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 (23 de agosto de 2023).

OBI

Olimpíada Brasileira de Informática

OBI2023

Caderno de Tarefas

Modalidade Programação • Nível Sênior • Fase 2 - Turno B

23 de agosto de 2023

A PROVA TEM DURAÇÃO DE 3 HORAS

Promoção:

SBC Sociedade Brasileira de Computação

Apoio:

FUNDAÇÃO BEHRING

Coordenação:

INSTITUTO DE COMPUTAÇÃO UNICAMP

Instruções

LEIA ATENTAMENTE ESTAS INSTRUÇÕES ANTES DE INICIAR A PROVA

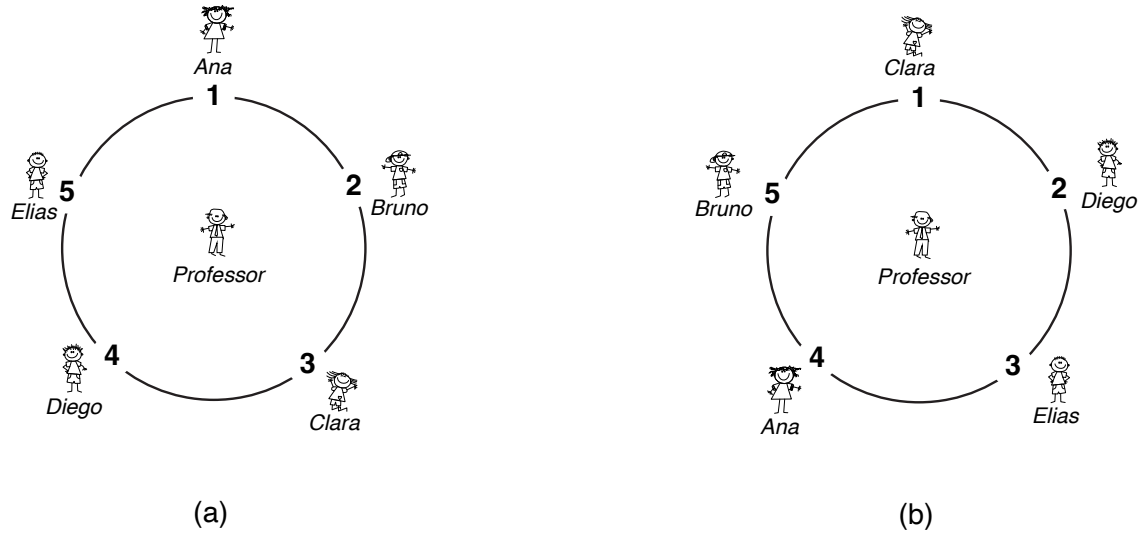
- Este caderno de tarefas é composto por 13 páginas (não contando a folha de rosto), numeradas de 1 a 13. Verifique se o caderno está completo.
- A prova deve ser feita individualmente.
- É proibido consultar a Internet, livros, anotações ou qualquer outro material durante a prova. É permitida a consulta 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.

Brincadeira de Roda

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

Bruno convidou seu professor e colegas de classe para uma conferência da OBI (*Organização de Brincadeiras Infantis*). Hoje, o tema da conferência é "Brincadeiras de Roda." As brincadeiras do dia seguem todas a mesma estrutura: Bruno e seus colegas formam uma roda enquanto o professor, que não faz parte da roda, fica no centro dela. O professor aponta para a posição de um dos alunos, indicando que aquela posição é representada pelo número 1. As outras posições são então numeradas no sentido horário, em ordem crescente. A figura (a) indica um possível cenário inicial da brincadeira para uma classe com 5 alunos.

Quando a brincadeira começa, os alunos devem ficar em suas posições (participando de outras dinâmicas) até o professor bater palmas: toda vez que ouvem palmas, todos os alunos se movem uma posição no sentido horário (ou seja, o aluno na posição 1 vai para a posição 2, o aluno na posição 2 vai para a posição 3, e assim por diante; o aluno na última posição se move para a posição 1). A figura (b) indica como fica a roda da figura (a) após o professor bater palmas três vezes.



