Competidor(a): _____

Número de inscrição: _____–_____ *(opcional)*

Este Caderno de Tarefas não pode ser levado para casa após a prova. Após a prova entregue este Caderno de Tarefas para seu professor guardar. Os professores poderão devolver os Cadernos de Tarefas aos competidores após o término do período de aplicação das provas (14 e 15 de agosto de 2025).

Olimpíada Brasileira de Informática

OBI2025

Caderno de Tarefas

Modalidade Programação • Nível 2 • Fase 2

14 e 15 de agosto de 2025

A PROVA TEM DURAÇÃO DE 3 horas

Promoção:

Sociedade Brasileira de Computação

Apoio:

Coordenação:

v1.0

Instruções

LEIA ATENTAMENTE ESTAS INSTRUÇÕES ANTES DE INICIAR A PROVA

- Este caderno de tarefas é composto por 10 páginas (não contando a folha de rosto), numeradas de 1 a 10. Verifique se o caderno está completo.
- A prova deve ser feita individualmente.
- É proibido consultar a Internet, livros, anotações ou qualquer outro material durante a prova. É permitida a consulta ao *help* do ambiente de programação se este estiver disponível.
- As tarefas têm o mesmo valor na correção.
- A correção é automatizada, portanto siga atentamente as exigências da tarefa quanto ao formato da entrada e saída de seu programa; em particular, seu programa não deve escrever frases como "Digite o dado de entrada:" ou similares.
- Não implemente nenhum recurso gráfico nas suas soluções (janelas, menus, etc.), nem utilize qualquer rotina para limpar a tela ou posicionar o cursor.
- As tarefas **não** estão necessariamente ordenadas, neste caderno, por ordem de dificuldade; procure resolver primeiro as questões mais fáceis.
- Preste muita atenção no nome dos arquivos fonte indicados nas tarefas. Soluções na linguagem C devem ser arquivos com sufixo *.c*; soluções na linguagem C++ devem ser arquivos com sufixo *.cc* ou *.cpp*; soluções na linguagem Java devem ser arquivos com sufixo *.java* e a classe principal deve ter o mesmo nome do arquivo fonte; soluções na linguagem Python 3 devem ser arquivos com sufixo *.py*; e soluções na linguagem Javascript devem ter arquivos com sufixo *.js*.
- Na linguagem Java, **não** use o comando *package*, e note que o nome de sua classe principal deve usar somente letras minúsculas (o mesmo nome do arquivo indicado nas tarefas).
- Você pode submeter até 50 soluções para cada tarefa. A pontuação total de cada tarefa é a melhor pontuação entre todas as submissões. Se a tarefa tem sub-tarefas, para cada sub-tarefa é considerada a melhor pontuação entre todas as submissões.
- Não utilize arquivos para entrada ou saída. Todos os dados devem ser lidos da entrada padrão (normalmente é o teclado) e escritos na saída padrão (normalmente é a tela). Utilize as funções padrão para entrada e saída de dados:
  - em C: *scanf, getchar, printf, putchar*;
  - em C++: as mesmas de C ou os objetos *cout* e *cin*.
  - em Java: qualquer classe ou função padrão, como por exemplo *Scanner, BufferedReader, BufferedWriter* e *System.out.println*
  - em Python: *read,readline,readlines,input,print,write*
  - em Javascript: *scanf, printf*
- Procure resolver a tarefa de maneira eficiente. Na correção, eficiência também será levada em conta. As soluções serão testadas com outras entradas além das apresentadas como exemplo nas tarefas.

Placar do Jogo

Nome do arquivo: `placar.c`, `placar.cpp`, `placar.pas`, `placar.java`, `placar.js` *ou* `placar.py`

Durante uma viagem para a Bolívia, Paulo e Camila encontraram um jogo de futebol de mesa. Como bons brasileiros, os dois já haviam jogado esse jogo no Brasil, e pretendiam simplesmente continuar o passeio. No entanto, um terrível impasse surgiu: Paulo dizia que o jogo se chamava Pebolim, mas Camila insistia que o nome era Totó! Diante desse importante dilema e sem a possibilidade de entendimento mútuo, a disputa foi inevitável.

