Willy e os Números Primos

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Durante o século XXI, apesar de terem existidos inúmeros matemáticos e inúmeros cientistas, o matemático Willy foi, sem dúvida, a mente mais importante do século para o avanço das tecnologias. Suas contribuições para sociedade se estendem a inúmeros ramos da ciência: na Matemática, na Física, na Química, na Informática, entre outras áreas.

Em um determinado momento de sua vida, Willy percebeu que estava se tornando uma pessoa sedentária pois tinha investido muito tempo em suas pesquisas e em seus cálculos. Então ele decidiu que, no começo de cada manhã, ele iria fazer uma corrida por sua cidade. A cidade de Willy pode ser vista como n estações de metrô e n - 1 ruas ligando-as. É garantido que, movendo-se apenas pelas ruas da cidade, é possível ir de qualquer estação de metro para qualquer outra. Para cada rua, devido a sua facilidade com cálculos, ele consegue estimar quanto tempo ele demora para percorre-la.

Inicialmente, Willy vai caminhando de sua casa até alguma estação de metrô. Esse percurso inicial não conta em seu tempo de corrida pois ele está fazendo apenas um aquecimento. Depois, ele começa a correr entre as estações de metrô. Devido a sua obsessão por números primos, ele só vai percorrer ruas tal que o tempo que ele demore para atravessa-la seja um número primo. Se não houver nenhuma rua com essa característica, Willy irá se negar a ir correr e irá continuar sedentário. Entretanto, se houver, ele quer percorrer o caminho entre duas estações que maximize a quantidade de tempo que ele fica correndo sem visitar duas vezes a mesma rua. Quando ele chegar na estação de metrô final, ele voltará caminhando para sua casa e, de modo análogo a parte inicial de seu treinamento, esse tempo não será contabilizado em seu tempo total de corrida.

Apesar dele ter contribuído com inúmeros algoritmos na informática, ele tem muita preguiça de escrever códigos. Portanto, ele pediu a sua ajuda para descobrir o caminho que ele deve percorrer. Willy irá te fornecer a configuração de sua cidade e a quantidade de tempo que ele irá gastar para percorrer cada rua. Ajude ele a descobrir o tempo máximo que ele pode ficar correndo por sua cidade.

Input

A primeira linha contém um número inteiro n ($1 \leq n \leq 10^5$), indicando o número de estações de metrô na cidade de Willy.

As seguintes n - 1 linhas contêm 3 números inteiros U_i, V_i, W_i ($1 \leq U_i, V_i \leq n$, $U_i \neq V_i$, $2 \leq W_i \leq 10^5$), indicando que há uma rua entre as estações de metrô U_i e V_i e Willy gastará W_i de tempo se ele atravessá-la.

Output

A saída deve conter um número inteiro: o tempo máximo que ele pode gastar correndo pela cidade. Se não houver rua em que Willy possa correr, imprima 0.

Examples

standard input	standard output
6 1 3 4 4 6 2 1 5 5 2 5 2 1 6 3	12
4 2 1 4 3 2 6 4 2 9	0

Note

No primeiro teste, o caminho ideal para Willy é começar na estação 4 e terminar na estação 2. O tempo desse caminho é $2 + 3 + 5 + 2 = 12$ e todos os horários são primos.

No segundo teste, não há ruas com tempos que são números primo. Então, a resposta é 0.

Barcos

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

Por causa de várias guerras e crises ocorridas no mundo no século XX, a cidade da Nlogonia sempre foi uma cidade que atraía muitos imigrantes. Seus habitantes eram pacíficos e nenhuma crise atingia-os. Navios de todas as partes do mundo traziam imigrantes para a cidade. Para isso, a cidade teve que planejar novos projetos de movimentação de barcos em sua baía. O projeto utilizado naquela época possui algumas regras:

- Cada porto pode comportar, no máximo, um navio
- Não podem haver dois navios não atracados simultaneamente na baía porque há um grande risco deles colidirem.
- Existem alguns pontos de risco na baía que os navios não podem passar por eles

De um modo simplificado, a baía da Nlogonia pode ser vista como um tabuleiro $n \times m$ onde cada célula possui valor 0 ou 1. Seja (i, j) a célula na i^{o}-ésima linha e na j^{o}-ésima coluna e $v_{i,j}$ seu valor. $v_{i,j}$ é igual a 1 se essa célula for um ponto de risco que os navios não podem passar, caso contrário, os návios podem trafegar por essa célula. Quando um barco está em uma célula, gastando 1 unidade de tempo, ele pode ir para todas as células que não são um ponto de risco e compartilham um lado com sua célula atual.

