Jhonny y el ascensor (ascensor)

Descripción

Jhonny es el administrador de uno de los edificios más altos de su ciudad, su edificio tiene $N + 1$ pisos, numerados de 0 a N, donde 0 representa la planta baja.

Como el edificio es tan alto, las personas que viven en él dependen mucho del uso del ascensor, es tanto su uso, que arruinaron varios botones del mismo. En vista de esto la administración del edificio decidió reprogramar el ascensor de la siguiente manera:

- El ascensor ahora cuenta con M botones.
- Cada boton puede subir o bajar una determinada cantidad de piso.
- Si un boton es presionado y como resultado este debe llevar niveles menores a 0 o a niveles mayores a N, estos botones no harán acción alguna. Por ejemplo si el edificio tiene 10 pisos, y actualmente estoy en el piso 8 y se apreta un boton que debe llevar 3 pisos arriba, este boton no hara ninguna acción y el ascensor se quedara en el piso 8.

Después de reprogramar los botones del ascensor, la totalidad de vecinos ya no utilizan el ascensor pues es muy facil perderse y no entienden bien la nueva programación. Sin embargo Jhonny, al ser el administrador del edificio, debe moverse rapidamente de un piso a otro en el edificio. Es por eso que Jhonny pide tu ayuda para dados dos pisos x e y, le indiques a Jhonny la minima cantidad de botones que se debe presionar para llegar del piso a x al piso y.

Implementación

Debes implementar el siguiente procedimiento en C++:

```
int cantidad_botones(
  int N, int M, int x, int y, std::vector<int> botones
);
```

o el siguiente procedimiento en Python:

```
def cantidad_botones(
    N: int, M: int, x: int, y: int, botones: List[int]
) -> int:
```

Parámetros

- N: la cantidad de pisos que tiene el edficio (recuerda que tambien existe el nivel 0).
- M: la cantidad de botones que ahora tiene el ascensro.
- x: el piso donde se encuentra Jhonny en este momento.
- y: el piso a donde debe ir Jhonny.
- botones: Un arreglo o lista, de longitud M con a_i enteros, si el entero es positivo significa que este boton permite que se suban a_i pisos, si es negativo significa que este boton permite que se bajen a_i pisos.

Retorna

El procedimiento debe devolver un entero que representa la cantidad mínima de botones que debe presionar Jhonny para llegar del piso x al piso y. Si es que no existe una forma de llegar el procedimiento debe retornar -1.

Ejemplos

Ejemplo 1

Considera la siguiente llamada:

```
cantidad_botones (20, 3, 7, 12, [3, -2, 4])
```

La respuesta debe ser 3 botones, pues para ir del piso 7 al piso 12, podemos usar el boton para subir 3 y estar en el piso 10, luego usar el boton para bajar 2 piso y estar en el piso 8 y finalmente usar el boton para subir 4 pisos para subir al piso 12.

Recuerda que puedes usar los botones las veces que quieras.

Límites

- $1 \leq N \leq 10^5$
- $1 \leq M \leq 100$
- $0 < |a_i| \leq N$
- $0 \leq x, y \leq N$

Subtareas

Subtarea 1 (3 puntos)

- $1 \leq N \leq 100$
- $N = M$
- $x = 0$
- $a_i = i$ para todo $1 \leq i \leq N$

Subtarea 2 (4 puntos)

- $M = 1$
- $x = 0$
- El único botón es $a_1 = 1$ (solo hay un botón que sube un piso)

Subtarea 3 (8 puntos)

- $M = 1$
- $x = 0$

Subtarea 4 (27 puntos)

- $N = 1000$
- $M = 2$

Subtarea 5 (58 puntos)

- Sin restricciones adicionales al problema original

Entrada

- Solo debes subir el archivo `ascensor.cpp` con tu solución.
- La entrada y salida serán manejadas por el archivo `grader.cpp` o `grader.py`.
- **No debes imprimir nada por la salida estándar** ni leer de la entrada estándar en tu archivo `ascensor.cpp`.
- No modifiques los archivos `grader.cpp` o `grader.py`, ya que el juez solo tomará en cuenta tu solución en `ascensor.cpp`.