Maratona Paulista de Programação 2026

2ª Edição

20 de Junho de 2026

Caderno de Problemas

Informações Gerais

Este caderno contém 14 problemas; as páginas estão numeradas de 1 a 21, não contando esta página de rosto. Verifique se o caderno está completo.

A) Sobre os nomes dos programas
1) Para soluções em C/C++ e Python, o nome do arquivo-fonte não é significativo, pode ser qualquer nome.
2) Se sua solução é em Java, ela deve ser chamada *codigo_de_problema*`.java` onde *codigo_de_problema* é a letra maiúscula que identifica o problema. Lembre que em Java o nome da classe principal deve ser igual ao nome do arquivo.

B) Sobre a entrada
1) A entrada de seu programa deve ser lida da *entrada padrão*.
2) A entrada é composta de um único caso de teste, descrito em um número de linhas que depende do problema.
3) Quando uma linha da entrada contém vários valores, estes são separados por um único espaço em branco; a entrada não contém nenhum outro espaço em branco.
4) Cada linha, incluindo a última, contém exatamente um caractere final-de-linha.
5) O final da entrada coincide com o final do arquivo.

C) Sobre a saída
1) A saída de seu programa deve ser escrita na *saída padrão*.
2) Quando uma linha da saída contém vários valores, estes devem ser separados por um único espaço em branco; a saída não deve conter nenhum outro espaço em branco.
3) Cada linha, incluindo a última, deve conter exatamente um caractere final-de-linha.

Problema A

After em Campinas

Houve um grande festival de vários estilos musicais na cidade de Campinas, atraindo visitantes de todo o país. Mas agora já são duas da manhã de domingo e o festival terminou. Ainda animados, vários grupos de visitantes decidiram procurar um after para continuar a noite.

Por sorte, eles encontraram uma lista de baladas disponíveis na cidade para aquela noite, todas com open bar. Para cada balada, a lista informa o valor da entrada e o estilo musical predominante.

Como os visitantes não conhecem bem a cidade, todos os grupos seguirão exatamente a ordem da lista para encontrar a primeira balada que eles possam ir. Cada grupo possui um limite de dinheiro que pode gastar por pessoa e também rejeita um determinado estilo musical. Assim, cada grupo escolherá a primeira balada da lista cuja entrada esteja dentro do seu limite e cujo estilo musical não seja o estilo rejeitado pelo grupo. Sua tarefa é ajudar a descobrir qual balada será escolhida por cada grupo de visitantes.

Entrada

A primeira linha contém dois inteiros N e Q ($1 \leq N, Q \leq 2 \cdot 10^5$), indicando, respectivamente, a quantidade de baladas na lista e a quantidade de grupos de visitantes.

As próximas N linhas descrevem as baladas, na ordem em que aparecem na lista. Cada uma dessas linhas contém dois inteiros C_i e S_i ($1 \leq C_i \leq 10^8$ e $1 \leq S_i \leq 10^8$), indicando, respectivamente, o valor da entrada da i-ésima balada e o identificador do seu estilo musical.

As baladas são fornecidas em ordem não crescente de valor de entrada, ou seja, $C_1 \geq C_2 \geq \cdots \geq C_N$. Em caso de empate no valor da entrada, a ordem entre as baladas é arbitrária.

As próximas Q linhas descrevem os grupos de visitantes. Cada uma dessas linhas contém dois inteiros D_i e O_i ($1 \leq D_i \leq 10^8$ e $1 \leq O_i \leq 10^8$) indicando, respectivamente, a quantidade máxima de dinheiro que o i-ésimo grupo pode gastar por pessoa e o identificador do estilo musical rejeitado por esse grupo.

Saída

Para cada grupo, imprima uma linha contendo o índice da balada escolhida.

Caso não exista uma balada válida para o grupo, imprima `-1`.

Exemplo de entrada 1	**Exemplo de saída 1**
5 6 100 1 80 2 80 1 50 3 20 2 90 1 80 2 80 1 50 3 19 1 100 4	2 3 2 5 -1 1

Exemplo de entrada 2	**Exemplo de saída 2**
6 7 50 1 40 2 40 3 25 2 10 4 5 1 50 1 40 2 30 2 10 4 4 1 5 2 100 5	2 3 5 6 -1 6 1

Problema B

Bilhete de metrô

Raphael utiliza esporadicamente o serviço de metrô da cidade de São Paulo e agora vai passar a utilizar um bilhete com um novo método de tarifa, o bilhete único com validade. Ao comprar esse tipo de bilhete, pode-se escolher uma duração de D dias para o bilhete com o custo de C reais para cada dia, ou seja, custo total de $D \cdot C$. Com esse bilhete, pode-se utilizar o metrô livremente por D dias consecutivos (no dia da compra e nos próximos $D - 1$ dias).

Porém Raphael precisa pedir carona por aplicativo para ir até a bilheteria onde se vende esse novo tipo de bilhete, pois apesar dos métodos de compra online, Raphael prefere pagar em dinheiro e receber uma passagem física. Essa carona por aplicativo tem um custo de K reais. Ou seja, no fim das contas, para Raphael conseguir comprar um bilhete que dura D dias, o custo final acaba sendo $D \cdot C + K$ reais.

Raphael já sabe a lista de N dias em que ele precisa utilizar o metrô e agora pediu sua ajuda para saber o menor custo possível para que ele possa utilizar o metrô todos esses dias apenas comprando bilhetes únicos com validade. Inicialmente Raphael não possui nenhum bilhete.