Durante o período em questão, existiam x navios querendo atracar nos y portos da Nlogonia. É garantido que existe pelo menos um porto para cada navio. Todos os navios aparecem na baía na linha 1 e todos os portos estão localizados na linha n. Os navios são numerados de 1 a x e vão entrando na baía por ordem crescente de seus números e, assim que o i^{o}-ésimo navio $(1 \leq i < x)$atracar em um porto, o $(i+1)^{o}$-ésimo navio irá entrar, no mesmo momento, na baía da cidade, até todos os navios estarem em algum porto. O navio i $(1 \leq i \leq x)$ irá entrar na baía na célula $(1, a_i)$ e o porto j $(1 \leq j \leq y)$ está localizado na célula (n, b_j). Sua missão é descobrir qual é a menor quantidade de tempo necessária para todos os navios atracarem em algum porto.

Input

A primeira linha contém 4 números inteiros n, m, x, y $(2 \leq n, m \leq 300, 1 \leq x \leq y \leq m)$, indicando o número de linhas e colunas do tabuleiro, o número de barcos e o número de portos.

As seguintes n linhas contêm m números inteiros $v_{i,j}$ $(v_{i,j} \in \{0, 1\})$, indicando o valor da célula (i, j).

As próximas x linhas contêm um número inteiro a_i $(1 \leq a_i \leq m)$, indicando a coluna em que o i-ésimo barco aparecerá na baía. É garantido que, para todos os i, j tais que $1 \leq i, j \leq x$ e $i \neq j$, $a_i \neq a_j$.

As seguintes linhas y contêm um número inteiro b_j $(1 \leq b_j \leq m)$, indicando a coluna em que o j-ésimo porto está localizado. É garantido que, para todos os i, j tais que $1 \leq i, j \leq y$ e $i \neq j$, $b_i \neq b_j$.

Output

A saída deve ter um número inteiro: a quantidade mínima de tempo necessária para todos os barcos atracarem em qualquer porto.

Examples

standard input	standard output
5 5 2 3 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 1 3 1 4 5	13
5 5 2 2 0 0 0 0 0 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 2 4 1 4	13
4 4 1 1 0 0 0 0 1 1 0 0 0 0 1 1 0 0 0 0 3 1	-1

Note

Uma resposta possível para o primeiro teste é o primeiro barco ancorar no último porto, gastando 8 unidades de tempo e o segundo barco ancorar no segundo porto, gastando mais 5 unidades de tempo, totalizando 13 unidades de tempo.

Árvores e Mandatos

Input file:	`standard input`
Output file:	`standard output`
Time limit:	5 seconds
Memory limit:	256 megabytes

Frederick é um senhor que adora cuidar de suas árvores. Ele possui n árvores dispostas em uma linha em seu jardim. Ele ficou muito contente quando percebeu que todas elas tinham atingido a marca de 10^{18} metros de altura!

Entretanto, devido a altura colossal de suas árvores, essas árvores podem cair e causar um grande estrago. Por isso, a prefeitura da cidade teve que intervir. Durante m dias, a prefeitura irá mandar um decreto para Frederick. Durante o dia i, o decreto da prefeitura terá 3 números: L_i, R_i e X_i. Isso significa que, no i^{o}-ésimo dia, ele terá que cortar alguns metros de algumas árvores localizadas no intervalo de L_i até R_i inclusive. A quantidade total de metros retirados das árvores no i^{o}-ésimo dia tem que ser exatamente X_i.

