SAET 2022 - Maratona de Programação

20 de Outubro de 2022

Instruções Importantes

- Use a opção **Runs** para enviar suas soluções. Os problemas podem resolvidos em qualquer ordem e linguagem (dentre C, C++ e Python, independentemente do problema);
- Suas soluções serão testadas com várias entradas, além das dada como exemplo. Por isso, sua solução pode não ser aceita mesmo se funcionar para os exemplos dados. Certifique-se que ela funciona para todas as entradas possíveis;
- A saída gerada deve ser *exatamente* conforme especificada. Em particular, **não** imprima instruções ("digite um número", "a resposta é", etc);
- É garantido que todas as entradas usadas para teste estarão de acordo com o enunciado, não sendo necessário testar se são válidas;
- Ao enviar uma solução, o sistema irá responder uma das seguintes respostas:
  - `Not answered yet`: a solução está sendo corrigida. Aguarde um pouco e atualize a página;
  - `YES`: solução aceita. Parabéns!
  - `Wrong Answer`: a saída impressa pelo seu programa não é a saída correta esperada, para alguma entrada de teste;
  - `Presentation Error`: a saída impressa está correta, exceto por espaços em branco e/ou quebras-de-linha faltando/sobrando;
  - `Time Limit Exceeded`: o tempo de execução do seu programa ultrapassou o tempo limite estipulado para o problema (ver tabela abaixo). O tempo de execução da sua solução precisa ser menor;
  - `Runtime Error`: seu programa gerou algum erro em tempo de execução ("crashou");
  - `Compile Error`: seu programa não compila.
- Todas as linhas, tanto na entrada quanto na saída, terminam com o caractere de fim-de-linha ($\backslash n$), mesmo quando houver apenas uma única linha na entrada e/ou saída;
- Sua solução deve processar cada arquivo de entrada no tempo máximo estipulado para cada problema, dado pela seguinte tabela:

Problema	**Nome**	**Tempo Limite (segundos)**
A	Funeral da Rainha	1
B	Herdeiros da Rainha	1
C	Rage Against the JVM	4
D	Quem não tem BIT caça com Seg	1
E	Logomarca	1
F	Resultado da Eleição	1
G	Álbum da Copa	1
H	Números Jares	1
I	Números Mares	1
J	Pesos na Barra	1

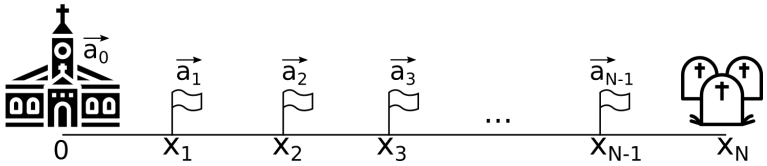
A: Funeral da Rainha

`Arquivo: funeral.[c|cpp|py]`

A rainha morreu.

Por ela ter sido uma pessoa muito relevante para a história de seu país, o funeral da rainha será aberto ao público e televisionado ao vivo para toda a nação. Uma procissão aberta acontecerá durante o funeral, levando o caixão com o corpo da monarca. A procissão sairá da Igreja Imperial (onde o corpo será velado) e irá, em linha reta, até o Cemitério Real (onde o corpo será enterrado).

Preocupada com a segurança do evento, a organização da procissão decidiu que a mesma não andará em velocidade constante durante todo o trajeto. Assim, a organização demarcou N pontos no trajeto, sendo o primeiro a x_1 metros da Igreja; o segundo a x_2 metros da Igreja; etc., até o último ponto, a x_N metros da Igreja, onde fica o Cemitério.



Então, a procissão sairá da Igreja, com velocidade inicial 0 (zero) e aceleração constante a_0 m/s^2, até o ponto x_1. Neste ponto, a aceleração é alterada para a_1 m/s^2, com a qual a procissão vai até o ponto x_2. Neste ponto, a aceleração é alterada para a_2 m/s^2 e a procissão segue até o ponto x_3; e assim por diante, até que a procissão chegue ao Cemitério.

Para ajudar a organização com o restante do funeral, sua tarefa é determinar em que horário a procissão chegará no Cemitério Real.