Por exemplo, se $C = 3$, $K = 10$ e Raphael precisa usar o metrô por 3 dias que são daqui a 2 dias, 3 dias e 8 dias, uma maneira possível seria comprar daqui a 2 dias apenas um bilhete com validade de 7 dias resultando no custo de $7 \cdot 3 + 10 = 31$. Porém, ele também poderia comprar um bilhete daqui a 2 dias com duração de 2 dias e mais um daqui a 8 dias com duração de apenas um dia, resultando num custo de $(2 \cdot 3 + 10) + (1 \cdot 3 + 10) = 29$, sendo assim uma opção mais barata.

Entrada

A primeira linha contém três inteiros N, C e K ($1 \leq N \leq 10^5$, $1 \leq C \leq 1000$, $1 \leq K \leq 10^9$). A segunda linha contém N inteiros distintos D_i em ordem crescente indicando quais dias Raphael precisa utilizar o metrô ($1 \leq D_i \leq 10^6$). D_i indica que daqui D_i dias Raphael precisa utilizar o metrô.

Saída

Imprima uma única linha com o menor custo possível que Raphael precisa gastar para utilizar o metro todos os dias que ele precisa apenas comprando bilhetes únicos com validade.

Exemplo de entrada 1	**Exemplo de saída 1**
3 3 10 2 3 8	29

Exemplo de entrada 2	**Exemplo de saída 2**
3 3 10 2 3 4	19

Problema C

Capital mundial da pizza

São Paulo é amplamente celebrada como a Capital Mundial da Pizza. A cidade produz quase 4 pizzas por segundo, superando a marca de 300 mil unidades diárias. Consome a segunda maior quantidade do prato no mundo, atrás apenas de Nova York.

Neste problema, você deve ajudar a calcular uma informação essencial quando se pede pizza por delivery. Se há N pizzas de 8 pedaços, e M pessoas para comer os pedaços, qual o maior número de pedaços que cada um pode comer se todas as pessoas comerem o mesmo número de pedaços?

Por exemplo, se há 2 pizzas e 3 pessoas, isso quer dizer que há 16 pedaços no total e que cada um come no máximo 5 pedaços (acaba sobrando 1 pedaço).

Entrada

A primeira e única linha da entrada contém dois inteiros N e M ($1 \leq N \leq 100, 1 \leq M \leq 20$) indicando respectivamente o número de pizzas de 8 pedaços e o número de pessoas para comer.

Saída

A saída deve ter apenas uma única linha com um inteiro indicando o número máximo de pedaços que cada um pode comer.

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

Exemplo de entrada 2	**Exemplo de saída 2**
`10 8`	`10`

Problema D

Desenhando SP

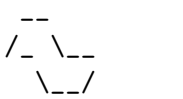
A ASCII art é uma forma de expressão artística usando apenas os caracteres disponíveis nas tabelas de código de página de computadores. Antes dos computadores já existia uma arte semelhante realizada com máquinas de escrever. Nesta tarefa você deverá desenhar o formato do estado de São Paulo seguindo os exemplos dados. O desenho deverá ser proporcional a um certo inteiro N.

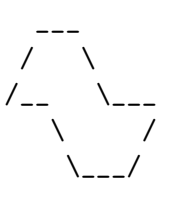
Entrada

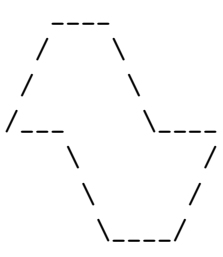
A entrada consiste de uma única linha que contém um inteiro N ($1 \leq N \leq 50$).

Saída

A saída deverá ter $2N + 1$ linhas contendo apenas caracteres de espaço ou / ou \ ou _ (na tabela ASCII, os valores desses caracteres são respectivamente 32, 47, 92 e 95). O desenho deve ter 2 lados formados por N caracteres /, 2 lados formados por N caracteres \, 3 lados formados por $N + 1$ caracteres _ e 1 lado formado por N caracteres _. Espaços devem ser adicionados para que o desenho fique de acordo com os exemplos de saída e não deve haver nenhuma linha terminada por caracteres de espaço. Neste problema, caracteres de espaço extras ou faltando serão considerados como resposta errada.

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

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

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

Problema E

Emergência espacial

A Estação de Pesquisa Orbital Artemis VII é composta por N módulos conectados por $N-1$ corredores pressurizados, de modo que é possível ir de qualquer módulo a qualquer outro percorrendo os corredores. Cada corredor tem um comprimento em metros.

Ao longo do tempo, engenheiros chegam à estação. Inicialmente, nenhum módulo possui engenheiros, e um mesmo módulo pode abrigar vários engenheiros simultaneamente. O número total de engenheiros na estação é sempre par.

Em caso de emergência, todos os engenheiros devem se reunir em pares. Cada engenheiro percorre os corredores para encontrar exatamente um outro engenheiro, de modo que todo engenheiro faça parte de exatamente um par. O **custo de uma resposta emergencial** é a menor distância total que os engenheiros precisam percorrer, somada sobre todos os pares. Isto é, a soma das distâncias entre os dois engenheiros de cada par, minimizada sobre todas as maneiras de formar os pares.

Você receberá uma sequência de Q eventos. Em cada evento, dois novos engenheiros chegam à estação e são alocados aos módulos x e y (não necessariamente distintos). Após cada evento, determine o custo de uma resposta emergencial considerando todos os engenheiros presentes na estação naquele momento.