A partida foi longa e intensa, de forma que as crianças perderam as contas de quantos gols fizeram. Prevendo que isso aconteceria, eles pediram a Afonso – um amigo que chamava o jogo de Pacau e, portanto, era um juiz imparcial – que marcasse o placar. Ao fim do jogo, Afonso havia anotado não só as quantidades P e C de gols que Paulo e Camila fizeram, respectivamente, como também os momentos (em minutos) em que cada gol foi marcado. Vale ressaltar que os gols ocorreram em minutos **distintos** e foram anotados em **ordem cronológica**.

Após o fim da partida, Paulo e Camila nem lembravam o porquê da disputa, mas estavam curiosos a respeito da evolução do placar ao longo do jogo. Por exemplo, suponha que Paulo fez dois gols, nos minutos 24 e 38, enquanto Camila fez três gols, nos minutos 1, 21 e 63. Os placares que existiram durante a partida são:

- 0×0: início da partida.
- 0×1: Camila marca um gol no minuto 1.
- 0×2: Camila marca um gol no minuto 21.
- 1×2: Paulo marca um gol no minuto 24.
- 2×2: Paulo marca um gol no minuto 38.
- 2×3: Camila marca um gol no minuto 63.

Perceba que o placar sempre é mostrado como a quantidade de gols de Paulo, seguida da quantidade de gols de Camila, nesta ordem.

Sua tarefa é: dadas as anotações de Afonso, ou seja, as quantidades de gols e os momentos em que os gols foram marcados para cada criança, determine a lista completa e cronologicamente ordenada de todos os placares que existiram durante a partida.

Entrada

A primeira linha da entrada contém um inteiro P, a quantidade de gols marcados por Paulo, seguido de P valores, $p_1,\ p_2,\ \ldots,\ a_P$, os momentos (em minutos) dos gols de Paulo, em ordem cronológica.

A segunda linha contém um inteiro C, a quantidade de gols marcados por Camila, seguido de C valores, $c_1,\ c_2,\ \ldots,\ c_C$, os momentos (em minutos) dos gols de Camila, em ordem cronológica.

Saída

Seu programa deverá imprimir $P + C + 1$ linhas, uma para cada placar que existiu em algum momento da partida, em ordem cronológica. Para cada placar, imprima dois números inteiros, a quantidade de gols de Paulo e a quantidade de gols de Camila, nesta ordem.

Restrições

É garantido que todo caso de teste satisfaz as restrições abaixo.

- $0 \leq P,\ C \leq 50$
- $1 \leq p_1 < p_2 < \ldots < p_P \leq 100$
- $1 \leq c_1 < c_2 < \ldots < c_C \leq 100$
- $p_i \neq c_j$ para todo $1 \leq i \leq P$ e $1 \leq j \leq C$ (ou seja, os minutos em que houve um gol são todos distintos)

Informações sobre a pontuação

A tarefa vale 100 pontos. Estes pontos estão distribuídos em subtarefas, cada uma com suas **restrições adicionais** às definidas acima.

- **Subtarefa 1 (0 pontos):** Esta subtarefa é composta apenas pelos exemplos mostrados abaixo. Ela não vale pontos, serve apenas para que você verifique se o seu programa imprime o resultado correto para os exemplos.
- **Subtarefa 2 (18 pontos):** $P = 1$ e $C = 1$.
- **Subtarefa 3 (16 pontos):** $P = 0$.
- **Subtarefa 4 (24 pontos):** Houve um gol a cada minuto da partida. Ou seja, houve um gol no minuto 1, um gol no minuto 2, e assim por diante até o minuto $P + C$.
- **Subtarefa 5 (42 pontos):** Sem restrições adicionais.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
2 24 38 3 1 21 63	0 0 0 1 0 2 1 2 2 2 2 3

Explicação do exemplo 1: Este é o exemplo mostrado no enunciado. A primeira linha indica que Paulo fez dois gols, nos minutos 24 e 38. A segunda linha indica que Camila fez três gols, nos minutos 1, 21 e 63. A saída lista os placares que existiram durante o jogo, **no formato especificado no enunciado**.

