Name : Akshay Daberao

Prismforce ASDE Algorithm Test

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 Abhimanyu 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.

Solution ->

Conditions:

- Abhimanyu starts with power p.
- He can skip up to a enemies.
- He can recharge his power up to b times.

- Each enemy in circle i has power k_i (for i = 1 to 11).

- Enemies in circles 3 (k3) and 7 (k7) can regenerate once with half of their initial power and attack from behind if Abhimanyu is in the next circle.

- If Abhimanyu’s power is less than the enemy’s power in a circle, he loses.

Algorithm:

The strategy involves exploring all possible sequences of actions (battling, skipping, recharging) using recursive approach. The main challenge is to decide the optimal combination of actions that allow Abhimanyu to cross all 11 circles.

Recursive Logic:

1] Base Case:

If Abhimanyu reaches the 12th circle (index >= 11), he has successfully crossed all circles, and we return true.

2] Regenerating Enemies:

When Abhimanyu reaches circle 4 (index 3), the enemy in circle 3 (k[2]) regenerates and its power is halved if it is still alive.
Similarly, when Abhimanyu reaches circle 8 (index 7), the enemy in circle 7 (k[6]) regenerates and its power is halved if it is still alive.

3] Three Possible Actions:

Battle the current enemy:
If Abhimanyu's current power is greater than or equal to the enemy's power at the current circle, he battles the enemy and moves to the next circle with reduced power.

Skip the current enemy:
If skips are available, he can skip the current enemy and move to the next circle without reducing his power.

Recharge and battle:
If recharges are available, he can recharge his power back to the initial value p and try to battle the current enemy again.

4] Recursive Function:

The function recursively explores each of the above actions, checking if any sequence of actions allows Abhimanyu to cross all circles.If battling, skipping, or recharging leads to successfully crossing all circles in any recursive call, the function returns true.If none of the actions result in a successful crossing, the function returns false.

Example Test case:
Consider an example with initial power p=20, 2 skips, and 1 recharge. The enemies' powers are increasing from 1 to 11.
Abhimanyu starts at the first circle with 20 power.

He can choose to:
Battle the enemy with power 1 (successfully since 20 >= 1), move to the next circle with 19 power.
Skip the enemy if he has skips remaining.
Recharge if he has recharges remaining and needs more power for subsequent enemies.
This process continues for each circle, recursively exploring all possible paths.
Special conditions are checked for regenerating enemies at circles 4 and 8.

Further we can see repeating cases in recursion so we will optimize it further using concept of Dynamic Programming.