**Homework 3C report**

**2014313303 홍태하(개발환경 : Linux)**

1. **알고리즘**

인풋으로 라쿤의 상태가 들어오는데 state배열에 저장하고 라쿤의 전체 수를 N에 저장한다. 그 후 라쿤의 수 두배 크기의 배열 dp를 선언한다. 이 배열에는 i번째 라쿤의 상태를 바꾸기 위한 최소의 숫자가 저장 되는데, dp[i][0]에는 i번째 라쿤부터 그 뒤의 모든 라쿤을 sleep 상태로 바꾸는 최소의 수가 저장 된다. Sleep 상태는 0으로 표현되므로 i부터 모두 0이 되도록 하는 최소의 수가 저장되는 것이다. dp[i][1]에는 i번째 라쿤만 깨어나고 그 뒤의 모든 라쿤이 sleep 상태가 되도록 하는 최소의 수가 저장 된다. Awake 상태는 1이므로 i번째 라쿤은 1이 되고 그 뒤의 모든 라쿤이 0이 되도록 하는 최소의 수가 저장되는 것이다. 이 과정을 맨 뒤 라쿤부터 맨 앞 라쿤까지 진행한다. 그렇게 하면 dp[0][0]이 모든 라쿤이 0인 상태가 되기 때문에 답이 된다.

그렇다면 dp[i][0]과 dp[i][1]은 어떻게 구하면 될까? 먼저 dp[i][0]을 구해보면 현재 i번째 라쿤의 상태에 따라 2가지로 나뉘게 된다. dp[i][0]은 i번째 라쿤부터 맨끝 라쿤까지 모두 0으로 만드는 것이기 때문에 만약 i번째 라쿤이 0이면 dp[i+1][0]으로 i+1번째 라쿤부터 맨끝 라쿤까지 0으로 만드는 방법의 수와 같게 된다. 하지만 i번째 라쿤이 1이라면 이것을 0으로 바꿔주고 그 뒤의 라쿤들을 0으로 바꿔야 한다. 1을 0으로 바꾸려면 바로 다음 라쿤이 1의 상태이고 그 뒤의 모든 라쿤이 0이어야 한다. 이것은 dp[i+1][1]의 상태와 같다. dp[i+1][1]은 i+1번째 라쿤만 1이고 그 뒤의 라쿤은 모두 0이므로 i번째 라쿤부터 보면 1100…00의 모양이 된다. 그 후 i번째 라쿤을 0으로 바꿔주면 0100…00의 모양이 되고 100..00은 Hint1에 따라 모두 0으로 바꾸는데 2^m-1번이 소요된다. 저기에서 m은 첫째수 1부터 마지막 0까지의 개수를 뜻한다. 따라서 i번째가 1일 때 dp[i][0]은 dp[i+1][1] + 1 + 2^(N-i-1) -1이 된다(라쿤의 전체 수는 N이고 라쿤의 위치인 i는 0부터 시작해서 N-1까지 있다고 한다. 2^(N-i-1)에서 N-i-1은 i+1부터 N-1까지의 개수를 뜻한다.).

dp[i][1]도 i번째 라쿤의 상태에 따라 2가지 경우로 나뉘게 된다. dp[i][1]은 i번째 라쿤만 1이고 그 뒤는 모두 0이 되는 경우이기 때문에 현재 i번째 라쿤이 1인 경우 뒤의 라쿤들만 전부 0으로 바꾸어 주면 된다. 이것은 dp[i+1][0]과 같다. 반면 i번째 라쿤이 0인 경우 이것을 1로 바꾸고 그 뒤의 라쿤을 0으로 바꾸어 주어야 한다. 라쿤의 상태를 바꾸기 위해서는 그 뒤의 라쿤이 1이고 그 뒤의 모든 라쿤이 0이 되어야 한다. 이것은 dp[i+1][1]과 같다. 그 후 i번째 라쿤을 1로 바꾸어 주고 위에서 설명한 것과 같이 i+1번째 라쿤부터 100…00의 모양이 되는데 이것을 전부 0으로 바꾸어 주면 된다. 따라서 i번째 라쿤이 0인 경우 dp[i][1]은 dp[i+1][1] +1 + 2^(N-i-1) -1이 된다.

배열 dp를 모두 채운 후 dp[0][0]이 모든 라쿤을 0의 상태로 만드는 최소의 수이다.

1. **코드 설명.**
2. **인풋 받기.**

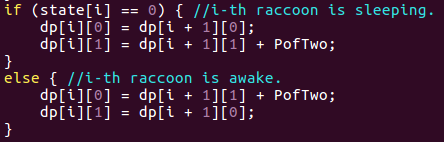
input이라는 char형 변수를 선언하여 getchar()를 통해 인풋을 받는다. 인풋은 0 또는 1뿐이므로 그 외의 것이 인풋으로 들어오면 인풋을 그만 받게 된다. 인풋은 input에 저장되고 이것은 바로 state배열에 integer의 형태로 저장된다. input에는 0 또는 1의 아스키 코드 값이 저장되어 있기 때문에 0의 아스키 코드 값을 빼주고 state배열에 저장하면 된다.

1. **배열 dp채우기.**

dp배열의 맨끝 라쿤부터 채우기 시작한다. 마지막 라쿤의 경우 마지막 라쿤이 0인 경우

dp[N-1][0]은 0이고 dp[N-1][1]은 1이 되고 마지막 라쿤이 1인 경우 0일 때와 반대가 되기 때문에

이렇게 하면 된다. 그 후부터는 알고리즘 부분에서 설명한 것과

같이 하면 된다. 밑의 코드처럼 하면 되는데 PofTwo는 2^(N-i-1)을 뜻한다. 저렇게 0번째 라쿤까지 모두 채우기 되면 끝이다. 여기서 한가지 주의할 점은 N은 최대 50까지 가능하다. 이는 라쿤이 최대 50마리라는 것인데 50마리의 라쿤이 100…00의 형태일 경우 모두 0으로 바꾸기 위해 2^50-1의 수가 필요하다. 이 수는 32비트인 int타입 변수가 저장할 수 없으므로 64비트인 long long타입 변수에 저장해야 한다. 따라서 배열 dp와 변수 PofTwo는 long long 타입으로 선언되어야 한다. 배열 dp는 동적 할당되었으므로 결과값 출력 후에는 free를 해준다.

1. **예시**

**1010** 의 인풋이 들어왔다고 하자. state배열에 순서대로 1010을 저장하고 배열 dp를 채우기 시작한다. 마지막 라쿤이 0이기 때문에 dp[3][0]은 0이고 dp[3][1]은 1이다. 그 후 위의 알고리즘에 따라 배열dp를 채우면 다음과 같다. 따라서 결과값은 12가 된다.

dp[i][j]