Bruno e alguns outros alunos foram beber água, enquanto a brincadeira continuou, com o professor batendo palmas algumas vezes. Bruno agora quer voltar para sua posição correta (ou seja, sua posição se não tivesse saído para beber água). Ele lembra a posição em que estava logo antes de sair, e conseguiu ouvir todas as vezes que o professor bateu palmas enquanto ele estava fora. Agora, ele pede sua ajuda: dados o número de estudantes na classe, a posição inicial de Bruno, e o número de vezes que o professor bateu palmas enquanto ele foi beber água, ajude Bruno a descobrir para qual posição na roda ele deve voltar.

Entrada

A primeira linha de entrada contém um inteiro N indicando o número de alunos na classe, incluindo Bruno. As posições na roda são numeradas de 1 a N no sentido horário. A segunda linha contém um inteiro I, indicando a posição inicial de Bruno (ou seja, a posição dele logo antes de sair para beber água). A terceira e última linha contém um inteiro P, indicando o número de vezes que o professor bateu palmas enquanto Bruno estava fora da roda.

Saída

Seu programa deve produzir uma única linha, contendo um único inteiro, a posição na roda para a qual Bruno deve voltar.

Restrições

- $3 \leq N \leq 100$
- $1 \leq I \leq N$
- $1 \leq P \leq 1\,000$

Informações sobre a pontuação

A tarefa vale 100 pontos.

Exemplos

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

Explicação do exemplo 1: esse é o exemplo da figura (b) do enunciado. Temos 5 estudantes na roda e Bruno estava na posição 2. Após o professor bater palmas três vezes, Bruno deve ir para a posição 5.

Exemplo de entrada 2	**Exemplo de saída 2**
10 7 30	7

Explicação do exemplo 2: Depois de 30 rodadas de palmas, todos os 10 estudantes estão de volta em suas posições iniciais (posições em que estavam quando Bruno saiu para beber água).

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

UPA

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

A Unicamp organiza anualmente, sempre em agosto, a UPA – Unicamp Portas Abertas –, um evento em que alunos de escolas da região podem visitar e conhecer as atividades da Universidade. O evento recebe em um dia dezenas de milhares alunos, que chegam em ônibus.

Um dos grandes problemas da organização da UPA é o estacionamento dos ônibus que trazem os alunos. A Unicamp reserva uma longa avenida para que os ônibus estacionem, um em seguida ao outro, todos paralelos à avenida; para cada ônibus são reservados 20 metros de avenida, com os espaços marcados no chão.

Para organizar o estacionamento, cada escola que chega de ônibus deve informar o horário de chegada e o horário de partida de seu ônibus, e os horários devem ser obedecidos rigorosamente. Dessa forma, o espaço de um ônibus que já partiu pode ser reutilizado por um outro ônibus que chega.

Dados os horários em que cada ônibus chega e parte, escreva um programa para calcular o menor espaço possível, em metros, que deve ser reservado para que todos os ônibus estacionem.

Entrada

A primeira linha contém um inteiro N, o número de ônibus que chegarão para a UPA. Os horários de chegada e partida são dados como números inteiros. Cada uma das N linhas seguintes descreve um ônibus e contém dois inteiros C_i e P_i, respectivamente o instante de chegada e o instante de partida do ônibus i. Dois ônibus A e B tais que $C_B \geq P_A$ ou $C_A \geq P_B$ podem utilizar o mesmo espaço de estacionamento. Assim, por exemplo, se o horário de chegada do ônibus B e a partida do ônibus A forem iguais, então o ônibus B pode ocupar o espaço que o ônibus A ocupou.