Entrada

A primeira linha contém dois inteiros N e Q ($1 \leq N, Q \leq 10^5$), o número de módulos e o número de eventos. A i-ésima das $N-1$ linhas seguintes contém três inteiros u_i, v_i e w_i ($1 \leq u_i, v_i \leq N$, $u_i \neq v_i$, $1 \leq w_i \leq 10^9$), descrevendo um corredor de comprimento w_i que conecta os módulos u_i e v_i. É garantido que os corredores conectam todos os módulos, ou seja, existe um caminho entre quaisquer dois deles. A i-ésima das Q linhas seguintes contém dois inteiros x_i e y_i ($1 \leq x_i, y_i \leq N$), os módulos aos quais os dois novos engenheiros do i-ésimo evento são alocados. Os valores x_i e y_i podem ser iguais.

Saída

O seu programa deve imprimir Q linhas. A i-ésima linha deve conter o custo de uma resposta emergencial após o i-ésimo evento.

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

Exemplo de entrada 2	**Exemplo de saída 2**
6 4 1 2 1 1 3 2 3 4 8 4 5 3 4 6 3 1 4 2 6 3 5 1 1	10 4 15 15

Explicação do exemplo 2:

- **Evento 1**: Há dois engenheiros na estação, um em 1 e um em 4, que formam o único par possível: o engenheiro de 1 se encontra com o de 4 percorrendo o caminho $1 \to 3 \to 4$, de distância $2 + 8 = 10$. Custo total: **10**.
- **Evento 2**: Agora há quatro engenheiros, em 1, 2, 4 e 6. O melhor emparelhamento possível desfaz o par do evento anterior:
  - o engenheiro de 1 com o de 2, pelo caminho $1 \to 2$, de distância 1;
  - o engenheiro de 4 com o de 6, pelo caminho $4 \to 6$, de distância 3.

  Custo total: $1 + 3 = \mathbf{4}$.
- **Evento 3**: Agora há um engenheiro em cada um dos seis módulos. O melhor emparelhamento é:
  - o engenheiro de 1 com o de 2, pelo caminho $1 \to 2$, de distância 1;
  - o engenheiro de 3 com o de 4, pelo caminho $3 \to 4$, de distância 8;
  - o engenheiro de 5 com o de 6, pelo caminho $5 \to 4 \to 6$, de distância $3 + 3 = 6$.

  Custo total: $1 + 8 + 6 = \mathbf{15}$.
- **Evento 4**: Agora os dois novos engenheiros chegam ao mesmo módulo, 1, que passa a ter três engenheiros (os demais módulos continuam com um cada). O melhor emparelhamento é:
  - os dois engenheiros que acabaram de chegar ao módulo 1 se emparelham entre si, sem percorrer nenhum corredor, de distância 0;
  - o engenheiro que já estava em 1 com o de 2, pelo caminho $1 \to 2$, de distância 1;
  - o engenheiro de 3 com o de 4, pelo caminho $3 \to 4$, de distância 8;
  - o engenheiro de 5 com o de 6, pelo caminho $5 \to 4 \to 6$, de distância $3 + 3 = 6$.

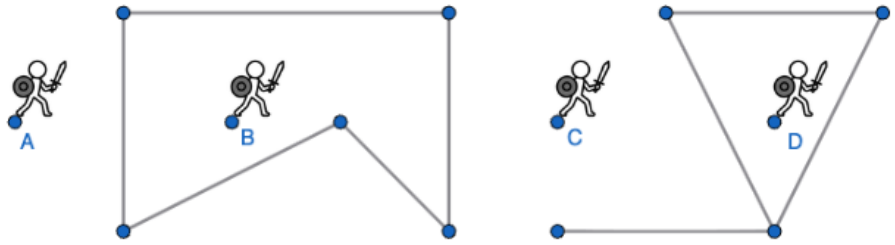
  Custo total: $0 + 1 + 8 + 6 = \mathbf{15}$.

Problema F

Farmando aura

Uma das gírias de internet mais populares no momento é "farmar aura", que significa acumular pontos de estilo, carisma, respeito ou presença. O termo mistura a expressão gamer "farmar" (repetir ações para acumular pontos) com aura (a energia, "vibe" ou postura de uma pessoa). Um dos jogos online onde essa gíria se popularizou se chama Fibia, um jogo onde o objetivo é explorar um vasto mundo em um mapa bidimensional onde o jogador tem uma visão superior do plano onde fica seu personagem.

Em um certo cenário do jogo, há N pontos e M segmentos ligando pares desses N pontos. Cada segmento representa uma parede no plano de jogo. Esses segmentos não se cruzam, podem se encostar apenas em suas extremidades em um dos N pontos (nenhum dos N pontos está dentro de um dos M segmentos). Dependendo da posição de um jogador nesse cenário, ele pode estar "preso" entre esses segmentos, ou seja, a posição do jogador está dentro de algum polígono delimitado por esses segmentos. Quando um jogador está nesse tipo de posição, ele está "perdendo aura", pois não conseguirá explorar toda a extensão do mapa a menos que utilize de alguma magia de teletransporte. Caso contrário, o jogador está numa posição "ganhando aura".



Na figura acima, podemos ver que os jogadores nas posições A e C estão "ganhando aura", enquanto os jogadores nas posições B e D estão "perdendo aura". Dado a posição de K jogadores no plano de jogo, sua tarefa vai ser dizer quem está perdendo aura e quem está ganhando aura.

Entrada

A primeira linha contém dois inteiros N e M ($2 \leq N \leq 1000$, $1 \leq M \leq \frac{N(N-1)}{2}$).

As próximas N linhas contém dois inteiros X_i e Y_i ($0 \leq X_i, Y_i \leq 10000$) indicando a posição dos N pontos em ordem no plano cartesiano do jogo (todos os pontos são distintos).