Como ele preza muito por suas árvores, ele te deu uma missão. Dados todos os m decretos da prefeitura, qual é a quantidade máxima de metros que ele terá que tirar de alguma das árvores se ele as cortar de modo ótima, ou seja, visando minimizar a quantidade máxima que ele teve que tirar de uma árvore após os m dias.

Input

A primeira linha contém dois números inteiros n, m ($1 \leq n, m \leq 10^5$), indicando o número de árvores no jardim de Frederick e o número de dias em que a prefeitura enviará um decreto a Frederick.

As seguintes linhas m contêm 3 números inteiros L_i, R_i, X_i ($1 \leq L_i \leq R_i \leq n$, $1 \leq X_i \leq 10^6$), indicando o decreto da prefeitura.

Output

A saída deve conter um número inteiro: número máximo de metros que Frederick terá que remover de alguma uma das árvores se ele as cortar de modo ótimo.

Examples

standard input	standard output
3 2 1 2 2 2 3 2	2
4 3 2 4 1 2 3 2 1 2 5	3

Note

No primeiro teste, uma possível sequência ótima de cortes seria remover 1 metro da primeira e da segunda árvore no primeiro dia e remover 1 metro da segunda e terceira árvore no segundo dia. No final, ele cortou 1 metro da primeira árvore, 2 metros da segunda árvore e 1 metro da terceira árvore. Portanto, o número máximo de metros que ele removeu de uma árvore foi de 2 metros.

Os 100

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Durante uma guerra nuclear na Terra, foram mandadas 12 estações espaciais para a órbita da Terra com algumas pessoas cada uma. Para uma maior chance de sobrevivência, essas 12 estações decidiram se unificar em apenas uma. Essa nova estação ficou conhecida como Arca.

Depois de um certo tempo, eles decidiram mandar 100 pessoas que viviam na Arca para a superfície terrestre para averiguar se a Terra ainda possuía condições razoáveis para ser habitada novamente. Esse grupo de pessoas estava sob o comando de uma mulher chamada Clarke.

Chegando na superfície terrestre, a partir de algumas missões de buscas, eles descobriram que existiam grupos de pessoas sobreviventes em n pontos da superfície (apenas nessa questão, suponha que a Terra pode ser vista como um plano). Entretanto, Clarke não confia tanto nessas equipes de busca. Com algumas peças da nave, Clarke conseguiu criar um drone que conseguia tirar fotos de áreas retangulares. Os lados dessas fotos são paralelos aos eixos. Com isso, ela podia tirar algumas fotos e verificar a veracidade das informações sobre os sobreviventes. Ela consegue verificar se realmente existe um grupo de sobreviventes em um dado ponto se, e somente se, ele for visível em, pelo menos, uma foto. Entretanto, devido a falha de algumas peças, seu drone apresentou um problema: quando ele tira foto de uma área retangular, apenas as bordas dessa foto são vísiveis. Seu interior apresenta vários borrões e manchas, impossibilitando a visibilidade dessa área interior. Para seu drone tirar uma foto, ele gasta uma unidade de energia. Durante sua movimentação, nenhuma unidade de energia é consumida. Como os recursos disponíveis são limitados, Clarke quer gastar a menor quantidade de energia possível para verificar se realmente existem grupos de terráqueos sobreviventes nos n pontos fornecidos pelas equipes de busca. Ajude Clarke a descobrir qual é essa menor quantidade de energia.

Input

A primeira linha contém um número inteiro n ($1 \leq n \leq 2000$), indicando o número de possíveis grupos de sobreviventes.

As n linhas a seguir contêm 2 números inteiros x_i, y_i ($10^{-9} \leq x_i, y_i \leq 10^9$), indicando a coordenada (x_i, y_i) de um possível grupo de terráqueos sobreviventes. É possível que haja vários grupos na mesma coordenada.

Output

A saída deve conter um número: a quantidade mínima de energia que Clarke precisa gastar para verificar todos os n pontos.