Saída

Seu programa deve produzir uma única linha, contendo um único inteiro, o menor comprimento da avenida, em metros, que deve ser reservado para o estacionamento de ônibus.

Restrições

- $1 \leq N \leq 100\ 000$
- $0 \leq C_i \leq 100\ 000$ para $1 \leq i \leq N$
- $0 \leq P_i \leq 100\ 000$ para $1 \leq i \leq N$
- $C_i < P_i$ para $1 \leq i \leq N$

Informações sobre a pontuação

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

- **Subtarefa 1 (12 pontos):** $P_i = 1 + C_i$ para $1 \leq i \leq N$

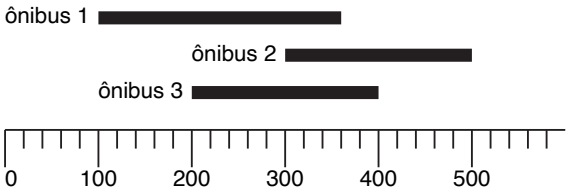
- **Subtarefa 2 (29 pontos):** $N \leq 1\,000$, $C_i \leq 1\,000$ e $P_i \leq 1\,000$.
- **Subtarefa 3 (27 pontos):** É garantido que a resposta é menor do que 1 000 metros.
- **Subtarefa 4 (32 pontos):** Nenhuma restrição adicional.

Seu programa pode resolver corretamente todas ou algumas das subtarefas acima *(elas não precisam ser resolvidas em ordem)*. Sua pontuação final na tarefa é a soma dos pontos de todas as subtarefas resolvidas corretamente por qualquer uma das suas submissões.

Exemplos

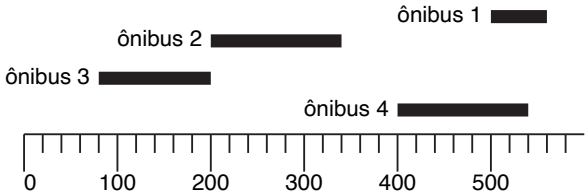
Exemplo de entrada 1	**Exemplo de saída 1**
3 100 360 300 500 200 400	60

Explicação do exemplo 1: há três ônibus. O primeiro chega no instante 100 e parte no instante 360, o segundo chega no instante 300 e parte no instante 500, e o terceiro chega no instante 200 e parte no instante 400. A figura abaixo ilustra o tempo em que cada ônibus utiliza o estacionamento. A resposta é 60 metros (espaço para 3 ônibus).



Exemplo de entrada 2	**Exemplo de saída 2**
4 500 560 200 340 80 200 400 540	40

Explicação do exemplo 2: há quatro ônibus. O primeiro chega no instante 500 e parte no instante 560, o segundo chega no instante 200 e parte no instante 340, o terceiro chega no instante 80 e parte no instante 200, e o quarto chega no instante 400 e parte no instante 540. A figura abaixo ilustra o tempo em que cada ônibus utiliza o estacionamento. A resposta é 40 metros (espaço para 2 ônibus) – por exemplo, os ônibus 1 e 2 podem usar um dos espaços e os ônibus 3 e 4 podem usar o outro.



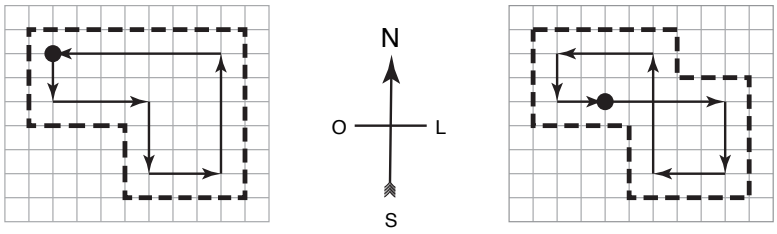
Corrida de Rua

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

A Quadradônia foi escolhida para sediar uma prova internacional de corrida de rua. As ruas da Quadradônia são todas alinhadas aos eixos Norte-Sul e Leste-Oeste. As quadras da cidade são quadradas, todas de mesma dimensão. A partida e a chegada da corrida são em uma mesma interseção de ruas.

