TROLL CODE

Integrantes:

- Roberth Gancino
- Anthony Goyes
- Danny lñaguazo

GitHub: https://github.com/NwDann/Taller01-MN.git

Nickname en CSACADEMY: "Reik Polamina"

Realmente costó moldear el código para este ejercicio debido a algunos problemas presentes para varias <u>cadenas</u> de bits. Entonces el primer resultado fue una mitad de nota:

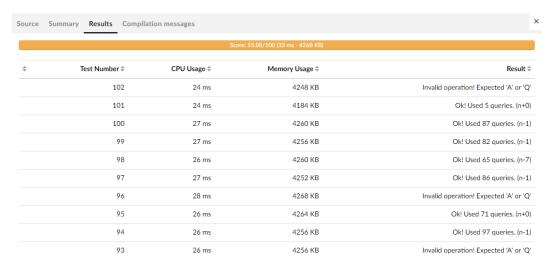


Ilustración 1: Puntuación de la resolución inicial.

Los casos que nos originaban errores fueron manejados correctamente mediante condiciones y consultas al "Troll". El resultado fue excelente:

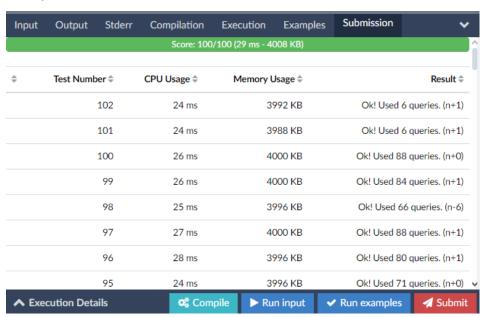


Ilustración 2: Puntuación de la resolución mejorada.

Finalmente, los siguientes códigos fueron los postulados como soluciones. En el segundo programa, se señalan las correcciones utilizadas para mejorar el algoritmo y robustecer el manejo de cadenas de bits.

CODIGO INICIAL (Score: 55.8/100, Fecha: 26/06/2024, Hora: 9:30)

```
def guessing():
    global sequence, prev_correctBits
    for i in range(N):
        sequence[i] = 1 - sequence[i] # Changing from 0 to 1 and from 1
        outPut = "Q " + " ".join(map(str, sequence))
        print(outPut)
        sys.stdout.flush()
        correctBits = int(input())
        if correctBits == N:
            return True # Correct sequence
        elif correctBits < N and correctBits > prev_correctBits:
            prev_correctBits = correctBits
        else:
            sequence[i] = 1 - sequence[i] # Reversing the change
    return False
import sys
N = int(input())
sequence = [0] * N
prev_correctBits = 0
flag = guessing()
if flag:
    outPut = "A " + " ".join(map(str, sequence))
    print(outPut)
    sys.stdout.flush()
else:
    print("Sequence not guessed")
```

CODIGO MEJORADO (Score: 100/100, Fecha: 26/06/2024, Hora: 16:31)

El código en rojo es el añadido, permitiendo ahorrar consultas en ciertos casos con la primera consulta dentro de la función "guessing". Después comprobamos si nos conviene trabajar con una secuencia de bits con la mayoría de ceros o unos.

```
def guessing(N):
    for i in range(N):
       sequence[i] = 1 - sequence[i] # Changing from 0 to 1 and from 1
       outPut = "Q " + " ".join(map(str, sequence))
       print(outPut)
       correctBits = int(input())
       sys.stdout.flush()
       if correctBits == N:
           return True # Correct sequence
       elif correctBits < N and correctBits > prev_correctBits:
           prev_correctBits = correctBits
       else:
           sequence[i] = 1 - sequence[i] # Reversing the change
    return False
             -----MAIN-----
import sys
N = int(input())
sequence = [0] * N
flag = guessing(N)
```

```
if flag:
    outPut = "A " + " ".join(map(str, sequence))
    print(outPut)
    sys.stdout.flush()
else:
    print("Sequence not guessed")
```