Examples

standard input	standard output
5 2 3 4 1 4 5 8 3 4 3	2
8 1 -1 2 -2 2 2 -1 1 -1 -1 -2 2 1 1 -2 -2	2

Note

Na primeira amostra, Clarke pode tirar uma foto com as seguintes áreas retangulares: um retângulo com ponta inferior esquerda igual a (2, 1) e ponta superior direita igual (8, 5), cobrindo todos os pontos, exceto o quinto ponto e outro retângulo com ponta inferior esquerda igual a (4, 0) e ponta superior direita igual a (10, 3), cobrindo o segundo, o quarto e o quinto ponto. Com essas fotos, todos os pontos foram visíveis em pelo menos uma foto.

Torneio de Programação

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

O prefeito da cidade de Byteland realiza um torneio de programação anualmente. Em 2020, a competição conta com n participantes. Para mudar um pouco a dinâmica do torneio, o prefeito decidiu mudar as regras dessa edição do torneio. Essas regras afetaram principalmente o modo de combate entre os partipantes: cada par de participantes irá se enfrentar uma única vez. Cada combate será realizado em uma certa linguagem de programação. Quando dois participantes se enfrentam utilizando uma certa linguagem, a pessoa com maior habilidade naquela linguagem vence a partida. Não haverá empates nesses combates pois cada participante possui uma habilidade única em cada linguagem.

Antes do início do torneio, o prefeito pediu para que seu assistente anotasse os resultados de cada batalha, a linguagem de programação em que ela foi realizada e que ele contasse a quantidade de linguagens utilizadas no torneio. Entretanto, ao fim do evento, o prefeito percebeu que seu assistente havia anotado apenas os resultados de cada confronto. Por isso, o prefeito te contratou para descobrir qual é a menor quantidade de linguagens de programação possível utilizadas no torneio de forma que seja possível chegar nos resultados dados considerando os jogadores com uma certa habilidade em cada linguagem e também exigiu que você desse uma possível configuração do torneio, ou seja, para cada combate, um número representando a linguagem de programação usada nele.

Input

A primeira linha contém um número inteiro n ($1 \leq n \leq 2000$), indicando o número de participantes no torneio.

As n linhas a seguir contêm n números inteiros $a_{i,j}$ ($a_{i,j} \in \{0, 1\}$), indicando o resultado do combate dos participantes i e j. $a_{i,j}$ será igual a 1 se, e somente se, o participante i venceu esse combate. É garantido que $a_{i,j} = 1\text{-}a_{j,i}$ para todos i e j tais que $1 \leq i, j \leq n$ e $i \neq j$. Também é garantido que $a_{i,i} = 0$ para todos os $1 \leq i \leq n$.

Output

A primeira linha deve conter o menor número de linguagens de programação usadas no torneio. As linguagens são numeradas de 1 ao menor número de linguagens de programação.

Depois disso, a saída deve ter mais n - 1 linhas. A i-ésima linha deve ter n - i inteiros. O inteiro j-ésimo inteiro na i-ésima linha deve ser o número da linguagem de programação usada no combate entre o participante i e o participante $i + j$.

Examples

standard input	standard output
3 0 1 0 0 0 1 1 0 0	2 1 2 1
3 0 1 1 0 0 1 0 0 0	1 1 1 1

Via Láctea

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

Jonn é um alienígena que vive em um planeta muito distante da Terra. Seu planeta pode ser visto como um tabuleiro $n \times m$, onde cada célula desse tabuleiro é a casa de um alienígena. Esse planeta de Jonn apresenta um formato muito peculiar: ele é um toróide, ou seja, além das linhas i e $i + 1$ serem adjacentes para $1 \leq i < n$ e das colunas j e $j + 1$ serem adjacentes para $1 \leq j < m$, as linhas 1 e n são adjacentes e as colunas 1 e m são adjacentes.

Jonn é o único entregador de leite desse planeta. Portanto, todas as manhãs, ele sai de sua casa, visita todas as outras casas e volta para sua. A casa de Jonn está na célula (x, y). Apesar de Jonn gostar de sua profissão, ele quer voltar o mais rápido possível para sua casa pois ele gosta muito de assistir seriados. Consequentemente, ele nunca visita a mesma casa mais de uma vez na mesma manhã. Quando Jonn está em uma casa, ele só pode visitar casas vizinhas a ela. Duas casas são consideradas vizinhas se elas estão na mesma linha e em colunas adjacentes ou na mesma coluna e em linhas adjacentes.

