorizar grandes sequências de números e calcular mentalmente várias propriedades a uma velocidade super-rápida.

Os caçadores de mitos, ao saber disso, resolveram verificar essa lenda diretamente com Feynman usando a sua máquina do tempo. Para verificar, eles iriam gerar uma sequência de números e perguntar de quantas formas podemos escolher exatamente 4 elementos dessa sequência que somem X. E a criação do teste foi designada para você, novo estagiário dos caçadores de mitos.

Sua tarefa é escrever um programa que, dado um conjunto de números e múltiplos valores de consulta, determine quantas quádruplas $\{i, j, k, l\}$ com $1 \leq i < j < k < l \leq N$ possuem soma $A_i + A_j + A_k + A_l$ igual aos valores consultados.

Entrada

A entrada consiste em um único caso de teste. A primeira linha contém um inteiro N ($4 \leq N \leq 1000$), representando a quantidade de números da sequência. A segunda linha contém N inteiros a_i ($0 \leq |a_i| \leq 1000$), separados por espaço. A terceira linha contém um inteiro Q ($1 \leq Q \leq 4000$), representando o número de consultas. Por fim, cada uma das próximas Q linhas contém um inteiro q_i ($0 \leq |q_i| \leq 4000$) cada, representando os valores-alvo consultados.

Saída

Para cada consulta, seu programa deve imprimir uma linha contendo um único inteiro: a quantidade de quádruplas cuja soma é igual a q_i.

Exemplo de entrada 1	**Exemplo de saída 1**
8 -1 23 4 -8 4 23 4 5 1 30	6

Explicação do exemplo 1:

As seis quádruplas que somam 30 são:

- $A_1, A_2, A_3, A_7 \rightarrow -1 + 23 + 4 + 4 = 30$
- $A_1, A_3, A_5, A_6 \rightarrow -1 + 4 + 4 + 23 = 30$
- $A_1, A_5, A_6, A_7 \rightarrow -1 + 4 + 23 + 4 = 30$
- $A_1, A_2, A_5, A_7 \rightarrow -1 + 23 + 4 + 4 = 30$
- $A_1, A_3, A_6, A_7 \rightarrow -1 + 4 + 23 + 4 = 30$
- $A_1, A_2, A_3, A_5 \rightarrow -1 + 23 + 4 + 4 = 30$

Problema G

Grover e Seus Caminhos Especiais

Tempo limite: 1 s	Limite de memória: 1 GiB

Grover é um cientista da computação indo-americano que quer propor um algoritmo de busca quântica que oferece uma aceleração quadrática em relação aos algoritmos de busca clássicos para bancos de dados não estruturados. Para isso, ele precisa resolver um problema em árvores que faz parte de sua pesquisa para desenvolver o seu algoritmo.

Grover tem uma árvore com N vértices e $N-1$ arestas não direcionadas, e tem como objetivo atribuir para cada vértice um valor v_i satisfazendo algumas restrições:

- $1 \leq v_i \leq 5$
- Existem exatamente cnt_x vértices com o valor $v_i = x$, para $1 \leq x \leq 5$
- Para cada vértice i existe um conjunto de valores que o valor de v_i pode assumir
- Além disso, nessa árvore existem P caminhos especiais para Grover, um caminho especial é representado por um par de vértices (X_i, Y_i) (os caminhos podem ter intersecção e um caminho pode aparecer mais de uma vez na entrada) e para esses caminhos os valores atribuídos aos vértices no caminho (na ordem de X_i para Y_i) têm que formar uma sequência estritamente crescente.

Se existir uma solução válida, imprima qualquer uma que satisfaça as restrições impostas por Grover, caso contrário, imprima -1.

Entrada

A primeira linha da entrada vai conter um inteiro N ($1 \leq N \leq 5 \cdot 10^4$), a quantidade de vértices da árvore. A segunda linha da entrada vai conter 5 números inteiros, os valores de $cnt_1, cnt_2, cnt_3, cnt_4, cnt_5$ respectivamente. É garantido que a soma $cnt_1 + cnt_2 + cnt_3 + cnt_4 + cnt_5 = N$.

