PF_CODING TEST – Ria Kabra

Q) Imagine Abhimanyu in Chakravyuha. There are 11 circles in the Chakravyuha surrounded by different enemies. Abhimanyu is located in the innermost circle and has to cross all the 11 circles to reach Pandavas army back.

Given:

Each circle is guarded by different enemy where enemy is equipped with k1, k2……k11 powers

Abhmanyu start from the innermost circle with p power

Abhimanyu has a boon to skip fighting enemy a times

Abhmanyu can recharge himself with power b times

Battling in each circle will result in loss of the same power from Abhimanyu as the enemy. If Abhmanyu enter the circle with energy less than the respective enemy, he will lose the battle k3 and k7 enemies are endured with power to regenerate themselves once with half of their initial power and can attack Abhimanyu from behind if he is battling in iteratively next circle

Write an algorithm to find if Abhimanyu can cross the Chakravyuh and test it with two sets of test cases.

A)

Algorithm

Input:

- p: Initial power of Abhimanyu.
- a: Number of times he can skip a battle.
- b: Number of times he can recharge.
- enemies: List of 11 integers representing enemy power in each circle.

Output:

- True if Abhimanyu successfully crosses all 11 circles.
- False if he cannot cross at any stage.

Algorithm:

1. **Initialize Counters:**
    - Set recharge_count = 0, this is for the number of recharges used.
    - Set skip_count = 0, for tracking the number of battles skipped.
2. **Loop Through Each Enemy (i from 0 to 10):**
    - **Checking is recharge is required**
        - If p < enemies[i] and recharge_count < b, recharge by adding enemies[i] to p.
        - Increase recharge_count by 1.

- **Check if you can skip:**
    - If p < enemies[i] and skip_count < a, skip this battle.
    - Increase skip_count by 1 and move to the next enemy.
    - If no skips are left, return False.
- **Fight the Enemy:**
    - Subtract enemies[i] from p.
    - If p < 0, return False (Abhimanyu loses).
- **Handle Special Regenerating Enemies (Index 2 and 6):**
    - If the enemy at index 2 or 6, it regenerates with half its power (enemies[i] // 2).
    - If p < regenerated power:
        - If skip_count < a, skip this second battle and increase skip_count.
        - Else, return False (Abhimanyu loses).
    - Otherwise, subtract the regenerated power from p.
- **Check if Abhimanyu Can Continue:**
    - If p < enemies[i+1] and no recharges are left (recharge_count >= b), return False.

3. **If All Circles Are Cleared, Return True.**

Code

```
def can_cross(p, a, b, enemies):
    recharge_count = 0
    skip_count = 0

    for i in range(len(enemies)):
        if p < enemies[i] and recharge_count < b:
            p += enemies[i]
            recharge_count += 1

        if p < enemies[i]:
            if skip_count < a:
```

```
            skip_count += 1
            continue
        else:
            return False

    p -= enemies[i]

    if p < 0:
        return False

    if i == 2 or i == 6:
        regen_power = enemies[i] // 2
        if p < regen_power:
            if skip_count < a:
                skip_count += 1
                continue
            else:
                return False
        p -= regen_power

    if i < len(enemies) - 1 and p < enemies[i + 1] and recharge_count >= b:
        return False
  return True
```

Tests

Input:

Testcase 1:

enemies_test1 = [3, 5, 3, 4, 6, 3, 4, 2, 5, 1, 3]

p1, a1, b1 = 30, 3, 3

Output: True

Testcase 2:

```
enemies_test2 = [4, 6, 5, 7, 6, 8, 5, 9, 7, 6, 8]

p2, a2, b2 = 10, 1, 1

print(can_cross(p2, a2, b2, enemies_test2))
```

Output: False