Entrada

A primeira linha contém dois inteiros N e a_0 ($1 \leq N \leq 10^4$, $0 < a_0 \leq 100$). As próximas $N-1$ linhas contém, cada uma, os inteiros x_i e a_i ($0 < x_i \leq 10^9$, $-100 \leq a_i \leq 100$). A última linha contém o inteiro x_N ($0 < x_N \leq 10^9$).

Os valores de x_i são dados em ordem estritamente crescente, iniciando por x_1. É garantido que a velocidade da procissão é sempre positiva durante todo o trajeto (inclusive em todos os trechos em que a aceleração é negativa), e que é igual a zero apenas no ponto inicial (na Igreja).

Saída

Imprima uma linha com um valor T, indicando que a procissão chegará no Cemitério Real T segundos após a saída da Igreja Imperial. Arredonde e imprima a resposta com duas casas decimais.

Exemplo de entrada	**Exemplo de saída**
1 3 100	8.16

Exemplo de entrada	**Exemplo de saída**
2 3 50 5 100	7.97

Exemplo de entrada	**Exemplo de saída**
5 10 60 5 120 -5 180 0 200 -6 295	11.64

B: Herdeiros da Rainha

`Arquivo: herdeiros.[c|cpp|py]`

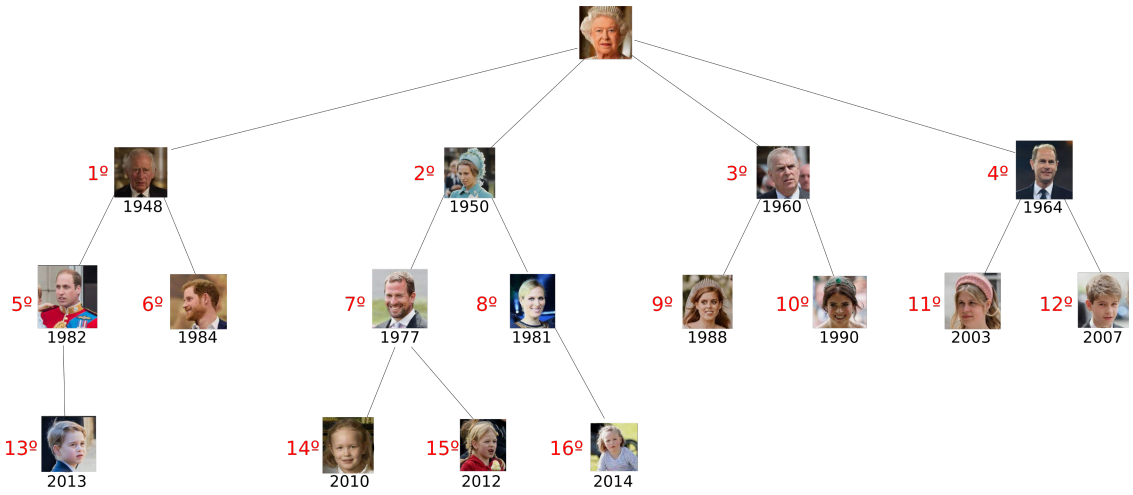
A rainha morreu.

Imediatamente após o falecimento da monarca, o povo começou a especular não apenas quem será o próximo rei, mas também quem sentará no trono depois dele, e depois dele, e depois dele, etc. Em pouco tempo, saber toda a linha de sucessão ao trono se tornou importantíssimo para a população.

Pelas regras da monarquia, o primeiro herdeiro ao trono é o filho mais velho da rainha. Em seguida, vem os demais filhos da rainha, em ordem decrescente de idade.

Após todos os filhos diretos da rainha, vem os filhos diretos do filho mais velho dela. Em seguida, vem os filhos diretos do segundo filho mais velho da rainha, e assim por diante.

Como exemplo, a figura abaixo apresenta uma possível árvore genealógica da família real, indicando o ano de nascimento de cada herdeiro da rainha, e sua ordem na linha de sucessão ao trono:



Dada a árvore genealógica da família real e o ano de nascimento de cada herdeiro, determine toda a linha de sucessão ao trono.

Entrada

A primeira linha contém um inteiro N ($1 \leq N \leq 10^5$), o número de herdeiros ao trono. Considere que os herdeiros são numerados de 1 a N. As próximas N linhas descrevem um herdeiro cada. A linha i ($1 \leq i \leq N$) contém dois inteiros P_i e D_i ($0 \leq P_i \leq N$, $0 \leq D_i \leq 10^5$), onde P_i é o identificador do pai do herdeiro i (ou 0 caso o herdeiro i seja filho direto da rainha), e D_i é o ano de nascimento do herdeiro i.