Para realizar a corrida será necessário cercar a área da cidade onde será feito o percurso. A cerca será colocada a *no mínimo* uma quadra de distância de qualquer ponto do percurso da corrida, e deve ser retilínea e alinhada aos eixos.

As figuras (a) e (b) abaixo mostram mapas da cidade ilustrando duas possibilidades de percurso e respectivas possibilidades de cercas.



Dado o percurso da corrida, sua tarefa é escrever um programa para determinar o menor comprimento possível da cerca, em número de quadras.

Entrada

A primeira linha contém um inteiro N, o número de segmentos que definem o percurso da corrida. As N linhas seguintes descrevem os segmentos, na ordem em que são percorridos na corrida, a partir do ponto de partida/chegada. Cada linha contém um inteiro C_i e um caractere D_i, indicando respectivamente o comprimento do segmento, em número de quadras, e a direção do segmento, onde '`N`' indica Norte, '`S`' indica Sul, '`L`' indica Leste e '`O`' indica Oeste.

Saída

Seu programa deve produzir uma única linha, contendo um único inteiro, o menor comprimento possível da cerca, em número de quadras.

Restrições

- $2 \leq N \leq 100\,000$
- $1 \leq C_i \leq 10\,000$ para $1 \leq i \leq N$
- D_i é '`N`', '`S`', '`L`' ou '`O`' para $1 \leq i \leq N$

Informações sobre a pontuação

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

- **Subtarefa 1 (11 pontos):** $N = 2$, ou seja, o percurso utiliza uma única rua. *(Veja o exemplo de entrada 1.)*
- **Subtarefa 2 (30 pontos):**

  - $D_i =$ '`N`' ou '`L`' para $1 \leq i \leq N-2$
  - $D_{N-1} =$ '`S`'
  - $D_N =$ '`O`'

  Ou seja, o percurso da corrida forma uma *escadinha. (Veja os exemplos de entrada 3 e 4.)*

- **Subtarefa 3 (25 pontos):**
  - $N \leq 100$
  - $C_i \leq 10$ para $1 \leq i \leq N$
- **Subtarefa 4 (34 pontos):** Nenhuma restrição adicional.

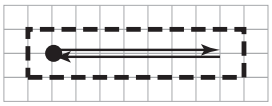
Seu programa pode resolver corretamente todas ou algumas das subtarefas acima *(elas não precisam ser resolvidas em ordem)*. Sua pontuação final na tarefa é a soma dos pontos de todas as subtarefas resolvidas corretamente por qualquer uma das suas submissões.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
2 7 L 7 O	22

Explicação do exemplo 1:

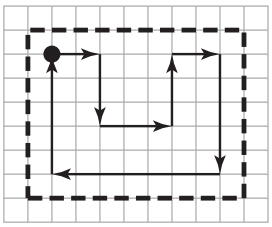
o percurso consiste de uma única rua. A figura abaixo ilustra o percurso e a menor cerca possível, que tem comprimento de 22 quadras.



Exemplo de entrada 2	**Exemplo de saída 2**
8 2 L 3 S 3 L 3 N 2 L 5 S 7 O 5 N	32

Explicação do exemplo 2:

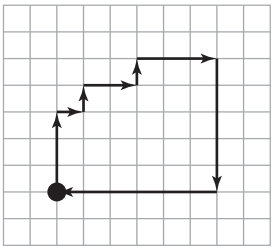
O percurso consiste de oito segmentos. A figura ao lado ilustra o percurso e a menor cerca possível.



