"중등부 3번. 로봇" 문제 풀이

작성자: 오주원

부분문제 1

$N = 1$이므로 점프대가 하나뿐이다. 로봇은 왼쪽으로 이동하다가 이 점프대에 도착하면 점프대를 작동시키는 것을 반복한다. 다만 초기에 로봇이 점프대보다 왼쪽에 있는 경우에는 로봇이 점프대에 도달할 수 없으므로, 단순히 왼쪽으로 T만큼 이동한 위치가 정답이 된다.

점프대는 작동될 때마다 파워가 두 배로 증가하므로, 동일한 점프대를 이용할 수 있는 횟수는 최대 $O(\log T)$번이다. 또한 점프대가 등장하기 전까지 왼쪽으로 이동하는 횟수는 단순히 현재 위치와 점프대의 위치 차이를 계산하여 $O(1)$에 구할 수 있다.

따라서, 로봇의 이동을 시뮬레이션할 때 왼쪽으로의 단순 이동은 한 번의 계산으로 처리하고, 점프대에 도착했을 때 점프와 파워를 증가시키는 행동을 반복하면 된다. 이 방법을 사용하면 로봇의 최종 위치를 $O(\log T)$ 시간에 구할 수 있다.

부분문제 2

부분문제 1과 달리 점프대가 두 개이므로 두 점프대 사이에서 출발하는 경우를 고려해야 한다. 특히 왼쪽 점프대를 사용하다가 점프 후 오른쪽 점프대의 위치를 넘어가는 상황까지 처리할 수 있도록 구현해야 한다.

부분문제 3

로봇이 이동하는 경로는 지문의 예시와 같은 방식으로 그대로 시뮬레이션할 수 있다. 매 초마다 현재 위치에 점프대가 존재하는지를 확인하고, 이에 따라 이동을 결정하면 된다. 각 초마다 점프대의 존재 여부를 확인하는 데 $O(N)$의 시간이 걸리므로, 전체적으로 T초 동안의 시뮬레이션은 $O(NT)$의 시간복잡도로 수행할 수 있다. 따라서 전체 시간복잡도는 $O(QNT)$이다.

부분문제 4

부분문제 3과 마찬가지로 로봇이 이동하는 경로는 지문의 예시처럼 그대로 시뮬레이션할 수 있다. 다만 매 초마다 현재 위치에 점프대가 존재하는지를 단순히 $O(N)$에 탐색하는 방식은 비효율적이다. 이를 개선하기 위해 정렬된 점프대의 위치에서 이분 탐색을 사용하면 한 번의 탐색을 $O(\log N)$에 수행할 수 있다. 혹은 배열이나 set과 같은 자료구조를 사용하여 점프대의 존재 여부를 $O(1)$ 또는 $O(\log N)$에 확인하도록 구현할 수도 있다. 이렇게 하면 전체 시간복잡도를 $O(QT \log N)$혹은 $O(QT)$로 줄일 수 있다.

부분문제 5

어떤 점프대에서 점프를 막 시작해서 다음 점프대의 위치를 넘어가 그 점프대에 도달하여 새로운 점프대를 사용하게 되는 과정을 **점프대 이동**이라고 정의하자. 이러한 **점프대 이동**은 최대 $N-1$번만 일어날 수 있다.

따라서 부분문제 2에서와 같이, 다음 점프대로 넘어가는 과정을 반복하며 시뮬레이션하면 된다. 다만 i번 점프대에서 **점프대 이동**이 발생했다고 해서 반드시 $i+1$번 점프대로 이동하는 것은 아니므로, 이 점을 주의해야 한다.

각 점프대에서 점프 횟수는 최대 $O(\log T)$에 불과하며, 이러한 과정을 N개의 모든 점프대에 대해 Q개의 각 시작 위치에 반복하므로, 이 방법의 시간 복잡도는 $O(NQ \log T)$이다.

하지만 Q개의 질의를 계산하기 전에, 각 점프대에 대해서 처음 도달한 이후 **점프대 이동이 발생하기까지 걸리는 시간**을 미리 전처리 해두면 더 빠른 시간으로 문제를 해결할 수 있다. 이 전처리는 각 점프대별로 최대 $O(\log 10^{17})$번의 시뮬레이션을 통해 $O(N \log 10^{17})$의 시간으로 필요한 값을 구할 수 있다. 이렇게 전처리가 끝나면, 각 시작 위치에 대한 계산에서 더 이상 점프 횟수를 일일이 시뮬레이션할 필요가 없으므로, 각 경우를 $O(N)$에 처리할 수 있어 전체 시간복잡도는 $O(NQ)$으로 줄어든다.

수식 정리를 통해 $O(1)$에 점프대 이동이 걸리는 시간을 계산하는 방법으로도 $O(NQ)$의 시간복잡도가 나오도록 최적화를 할 수 있다.