Não haverá dois herdeiros que nasceram no mesmo ano.

Saída

Imprima N linhas com os identificadores dos herdeiros, um por linha, na ordem da linha de sucessão ao trono.

Exemplo de entrada	**Exemplo de saída**
5 0 1961 0 1933 0 1957 0 2022 0 1970	2 3 1 5 4

Exemplo de entrada	**Exemplo de saída**
16 2 1990 0 1960 5 1977 13 2014 0 1950 0 1964 9 1984 6 2007 0 1948 15 2013 3 2012 6 2003 5 1981 2 1988 9 1982 3 2010	9 5 2 6 15 7 3 13 14 1 12 8 10 16 11 4

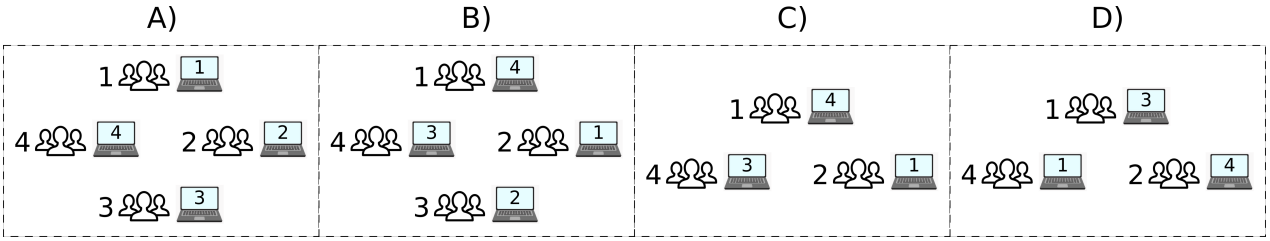
C: Rage Against the JVM

`Arquivo: rage.[c|cpp|py]`

Depois de uma viagem para lá de conturbada, finalmente as equipes chegaram no local da Maratona de Programação deste ano. Ao chegar, se depararam com um problema: as máquinas, rodando nas problemáticas JVMs (*Java Virtual Machine*), estão travando demais.

Inicialmente, as equipes estão posicionadas em círculo, da equipe 1 à equipe N, em sentido horário, e a equipe i está utilizando a máquina i. Quando acontece algum problema com as JVMs, a organização reinicia as máquinas e faz um rodízio, no qual a máquina que cada equipe está usando passa para a equipe que está a sua esquerda no círculo.

Cansadas, algumas equipes estão desistindo da competição e indo embora. Quando uma equipe desiste, a máquina que ela estava utilizando é descartada. Como exemplo, a figura abaixo apresenta: A) a posição inicial de $N = 4$ equipes e suas máquinas; B) o rodízio de máquinas após um problema; C) a equipe 3 desistiu; D) o rodízio após outro problema.



Dada a sequência de problemas e de desistências que ocorreram, determine quais equipes sobraram e com quais máquinas elas ficaram ao final da prova.

Entrada

A primeira linha contém dois inteiros N e M ($1 \leq N, M \leq 10^5$), o número de equipes e de acontecimentos, respectivamente. A próxima linha contém M inteiros A_i indicando os acontecimentos, na ordem em que aconteceram. $A_i = 0$ indica um problema ocorrido e um rodizio feito, enquanto $1 \leq A_i \leq N$ indica a desistência da equipe A_i. Sobrará no mínimo uma equipe ao final da competição, e nenhuma equipe desiste mais de uma vez.

Saída

Para cada equipe que sobrou, imprima uma linha com i c , indicando que a equipe i ficou com a máquina c no final da prova. Imprima em ordem crescente dos números das equipes.

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

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

D: Quem não tem BIT caça com Seg

`Arquivo: bit.[c|cpp|py]`

Enquanto preparava suas anotações para levar para a Maratona de Programação, Douglas percebeu um fato interessante: algumas estruturas de dados fazem tudo o que outras estruturas fazem e mais um pouco, e logo poderiam substitui-las sem prejuizo. É o caso, por exemplo, da BIT (*Binary Indexed Tree*) e da Seg (*Segment Tree*). A Seg tem todas as funcionalidades que a BIT tem (além das funcionalidades que só a Seg possui). Por isso, conhecer a BIT não é *realmente* necessário quando se já conhece a Seg[1], pois todo problema que pode ser resolvido com a BIT pode ser resolvido com a Seg em seu lugar. Neste caso, dizemos que a Seg *pode substituir* a BIT.