As próximas M linhas contém dois inteiros A_i e B_i ($1 \leq A_i, B_i \leq N$, $A_i \neq B_i$) indicando M pares distintos de pontos que estão conectados por segmentos. É garantido que nenhum par de segmentos se cruza, apenas podem se tocar em suas extremidades. Nenhum segmento toca alguns dos outros $N-2$ pontos fora suas extremidades.

A próxima linha contém um inteiro K ($1 \leq K \leq 1000$).

As próximas K linhas possuem dois inteiros KX_i e KY_i ($0 \leq KX_i, KY_i \leq 10000$) indicando as coordenadas dos K pontos em ordem. É garantido que os K pontos não estão contidos em nenhum dos M segmentos e que não são iguais a nenhum dos N pontos.

Saída

Seu programa deve produzir uma única linha com uma sequência de K caracteres "`P`" ou "`G`" indicando em ordem quais jogadores estão "perdendo aura" ou "ganhando aura".

Exemplo de entrada 1	**Exemplo de saída 1**
3 3 0 0 10 0 0 10 1 2 2 3 3 1 2 1 1 5 6	PG

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

Explicação do exemplo 2: Esse exemplo representa a imagem no enunciado.

Problema G

Gestão do desfile

A cidade de Grafos do Jordão está sendo preparada para um grande festival. Estão sendo preparadas $N-1$ avenidas que ligam N pontos diferentes da cidade para a realização do mesmo. Da maneira que foram escolhidas essas avenidas, existe um e apenas um único caminho entre todos os pares desses N pontos da cidade somente utilizando essas $N-1$ avenidas. Para deixar essas avenidas mais bonitas para o festival, foram escolhidas M cores diferentes para pintar algumas dessas avenidas.

O processo de pintura das avenidas foi feito da seguinte maneira: foram escolhidos M pares dos N pontos da cidade, primeiro foi pintado com a cor 1 todas as avenidas no caminho entre o primeiro par de pontos, depois com a cor 2 todas as avenidas no caminho entre o segundo par de pontos, e assim por diante. Note que esse processo pode acabar trocando a cor de avenidas que já foram pintadas anteriormente, quando uma avenida é pintada novamente, a cor anterior é totalmente apagada e a avenida fica pintada somente com a nova cor. Inicialmente nenhuma avenida está pintada e ao final do processo de pintura, é possível que haja algumas avenida que não são pintadas.

Após todo esse processo de pintura, planeja-se também um desfile de encerramento desse festival. A organização do evento quer fazer um desfile começando em um ponto A e indo até um ponto B de maneira que todas avenidas no caminho de A para B estejam pintadas e com a mesma cor. Eles precisam de sua ajuda para saber qual o maior caminho possível para realização desse desfile. Além disso, há também uma lista de Q eventos que podem interferir nessa escolha do local do desfile.

Para a realização do desfile entre dois pontos A e B, além da condição de mesma cor em todas as avenidas, nenhum ponto intermediário que esteja no caminho entre A e B pode ter algum tipo de bloqueio que impeça a passagem do desfile. Inicialmente, todos os N pontos da cidade estão desbloqueados para a passagem do desfile. Porém há uma sequência de Q eventos onde um certo ponto P_i é bloqueado ou desbloqueado. Seu programa também deve responder após cada evento dessa sequência, a distância do novo maior caminho possível para o desfile.

Entrada

A primeira linha contém um inteiro N ($2 \leq N \leq 10^5$) indicando o número de pontos da cidade.

As próximas $N-1$ linhas contém três inteiros A_i, B_i e C_i indicando que há uma avenida entre os pontos A_i e B_i com distância C_i ($1 \leq A_i, B_i \leq N$, $A_i \neq B_i$, $1 \leq C_i \leq 10^4$). É garantindo que existe caminho e apenas um único caminho entre qualquer par desses N pontos.

A próxima linha contém um inteiro M ($1 \leq M \leq 2 \cdot 10^5$) indicando o número de cores diferentes utilizadas no processo de pintura.

As próximas M linhas contém dois inteiros D_i e E_i ($1 \leq D_i, E_i \leq N$, $D_i \neq E_i$) indicando que todas as avenidas no caminho entre D_i e E_i são pintadas com a cor i.

A próxima linha contém um inteiro Q ($1 \leq Q \leq 10^5$) indicando o número de eventos de bloqueio ou desbloqueio de pontos da cidade.

A próxima linha contém Q inteiros P_i ($1 \leq P_i \leq N$) indicando a sequência de eventos onde se troca o estado do ponto P_i, o ponto P_i é bloqueado caso ele esteja desbloqueado ou passa estar desbloqueado caso já esteja bloqueado. Inicialmente todos os pontos estão desbloqueados.

Saída

Seu programa deve produzir uma única linha com $Q+1$ inteiros. O primeiro inteiro deve indicar o tamanho do maior caminho possível para o desfile após todo o processo de pintura. Os próximos Q inteiros devem indicar o tamanho do maior caminho possível após cada um dos Q eventos em sequência.

Exemplo de entrada 1	**Exemplo de saída 1**
4 1 2 11 2 3 12 3 4 13 1 1 4 3 2 3 2	36 25 13 23

Exemplo de entrada 2	**Exemplo de saída 2**
8 1 2 11 2 3 12 5 2 13 4 2 14 5 6 15 5 7 16 5 8 100 3 1 7 1 6 4 3 5 6 5 2 7 5	39 39 26 16 16 28

Problema H

Horário de rodízio

Reinaldo é um motorista que trabalha na cidade de São Paulo começando sua árdua rotina todo dia às 5 da manhã. Ele precisa passar por N locais diferentes em ordem para acabar seu trabalho. Apesar do trânsito de São Paulo, Reinaldo sempre leva o mesmo tempo de T_1 minutos para sair de sua casa e chegar no primeiro local, leva T_2 minutos para ir do primeiro local até o segundo local, e assim por diante.

