Qn Link : <https://www.desiqna.in/16115/google-interview-problem-dynamic-programming-cities-october>

Question Summary :

* Given two array of size “N” find the maximum sum can be obtained by doing the following opr.
  + Can pick value from “A” cost A[i]
  + Can pick value from “B” cost B[i]
  + Can travel A → B or B →A cost = 0

Observation :

* For index i , dp[i] is the best answer.
* For each index we can do three opreration , do you think we can do it in one state ?
  + Whenever you need state , then only add it to the soln .
* What are states?
  + What if i pick from A for index “i”
  + What if i pick from B for index “i”
* We don’t consider travel as a state because , what happen if we travel , we pick the element from b[i - 2] , we use i - 1th index for travel purpose. So we can include it in the formula

Recurrence relation :

dp[i][a] = max (dp[i - 1][a] , dp[i - 2][b]) + a[i]

dp[i][b] = max (dp[i - 1] [b] , dp[i - 2][a]) + b[i]

For an index we have only two possibilities

Either we came from the same array → i - 1

We came from another array → i - 2

Code :

class Solution {

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

int [][] dp = new int[n + 1] [3];

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

dp[1][2] = B[0];

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

dp[i][1] = Math.max(dp[i - 1][1] , dp[i - 2][2]) + A[i];

dp[i][2] = Math.max(dp[i - 1][2] , dp[i - 2][1]) + B[i];

}

return Math.max(dp[n][1] , dp[n][2]);

}

}

Follow Up :

What if there is 3 cities

class Solution {

public int maxMoney(int [] A , int [] B ,int [] C int n){

int [][] dp = new int[n + 1] [4];

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

dp[1][2] = B[0];

dp[1][3] = C[0]

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

dp[i][1] = Math.max(dp[i - 1][1] , dp[i - 2][2] , dp[i - 2][3]) + A[i];

dp[i][2] = Math.max(dp[i - 1][2] , dp[i - 2][1] , dp[i - 2][3]) + B[i];

dp[i][3] = Math.max(dp[i - 1][3] , dp[i - 2][1] , dp[i - 2][2]) + C[i];

}

return Math.max(dp[n][1] , dp[n][2]);

}

}