Ajude Jonn a construir algum caminho válido para sua entrega, ou seja, um caminho que começe em sua casa, passe por todas as outras casas do planeta e volte para sua casa e as casas adjacentes nesse caminho sejam vizinhas.

Input

A única linha da entrada contém quatro números inteiros n, m, x, y ($2 \leq n, m \leq 2000$, $1 \leq x \leq n$, $1 \leq y \leq m$), indicando o número de linhas e colunas do tabuleiro, e as coordenadas da casa de Jonn.

Output

Sua saída deve ter $n \cdot m + 1$ linhas, cada uma contendo a casa que Jonn deve visitar em ordem. A primeira e a última casa devem ser a casa dos Jonn. Se houver mais de uma resposta, qualquer uma será aceita. (Cuidado! A saída para esse problema é grande.)

Examples

standard input	standard output
2 2 1 1	1 1 2 1 2 2 1 2 1 1
2 3 1 2	1 2 2 2 2 3 2 1 1 1 1 3 1 2

Torneio de Programação novamente

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

O prefeito de Byteland, dado o desastre que foi a edição de 2020 por causa dos erros de seu assistente, decidiu voltar as regras originais do torneio de programação. Segundo essas regras, inicialmente, os participantes são dispostos em uma linha, sendo essa ordem decidida pelo prefeito, e os combates devem ser realizados da seguinte maneira: na primeira rodada, o primeiro participante vai batalhar contra o segundo participante, o terceiro contra o quarto, e assim por diante. Na segunda rodada, o vencedor da primeira batalha da primeira rodada joga contra o vencedor da segunda batalha da primeira rodada, o vencedor da terceira contra o vencedor da quarta, e assim por diante. Esse padrão se repete até restar apenas um jogador.

Com esse estilo de batalhas, os participantes ficam ansiosos para saber quando vão batalhar contra algum oponente e ficam fazendo essas perguntas para o prefeito. As vezes, o prefeito também decide a ordem de dois participantes na configuração atual. O prefeito te contratou novamente para ajuda-lo a lidar com essas perguntas e com essas mudanças.

Input

A primeira linha contém 2 números inteiros n, q ($1 \leq n, q \leq 10^5$), indicando o número de participantes do torneio e o número de consultas. É garantido que n é uma potência de dois.

A próxima linha contém n números inteiros a_1, a_2, ..., a_n indicando a disposição inicial dos participantes. É garantido que a sequência a_1, a_2, ..., a_n é uma permutação dos n participantes.

Cada uma das seguintes linhas q descreve as perguntas e as mudanças e tem o seguinte formato:

- 1 x y - Troque a posição dos participantes x e y ($1 \leq x, y \leq n$, $x \neq y$);
- 2 x y - Responda a rodada que os participantes x e y irão lutar entre si se vencerem todas as batalhas até esta rodada ($1 \leq x, y \leq n$, $x \neq y$).

Output

Para cada consulta do tipo 2, imprima a rodada em que os participantes x e y se enfrentarão se vencerem todas as batalhas até essa rodada.

Examples

standard input	standard output
8 5 3 7 5 1 8 4 6 2 2 3 5 2 1 8 2 6 2 2 1 5 2 8 4	2 3 1 1 1
4 5 2 1 3 4 2 1 3 1 1 4 2 1 3 2 2 4 2 1 4	2 1 1 2

Max e Min

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	256 megabytes

É dada uma sequência a de n inteiros positivos. Uma divisão dessa sequência é uma partição dessa sequência em exatamente k blocos contínuos disjuntos dessa sequência. O valor de uma divisão é a soma dos custos dos k intervalos dessa divisão, sendo que o custo de um intervalo é dado pela diferença entre o maior e o menor valor contidos nesse intervalo. Sua tarefa é achar o maior valor possível de uma divisão dessa sequência.