O problema é que durante um dia da semana, a cidade de São Paulo possui um sistema de rodízio de placas que não permite que Reinaldo esteja dirigindo entre 7 e 10 horas da manhã e também entre 5 horas da tarde e 8 horas da noite. Nesses horários em que Reinaldo não pode dirigir, ele para o carro onde estiver e continua dirigindo após o término do rodízio. Você consegue ajudar Reinaldo a descobrir quanto tempo leva para que ele termine sua rotina de trabalho em um dia com rodízio de placas?

Entrada

A primeira linha contém um inteiro N ($1 \leq N \leq 12$) e a segunda linha contém N inteiros T_i ($1 \leq T_i \leq 120$ minutos) que são os tempos de deslocamento entre os locais.

Saída

A saída deve ter uma linha com um inteiro indicando quantos minutos Reinaldo leva para acabar sua rotina de trabalho.

Exemplo de entrada 1	**Exemplo de saída 1**
3 60 30 30	120

Exemplo de entrada 2	**Exemplo de saída 2**
3 60 30 31	301

Exemplo de entrada 3	**Exemplo de saída 3**
5 70 60 30 40 30	410

Exemplo de entrada 4	**Exemplo de saída 4**
6 100 100 100 100 100 100	960

Explicação do exemplo 1: Os três percursos totalizam 2 horas (120 minutos), então Reinaldo começa as 5 da manhã e finaliza as 7 da manhã sem precisar parar por conta do rodízio.

Explicação do exemplo 2: Os três percursos totalizam 121 minutos, obrigando Reinaldo a parar entre as 7 e 10 horas da manhã, totalizando 301 minutos.

Problema I

Inserção de dígitos

Em uma unidade do Poupatempo de São Paulo, há um totem de auto atendimento que, dentre muitas outras informações, requer dos usuários que eles digitem seu número de celular. Infelizmente, o sistema não verifica se o número de celular digitado é válido, então os usuários podem digitar qualquer sequência de 1 a 20 dígitos e o sistema aceita e salva.

Para piorar a situação do cadastro de celular, funcionários do poupatempo perceberam que as teclas de dígitos mais usadas para digitar números de celular, que são 1 e 9, começaram a falhar. Quando um usuário aperta o 1 ou o 9, há uma chance de que dessas teclas não funcionem e de que o usuário salve o número sem algum desses dígitos.

O Poupatempo te contratou para desenvolver um programa que, dado um número de celular cadastrado para um usuário, imprima todas os possíveis números de celular do Estado de São Paulo válidos que podem ter sido digitados por aquele usuário, considerando que as teclas 1 e 9 podem ter falhado enquanto o usuário usava o totem.

Um número de celular do Estado de São Paulo é válido se possui 11 dígitos começando com dois dígitos que representem um dos DDDs de São Paulo, seguido de um dígito 9, seguidos de mais 8 dígitos quaisquer. A seguir está a lista de todos os DDDs válidos do estado de São Paulo:

- 11: São Paulo (capital), Grande São Paulo (Guarulhos, Osasco, ABC Paulista etc.) e região de Jundiaí e Mogi das Cruzes.
- 12: Vale do Paraíba, São José dos Campos, Taubaté, Guaratinguetá e litoral norte.
- 13: Baixada Santista, Santos, São Vicente, Guarujá, Praia Grande e litoral sul.
- 14: Bauru, Marília, Botucatu, Ourinhos e Jaú.
- 15: Sorocaba, Itapetininga e região.
- 16: Ribeirão Preto, Franca, Araraquara e São Carlos.
- 17: São José do Rio Preto, Catanduva e Votuporanga.
- 18: Presidente Prudente, Assis, Dracena e Araçatuba.
- 19: Campinas, Piracicaba, Limeira, Americana e Rio Claro.

Por exemplo, se o número cadastrado foi "`11983609574`", o único número válido que a pessoa pode ter digitado é o próprio número salvo. Porém, se o número cadastrado foi "`1922222222`", há dois números válidos que a pessoa pode ter digitado corretamente, ou "`19922222222`" se o terceiro digito falhou, ou "`11922222222`" se o segundo digito falhou.

Entrada

A entrada consiste de apenas uma linha com uma sequência de 1 a 20 dígitos de 0 a 9.

Saída

A primeira linha deve ter o número N de números válidos possíveis. As N linhas seguintes devem conter um número válidos possível cada. Os números devem ser impressos do menor para o maior.

Exemplo de entrada 1	**Exemplo de saída 1**
1922222222	2 11922222222 19922222222

Exemplo de entrada 2	**Exemplo de saída 2**
10913574286	0

Exemplo de entrada 3	**Exemplo de saída 3**
23456789	17 11923456789 12913456789 12931456789 12934156789 12934516789 12934561789 12934567189 12934567819 12934567891 12934567899 12934567989 12934569789 12934596789 12934956789 12939456789 12993456789 19923456789

Problema J

Jogando com intervalos

Ana e Beto criaram um jogo utilizando intervalos de números inteiros. Eles começam o jogo com um intervalo de A até B ($A < B$) e em cada jogada eles devem escolher entre somar um ao valor de A ou subtrair um do valor de B. Ana começa o jogo e eles jogam em turnos alternados, a segunda jogada é de Beto, a terceira jogada é de Ana e assim por diante.

