

A

dp[l][r] = sum[l][r]-min(dp[l+1][r],dp[l][r-1]) (l < r)

dp[l][r] = 0 (l>r)

dp[l][r] = a[l][r] (l = r)

sum[l][r] means sum of integers between interval[l,r]

dp[l][r] means player get maxim value in interval[l,r]

B

int FirstPlayerSum(A, l, r)

{

if (l == r)

return dp[l][r] = A[l][r]; // if l==r, there is only one number to choose

if (vis[l][r]) // if vis[l][r] == true, states dp[l][r] has been calculated

return dp[l][r];

vis[l][r] = true; // if vis[l][r] == false, calculate dp[l][r] and set vis[l][r] as true

dp[l][r] = sum[l][r] - min(FirstPlayerSum(A, l + 1, r), FirstPlayerSum(A, l, r - 1)); // dp[l][r] will be transferred from [l+1,r] or [l,r-1], and the sum of [l,r] is sum[l][r]. so dp[l][r] is sum[l][r] minus other player’s maximum value in [l+1,r] or [l,r-1]. We can obviously find a conclusion that sum of two players’ value is sum[l][r]. Thus dp[l][r] = sum[l][r] – other’s value, finally,dp[l][r] = sum[l][r]-min(dp[l+1][r],dp[l][r-1]);

return dp[l][r];

}

C

Due to array dp in algorithm, there are most n^2 probable situations need to calculate. In addition, every situation will only be calculated once. So the time complexity of algorithm is O(n^2).