Input

A primeira linha contém dois números inteiros n, k ($1 \leq n \leq 20000$, $1 \leq k \leq 100$, $k \leq n$), indicando o tamanho da sequência a e o número de blocos que ela precisa ser dividida.

A segunda linha contém n números inteiros a_1, a_2, ..., a_n ($1 \leq a_i \leq 10^6$, para todo i tal que $1 \leq i \leq n$), indicando os números da sequência a.

Output

A saída deve ter um número inteiro: o maior valor possível de uma divisão dessa sequência.

Examples

standard input	standard output
4 1 1 4 3 8	7
4 2 1 4 3 8	8

Hospital

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

O Hospital de Seattle Grace é um hospital com um intenso fluxo de pacientes todos os dias. Em razão disso, foram instalados n elevadores nesse hospital. Coincidentemente, ele também possui n andares. Apesar da quantidade de elevadores, a Doutora Bailey ainda fica preocupado com a velocidade de locomoção entre diferentes alas e andares do hospital, dado o grande tamanho da edificação. Por isso, ele deu chamou sua interna Cristina para tentar resolver esse problema.

Inicialmente, o i^{o}-ésimo elevador está no andar h_i. Bailey quer que Cristina mude alguns elevadores de andares de tal modo que, em cada andar, tenha exatamente um elevador. Apesar desse hospital ser moderno, seus elevadores são antigos e só possuem dois botões: um botão para subir um andar e outro para descer um andar. Cristina pediu sua ajuda para determinar qual é a menor quantidade de vezes que ela tem que apertar algum botão para cumprir a tarefa que lhe foi dada.

Considere que, para sua locomoção entre os andares, Cristina utiliza as escadas do hospital, não influenciando, portanto, a configuração dos elevadores durante suas trocas de andares e/ou elevadores.

Input

A primeira linha contém um número inteiro n ($1 \leq n \leq 10^5$), indicando o número de elevadores no hospital.

A segunda linha contém n números inteiros $h_1, h_2, \ldots, h_n$ ($1 \leq h_i \leq n$, para todo o i tal que $1 \leq i \leq n$), indicando o andar inicial de cada elevador.

Output

A saída deve conter apenas um número: o número mínimo de vezes que Cristina precisa pressionar um botão para concluir sua tarefa.

Examples

standard input	standard output
4 3 4 4 3	4
3 1 1 1	3

Note

No primeiro teste, uma resposta possível é mover o primeiro elevador para o primeiro andar, mover o segundo elevador para o terceiro andar e mover o quarto elevador para o segundo andar. O número total de movimentos será $2 + 1 + 0 + 1 = 4$.

No segundo teste, uma resposta possível para essa configuração é mover o segundo elevador para o segundo andar e o terceiro elevador para o terceiro andar. O número total de jogadas será $0 + 1 + 2 = 3$.

Experimentos com luz

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	256 megabytes

Um dos estudos de Sir Isaac Maliki, no século XVII, era sobre a luz. Em seu experimento, ele tinha uma faixa de n segmentos coloridos. Cada um desses segmentos possuia, inicialmente, cor c_i. Feito alguns testes, ele percebeu algo muito interessante: quando ele incidia sobre essa faixa uma luz de cor c, todos os segmentos que possuiam cor c refletiam essa cor e os segmentos que não possuíam essa cor ficavam pretos. Depois dessa descoberta, ele começou a brincar com esse fato. Ele fez q operações nessa faixa: uma das operações era trocar a cor de um segmento da fita e a outra operação era iluminar a fita com uma luz de uma certa cor e contar quantos intervalos contínuos dela não eram pretos. Ele anotou todas essas operações em um papel, juntamente com sua observação.

Já no século XXI, alguns pesquisadores de uma universidade acharam esse papel de Maliki. Entretanto, havia uma parte do papel que estava desgastada e ninguém conseguia decrifrá-la: as respostas do segundo tipo de operação. A tarefa de descobrir essas respostas foi passada para o instituto de computação dessa universidade e você, por sorte, faz parte desse instituto. Ajude-os escrevendo um programa que simule o experimento de Maliki, supondo que o fato descoberto por ele seja verdadeiro e suportando as consultas e as mudanças feitas.

