Solution Approach

- First, I took the initial power, the number of times Abhimanyu can recharge his power, and the power of each enemy into an array.

- I then approached a recursive solution, as for each enemy there is a choice to either defeat that enemy if his power is less than Abhimanyu's current power, recharge his power and defeat the enemy, or apply the one-time boon to skip that enemy.

- In the recursion, there is a necessary condition: if the enemy's power is greater than Abhimanyu's maximum initial power, he must apply the boon and skip that battle.

- Furthermore, there is a subtle condition: if the 3rd or 7th enemy is not skipped, they will attack from behind with half of their power, but only if Abhimanyu is battling the 4th or 8th enemy, respectively.

- Additionally, I created a cached array to memoize the recurrence values, reducing the overhead of traversing the same recurrence tree multiple times.

- Finally, after processing the data, if the returned value turns out to be true, then Abhimanyu will be able to defeat all enemies and break through the Chakravyuha. Otherwise, he will be defeated by the enemies.

Assumptions

- I have assumed that all input values will be integers and will fit within a 32-bit size.

- For rule no. 5 in the question, I have assumed that if we skip the 3rd or 7th enemy, they will not attack from behind. Otherwise, they will regenerate themselves with half of their power and attack from behind.

- For rule no. 5 in the question, I have assumed that if we skip the 4th or 8th enemy, then the 3rd or 7th enemies will not attack from behind.

Time and Space Complexity

- The worst-case time and space complexity will be: O (noOfEnemies * power * powerRechargeTimes * 8).
- Where:
  - noOfEnemies = the number of enemy circles present (as stated in the question, it’s 11).
  - power = the initial power of Abhimanyu.
  - powerRechargeTimes = the number of times Abhimanyu can recharge his power.
  - 8 refers to the total number of combinations for three variables: isBoonApplied, isThirdEnemySkipped, and isSeventhEnemySkipped. Each of these is a boolean, resulting in a maximum of two possibilities (true and false).