Exemplo de entrada 2	**Exemplo de saída 2**
0 2 60 70	0 0 0 1 0 2

Explicação do exemplo 2: Neste caso Paulo não marcou gols, enquanto Camila marcou 2 gols, um no minuto 60 e o outro no minuto 70.

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

Explicação do exemplo 3: Neste exemplo nenhum gol foi marcado, por isso o único placar durante a partida foi 0×0.

Mania de Ímpar

Nome do arquivo: `mania.c`, `mania.cpp`, `mania.pas`, `mania.java`, `mania.js` *ou* `mania.py`

Bel é uma garota muito estudiosa e inteligente. No entanto, como muitas pessoas geniais, ela tem algumas manias peculiares, sendo que a principal delas é fazer as coisas em quantidades ímpares. Isso geralmente não atrapalha a sua vida, mas às vezes cria situações interessantes. Por exemplo, Bel visitou sua tia e, como de costume, levou em sua mochila um tênis e uma meia extras, bateu na porta três vezes e tomou cinco copos de água.

Nesse dia, elas decidiram que fariam cookies. Quando a menina chegou, a massa já havia sido preparada e disposta em N linhas e M colunas em uma bandeja. A fim de atender à mania da sobrinha, a tia de Bel havia posicionado os biscoitos de forma que N **e** M **são ímpares**, mas não teve tempo de colocar as quantidades corretas de gotas de chocolate em cada cookie. Dessa forma, o biscoito na linha i e coluna j possui $G_{i,j}$ gotas de chocolate.

Bel decidiu modificar os cookies para estarem de acordo com sua mania. Ela considera que uma bandeja está *organizada* se, para todo par de cookies **adjacentes**, a soma das quantidades de gotas de chocolate nos dois cookies é **ímpar**. Um cookie é adjacente a outro se está imediatamente à esquerda, à direita, acima ou abaixo dele. Bel pretende adicionar gotas de chocolate em alguns cookies para deixar a bandeja organizada. Porém, para economizar os ingredientes da tia, ela deseja fazer isso adicionando o **mínimo** de gotas possível.

Sua tarefa é: dadas os valores N e M, bem como as quantidades $G_{i,j}$ de gotas de chocolate em cada cookie, determine a quantidade mínima de gotas que precisam ser adicionadas para que a bandeja esteja organizada, isto é, para que a soma das quantidades de gotas em dois cookies adjacentes seja sempre ímpar. Além disso, você deve descrever a configuração final da bandeja organizada, isto é, indicar a quantidade de gotas de chocolate em cada cookie após as adições de Bel.

Entrada

A primeira linha da entrada contém dois inteiros, N e M, a quantidade de linhas e a quantidade de colunas na bandeja.

As próximas N linhas contém M inteiros cada. A i-ésima destas linhas contém os inteiros $G_{i,1}$, $G_{i,2}$, ..., $G_{i,M}$, as quantidades de gotas de chocolate nos cookies da i-ésima linha.

Saída

A primeira linha da saída deve conter um único inteiro, o mínimo de gotas de chocolate que precisam ser adicionadas para que a bandeja esteja organizada.

As N linhas seguintes devem conter M inteiros cada, indicando a configuração final da bandeja de cookies na solução ótima, no mesmo formato da entrada.

Restrições

É garantido que todo caso de teste satisfaz as restrições abaixo.

- $1 \leq N, M \leq 100$
- N e M são ímpares
- $1 \leq G_{i,j} \leq 1\,000$ para todo $1 \leq i \leq N$ e $1 \leq j \leq M$

Informações sobre a pontuação

A tarefa vale 100 pontos. Estes pontos estão distribuídos em subtarefas, cada uma com suas **restrições adicionais** às definidas acima.

