Question :

* You are given an two arrays of size “N”.You need to find the number of journey whose sum is odd and even to reach index “N”.
* For each index i,
  + A[i] = a[i] → a[i + 1] or A[i] = a[i] → b[i + 1]
  + B[i] = b[i] → b[i + 1] or B[i] = b[i] → a[i + 1]

Observation :

* For index i , if the value is even then the number of odd journey will be dp[i - 1][odd] and even journey will be dp[i - 1] [even].
* For index i , if the value is odd then the number of odd journey will be dp[i - 1][even] and the even journey will be dp[i - 1] [odd].
* For each index we have four possibilities
  + Value is Odd , comes from A or comes from B
  + Value is even , comes from A or comes from B.
* So i will create a 3 D dp array to store values

, dp[i][a][odd] and dp[i][a][even] →Indicates that the for index i , the value is supplied by array A

, dp[i][b][odd] and dp[i][b][even] → Inidicate that for the index i , the value is supplied by array A.

So for index i , I can do a[i - 1] or b[i - 1]

The same Logic in Session 9 of google OA.

Recurrence Relation :

If a[i] is odd , I assume that i can be from A and B

dp[i][a][even] = dp[i - 1][a][odd] + dp[i - 1][b][odd]

dp[i][a][odd] = dp[i - 1][a][even] + dp[i - 1][b][even]

If a[i] is even ,

dp[i][a][even] = dp[i - 1][a][even] + dp[i - 1][b][even]

dp[i][a][odd] = dp[i - 1][a][odd] + dp[i - 1][b][odd]

If b[i] is odd ,

,dp[i][b][even] = dp[i - 1][b][odd] + dp[i - 1][a][odd]

,dp[i][b][odd] = dp[i - 1][b][even] + dp[i - 1][a][even]

If b[i] is even ,

,dp[i][b][even] = dp[i - 1][b][even] + dp[i - 1][a][even]

,dp[i][b][odd] = dp[i - 1][b][odd] + dp[i - 1][a][odd]]

Step 1 : Create an 3 D dp array of size “N” , “2” , “2”

Step 2 : Fill the base condition in the dp array

Step 3 : Use the recurrence relation and find the answer.

Step 4: Return the sum of dp[n][0][0] + dp[n][1][0] and dp[n][0][1] + dp[n][1][1]

Code :

public int[] sumofJourney(int [] A , int [] B , int n){

int [][][] dp = new int[n + 1] [2][2];

/\*

dp [index][A / B] [odd / Even]

A --> 0

B --> 1

Even --> 0

Odd --> 1

\*/

//If index is even then

// the number of ways to reach index 0 with even sum 1 and odd sum is 0

if(a[i] % 2 == 0){

dp[1][0][0] = 1;

dp[1][0][1] = 0;

}//else the number of ways to reach index 0 with even sum 0 and odd sum is 1

else{

dp[1][0][0] = 0;

dp[1][0][1] = 1;

}

// the same base case will be followed for array B

// the number of ways to reach index 0 with even sum 1 and odd sum is 0

if(b[i] % 2 == 0){

dp[1][1][0] = 1;

dp[1][1][1] = 0;

}//else the number of ways to reach index 0 with even sum 0 and odd sum is 1

else{

dp[1][1][0] = 0;

dp[1][1][1] = 1;

}

for(int i = 1 ; i < n ; i++){

if(a[i] % 2 == 0){

dp[i][0][0] = dp[i - 1][0][0] + dp[i - 1][1][0];

dp[i][0][1] = dp[i - 1][0][1] + dp[i - 1][1][1];

}else {

dp[i][0][0] = dp[i - 1][0][1] + dp[i - 1][1][1];

dp[i][0][1] = dp[i - 1][0][0] + dp[i - 1][1][0];

}

if(b[i] % 2 == 0){

dp[i][1][0] = dp[i - 1][0][0] + dp[i - 1][1][0];

dp[i][1][1] = dp[i - 1][0][1] + dp[i - 1][1][1];

}else{

dp[i][1][0] = dp[i - 1][0][1] + dp[i - 1][1][1];

dp[i][1][1] = dp[i - 1][0][0] + dp[i - 1][1][0];

}

}

int oddJourney = dp[n][0][1] + dp[1][1];

int evenJourney = dp[n][0][0] + dp[n][1][0];

return new int[]{oddJourney , evenJourney};

}

}