As próximas N linhas da entrada começam com um número inteiro M_i ($1 \leq M_i \leq 5$), seguidos de M_i inteiros distintos entre 1 e 5, indicando os valores que o i-ésimo vértice pode assumir. Seguem $N-1$ linhas, cada uma com dois números U, V ($1 \leq U, V \leq N$) indicando uma aresta da árvore.

Em seguida, haverá uma linha com um único número inteiro P ($1 \leq P \leq 5$), indicando a quantidade de caminhos especiais. Por fim, as próximas P linhas da entrada vão conter dois números X_i, Y_i ($1 \leq X_i, Y_i \leq N$), indicando os caminhos especiais.

Saída

Caso exista solução imprima N inteiros v_1, v_2, ..., v_n indicando uma solução válida para o problema. Caso contrário, imprima -1. Se existirem múltiplas respostas válidas, qualquer uma será aceita.

Exemplo de entrada 1	**Exemplo de saída 1**
10 1 1 4 2 2 5 1 2 3 4 5 5 1 2 3 4 5 5 1 2 3 4 5 5 1 2 3 4 5 5 1 2 3 4 5 5 1 2 3 4 5 5 1 2 3 4 5 5 1 2 3 4 5 5 1 2 3 4 5 5 1 2 3 4 5 5 6 5 4 5 3 5 8 3 10 10 9 9 1 3 7 10 2 5 7 1 3 10 10 2 5 6 7 6	5 4 2 3 3 5 1 3 4 3

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

Exemplo de entrada 3	**Exemplo de saída 3**
1 0 0 0 0 1 4 1 2 3 4 1 1 1	-1

Exemplo de entrada 4	**Exemplo de saída 4**
10 2 1 4 1 2 5 1 2 3 4 5 4 1 2 3 5 1 3 3 2 4 5 2 4 5 1 3 5 1 2 3 4 5 3 1 2 3 1 4 2 1 5 8 5 5 2 8 9 9 10 8 4 4 1 1 6 9 3 4 7 4 7 10 1 4 6 6 7 10	1 3 3 2 5 3 1 3 4 5

Problema H

Harmonia Palíndromica Binária

Tempo limite: 0,5 s	Limite de memória: 1 GiB

Várias alternativas quânticas a algoritmos tradicionais vem sendo pesquisadas para resolver os mais diversos problemas. Usando portas lógicas quânticas, é possível ganhar desempenho em alguns tipos de tarefas específicas, levando a ganhos por exemplo em algoritmos de fatorização de números e busca em conjuntos de dados.

Shor está trabalhando em um algoritmo quântico para reconhecimento de palíndromos em sequências binárias. Para tanto, ele precisa gerar uma grande quantidade de exemplos, e decidiu que ele precisaria saber para um determinado inteiro positivo X, qual seria o maior inteiro Y menor ou igual a X cuja representação em binário fosse palíndroma.

Shor tem muita experiência com algoritmos quânticos, mas está com dificuldade em fazer um algoritmo clássico que resolva esse problema, você pode ajudar?

Entrada

A única linha de entrada contém o inteiro X ($1 \leq X \leq 10^{18}$).

Saída

Seu programa deve produzir uma única linha com o inteiro Y correspondente.

Exemplo de entrada 1	**Exemplo de saída 1**
`9945`	`9945`

Exemplo de entrada 2	**Exemplo de saída 2**
`11`	`9`

Exemplo de entrada 3	**Exemplo de saída 3**
`154`	`153`

Problema I

Inspecionando o Emaranhamento

Tempo limite: 1 s	Limite de memória: 1 GiB

No laboratório do Instituto de Física Quântica, uma equipe de cientistas está conduzindo um novo experimento. A ideia é reconstruir a evolução de um conjunto de partículas emaranhadas ao longo do tempo. O problema é que o experimento não pode ser observado diretamente.