- **Subtarefa 1 (0 pontos):** Esta subtarefa é composta apenas pelos exemplos mostrados abaixo. Ela não vale pontos, serve apenas para que você verifique se o seu programa imprime o resultado correto para os exemplos.
- **Subtarefa 2 (15 pontos):** $N = 1$ e $M = 3$.
- **Subtarefa 3 (31 pontos):** $N = 1$.
- **Subtarefa 4 (54 pontos):** Sem restrições adicionais.

Exemplos

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

Explicação do exemplo 1: A entrada descreve uma bandeja 3×3. A bandeja inicial não está organizada pois a soma dos dois primeiros cookies na segunda linha, $2+2$, é par. Portanto, Bel precisa adicionar alguma quantidade de gotas de chocolate. De fato, basta que Bel adicione uma gota no cookie central, de forma que ele possua 3 gotas. É possível verificar que, agora, a soma das gotas em dois cookies adjacentes é sempre ímpar.

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

Exemplo de entrada 3	**Exemplo de saída 3**
1 5 1 2 3 4 5	0 1 2 3 4 5

Explicação do exemplo 3: Nesse exemplo, a bandeja de cookies já está organizada, logo, Bel não precisa adicionar nenhuma gota de chocolate.

Um Desafio Muito Distinto

Nome do arquivo: `distinto.c`, `distinto.cpp`, `distinto.pas`, `distinto.java`, `distinto.js` *ou* `distinto.py`

João e Maria são dois irmãos que adoram bolinhas de gude e matemática. Por isso, eles gostaram muito do novo lançamento da OBI (*Organização de Brincadeiras Infantis*), o *Desafio Muito Distinto*. Em cada partida do desafio, Maria usa bolinhas de gude para avaliar as habilidades de memória e matemática de João.

No início de uma partida, Maria coloca uma quantidade muito grande de bolinhas de gude em uma mesa e dá uma caixa **vazia** para João. Maria também escolhe três inteiros positivos L, A e B que definem as regras da partida:

- João deve usar as rodadas do desafio para mover bolinhas da mesa para a caixa. Se, ao final de uma rodada, a caixa possuir L ou mais bolinhas, a partida acaba.
- Em cada rodada, João poderá mover no mínimo A e no máximo B bolinhas de gude da mesa para a caixa.
- Uma rodada se inicia com João escolhendo um inteiro x, tal que $A \leq x \leq B$, de bolinhas de gude que ele deseja mover. Porém, João **não** pode escolher o mesmo valor de x mais de uma vez **na mesma partida**. Em outras palavras, os valores que João escolhe para x em cada rodada precisam ser todos **distintos**.
- Caso João não consiga escolher algum valor válido para x, a partida termina. Caso João escolha um x válido, ele completa a rodada movendo as x bolinhas de gude para a caixa. Vale ressaltar que a mesa possui muitas bolinhas de gude, de modo que sempre existem bolinhas suficientes para João mover.

Memorizar quantidades, contar e mover bolinhas é bastante trabalho. Por isso, Maria decidiu que, após cada partida, ela irá presentear João com uma quantidade de chocolates igual ao número de rodadas que ele **completou** na partida. Deste modo, a meta de João em toda partida é maximizar o número de rodadas que ele completa. Observe que é irrelevante para João se a caixa possui mais ou menos de L bolinhas de gude ao fim da partida.

João quer saber quantos chocolates ele conseguiria ganhar se jogar o desafio da melhor maneira possível. Além disso, como os valores de L, A e B escolhidos por Maria podem variar de uma partida para a outra, ele deseja consultar a resposta para vários valores diferentes de L, A e B.

Sua tarefa é: dada a quantidade P de partidas que João deseja consultar e os parâmetros L, A e B de cada partida, determine, para cada partida, o número máximo de rodadas que João consegue completar. Observe que as partidas são **independentes**, ou seja, é **permitido** que João repita o mesmo valor de x em partidas diferentes.

Entrada

A primeira linha da entrada contém um único inteiro P, a quantidade de partidas que João deseja consultar.

As próximas P linhas descrevem as consultas. A i-ésima destas linhas possui três inteiros L_i, A_i e B_i indicando os parâmetros L, A e B, respectivamente, escolhidos para a i-ésima partida.