Exemplo de entrada 3	**Exemplo de saída 3**
8 3 N 1 L 1 N 2 L 1 N 3 L 5 S 6 O	30

Explicação do exemplo 3:

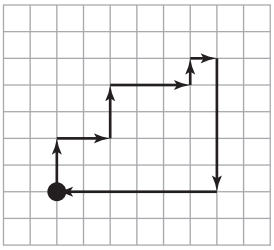
A figura ao lado ilustra o percurso, que tem 8 segmentos. A menor cerca possível para esse percurso tem comprimento 30 quadras. Observe que esse exemplo e o próximo satisfazem às restrições da subtarefa 2.



Exemplo de entrada 4	**Exemplo de saída 4**
8 2 N 2 L 2 N 3 L 1 N 1 L 5 S 6 O	30

Explicação do exemplo 4:

A figura ao lado ilustra o percurso, que tem 8 segmentos. A menor cerca possível para esse percurso tem comprimento 30 quadras. Observe que esse exemplo satisfaz às restrições da subtarefa 2.



Exemplo de entrada 5	**Exemplo de saída 5**
7 5 L 3 S 3 O 5 N 4 O 2 S 2 L	32

Explicação do exemplo 5: esse é o exemplo da figura (b) do enunciado. A menor cerca possível para esse percurso é 32 quadras.

Fortunas

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

A família Silva tem a tradição de investir em grandes empreendimentos. A família é um *matriarcado*, ou seja, são as mulheres da família que comandam as decisões. Vamos chamar de matriarca cada mulher integrante da família. Elas são sócias de empresas como bancos, mineradoras, construtoras e petroleiras. Além de saberem analisar muito bem diversos setores da economia, elas são muito unidas e investem sempre de maneira conjunta, em família.

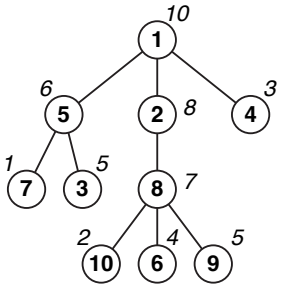
Apesar de investirem juntas, cada matriarca da família possui sua própria fortuna e administra seu próprio patrimônio. Um detalhe curioso é que, nessa família, a fortuna de uma matriarca é sempre menor que a fortuna de sua mãe, como consequência da experiência administrativa dos mais velhos.

Toda vez que uma matriarca da família pretende investir em algum empreendimento novo, ela convoca uma reunião com outras matriarcas para compartilhar conhecimento. Quando uma matriarca V convoca uma reunião, ela define um intervalo $[E, D]$ de valores de fortunas, de modo que são convocadas todas as matriarcas da família que satisfazem as seguintes condições:

- A matriarca V é convocada para sua própria reunião (chamamos ela de *anfitriã*), e é garantido que sua fortuna está no intervalo $[E, D]$ (ou seja, sua fortuna é maior ou igual a E e menor ou igual a D);
- Toda matriarca convocada precisa ter fortuna no intervalo $[E, D]$;
- Toda matriarca convocada além da anfitriã precisa ter relação de parentesco direto (ou seja, ser mãe ou filha imediata) com alguma outra matriarca convocada.

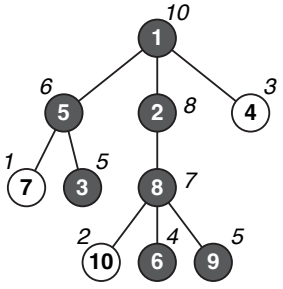
Por exemplo, considere a família mostrada na figura ao lado, com 10 matriarcas, identificadas pelos números de 1 a 10, com 1 sendo a matriarca mais velha.

A fortuna de cada matriarca está indicada ao lado dela, em milhões de reais.



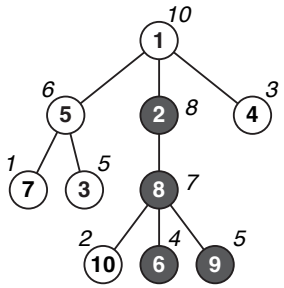
Se a matriarca 2 convocar uma reunião e definir o intervalo de fortunas como $[4, 10]$, as matriarcas convocadas serão 1, 2, 3, 5, 6, 8 e 9.