Por isso o laboratório possui uma rede de N sensores quânticos distribuídos no espaço. Os sensores detectam várias propriedades das partículas (*spin*, polarização, momento angular etc.). No entanto por razões técnicas e energéticas, um mesmo sensor não pode ser ativado por um tempo muito curto (para evitar ruído) e nem permanecer ativo por muito tempo (para evitar o superaquecimento).

O experimento vai durar T segundos. Para que não seja gasta energia à toa e que toda a informação seja coletada, em cada um dos segundos exatamente um dos sensores deve estar ligado coletando informação e nenhum sensor deve permanecer ligado após o termino do experimento.

Cada sensor i no tempo j fornece uma pontuação de confiabilidade $c(i, j)$, refletindo a qualidade da medição naquele instante. O desafio da equipe é selecionar quais sensores ativar ao longo do experimento, garantindo que após um sensor ser acionado ele deve ficar ativo por pelo menos L segundos e no máximo U segundos. Além disso um sensor pode ser utilizado diversas vezes durante o experimento, desde que o tempo para permanecer ativo seja respeitado.

A confiabilidade final do experimento é igual a soma da confiabilidade dos sensores escolhidos a cada segundo durante o teste. O objetivo da equipe é escolher os sensores de forma a maximizar a confiabilidade da reconstrução do estado quântico. Como os físicos estão muito ocupados estudando outros assuntos quânticos, eles enviaram este problema para os times da Fase Zero da Maratona SBC de Programação resolver.

Entrada

A primeira linha contém dois inteiros N ($1 \leq N \leq 5 \cdot 10^3$) e T ($1 \leq T \leq 100$) representando respectivamente a quantidade de sensores e o tempo do experimento.

A seguir N linhas, cada uma contendo T inteiros sendo que na i-ésima linha o j-ésimo inteiro será $c(i, j)$ ($1 \leq c(i, j) \leq 5 \cdot 10^7$).

A última linha contém dois inteiros L e U ($1 \leq L \leq U \leq T$) representando o mínimo e o máximo de tempo que cada sensor pode ficar ativo.

Saída

Seu programa deve produzir uma única linha com um inteiro representando a maior confiabilidade possível. Caso não tenho solução para as restrições dadas, imprima -1.

Exemplo de entrada 1	**Exemplo de saída 1**
3 5 2 3 2 1 2 1 1 5 1 2 1 2 2 1 5 1 5	16

Exemplo de entrada 2	**Exemplo de saída 2**
3 5 2 3 2 1 2 1 1 5 1 2 1 2 2 1 5 2 3	13

Exemplo de entrada 3	**Exemplo de saída 3**
3 5 2 3 2 1 2 1 1 5 1 2 1 2 2 1 5 2 2	-1

Problema J

Jornada das Partículas

Tempo limite: 1 s	Limite de memória: 1 GiB

Um experimento no Grande Colisor de Hádrons está sendo montado para testar propriedades de fase de várias partículas diferentes. São N filtros colocados em um círculo em que a i-ésimo filtro deixa passar partículas com fase menor ou igual a seu limite A_i.

Um experimento se inicia com a liberação de uma partícula na posição do i-ésimo filtro, com fase inicializada em A_i. Ao passar pelo filtro i, incluindo o primeiro, a partícula:

- É filtrada se a fase da partícula for maior que o limite A_i finalizando o experimento.
- Caso contrário, a partícula passa pelo filtro, aumenta a sua fase em K, e vai para o próximo filtro à direita, o $(i \mod N) + 1$-ésimo filtro.

Queremos fazer uma simulação do resultado desse experimento para poder comparar com os resultados verificados empiricamente. Com esse objetivo, precisamos criar um vetor B, de tamanho N, em que B_i $(1 \leq B_i \leq N)$ é a posição do filtro que parou a partícula que iniciou na posição i.