Saída

Seu programa deverá imprimir P linhas, uma para cada partida, na mesma ordem da entrada. Para cada partida, imprima um único inteiro: o número máximo de rodadas que João consegue completar se ele jogar de forma ótima.

Restrições

É garantido que todo caso de teste satisfaz as restrições abaixo.

- $1 \leq P \leq 50\,000$
- $1 \leq L_i \leq 1\,000\,000\,000\,000$ para todo $1 \leq i \leq P$
- $1 \leq A_i \leq B_i \leq 2\,000\,000\,000$ para todo $1 \leq i \leq P$

Para competidores que utilizam C++ ou Java: Observe que alguns valores na entrada podem ser muito grandes para caberem em um inteiro de 32 bits. É recomendado o uso de inteiros de 64 bits (`long long` em C++; `long` em Java). *(Competidores usando Python ou JavaScript podem ignorar este aviso.)*

Informações sobre a pontuação

A tarefa vale 100 pontos. Estes pontos estão distribuídos em subtarefas, cada uma com suas **restrições adicionais** às definidas acima.

- **Subtarefa 1 (0 pontos)**: Esta subtarefa é composta apenas pelos exemplos mostrados abaixo. Ela não vale pontos, serve apenas para que você verifique se o seu programa imprime o resultado correto para os exemplos.
- **Subtarefa 2 (11 pontos):** $P = 1$, $A_1 = 1$ e $B_1 = L_1$.
- **Subtarefa 3 (12 pontos):** $P = 1$.
- **Subtarefa 4 (26 pontos):** $P \leq 80$.
- **Subtarefa 5 (20 pontos):** $A_i = 1$ e $B_i = L_i$ para todo $1 \leq i \leq P$.
- **Subtarefa 6 (31 pontos):** Sem restrições adicionais.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
2 7 3 7 8 3 7	2 3

Explicação do exemplo 1: Neste exemplo existem duas partidas.

Na primeira partida, Maria escolhe $L = 7$, $A = 3$ e $B = 7$. João consegue ganhar 2 chocolates da seguinte maneira:

- Na primeira rodada, João move 6 bolinhas da mesa para a caixa. 6 é uma escolha válida pois $A \leq 6 \leq B$ e João ainda não usou essa quantidade nessa partida. Como há menos de 7 bolinhas na caixa, a partida continua.
- Na segunda rodada, João move 3 bolinhas (perceba que ele não poderia mover 6 novamente). Agora a partida termina, pois há $6 + 3 = 9$ bolinhas na caixa e $9 \geq L$.
- Portanto, João consegue ganhar dois chocolates, pois completou duas rodadas. Perceba que João poderia ter movido 7 bolinhas na primeira rodada; porém, esta não seria a melhor maneira de jogar o desafio pois a partida terminaria após a primeira rodada.

Na segunda partida, Maria escolhe $L = 8$, $A = 3$ e $B = 7$. João consegue ganhar 3 chocolates:

- Na primeira rodada, ele move 4 bolinhas para a caixa.
- Na segunda rodada, ele move 3 bolinhas. Há $4 + 3 = 7$ bolinhas na caixa, e $7 < L$, então a partida continua.
- Na terceira rodada, João move 6 bolinhas. Como $4 + 3 + 6 = 13 \geq L$, a partida termina.
- Observe que a ordem das jogadas de João importa: se ele movesse 6 bolinhas na primeira rodada e depois 4 na segunda rodada, a partida acabaria em duas rodadas.

É possível verificar que 2 e 3 chocolates, respectivamente, são as respostas ótimas.

Exemplo de entrada 2	**Exemplo de saída 2**
1 10 1 3	3

Explicação do exemplo 2: Neste caso há apenas $P = 1$ partida com $L = 10$, $A = 1$ e $B = 3$. João consegue 3 chocolates nesta partida: primeiro ele move 2 bolinhas para a caixa na primeira rodada, depois 1 bolinha na segunda rodada, e, por fim, 3 bolinhas na terceira rodada. Ao final da partida, haverá $2 + 1 + 3 = 6$ bolinhas na caixa, o que é menos que L, mas a partida acaba pois não há mais nenhum x válido para João jogar.