Observe que a matriarca 4 não será convocada pois possui fortuna igual a 3 (menor do que o limite inferior 4).



Por outro lado, uma reunião convocada pela matriarca 8 com o intervalo de fortunas $[4, 9]$ terá as matriarcas 2, 6, 8 e 9.

Apesar de a matriarca 5 possuir fortuna no intervalo $[4, 9]$, ela não foi convocada pois não possui relação de parentesco direto com nenhuma outra matriarca convocada.



Sua tarefa é: dada a estrutura da família, a fortuna de cada matriarca, e quem convocou e os limites de fortuna de cada reunião, determine para quantas reuniões cada matriarca foi convocada.

Entrada

A primeira linha de entrada contém dois inteiros N e M representando o número de matriarcas da família e o número de reuniões, respectivamente. As matriarcas da família são identificadas por inteiros de 1 a N, com o número 1 representando a matriarca mais velha da família.

As próximas N linhas descrevem a estrutura da família. A i-ésima dessas linhas contém dois inteiros A_i e B_i representando a fortuna da i-ésima matriarca e o identificador de sua mãe, respectivamente. A matriarca mais velha da família (matriarca 1) é a única matriarca com $B_i = i$. É garantido que a fortuna de uma matriarca é sempre menor que a fortuna de sua mãe.

As próximas M linhas descrevem as reuniões. A j-ésima dessas linhas contém três inteiros O_j, E_j e D_j, indicando que a matriarca O_j foi anfitriã de uma reunião com intervalo de fortunas $[E_j, D_j]$.

Saída

Seu programa deverá imprimir uma única linha contendo N inteiros separados por espaços. O i-ésimo desses números deve ser a quantidade de reuniões para as quais a matriarca i foi convocada.

Restrições

- $1 \leq N \leq 100\ 000$
- $1 \leq M \leq 100\ 000$
- $1 \leq A_i \leq 100\ 000$
- $A_i < A_{B_i}$ para todo $i > 1$
- $1 \leq B_i \leq N$
- $1 \leq O_j \leq N$
- $1 \leq E_j \leq A_{O_j} \leq D_j \leq 100\ 000$

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 (13 pontos):** $M = 1$
- **Subtarefa 2 (21 pontos):** $B_i = i - 1$ para todo $i > 1$ – ou seja, a mãe da matriarca i é a matriarca $i - 1$. *(Veja o exemplo de entrada 2.)*

- **Subtarefa 3 (25 pontos):** $E_j = 1$ e $D_j = A_{O_j}$ para todo $1 \leq j \leq M$
- **Subtarefa 4 (21 pontos):**
  - $D_j = A_{O_j}$ para todo $1 \leq j \leq M$
  - $A_i \leq 200$ para todo $1 \leq i \leq N$
- **Subtarefa 5 (20 pontos):** Nenhuma restrição adicional.

Seu programa pode resolver corretamente todas ou algumas das subtarefas acima *(elas não precisam ser resolvidas em ordem)*. Sua pontuação final na tarefa é a soma dos pontos de todas as subtarefas resolvidas corretamente por qualquer uma das suas submissões.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
10 2 10 1 8 1 5 5 3 1 6 1 4 8 1 5 7 2 5 8 2 8 2 4 10 8 4 9	1 2 1 0 1 2 0 2 2 0

Explicação do exemplo 1: Este exemplo corresponde ao exemplo mostrado no enunciado.

Exemplo de entrada 2	**Exemplo de saída 2**
8 5 42 1 31 1 29 2 27 3 18 4 15 5 12 6 5 7 1 17 42 4 18 30 8 5 28 4 16 40 6 12 18	1 2 3 4 5 2 2 1

Explicação do exemplo 2: A seguir temos uma visualização da estrutura da família: