O Desafio dos Prisioneiros

Numa prisão existem 500 prisioneiros. Um dia, o diretor da prisão oferece-lhes uma hipótese de eles se libertarem a si próprios. Ele coloca dois sacos com dinheiro numa sala: o saco A e o saco B. Cada saco contém entre 1 a N moedas, inclusive. O número de moedas no saco A é **diferente** do número de moedas no saco B. O diretor apresenta um desafio aos prisioneiros. O objetivo dos prisioneiros é identificar o saco com menos moedas.

Na sala, para além dos sacos de dinheiro, existe um quadro branco. Um único número deve estar escrito no quadro em qualquer altura. Inicialmente, o número no quadro é 0.

Então, o diretor pede aos prisioneiros para entrarem na sala, um por um. O prisioneiro que entra na sala não sabe quais ou quantos prisioneiros entraram na sala antes deles. De cada vez que um prisioneiro entra na sala, ele lê o número que está presentemente escrito no quadro. Depois de ler o número, ele deve escolher o saco A ou o saco B. De seguida, o prisioneiro **inspeciona** o saco escolhido, ficando então a conhecer o número de moedas dentro dele. Então, os prisioneiros têm de fazer uma das duas seguintes **ações**:

- Apagar o número no quadro e escrever um novo número não negativo e sair da sala. Nota que podem optar por mudar ou manter o número atual. O desafio continua depois disso (a não ser que todos os 500 prisioneiros já tenham entrado na sala).
- Identificar um saco como sendo aquele que contém menos moedas. Isto termina imediatamente o desafio.

O diretor nunca irá pedir a um prisioneiro que saiu da sala para entrar novamente.

Os prisioneiros vencem o desafio se um deles identificar corretamente o saco com menos moedas. Eles perdem o desafio se algum deles identificar o saco incorretamente, ou se todos os 500 prisioneiros entraram na sala e não tentaram identificar o saco com menos moedas.

Antes do desafio começar, os prisioneiros reúnem-se no corredor da prisão e decidem uma **estratégia** comum para o desafio assente em três passos:

- Eles escolhem um número não negativo x, que é o maior número que eles alguma querem escrever no quadro.
- Eles decidem, para um qualquer número i escrito no quadro ($0 \leq i \leq x$), qual saco deve ser inspecionado por um prisioneiro que veja o número i escrito no quadro quando entra na sala.

- Eles decidem que ação deve um prisioneiro na sala fazer depois de saber o número de moedas no saco escolhido. Especificamente, para qualquer número i escrito no quadro ($0 \leq i \leq x$) e qualquer número de moedas j vistos no saco inspecionado ($1 \leq j \leq N$), eles decidem
    - qual número entre 0 e x (inclusive) deve ser escrito no quadro, ou
    - qual saco deve ser identificado como sendo o que tem menos moedas.

Depois de ganhar o desafio, o diretor libertará os prisioneiros depois de cumprirem pena durante mais x dias.

A tua tarefa é desenhar uma estratégia para os prisioneiros que garanta que ganham o desafio (independentemente do número de moedas no saco A e no saco B). A pontuação da tua solução depende do valor de x (verifica a secção das Subtarefas para mais detalhes)

Detalhes de Implementação

Deves implementar a seguinte função:

```
int[][] devise_strategy(int N)
```

- N: o número máximo de moedas em cada saco.
- Esta função deve devolver um array s de arrays de $N + 1$ inteiros, representando a tua estratégia. O valor de x é o tamanho do array s menos um. Para cada i tal que $0 \leq i \leq x$, o array $s[i]$ representa o que um prisioneiro deve fazer se ele vir o número i no quadro ao entrar na sala:
    1. O valor de $s[i][0]$ é 0 se o prisioneiro deve inspecionar o saco A, ou 1 se o prisioneiro deve inspecionar o saco B.
    2. Seja j o número de moedas visto no saco escolhido. O prisioneiro deve então fazer a seguinte ação:
        - Se o valor de $s[i][j]$ for -1, o prisioneiro deve identificar o saco A como sendo o que tem menos moedas.
        - Se o valor de $s[i][j]$ for -2, o prisioneiro deve identificar o saco B como sendo o que tem menos moedas.
        - Se o valor de $s[i][j]$ for um inteiro não negativo, o prisioneiro deve escrever esse número no quadro. Nota que $s[i][j]$ deve ser no máximo x.
- Esta função é chamada exatamente uma vez.

Exemplo

Considera a seguinte chamada:

```
devise_strategy(3)
```

Seja v o número que o prisioneiro vê no quadro depois de entrar na sala. Uma das estratégias corretas é a seguinte:

- Se $v = 0$ (incluindo o número inicial), inspecionar o saco A.
  - Se ele contém 1 moeda, identificar o saco A como sendo o que tem menos moedas.
  - Se ele contém 3 moedas, identificar o saco B como sendo o que tem menos moedas.
  - Se ele contém 2 moedas, escrever 1 no quadro (depois de apagar o 0).
- Se $v = 1$, inspecionar o saco B.
  - Se ele contém 1 moeda, identificar o saco B como sendo o que tem menos moedas.
  - Se ele contém 3 moedas, identificar o saco A como sendo o que tem menos moedas.
  - Se ele contém 2 moedas, escrever 0 no quadro (depois de apagar o 1). Nota que este caso nunca vai acontecer, uma vez que podemos concluir que ambos os sacos contêm duas moedas, o que não é permitido.

Para indicar esta estratégia, a função deve devolver `[[0, -1, 1, -2], [1, -2, 0, -1]]`. O tamanho do array devolvido é 2 e por isso neste caso o valor de x é $2 - 1 = 1$.

Restrições

- $2 \leq N \leq 5000$

Subtarefas

1. (5 pontos) $N \leq 500$, o valor de x não pode ser mais do que 500.
2. (5 pontos) $N \leq 500$, o valor de x não pode ser mais do que 70.
3. (90 pontos) O valor de x não pode ser mais do que 60.

Se em qualquer um dos casos o array devolvido por `devise_strategy` não representar uma estratégia correta, a pontuação da tua submissão será 0.

Na subtarefa 3 podes obter pontuação parcial. Seja m o maior valor de x dos arrays devolvidos em todos os casos de teste desta subtarefa. A tua pontuação nesta subtarefa é calculada de acordo com a seguinte tabela:

Condição	Pontos
$40 \leq m \leq 60$	20
$26 \leq m \leq 39$	$25 + 1.5 \times (40 - m)$
$m = 25$	50
$m = 24$	55
$m = 23$	62
$m = 22$	70
$m = 21$	80
$m \leq 20$	90

Avaliador Exemplo

O avaliador exemplo lê o input no seguinte formato:

- linha 1: N
- linha $2 + k$ ($0 \leq k$): $A[k]$ $B[k]$
- última linha: -1

Cada linha exceto a última representa um cenário. Referimo-nos ao cenário descrito na linha $2 + k$ como o cenário k. No cenário k o saco A contém $A[k]$ moedas e o saco B contém $B[k]$ moedas.

O avaliador exemplo começa por chamar `devise_strategy(N)`. O valor de x é o tamanho do array devolvido menos um. Então, se o avaliador exemplo deteta que o array devolvido por `devise_strategy` não está conforme as restrições descritas nos Detalhes de Implementação, ele imprime uma das seguintes mensagens de erro e termina:

- `s is an empty array`: s é um array vazio (que não representa uma estratégia válida).
- `s[i] contains incorrect length`: existe um índice i ($0 \leq i \leq x$) tal que o tamanho de $s[i]$ não é $N + 1$.
- `First element of s[i] is non-binary`: existe um índice i ($0 \leq i \leq x$) tal que $s[i][0]$ não é 0 nem 1.
- `s[i][j] contains incorrect value`: existem índices i, j ($0 \leq i \leq x, 1 \leq j \leq N$) tal que $s[i][j]$ não está entre -2 e x.

Se isto não acontecer, o avaliador exemplo produz dois outputs.

Em primeiro lugar, o avaliador exemplo imprime o output da tua estratégia no seguinte formato:

- linha $1 + k$ ($0 \leq k$): output da tua estratégia para o cenário k. Se aplicar a estratégia resultar num prisioneiro a identificar o saco A como sendo o que tem menos moedas, o output é o carácter `A`. Se aplicar a estratégia resultar num prisioneiro a identificar o saco A como sendo

o que tem menos moedas, o output é o carácter B. Se aplicar a estratégia não resultar num prisioneiro a identificar o saco com menos moedas, o output é o carácter X.

Em segundo lugar, o avaliador exemplo escreve um ficheiro `log.txt` no diretório corrente no seguinte formato:

- linha $1 + k$ ($0 \leq k$): $w[k][0]$ $w[k][1]$ $\ldots$

A sequência no linha $1 + k$ corresponde ao cenário k e descreve os números escritos no quadro. Mais especificamente, $w[k][l]$ é o número escrito pelo $(l + 1)$-ésimo prisioneiro a entrar na sala.