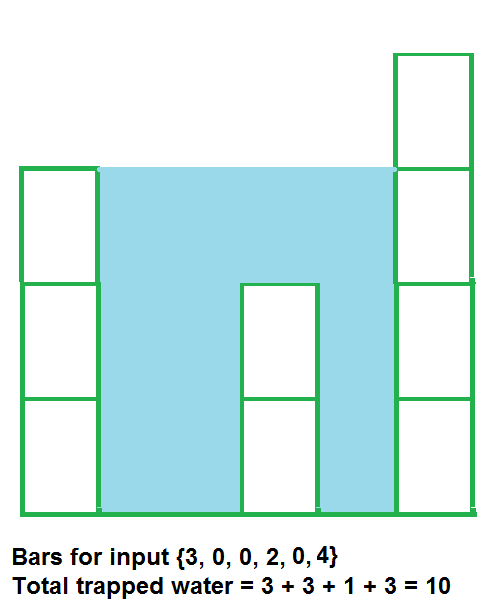
Medium Dynamic Programming

# **Trapping Rain Water**



#define lli long long int

class Solution{

// Function to find the trapped water between the blocks.

public:

long long trappingWater(int arr[], int n){

lli ans = 0, l = 0, r = n-1, lmax = 0, rmax = 0;

while(l <= r) {

if(arr[l] < arr[r]) {

if(arr[l] >= lmax) lmax = arr[l];

else ans += (lmax - arr[l]);

l++;

}

else {

if(arr[r] >= rmax) rmax = arr[r];

else ans += (rmax - arr[r]);

r--;

}

}

return ans;

}

};

# Maximum Product Subarray

#define lli long long int

class Solution{

lli max(lli a, lli b) {

return a > b ? a : b;

}

lli min(lli a, lli b) {

return a < b ? a : b;

}

public:

long long maxProduct(vector<int> arr, int n) {

lli ans = arr[0], ma = arr[0], mi = arr[0];

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

if(arr[i] < 0) swap(ma, mi);

ma = max(arr[i], ma \* arr[i]);

mi = min(arr[i], mi \* arr[i]);

ans = max(ans, ma);

}

return ans;

}

};