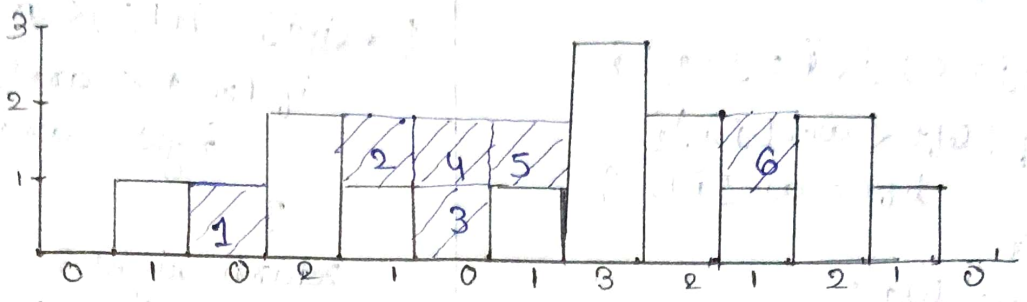
Array

Trapping Rain Water Problem



In this, we will be having an array which contains the positive or non-negative numbers. which tells the height of the building

Brute force Approach

In this Approach, we will iterate through each element ~~and~~ and then find the leftMax & RightMax for that element. if leftMax and RightMax is greater then the element then we will substract the Minimum among the leftMax & RightMax with the element and add then in another variable name Total After the all the iteration, the variable Total contains the result of the asking question.

In a simple way, we can say that

```
if (arr[i] < leftMax && arr[i] < RightMax){
```

$$\sum_{i=0}^{n-1} \min(leftMax, RightMax) - arr[i]$$

left Max function

```
int leftMax(int i, int arr[]){
    int left = arr[0];
for(int j = 0; j < i; j++){
    if (left < arr[j]){
        left = arr[j]; }
    }
  return left;
}
```

Right Max function

```
int rightMax (int i ; int arr[]){
    int right = arr[i+1];
  int n = sizeof(arr) / sizeof(arr[0])
for (int j = i+1; j < n ; j++){
    if (right < arr[j])
        right = arr[j];
    }
  return right;
}
```

```
// Function to return maximum water logged in.
  int maxWater(int arr[], int n){
      int n = sizeof(arr) / sizeof(arr[0]);
      int Total = 0;
    for (int i = 1; i < n-1; i++){
        left = leftMax (i, arr[]);
        right = rightMax (i, arr[]);
        if (left > arr[i] && right > arr[i]){
            Total = (left > right ? right : left) - arr[i];
        }
    return Total;
  }
// Driver code
  int main(){
      int arr[] = {2,1,5,3,1,0,4};
      printf("%d", maxWater(arr));
    return 0;
  }
```

Better Approach

In better approach, we will be using two array of same size named as prefix and suffix

In prefix, the height of building will be the element till that elements. and for prefix iteration will be take from left to right.

Same in suffix, but the difference is that it will iterate from right to left.

```
int waterMax(int arr[]){
    int n = sizeof(arr)/sizeof(arr[0]);
    int prefix[n]; int suffix[n];
    prefix[0] = arr[0]; suffix[n-1] = arr[n-1];
    for (i=1; i<n; i++){
        if (prefix[i-1] < arr[i])
            prefix[i] = arr[i];
        else
            prefix[i] = prefix[i-1];
    }
    for(i=n-2; i>=0; i--){
        if (suffix[i+1] < arr[i]){
            suffix[i] = arr[i];}
        else
            suffix[i] = suffix[i+1];
    }
    int result=0
    for(int i=0; i<n; i++){
        if(arr[i] < prefix[i] && arr[i] <suffix[i]){
            result+= (prefix[i]<suffix[i] ? prefix[i]:suffix[i])-
                                                arr[i];
        }
    }
    return result;
}
```

optimal approach

In this approach, We will use two pointer approach, in which one pointer will be on the left size and other at the right size, & check the items on each size if then the smallest value pointer will iterate and check whether the next node is heigher or not.

let discuss with the code :-

```
int WaterMax (int arr[]){

// Variable we will be using
// Two pointer as l & r
  l=0 ;  r= sizeof (arr)/(sizeof (arr[0]) ;
// leftMax & Right max & Total vauible
  lm = rm = Total =0;

while ( l<r){
   if ( arr[l] < arr[r]){
      if ( lm < arr[l]){
           lm = arr[l];}
      else
        Total += lm - arr[l];
      l++;   // incrementing left pointer
   }
   else {
      if ( rm < arr[r])
           rm = arr[r];
      else
        Total += rm - arr[r];
      r=r-1;  // decrementing right pointer
   }
return Total ;
}
```

In this we will iterate that pointer which is smaller then we will compare it will left max value if the 1st if condition i.e (l < r) is true

In this, if we will compare it with left if left max is greater then means water will be logged in then we will substract it with that item and added it into total

if not then we will replace leftMax with item value and increment left pointer

Same will go with right pointer only difference is that it will decrement at the last.