Entrada

A primeira linha contém dois inteiros N $(1 \leq N \leq 2 \cdot 10^5)$ e K $(1 \leq K \leq 10^9)$.

A segunda linha contém N inteiros $A_1, A_2, \ldots, A_N$ $(-10^9 \leq A_i \leq 10^9)$.

Saída

Imprima uma linha contendo N inteiros $B_1, B_2, \ldots, B_N$.

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

Exemplo de entrada 2	**Exemplo de saída 2**
5 5 -10 -5 0 5 10	1 1 1 1 1

Exemplo de entrada 3	**Exemplo de saída 3**
6 2 12 4 5 9 2 3	2 3 5 5 6 2

Problema K

K Elementos Perdidos

Tempo limite: 2 s	Limite de memória: 1 GiB

No núcleo da Estação Q-42, os cientistas criaram o mais avançado simulador de estados quânticos combinatoriais: o Entrelaçador de Caminhos Aumentados, também conhecido como ECA. O ECA foi programado para estudar a caminhada quântica em espaços de permutação. Na caminhada quântica, o caminhante evolui em uma superposição de posições, seguindo as regras da mecânica quântica.

A entrada do simulador é uma permutação A de tamanho N. Cada índice i da permutação tem um inteiro A_i associado, representando um nó em um espaço de Hilbert. Definimos uma caminhada quântica que segue o critério crescente de evolução como uma subsequência de índices $i_1 < i_2 \ldots < i_M$ $(M > 0)$ em que $A_{i_1} < A_{i_2} \ldots < A_{i_M}$. Cada nó i também tem uma carga de energia quântica armazenada na posição B_i do vetor B.

Ao simular todas as possíveis caminhadas quânticas que seguem o critério de crescimento de evolução, o ECA colapsava o estado de cada caminho em uma soma das energias quânticas dos nós visitados. Cada uma dessas somas era registrada em um vetor C, representando todas as amplitudes de caminhos válidos colapsados em energia clássica.

Para organizar os dados, o vetor C foi ordenado em ordem não crescente, mas algo inesperado aconteceu: uma decoerência fez com que o vetor C se perdesse no meio da simulação. Seu objetivo como analista da Estação Q-42 é reconstruir os K maiores valores do vetor C, ou reportar caso C não tenha aquele elemento.

Entrada

A primeira linha contém dois inteiros N $(1 \leq N \leq 10^4)$ e K $(1 \leq K \leq 10^5)$.

A segunda linha contém N inteiros $A_1, A_2, \ldots, A_N$ $(1 \leq A_i \leq N)$, é garantido que A é uma permutação.

A terceira linha contém N inteiros $B_1, B_2, \ldots, B_N$ $(1 \leq B_i \leq 10^4)$.

Saída

Imprima uma linha contendo K inteiros C_i $(1 \leq i \leq K)$. Caso um determinado C_i não exista imprima -1 em seu lugar.

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

Explicação do exemplo 1:

As subsequências crescentes de A são:

- $\{A_1\} = \{2\} \rightarrow B_1 = 1$
- $\{A_2\} = \{1\} \rightarrow B_2 = 2$
- $\{A_3\} = \{3\} \rightarrow B_3 = 1$
- $\{A_1, A_3\} = \{2, 3\} \rightarrow B_1 + B_3 = 2$
- $\{A_2, A_3\} = \{1, 3\} \rightarrow B_2 + B_3 = 3$

Logo, o vetor C após a ordenação equivale a $\{3, 2, 2, 1, 1\}$. Como $K = 4$, somente é necessário imprimir $\{C_1, \ldots, C_4\}$.

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

`4 16`
`1 2 3 4`
`1 1 1 1`
	`4 3 3 3 3 2 2 2 2 2 2 1 1 1 1 -1`

Explicação do exemplo 2:

Todas as $2^4 - 1 = 15$ sequências de A são crescentes. Como $K = 16$, então $C_{16} = -1$.