Há também N intervalos especiais de números inteiros que influenciam nesse jogo. O objetivo de Beto é ganhar o maior número possível de pontos fazendo com que o intervalo $[A, B]$ fique igual há um desses N intervalos após sua jogada, nesse caso o jogo acaba e Beto ganha um número de pontos igual ao tamanho do intervalo, ou seja, $B - A$ pontos. Por outro lado, o objetivo de Ana é fazer com que Beto ganhe o menor número possível de pontos. Se o valor de A ficar igual ao valor de B após a jogada de qualquer um dos jogadores, o jogo acaba e Beto ganha 0 pontos. Note que se $[A, B]$ ficar igual a um dos intervalos especiais após a jogada de Ana ou se $[A, B]$ for inicialmente já igual a um dos intervalos especiais, o jogo segue normalmente.

Por exemplo, se temos apenas o intervalo $[1, 2]$ como o único intervalo especial e o jogo começa com $[A, B] = [0, 3]$, Ana pode fazer $[A, B]$ virar $[1, 3]$ ou $[0, 2]$ na primeira jogada, porém em ambos os casos, Beto pode ganhar 1 ponto fazendo com que $[A, B]$ fique igual a $[1, 2]$ na segunda jogada. Se $[A, B]$ for inicialmente igual $[0, 2]$, $[1, 3]$ ou $[1, 2]$, em todos esses casos o jogo acaba com Beto ganhando 0 pontos.

Considerando que Ana e Beto jogam de maneira ótima, eles querem sua ajuda para responder a seguinte pergunta: dado um inteiro positivo K, se sortearmos um intervalo inicial $[A, B]$ para o jogo tal que $0 \leq A < B \leq K$, qual é o valor esperado de pontos que Beto ganha? Em outras palavras, calcule a média aritmética de todos os resultados possíveis.

Entrada

A primeira linha contém dois inteiros N e K ($1 \leq N \leq 10^5$, $1 \leq K \leq 10^6$). As próximas N linhas contém dois inteiros X_i e Y_i para indicar cada um dos N intervalos distintos especiais $[X_i, Y_i]$ ($0 \leq X_i < Y_i \leq K$).

Saída

Seu programa deve produzir uma única linha com dois inteiros não negativos primos entre si P e Q tal que $\frac{P}{Q}$ seja igual ao valor esperado de pontos que Beto ganha no jogo quando se sorteia um intervalo inicial $[A, B]$ para o jogo tal que $0 \leq A < B \leq K$ (todos os intervalos possíveis dentro dessas condições possuem mesma probabilidade de serem escolhidos).

Exemplo de entrada 1	**Exemplo de saída 1**
1 3 1 2	1 6

Exemplo de entrada 2	**Exemplo de saída 2**
3 10 4 8 2 8 7 9	24 55

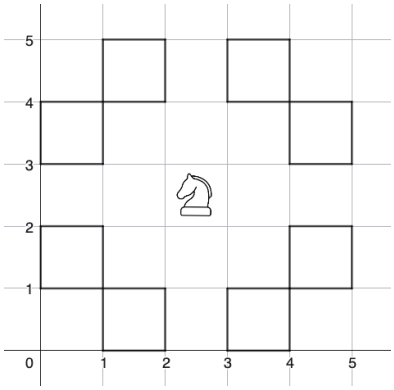
Explicação do exemplo 1: Há 6 possibilidades de início: $[0, 1]$, $[0, 2]$, $[0, 3]$, $[1, 2]$, $[1, 3]$ e $[2, 3]$. Apenas começando com $[0, 3]$ Beto ganha 1 ponto, portanto a resposta é $\frac{1}{6}$.

Problema K

Knight number

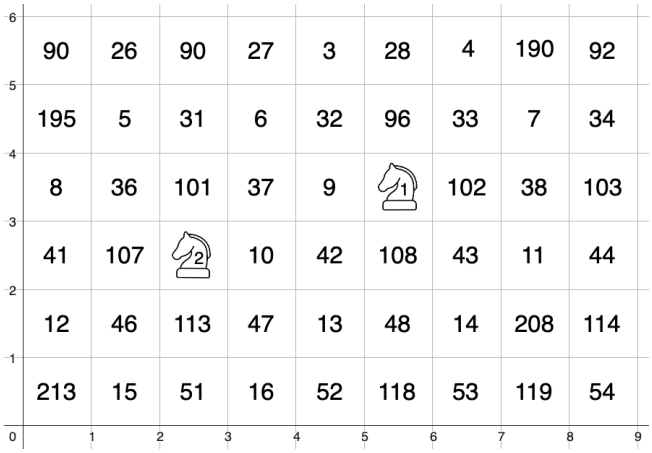
Marquinhos Carlsen é um grande aficionado pelo jogo de xadrez. Apesar do xadrez ser jogado em um tabuleiro quadriculado 8 por 8, Marquinhos gosta de fazer análises matemáticas imaginando um tabuleiro infinito de xadrez colocado em um plano cartesiano. Sua peça de xadrez favorita é o cavalo (*knight* em inglês), pois ela tem um padrão de movimento muito interessante. Imaginando-se um tabuleiro infinito, o cavalo sempre vai possuir 8 movimentos possíveis em formato de "L", a cada jogada o cavalo deve se mover 2 casas para alguma direção e mais 1 casa pra outra direção perpendicular.

Na imagem abaixo, é possível ver que se um cavalo está na posição $(2, 2)$, ele pode-se mover para as posições $(1, 4)$, $(3, 4)$, $(0, 3)$, $(4, 3)$, $(0, 1)$, $(4, 1)$, $(1, 0)$ e $(3, 0)$. As casas são representadas por seu canto inferior esquerdo no plano cartesiano.