Input

A primeira linha contém dois números inteiros n, q ($1 \leq n, q \leq 10^6$), indicando o tamanho da faixa e o número de operações realizadas pelo Maliki.

A segunda linha contém n números inteiros c_i ($1 \leq c_i \leq n$), indicando a cor inicial do i-ésimo segmento da faixa.

Cada uma das seguintes q linhas descreve as perguntas e as alterações e tem o seguinte formato:

- 1 x c - Altere a cor do segmento x da faixa para a cor c ($1 \leq x, c \leq n$). Pode ser que o segmento x já possua cor c, então a configuração da fita não será alterada.
- 2 c - Conte quantos intervalos contínuos da faixa atual não serão pretos se a iluminarmos com uma luz de cor c ($1 \leq c \leq n$).

Output

Para cada consulta do tipo 2, imprima o número de intervalos contínuos da faixa atual que não é preto, quando iluminados com uma luz de cor c.

Example

standard input	standard output
5 5 1 1 3 2 1 2 1 2 3 1 3 1 2 1 2 3	2 1 2 0

Você consegue responder a essas perguntas X?

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

São dados n vetores de tamanho 2^i $(1 \leq i \leq 16)$ e q consultas. Há 4 tipos de consultas:

- Tipo 1: x, y, v - Somar v no elemento de índice y do bloco x;
- Tipo 2: x - A resposta para essa consulta é o intervalo de soma máxima do bloco x;
- Tipo 3: x, y - Colar blocos x e y, ambos de mesmo tamanho. O vetor x ficará na metade da esquerda e o vetor y ficará na metade direita do novo vetor. Os vetores x e y serão **destruídos**;
- Tipo 4: x - Partir ao meio o bloco x, criando dois novos vetores e **destruindo** o vetor x.

Em uma consulta do tipo 3, o novo vetor recebe o índice do menor inteiro positivo que ainda não foi usado. Da mesma forma, em uma consulta do tipo 4, ao partir ao meio um vetor, o vetor da metade da esquerda receberá o menor inteiro positivo não usado e o vetor da metade da direita receberá o segundo menor inteiro não usado.

Input

A primeira linha contém 2 números inteiros n, q $(1 \leq n, q \leq 10^5)$, indicando o número de vetores e o número de consultas.

Cada uma das n linhas a seguir começa com um número inteiro s_i $(1 \leq s_i \leq 2^{16})$, indicando que o tamanho do i-ésimo vetor é s_i, seguido por s_i inteiros. O j-ésimo número $v_{i,j}$ $(-10^6 \leq v_{i,j} \leq 10^6)$, indicando j-ésimo número do i-ésimo vetor. É garantido que $\sum_{i=1}^{N} s_i \leq 10^5$.

As seguintes q linhas descrevem uma consulta e têm o seguinte formato:

- 1 x y v - Adicione v $(-10^6 \leq v \leq 10^6)$ no elemento do índice y $(1 \leq y \leq s_x)$ do vetor x;
- 2 x - A resposta a esta consulta é a soma do intervalo máximo de soma no vetor x;
- 3 x y - Cole os vetores x e y $(x \neq y)$, ambos com o mesmo tamanho. O vetor x estará na metade esquerda e o vetor y estará na metade direita do novo vetor. Os vetores x e y serão destruídos após esta consulta;
- 4 x - Quebre o vetor x no meio, criando dois novos vetores e destruindo o vetor x. Nesta consulta, é garantido que $s_x \geq 2$.

.

Em todas as consultas, é garantido que o vetor usada nessa consulta exista neste momento.

Output

Para cada consulta do tipo 2, imprima a soma do intervalo máximo de soma no vetor x.

Example

standard input	standard output
3 6 2 1 -3 1 -2 4 1 2 5 -4 2 3 4 3 3 5 4 2 6 2 1 2 2	8 5 1 -2