Problema L

Linha de Produção de qPhones

Tempo limite: 0,5 s	Limite de memória: 1 GiB

A Sociedade Brasileira de Celulares (SBC) está desenvolvendo um novo modelo de smartphones que utiliza computação quântica, os qPhones. Diferente dos aparelhos tradicionais, que armazenam bits, essa nova arquitetura utilizará qubits.

Um qubit (quantum bit) é a unidade básica da informação quântica, assim como o bit na computação clássica. Porém, enquanto que um bit clássico assume apenas um estado por vez (0 ou 1), um qubit pode existir em superposição, assumindo múltiplos estados simultaneamente como uma "mistura quântica" de 0 e 1, cada um com uma probabilidade de ser medida ao ser observado.

Deste modo, caso um dispositivo consiga armazenar qubits, todas as suas combinações podem ser representadas simultaneamente, por conta da superposição. Por exemplo, se um dispositivo armazena 3 qubits, podemos ter a representação de $2^3 = 8$ estados clássicos superpostos, sendo eles 000, 001, 010, 011, 100, 101, 110 e 111.

Na prática, podemos assumir que para simular um 1 qubit precisamos de 2 bits, para simular 2 qubits precisamos de 4 bits, para simular 3 qubits precisamos de 8 bits, e assim por diante. Assim, para simular completamente a memória de um celular clássico com M megabytes (MB), os engenheiros da SBC precisam garantir que os qubits do novo aparelho sejam capazes de representar ao menos os M megabytes. Considere que 1 MB é equivalente a 10^6 bytes.

Você foi recentemente contratado pela SBC para ajudar na linha de produção dos novos smartphones quânticos. Sua tarefa é, dado o valor de memória de M megabytes, determinar o número mínimo de qubits necessários para simular todos os estados possíveis de um dispositivo clássico com essa quantidade de memória.

Entrada

A entrada consiste de uma única linha que contém um inteiro M ($1 \leq M \leq 10^{10}$) representando a quantidade de memória em MB de um dispositivo tradicional.

Saída

Seu programa deve produzir uma única linha com um inteiro representando o número mínimo de qubits necessários para simular todos os estados possíveis de um dispositivo clássico com M megabytes de memória.

Exemplo de entrada 1	**Exemplo de saída 1**
1	23

Exemplo de entrada 2	**Exemplo de saída 2**
17	28

Problema M

Movimentação Assustadora à Distância

Tempo limite: 1 s	Limite de memória: 1 GiB

Charles é um grande físico, criptógrafo e cientista da computação, sendo responsável por grandes contribuições que incluem trabalhos fundamentais sobre a relação entre física e informação. Durante seus estudos sobre teletransporte quântico, Charles descobriu um campo quântico com N posições numeradas de 1 até N. Em um experimento, Charles pode colocar uma partícula inicialmente em qualquer posição do campo quântico. Em cada instante de tempo, a partícula pode decidir se teletransportar para uma posição maior do que a sua posição atual ou permanecer parada e finalizar o trajeto. Desta forma, existem $2^N - 1$ trajetos possíveis.

Seja um trajeto uma sequência de posições visitadas por uma partícula em um experimento. Cada posição i ($1 \leq i \leq N$) do campo quântico possui um coeficiente A_i associado. Charles define a beleza de um trajeto sendo igual ao máximo divisor comum de todos os coeficientes de posições visitada no trajeto.

Charles irá fazer várias operações em sequência, são elas:

- Operação `1 X`: Considere que todos os trajetos possíveis têm a mesma probabilidade de serem feitos. Qual a probabilidade do trajeto feito ter beleza igual a X?
- Operação `2 I X`: Atualize o valor do coeficiente A_i para ser X.

Você pode ajudar Charles com os seus experimentos?

Entrada

A primeira linha da entrada contém o inteiro N ($1 \leq N \leq 10^5$), a quantidade de posições do campo quântico. Seguem na segunda linha N inteiros positivos $A_1, A_2, \ldots, A_N$ ($1 \leq A_i \leq 10^5$).