Exemplo de entrada 3	**Exemplo de saída 3**
4 20 10 11 40 1 10 40 10 20 5 5 5	2 9 4 1

Feira de Artesanato

Nome do arquivo: `feira.c`, `feira.cpp`, `feira.pas`, `feira.java`, `feira.js` *ou* `feira.py`

A tradicional feira anual de artesanatos da sua cidade está chegando. O dono de uma das barracas mais populares da feira pediu a sua ajuda para registrar o lucro da barraca ao fim do dia.

Existem T tipos de objetos, numerados de 1 a T, que podem ser vendidos na barraca. No início do dia, você registra todo o estoque atual da barraca, que contém N objetos numerados de 1 a N. O i-ésimo objeto possui tipo t_i e preço p_i (em reais). Observe que a barraca pode possuir mais de um objeto do mesmo tipo em estoque, e **não** é garantido que todos os T tipos estão em estoque.

Durante o dia, C clientes vão visitar a barraca, um de cada vez. Todo cliente vai comprar no máximo um objeto, pagando à barraca o preço dele. Cada cliente pode ser decidido ou indeciso:

- Um cliente *decidido* sabe qual tipo de objeto ele deseja comprar. Ao visitar a barraca, um cliente decidido compra o objeto do tipo desejado que possui o menor preço. Caso existam mais de um objeto com o tipo desejado e preço mínimo, ele compra qualquer um destes objetos, mas somente um. Caso não exista nenhum objeto com o tipo desejado disponível, o cliente decidido vai embora sem comprar nada.
- Um cliente *indeciso* se importa mais com o preço do objeto do que com o tipo, usando o tipo do objeto apenas como critério de desempate. Mais especificamente, ao visitar a barraca, um cliente indeciso compra o objeto que possui menor preço entre todos os objetos disponíveis. Caso existam vários objetos com preço mínimo, ele compra o objeto cujo tipo é mínimo. O cliente indeciso só vai embora sem comprar nada caso não existam mais objetos disponíveis.

Vale ressaltar que cada um dos N objetos só pode ser comprado uma vez, e um objeto que é comprado é removido do estoque.

Sua tarefa é calcular o valor total que a barraca arrecadou com vendas após as visitas dos C clientes.

Entrada

A primeira linha da entrada possui dois inteiros N e T representando, respectivamente, a quantidade de objetos em estoque no início do dia e o número de tipos de objetos.

A segunda linha da entrada possui N inteiros $t_1,\ t_2,\ \ldots,\ t_N$, os tipos dos N objetos.

A terceira linha da entrada possui N inteiros $p_1,\ p_2,\ \ldots,\ p_N$, os preços em reais dos N objetos, na mesma ordem da linha anterior.

A quarta linha da entrada possui um único inteiro C, o número de clientes que visitaram a barraca.

A quinta linha da entrada possui C inteiros $u_1,\ u_2,\ \ldots,\ u_C$ e descrevem os clientes **na ordem em que visitaram a barraca**. Mais especificamente:

- Se o j-ésimo cliente é decidido, u_j é o tipo de objeto desejado.
- Se o j-ésimo cliente é indeciso, $u_j = 0$.

Saída

Seu programa deverá imprimir uma única linha contendo um único inteiro, o total em reais que a barraca recebeu ao longo do dia.

Restrições

É garantido que todo caso de teste satisfaz as restrições abaixo.

- $1 \leq N \leq 100\,000$
- $1 \leq T \leq 100\,000$
- $1 \leq C \leq 100\,000$
- $1 \leq t_i \leq T$ para $1 \leq i \leq N$
- $1 \leq p_i \leq 1\,000\,000$ para $1 \leq i \leq N$
- $0 \leq u_j \leq T$ para $1 \leq j \leq C$

Para competidores que utilizam C++ ou Java: Observe que a resposta pode ser muito grande para caber em um inteiro de 32 bits. É recomendado o uso de um inteiro de 64 bits (`long long` em C++; `long` em Java). *(Competidores usando Python ou JavaScript podem ignorar este aviso.)*

Informações sobre a pontuação

A tarefa vale 100 pontos. Estes pontos estão distribuídos em subtarefas, cada uma com suas **restrições adicionais** às definidas acima.