부분문제 6

$dp[i][j]$를 i번째 점프대에 처음 도달한 이후 j번의 **점프대 이동이 발생하기까지 걸리는 시간**이라고 정의하고, $nxt[i][j]$를 i번째 점프대에서 j번의 **점프대 이동이 발생한 이후의 도달하게 되는 점프대의 번호**로 정의하자.

이 값들은 부분문제 5의 사용한 방법으로 $dp[*][1]$와 $nxt[*][1]$을 먼저 구한 뒤, 점프대 번호 i가 감소하는 순서대로 $dp[i][*]$와 $nxt[i][*]$를 채워 나가면 된다. 이를 통해 각 점프대에서 여러 번의 **점프대 이동**이 발생한 이후의 걸린 시간과 도달하는 점프대의 번호를 미리 구할 수 있다.

이 값들을 미리 구해놓으면 더 이상 매번 $O(N)$번의 점프대 이동을 직접 시뮬레이션할 필요가 없다. 로봇이 처음으로 도달하는 점프대의 번호를 k라고 할 때, $dp[k][i]$값은 i가 증가할수록 증가하므로, $dp[k][i]$가 T 이하인 최대 i를 이분 탐색으로 $O(\log N)$에 찾을 수 있다. 이렇게 하면 **점프대 이동**이 더 이상 일어나지 않을 때까지의 과정을 스킵할 수 있으며, 이후 남은 시간 동안은 단일 점프대에서의 이동만 단순 시뮬레이션하면 된다.

전처리 과정에서 $dp[*][1]$와 $nxt[*][1]$을 구하는 데 $O(N \log 10^{17})$의 시간이 필요하며, $dp[*][*]$와 $nxt[*][*]$을 모두 구하는 데 $O(N^2)$의 시간이 소요된다. 전처리 이후에는 각 시작 위치에 대해 처음 도달하는 점프대를 찾는데 $O(\log N)$, 그리고 그 점프대에서 이분 탐색을 통해 최대 **점프대 이동** 횟수를 찾는 데 $O(\log N)$, 마지막으로 남은 시간동안 단일 점프대에서 시뮬레이션을 수행하는 데 $O(\log T)$가 걸린다. 따라서 전체 시간복잡도는 $O(N \log 10^{17} + N^2 + Q(\log N + \log T))$이다.

부분문제 7

$dp[i][j]$를 i번째 점프대에 처음 도달한 이후 2^j번의 **점프대 이동이 발생하기까지 걸리는 시간**이라고 정의하고, $nxt[i][j]$를 i번째 점프대에서 2^j번의 **점프대 이동이 발생한 이후의 도달하게 되는 점프대의 번호**로 정의하자.

우선 $dp[i][0]$과 $nxt[i][0]$은 부분문제 5에서 사용한 방식과 동일하게 초기화한다. 이후 j가 증가하는 순서로 아래 점화식을 이용해 값을 채울 수 있다.

- $dp[i][j] = dp[i][j-1] + dp[nxt[i][j-1]][j-1]$
- $nxt[i][j] = nxt[nxt[i][j-1]][j-1]$

T초 동안의 점프대 이동을 효율적으로 계산하기 위해서는, 매 단계에서 현재 남은 시간 안에서 최대한 많이 **점프대 이동**을 수행할 수 있도록 선택하는 것이 핵심이다. 현재 위치한 점프대가 k, 남은 시간이 T_{rem}이라고 하자. $dp[k][j] \le T_{rem}$인 j 중에서 가장 큰 j를 찾아 이동하면 한 번에 2^j번의 점프대 이동을 처리할

수 있으며, 이동 후에는 T_{rem}에서 $dp[k][j]$를 빼고, 점프대 번호 k를 $nxt[k][j]$로 갱신한다. 이 과정을 남은 시간이 허용하는 동안 반복하면, T초 동안 발생하는 모든 점프대 이동을 빠르게 처리할 수 있고, 이후 남은 시간 동안은 단일 점프대에서의 이동만 $O(\log T)$의 시간으로 단순 시뮬레이션하면 된다.

이 방법은 $O(\log^2 N)$이 아닌 $O(\log N)$으로 구현할 수 있는데, 한 번의 선택에서 반드시 j가 감소하는 방향으로 진행하는 것을 이용하면 된다. 만약 $dp[k][j]$를 선택한 뒤에도 j 이상인 어떤 j'에 대해 $dp[k][j']$을 선택할 수 있었다면, 이는 처음부터 $dp[k][j+1]$을 선택했어야 하므로 모순이다. 따라서 j는 매 단계마다 감소하며, 최대 $\log N$번의 선택으로 모든 **점프대 이동**을 처리할 수 있다.

이 방식에서 $dp[*][0]$와 $nxt[*][0]$을 구하는 데 $O(N \log 10^{17})$, $dp[*][*]$와 $nxt[*][*]$을 모두 채우는 데 $O(N \log N)$이 소요된다. 이후 각 시작 위치에 대해서는 $O(\log N + \log T)$만에 계산할 수 있으므로, 전체 시간복잡도는 $O(N \log 10^{17} + Q(\log N + \log T))$이다.