A terceira linha da entrada contém o inteiro Q ($1 \leq Q \leq 10^5$), a quantidade de consultas. As próximas Q linhas vão conter operações. Cada linha é uma operação que é identificada pelo inteiro T ($1 \leq T \leq 2$). Operações com $T = 1$ são seguidas de um inteiro X e operações do tipo 2 são seguidas dos inteiros I ($1 \leq I \leq N$) e X. Em ambas a operações, $1 \leq X \leq 10^5$.

Saída

Para cada experimento feito por Charles, imprima a probabilidade $\frac{P}{Q}$ do experimento ter beleza igual a X na forma $P \cdot Q^{-1} \pmod{998244353}$. É garantido que P é um inteiro não negativo e Q é um inteiro positivo, e que Q^{-1} com a propriedade $QQ^{-1} = 1 \pmod{998244353}$ existe.

Exemplo de entrada 1	**Exemplo de saída 1**
4 1 2 4 8 6 1 1 1 2 1 4 1 8 1 3 1 5	931694730 465847365 732045859 865145106 0 0

Explicação do exemplo 1:

Os 15 possíveis trajetos que podem ser realizados em um experimento são:

- $\{a_1\} = \{1\} \rightarrow \gcd(1) = 1$
- $\{a_2\} = \{2\} \rightarrow \gcd(2) = 2$
- $\{a_3\} = \{4\} \rightarrow \gcd(4) = 4$
- $\{a_4\} = \{8\} \rightarrow \gcd(8) = 8$
- $\{a_1, a_2\} = \{1, 2\} \rightarrow \gcd(1, 2) = 1$
- $\{a_1, a_3\} = \{1, 4\} \rightarrow \gcd(1, 4) = 1$
- $\{a_1, a_4\} = \{1, 8\} \rightarrow \gcd(1, 8) = 1$
- $\{a_2, a_3\} = \{2, 4\} \rightarrow \gcd(2, 4) = 2$
- $\{a_2, a_4\} = \{2, 8\} \rightarrow \gcd(2, 8) = 2$
- $\{a_3, a_4\} = \{4, 8\} \rightarrow \gcd(4, 8) = 4$
- $\{a_1, a_2, a_3\} = \{1, 2, 4\} \rightarrow \gcd(1, 2, 4) = 1$
- $\{a_1, a_2, a_4\} = \{1, 2, 8\} \rightarrow \gcd(1, 2, 8) = 1$
- $\{a_1, a_3, a_4\} = \{1, 4, 8\} \rightarrow \gcd(1, 4, 8) = 1$
- $\{a_2, a_3, a_4\} = \{2, 4, 8\} \rightarrow \gcd(2, 4, 8) = 2$
- $\{a_1, a_2, a_3, a_4\} = \{1, 2, 4, 8\} \rightarrow \gcd(1, 2, 4, 8) = 1$

onde $\gcd(X)$ é o maior divisor comum do conjunto X.
Assim:

- A probabilidade da beleza de um experimento ser igual a 1 é $\frac{8}{15}$.
- A probabilidade da beleza de um experimento ser igual a 2 é $\frac{4}{15}$.
- A probabilidade da beleza de um experimento ser igual a 4 é $\frac{2}{15}$.
- A probabilidade da beleza de um experimento ser igual a 8 é $\frac{1}{15}$.
- A probabilidade da beleza de um experimento ser igual a 3 é $\frac{0}{15}$.
- A probabilidade da beleza de um experimento ser igual a 5 é $\frac{0}{15}$.

Exemplo de entrada 2	**Exemplo de saída 2**
3 18 29 15 5 1 12 1 25 1 28 2 1 25 1 5	0 0 0 855638017

Explicação do exemplo 2:

Após a operação `2 1 25` a sequência de coeficientes passa a ser $(25, 29, 15)$. Antes da operação do tipo 2, a sequência de coeficientes era $(18, 29, 15)$.