Em uma de suas análises matemáticas, Marquinhos posicionou N cavalos nesse plano com tabuleiro infinito e definiu a "distância de cavalo"de cada posição do tabuleiro como o menor número de jogadas que algum dos cavalos consegue alcançar a posição. Com isso, ele também definiu o "knight number"de todas as posições, que é um número inteiro positivo único para cada posição que é o resultado da seguinte ordenação de todas as posições do tabuleiro: primeiro se ordena as posições em ordem crescente de suas distâncias de cavalo, em caso de empate ordena-se em ordem decrescente do valor no eixo vertical, e ainda em caso de empate ordena-se em ordem crescente da posição no eixo horizontal.

Na imagem abaixo está um exemplo onde se posiciona dois cavalos nas posições $(2, 2)$ e $(5, 3)$. Em cada posição visível na imagem está escrito o valor de seu "knight number", porém note que há infinitas posições fora da imagem também que não são mostradas, incluindo posições com coordenadas negativas.



Marquinhos quer sua ajuda para analisar o "knight number". Ele tem uma lista de Q números inteiros positivos e quer saber para cada um deles qual posição do tabuleiro infinito tem o "knight number"de mesmo valor.

Entrada

A primeira linha contém um inteiro N ($1 \leq N \leq 10$) indicando o número de cavalos.

As próximas N linhas contém dois inteiros X_i e Y_i ($0 \leq X_i, Y_i \leq 10^9$) indicando a posição dos N cavalos no tabuleiro. As posições são indicadas pelo ponto inferior esquerdo assim como nas imagens e todos os cavalos estão em posições distintas.

A próxima linha contém um inteiro Q ($1 \leq Q \leq 10$).

A próxima linha contém Q inteiros K_i ($1 \leq K_i \leq 10^9$) indicando para quais "knight numbers"deve se descobrir suas posições no tabuleiro.

Saída

Seu programa deve imprimir Q linhas, cada uma delas com dois inteiros A_i e B_i de tal maneira que o "knight number"da posição (A_i, B_i) seja igual a K_i.

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

Exemplo de entrada 2	**Exemplo de saída 2**
2 2 2 5 3 7 213 190 7 25 1000 10000 100000	0 0 7 5 7 4 -1 5 -1 18 -51 14 121 138

Problema L

Lado esquerdo ou direito

Os organizadores da Maratona Paulista de Programação decidiram criar um sistema especial para controlar a entrada das equipes na área de competição. Há N equipes inscritas na competição, cada uma com um identificador único entre 0 e $N-1$.

Para organizar o fluxo de participantes, é construída uma rede de postos de controle. Cada posto possui um número distinto v entre 0 e $N-1$ e conta, possivelmente, com um corredor à esquerda e, possivelmente, com um corredor à direita, podendo assim ter 0, 1 ou 2 corredores no total.

A estrutura dos postos deve satisfazer a seguinte propriedade: para cada posto com número v, todos os postos alcançáveis a partir do corredor da esquerda possuem números menores que v, enquanto todos os postos alcançáveis a partir do corredor da direita possuem números maiores que v.

O objetivo de cada equipe é chegar ao posto de controle cujo número coincide com seu identificador. Quando uma equipe chega a um posto com número v, seu identificador id é comparado com ele:

- se $id < v$, a equipe segue pelo corredor da esquerda;
- se $id > v$, a equipe segue pelo corredor da direita;
- se $id = v$, a equipe permanece no posto.

Cada corredor leva a outro posto de controle, onde o mesmo processo é repetido. Dessa forma, uma equipe pode atravessar vários postos antes de chegar ao seu destino final.

Durante o dia, as equipes chegam na ordem definida por uma sequência P. Para analisar a eficiência de um posto específico u, são observadas apenas as equipes que passam por ele **sem permanecer nele**. Para cada uma dessas equipes:

- registra-se `E` se ela foi enviada para o corredor da esquerda de u;
- registra-se `D` se ela foi enviada para o corredor da direita de u.

Isso gera uma sequência S formada pelos caracteres `E` e `D`. A contribuição de um posto é a quantidade de pares de posições adjacentes cujos caracteres são diferentes. Em outras palavras, a quantidade de posições i tais que $S_i \neq S_{i+1}$. A pontuação total do sistema é a soma das contribuições de todos os postos.

Rouse conhece a sequência P e deseja determinar a máxima pontuação total possível ao escolher a melhor estrutura de postos de controle. Você pode ajudá-la?

Entrada

A primeira linha contém um inteiro N ($1 \leq N \leq 500$), o número de equipes. A segunda linha contém N inteiros diferentes $P_1, \ldots, P_N$ ($0 \leq P_i < N$), representando a ordem de chegada das equipes.

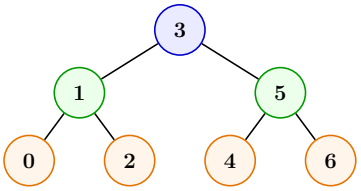
Saída

Seu programa deve produzir uma única linha com a máxima pontuação total possível entre todas as estruturas de postos de controle válidas.

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

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

Explicação do exemplo 1:
Rouse escolhe a seguinte estrutura de postos de controle.