Após estudar todas as N estruturas de dados que existem, Douglas descobriu quais estruturas podem substituir quais outras. Agora Douglas está curioso: qual é a estrutura de dados que pode substituir (direta ou indiretamente) a maior quantidade de outras estruturas?

Entrada

A primeira linha contém dois inteiros N e M ($1 \leq N \leq 2000, 0 \leq M \leq min(\frac{N^2-N}{2}, 2000)$), o número de estruturas e de substituições, respectivamente. As estruturas são numeradas de 1 a N. As próximas M linhas contém dois inteiros A e B cada ($1 \leq A, B \leq N, A \neq B$), indicando que a estrutura A pode substituir a estrutura B.

É garantido que a relação de substituições não contém ciclos.

Saída

Imprima uma linha com dois inteiros E e S, onde E é a estrutura de dados que pode substituir (direta ou indiretamente) a maior quantidade de outras estruturas, enquanto S é a quantidade de outras estruturas que E pode substituir (direta ou indiretamente).

Se houver mais de uma estrutura que pode substituir a maior quantidade de estruturas, imprima a de menor número identificador.

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

[1] embora é recomendado, pois o código da BIT é mais compacto que o da Seg, tornando-a mais prática.

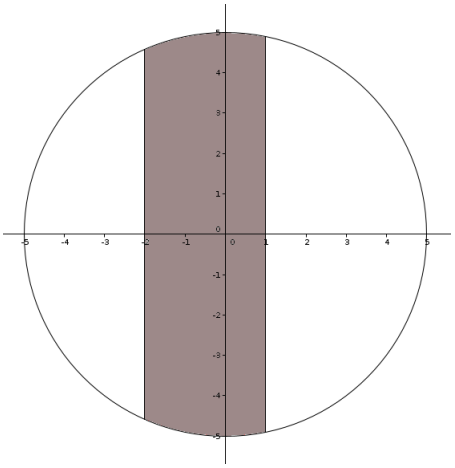
E: Logomarca

`Arquivo: logomarca.[c|cpp|py]`

Rodriguinho está muito animado com a empresa de serviços em TI que está abrindo. Após já ter decidido o nome da empresa e o endereço onde o escritório será instalado, Rodriguinho está agora definindo uma logomarca para a empresa.

Minimalista que é, ele decidiu que a logo consistirá apenas de um círculo cortado por duas retas verticais, com a área entre elas pintadas de alguma cor (a cor vai mudar a cada mês).

Para simplificar, considere que o círculo tem centro na origem do plano cartesiano e tem raio r, e que as retas verticais cortam o eixo X nos pontos x_1 e x_2. Como exemplo, a figura abaixo apresenta uma logo com $r = 5, x_1 = -2, x_2 = 1$:



Rodriguinho está preocupado com a quantidade de tinta que terá de comprar todos os meses. Por isso, ele pediu sua ajuda para determinar a área da região pintada na sua logomarca.

Entrada

A única linha da entrada contém três inteiros r, x_1 e x_2 ($1 \leq r \leq 50$, $-r \leq x_1 < x_2 \leq r$).

Saída

Imprima uma linha contendo a área da região pintada, arredondada com três casas decimais.

Exemplo de entrada	**Exemplo de saída**
`5 -2 1`	`29.386`

Exemplo de entrada	**Exemplo de saída**
`5 -4 -2`	`15.729`

Exemplo de entrada	**Exemplo de saída**
`16 11 12`	`22.233`

F: Resultado da Eleição

`Arquivo: resultado.[c|cpp|py]`

Este ano temos eleições gerais para escolher o novo chefe de governo da Nlogonia.

Cada um dos N eleitores do país tem três opções de como votar no dia da eleição: votar em um dos M candidatos que estão concorrendo ao cargo; votar em branco; ou anular seu voto.

Um *voto válido* é um voto dado em algum dos M candidatos (isto é, um voto que não é nem em branco nem nulo). Se algum candidato receber mais que 50% de todos os votos válidos, ele vence a eleição. Caso contrário, os dois candidatos que mais receberem votos válidos irão disputar uma nova eleição, em segundo turno.