- **Subtarefa 1 (0 pontos)**: Esta subtarefa é composta apenas pelos exemplos mostrados abaixo. Ela não vale pontos, serve apenas para que você verifique se o seu programa imprime o resultado correto para os exemplos.
- **Subtarefa 2 (17 pontos):** $N \leq 100$, $T \leq 100$ e $C \leq 100$.
- **Subtarefa 3 (10 pontos):** Todos os clientes são indecisos.
- **Subtarefa 4 (13 pontos):** Todos os clientes são decididos.
- **Subtarefa 5 (12 pontos):** $T = 2$.
- **Subtarefa 6 (15 pontos):** Todos os objetos possuem preço 1.
- **Subtarefa 7 (14 pontos):** Não existem dois objetos com o mesmo preço.
- **Subtarefa 8 (19 pontos):** Sem restrições adicionais.

Exemplos

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

`8 10`
`4 2 3 1 10 1 1 4`
`34 50 156 81 97 12 3 3`
`7`
`0 1 0 1 5 4 1`
	`133`

Explicação do exemplo 1: A entrada indica que existem 10 tipos diferentes de objetos e que a barraca possui 8 objetos com os seguintes tipos e preços:

Objeto	**Tipo**	**Preço**
1	4	34
2	2	50
3	3	156
4	1	81
5	10	97
6	1	12
7	1	3
8	4	3

Durante o dia, 7 clientes visitam a barraca:

- O primeiro cliente é indeciso, então ele vai comprar um objeto com preço mínimo. Existem dois objetos com preço mínimo 3: o objeto 7, que é do tipo 1, e o objeto 8, que é do tipo 4. Entre esses dois, o cliente indeciso prefere o de menor tipo, e portanto **ele compra o objeto** 7.
- O segundo cliente é decidido pelo tipo 1, então ele vai comprar o objeto do tipo 1 com preço mínimo. O objeto 7 não está mais disponível, logo o mais barato do tipo 1 é o objeto 6. Portanto, **ele compra o objeto** 6.
- O terceiro cliente é indeciso. Dentre os objetos ainda disponíveis, o de menor preço é o objeto 8. Portanto, **ele compra o objeto** 8.
- O quarto cliente também é decidido pelo tipo 1. O objeto mais barato do tipo 1 ainda disponível é o objeto 4. Portanto, **ele compra o objeto** 4.
- O quinto cliente é decidido pelo tipo 5. Não existe nenhum objeto em estoque do tipo 5. Portanto, ele não compra nada.
- O sexto cliente é decidido pelo tipo 4, cujo objeto mais barato ainda disponível é o objeto 1. Portanto, **ele compra o objeto** 1.
- O sétimo cliente é decidido pelo tipo 1. Porém, não existe mais nenhum objeto do tipo 1 disponível. Portanto, ele não compra nada.

Ao fim do dia, a barraca arrecadou com a venda dos objetos 1, 4, 6, 7 e 8. O total recebido pela barraca foi $34 + 81 + 12 + 3 + 3 = 133$. Logo, a saída correta é apenas o inteiro `133`.

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

`7 2`
`1 1 2 1 2 2 1`
`7 3 4 1 8 5 10`
`8`
`0 2 0 0 1 1 1 1`
	`30`

Explicação do exemplo 2: (Este exemplo satisfaz as restrições das subtarefas 2, 5 e 7.) A ordem de compra dos objetos é: cliente 1 compra o objeto 4, cliente 2 compra o objeto 3, cliente 3 compra o objeto 2, cliente 4 compra o objeto 6, cliente 5 compra o objeto 1, cliente 6 compra o objeto 7, e clientes 7 e 8 não compram nada. Ao fim do dia, todos os objetos foram vendidos exceto o objeto 5. O total recebido pela barraca é $7 + 3 + 4 + 1 + 5 + 10 = 30$.