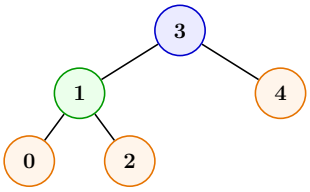


As contribuições dos postos são as seguintes:

- No posto 1, as equipes de identificadores 0 e 2 passam por ele, gerando a sequência `ED`. Como há uma mudança entre caracteres consecutivos, sua contribuição é 1.
- No posto 3, as equipes de identificadores 0, 1, 2, 4, 5 e 6 passam por ele, gerando a sequência `EDEDED`. Como há 5 mudanças entre caracteres consecutivos, sua contribuição é 5.
- No posto 5, as equipes de identificadores 4 e 6 passam por ele, gerando a sequência `ED`. Como há uma mudança entre caracteres consecutivos, sua contribuição é 1.

Os outros postos têm contribuição 0, pois nenhuma equipe passa por eles. Portanto, a máxima soma total das contribuições é 7.

Explicação do exemplo 2:
Rouse escolhe a seguinte estrutura de postos de controle.



As contribuições dos postos são as seguintes:

- No posto 1, as equipes de identificadores 0 e 2 passam por ele, gerando a sequência `DE`. Como há uma mudança entre caracteres consecutivos, sua contribuição é 1.
- No posto 3, as equipes de identificadores 0, 1, 2 e 4 passam por ele, gerando a sequência `DEEE`. Como há apenas uma mudança entre caracteres consecutivos, sua contribuição é 1.

Os outros postos têm contribuição 0, pois nenhuma equipe passa por eles. Portanto, a máxima soma total das contribuições é 2.

Problema M

Multipalavra

Durante uma longa viagem de ônibus por São Paulo, Miguel gosta de observar as placas das ruas pela janela. O trajeto parece não acabar nunca: o ônibus cruza avenidas compridas como a Avenida Sapopemba, a Avenida Aricanduva e a Avenida Raimundo Pereira de Magalhães, enquanto Miguel tenta se distrair com os nomes que aparecem pelo caminho.

Em uma parte do trajeto, ao passar por uma região da cidade, ele vê a mesma sequência de letras s repetida p vezes. Mais tarde, já depois de muitas paradas, passa a ver a sequência t repetida q vezes.

Na volta para casa, porém, o caminho é diferente: primeiro aparecem as placas com t repetida q vezes, e só depois as placas com s repetida p vezes. Miguel decidiu comparar mentalmente as duas gigantescas sequências de letras usando ordem lexicográfica, mas elas são grandes demais para serem escritas completamente.

Dada uma string u e um inteiro não negativo k, seja u^k a string formada pela concatenação de k cópias de u. Em particular, u^0 é a string vazia.

Lembre que, em ordem lexicográfica, uma string a é menor que uma string b se, na primeira posição em que elas diferem, o caractere de a é menor que o caractere de b. Se uma das strings termina antes dessa primeira diferença existir, então a string mais curta é considerada menor.

Sua tarefa é comparar lexicograficamente as strings $s^p t^q$ e $t^q s^p$.

Entrada

A primeira linha da entrada contém as strings s e t. A segunda linha contém os inteiros p e q. As strings s e t são não vazias, contêm apenas letras minúsculas do alfabeto inglês, e satisfazem $1 \leq |s|, |t| \leq 200\,000$. Os inteiros satisfazem $0 \leq p, q \leq 10^9$.

Saída

A saída deve conter uma única linha com um caractere: "=", se $s^p t^q$ e $t^q s^p$ são iguais; "<", se $s^p t^q$ é lexicograficamente menor que $t^q s^p$; ou ">", caso contrário.

Exemplo de entrada 1	**Exemplo de saída 1**
`ab aba` `1 1`	`>`

Exemplo de entrada 2	**Exemplo de saída 2**
`a b` `2 3`	`<`

Exemplo de entrada 3	**Exemplo de saída 3**
`abc abcabc` `3 2`	`=`

Problema N

Níveis de reatividade

Um laboratório de pesquisa desenvolveu o misterioso *Composto S*, uma substância instável capaz de alterar a estrutura de amostras biológicas. O laboratório possui um lote original A contendo N amostras, cada uma com um nível de reatividade representado por um número inteiro não-negativo.

Antes do experimento, os cientistas registraram duas métricas cruciais de segurança:

- S: A reatividade total do lote (a soma dos níveis de reatividade de todas as amostras em A).
- M: O nível máximo de reatividade presente em uma única amostra de A.

Durante o experimento, o lote inteiro foi exposto ao *Composto S*. A reação em cadeia fez com que cada amostra sofresse uma mutação: o seu novo nível de reatividade passou a ser a soma dos níveis de reatividade de todas as **outras** amostras do lote original.

Para tentar estabilizar a reação, os cientistas injetaram duas amostras de controle no lote, possuindo exatamente os níveis de reatividade globais S e M. Devido à volatilidade do processo, as $N+2$ amostras resultantes foram completamente misturadas, formando o lote caótico B.

Sua tarefa é, dado o lote misturado B, descobrir os valores originais de S e M, e reconstruir os níveis de reatividade das amostras do lote original A, imprimindo-os em ordem não-decrescente.

Entrada

A primeira linha contém um inteiro N ($1 \leq N \leq 10^5$), representando o tamanho do array original A.

A segunda linha contém $N + 2$ inteiros embaralhados B_i ($0 \leq B_i \leq 10^{14}$), representando os elementos do array modificado e injetado.

Saída

A saída deve conter duas linhas. A primeira linha deve conter dois inteiros: os valores de S e M. A segunda linha deve conter os N inteiros do array original A ordenados de forma não-decrescente, separados por espaços.

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

Exemplo de entrada 2	**Exemplo de saída 2**
4 0 0 0 0 0 0	0 0 0 0 0 0