Chegou a hora de totalizar os votos! Dados os votos registrados nas urnas, algum candidato venceu a eleição? Ou haverá segundo turno?

Entrada

A primeira linha contém dois inteiros N e M ($1 \leq N, M \leq 10^5$), o número de eleitores e de candidatos, respectivamente. Os candidatos são numerados de 1 a M. A próxima linha contém N votos. Cada voto pode ser um número de um cadidato, a letra `B` (voto em branco) ou a letra `N` (voto nulo).

É garantido que a entrada contém no mínimo 1 voto válido, e que, se houver segundo turno, não haverá empate nem na primeira nem na segunda colocação.

Saída

Se não houver segundo turno, imprima uma linha contendo `vencedor` C, sendo C o número do candidato vencedor. Caso contrário, imprima uma linha contendo `segundo turno entre` A `e` B, onde A e B são os números do candidato mais votado e o segundo mais votado, respectivamente.

Exemplo de entrada	**Exemplo de saída**
`7 100` `92 92 B 7 92 7 N`	`vencedor 92`

Exemplo de entrada	**Exemplo de saída**
`11 100` `B 95 N 95 95 99 99 B 95 98 99`	`segundo turno entre 95 e 99`

Exemplo de entrada	**Exemplo de saída**
`10 10` `9 B 5 N 5 9 B 5 9 5`	`vencedor 5`

G: Álbum da Copa

`Arquivo: album.[c|cpp|py]`

No mês que vem começa a grande Copa do Mundo de Dados Estruturados de 2022, que este ano acontece na grande cidade de Touledow. Para motivar as pessoas a acompanherem o torneio[2], diversas ações de publicidade e *merchandising* estão sendo feitas durante o ano todo.

Em uma dessas ações, a organização lançou o Álbum de Figurinhas Oficial da Copa®. O álbum é composto por N figurinhas distintas, numeradas de 1 a N. Para completar o álbum, é necessário obter pelo menos uma cópia de cada uma dessas N figurinhas.

Ana tem uma coleção de figurinhas que pode não conter todas as figurinhas de 1 a N, e pode conter algumas figurinhas repetidas. Beto também tem uma coleção de figurinhas com as mesmas condições.

Para poderem completar seus álbuns, Ana e Beto irão trocar figurinhas. Em uma troca, Ana dá uma figurinha para Beto e, ao mesmo tempo, Beto dá uma figurinha para Ana. Note que apenas *trocas* são permitidas – uma pessoa não irá dar uma figurinha para outra sem receber uma outra figurinha em troca.

Determine o menor número de trocas necessário para que ambos completem seus álbuns, ou indique que isso não é possível.

Entrada

A primeira linha contém o inteiro N ($1 \leq N \leq 1000$).

A próxima linha contém N_A ($1 \leq N_A \leq 1000$), o número de figurinhas de Ana.

A próxima linha contém N_A inteiros de 1 a N, indicando quais figurinhas Ana tem.

A próxima linha contém N_B ($1 \leq N_B \leq 1000$), o número de figurinhas de Beto.

A próxima linha contém N_B inteiros de 1 a N, indicando quais figurinhas Beto tem.

Saída

Imprima uma única linha contendo o número mínimo de trocas necessárias, ou `impossivel` se não for possível que ambos completem seus álbuns trocando figurinhas.

[2] o que os jovens de hoje chamam de "hypar" o evento

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

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

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

Exemplo de entrada	Exemplo de saída
10 13 9 1 7 1 3 3 5 3 5 9 7 7 9 12 6 2 4 2 10 4 8 4 6 8 6 10	5

H: Números Jares

`Arquivo: jares.[c|cpp|py]`

Sempre que o prof. Racirdinho fala sobre números pares e números ímpares durante as aulas de Teoria da Poncutação, uma dúvida geral surge: "o número 0 (zero) é par?".

Embora o professor insista que zero seja sim um número par, Joãozinho ainda está desconfiado desta informação, e prefere não tomar isso como verdade. Para não gerar conflitos, Joãozinho decidiu criar sua própria definição de números pares.

Um número é *jar* (abreviação para "Joãozinho-par") se é positivo, é múltiplo de dois, e não termina com zero em sua representação decimal. Desta forma, os primeiros números jares são 2, 4, 6, 8, 12, 14, 16, 18, 22, ... Dado um número inteiro, determine se ele é um número jar.

