Tercer Problema

Giovanni Alejandri Espinosa

321037293

¿Cómo se llegó a la solución? ¿Cuál fue su elaboración?

Paso 1

El problema pide determinar si es posible llegar al final de una lista de enteros, donde cada número representa la cantidad máxima de posiciones que se pueden avanzar desde esa posición. Para resolverlo, decidí utilizar un enfoque iterativo, controlando el "alcance" máximo posible en cada posición de la lista.

Paso 2

Definí la función `saltos` que recorre la lista y mantiene un control del mayor índice alcanzable en cada iteración. Si en algún momento el índice actual excede el alcance posible, significa que no es posible avanzar más, por lo que se retorna `False`. Si el alcance es suficiente para llegar al final, la función retorna `True`.

Paso 3

En cada iteración, se compara el alcance actual con el máximo posible desde la posición actual, y se actualiza si es necesario. Si en algún punto el alcance cubre o supera el último índice de la lista, la función termina con éxito.

Explicación detallada del código:

- `alcance = 0`: Esta variable controla hasta dónde es posible avanzar en la lista en cada iteración.
- `for i in range(len(mi`$_l ista$`))` : $Serecorrelalistadeizquierdaaderecha, posiciónporposición.$`if i > al`
  $Sielíndiceactualexcedeelalcancemáximoposible, seretorna$`False`$, yaquenoesposibleavanzarmás.$

- `alcance = max(alcance, i + mi`$_l ista[i]$) : $Aquíseactualizaelalcancemáximoposibledesdelaposiciónactual.$`if`

Ejemplos añadidos

```
from saltos import saltos

def test_saltos():
    # Prueba 1: Deber a devolver True
    assert saltos([1, 1, 2]) == True, "Error en el caso 1"

    # Prueba 2: Deber a devolver False
    assert saltos([3, 2, 1, 0, 4]) == False, "Error en el caso 2"

    # Prueba 3: Deber a devolver True
    assert saltos([3, 3, 1, 2, 0, 1]) == True, "Error en el caso 3"

    # Prueba 4: Deber a devolver False (lista vac a)
    assert saltos([]) == False, "Error en el caso lista vac a"

    # Prueba 5: Deber a devolver True (ya estamos en el  nico  ndice )
    assert saltos([0]) == True, "Error en el caso un solo elemento"

if __name__ == "__main__":
    test_saltos()
```

En los ejemplos añadidos se comprueban casos básicos, como una lista con saltos que permiten llegar al final, listas que no lo permiten, listas vacías y una lista con un único elemento.

Conclusión

La solución fue implementada de manera eficiente utilizando un enfoque iterativo con control de alcance. Esto permite verificar si es posible llegar al final de la lista sin necesidad de hacer retrocesos o complicaciones adicionales. Las pruebas añadidas cubren diferentes escenarios, asegurando que la función funcione correctamente para una variedad de casos. La función responde adecuadamente tanto en listas vacías como en listas con un solo elemento, y los resultados son precisos en todos los casos probados.