Entrada

A única linha da entrada contém um inteiro N ($-10^9 \leq N \leq 10^9$).

Saída

Imprima uma única linha contendo `sim` se N é um número jar, ou `nao` caso contrário.

Exemplo de entrada	**Exemplo de saída**
`0`	`nao`

Exemplo de entrada	**Exemplo de saída**
`42`	`sim`

Exemplo de entrada	**Exemplo de saída**
`100`	`nao`

Exemplo de entrada	**Exemplo de saída**
`102`	`sim`

Exemplo de entrada	**Exemplo de saída**
`642487`	`nao`

Exemplo de entrada	**Exemplo de saída**
`-2`	`nao`

I: Números Mares

`Arquivo: mares.[c|cpp|py]`

Sempre que o prof. Racirdinho fala sobre números pares e números ímpares durante as aulas de Teoria da Poncutação, uma dúvida geral surge: "o número 0 (zero) é par?".

Embora o professor insista que zero seja sim um número par, Mariazinha ainda está desconfiada desta informação, e prefere não tomar isso como verdade. Para não gerar conflitos, Mariazinha decidiu criar sua própria definição de números pares.

Um número é *mar* (abreviação para "Mariazinha-par") se é positivo, é múltiplo de dois, e não contém nenhum 0 (zero) em sua representação decimal. Como exemplo, os números 2, 42, e 796 são mares, enquanto 0, 20, 402 e 7096 não são mares.

Os primeiros números mares são, nesta ordem, 2, 4, 6, 8, 12, 14, 16, 18, 22, 24, etc. Sua tarefa é, para várias consultas, determinar, dado um inteiro N, qual é o $N-$ésimo número mar.

Entrada

A primeira linha contém um inteiro Q ($1 \leq Q \leq 100$), o número de consultas. As próximas Q linhas descrevem uma consulta cada. Cada linha contém um inteiro N ($1 \leq N \leq 10^{15}$).

Saída

Para cada consulta, imprima uma linha contendo o $N-$ésimo número mar.

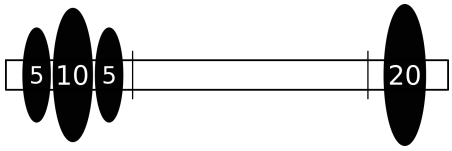
Exemplo de entrada	**Exemplo de saída**
6 1 2 10 100 38 1000	2 4 24 268 94 2968

J: Pesos na Barra

`Arquivo: pesos.[c|cpp|py]`

"Em pleno 2022; ano da tecnologia; ano da eleição; ano da copa; ano em que empresários bilionários estão construindo foguetes; neste ano, é necessário malhar bastante na academia". Com essa filosofia em mente, Gustavo vai à academia treinar todos os dias depois do trabalho.

Diversos exercícios da academia envolvem encaixar anilhas (pesos em forma de disco) nas duas pontas de uma barra de ferro, para então erguer a barra em uma posição que favorece determinado músculo. Para que o exercício seja executado corretamente, é necessário que o peso total em uma ponta da barra seja igual ao peso total na outra ponta da barra, como no exemplo abaixo:



Gustavo quer treinar o máximo possível e, por isso, quer usar *todas* as anilhas disponíveis na academia. Dadas as anilhas disponíveis e seus pesos, é possível distribuir *todas* as anilhas nas duas pontas da barra de forma que o peso total seja o mesmo em ambas?

Entrada

A primeira linha contém o inteiro N ($1 \leq N \leq 100$), o número de anilhas disponíveis. A segunda linha contém N inteiros P_i ($1 \leq P_i \leq 100$), o peso de cada anilha, em quilogramas.

Considere que a barra é comprida o bastante de forma que é possível colocar qualquer quantidade de anilhas, em ambas as pontas.

Saída

Imprima uma única linha contendo `SIM` se é possível distribuir todas as anilhas nas duas pontas da barra de forma que o peso total seja o mesmo em ambas, ou `NAO` caso contrário.

Exemplo de entrada	**Exemplo de saída**
`4` `5 20 5 10`	`SIM`

Exemplo de entrada	**Exemplo de saída**
`3` `20 5 10`	`NAO`

Exemplo de entrada	**Exemplo de saída**
`6` `5 5 10 10 15 15`	`SIM`