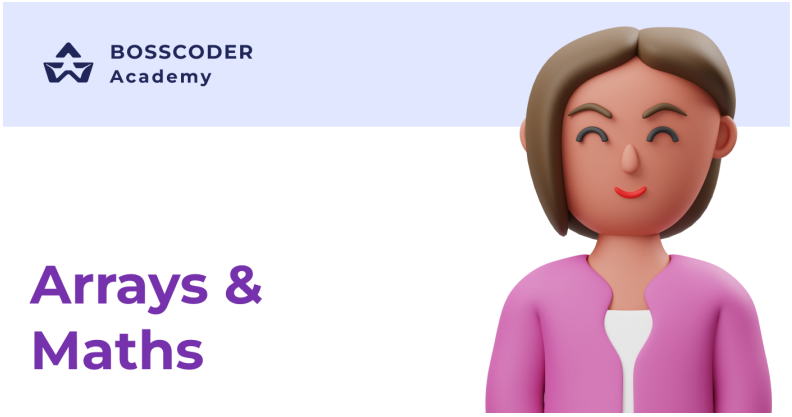
Arrays & Maths

Gyanu Mayank

30th January, 2023



Agenda

1. Introduction to Array/Dynamic size array
2. Problem-solving

Array

An array is a data structure which stores homogenous (similar) data items. An array variable is used to store more than one data item of the same data type at contiguous memory locations.

Limitation of Array

1. The array is a static data structure with a fixed size and the size of the array should be known in advance so, the size of the array cannot be modified further and hence no modification can be done during runtime.

wastage.

We can overcome this problem of fixed size by using **Dynamic Arrays.**

Dynamic Array
A dynamic array expands as we add more elements. So we *don't* need to determine the size ahead of time. Usually, the size gets doubled when we want to do insertion and no space is left.
Example of **Dynamic Arrays**
1. C++ - Vector
2. Java - ArrayList
3. Python - List

Time Complexity
Insertion
1. Best Case- O(1)
2. Worst Case-O(n) [If the dynamic array doesn't have any room for the new item, it will need to expand, which takes $O(n)$ time.]

Deletion - O(n)

Problems

1. Max Value
Given an array of N positive integers. The task is to find the maximum value of |arr[i] – arr[j]| + |i – j|, where 0 <= i, j <= N – 1 and arr[i], arr[j] belong to the array.
Example
Input : N = 4, arr[] = { 4, 5, 6, 8 }
Output: 4
Explanation:

value.

Brute Force Method

We can iterate in two for loops and get the maximum value for the expression.

Implementation

```
#include <bits/stdc++.h>
using namespace std;

int findMax(int arr[], int n)
{
  int ans = 0;
  for (int i = 0; i < n; i++)
    for (int j = 0; j < n; j++)
      ans = max(ans, abs(arr[i] - arr[j]) +
  return ans;
}

int main()
{
  int n;
  cin >> n;
  int arr[n];
  for (int i = 0; i < n; i++)
    cin >> arr[i];
  cout << findMax(arr, n) << endl;
  return 0;
}
```

Optimised Approach

After removing the mod signs the following 2 equation is formed and we have to find the maximum value of them-

1.(arr[i] + i) – (arr[j] + j)

maximum difference between the two values of these two arrays.

Implementation

```
#include <bits/stdc++.h>
using namespace std;

int findMax(int arr[], int n)
{
  int temp1, temp2;
  int max1 = INT_MIN, max2 = INT_MIN;
  int min1 = INT_MAX, min2 = INT_MAX;
  for (int i = 0; i < n; i++)
  {
    temp1 = arr[i] + i;
    temp2 = arr[i] - i;
    max1 = max(max1, temp1);
    min1 = min(min1, temp1);
    max2 = max(max2, temp2);
    min2 = min(min2, temp2);
  }
  return max((max1 - min1), (max2 - min2));
}

int main()
{
  int n;
  cin >> n;
  int arr[n];
  for (int i = 0; i < n; i++)
    cin >> arr[i];
  cout << findMax(arr, n) << endl;
  return 0;
}
```

2. [Trapping Rain Water]

Given **n** non-negative integers representing an elevation map where the width of each bar is **1**, compute how much water it can trap after rain.

Example

Input: n=12 height= [0,1,0,2,1,0,1,3,2,1,2,1]

Output: 6

Explanation: The above elevation map (black section) is represented by an array [0,1,0,2,1,0,1,3,2,1,2,1]. In this case, 6 units of rain water (blue section) are trapped.

Brute force Method

For each index, find the amount of water that can be stored and we have to sum it up. If we observe the amount the water stored at a particular index is the minimum or maximum elevation to the left and right of the index minus the elevation at that index.

Implementation

```
#include <bits/stdc++.h>
using namespace std;

int trap(vector<int> &arr)
```

```
  int waterTrapped = 0;
  for (int i = 0; i < n; i++)
  {
    int j = i;
    int leftMax = 0, rightMax = 0;
    while (j >= 0)
    {
      leftMax = max(leftMax, arr[j]);
      j--;
    }
    j = i;
    while (j < n)
    {
      rightMax = max(rightMax, arr[j]);
      j++;
    }
    waterTrapped += min(leftMax, rightMax)
  }
  return waterTrapped;
}

int main()
{
  int n;
  cin >> n;
  vector<int> arr(n);
  for (int i = 0; i < n; i++)
    cin >> arr[i];
  cout << trap(arr) << endl;
}
```

Better Approach

We can take 2 array prefix and suffix array and precompute the leftMax and rightMax for

trapped at each index.

Implementation

```
#include <bits/stdc++.h>
using namespace std;

int trap(vector<int> &arr)
{
  int n = arr.size();
  int prefix[n], suffix[n];
  prefix[0] = arr[0];
  for (int i = 1; i < n; i++)
  {
    prefix[i] = max(prefix[i - 1], arr[i]);
  }
  suffix[n - 1] = arr[n - 1];
  for (int i = n - 2; i >= 0; i--)
  {
    suffix[i] = max(suffix[i + 1], arr[i]);
  }
  int waterTrapped = 0;
  for (int i = 0; i < n; i++)
  {
    waterTrapped += min(prefix[i], suffix[i
  }
  return waterTrapped;
}

int main()
{
  int n;
  cin >> n;
  vector<int> arr(n);
  for (int i = 0; i < n; i++)
```

```
}
```

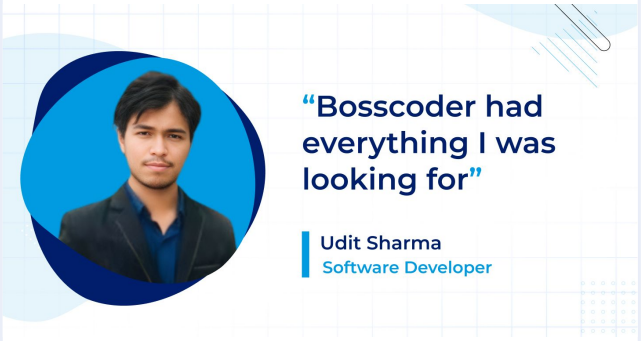
Optimised Solution

We can take 2 pointers l(left pointer) and r(right pointer) pointing to the 0th and last index respectively. Take two variables leftMax and rightMax and initialise them to 0. If height[l] is less than or equal to height[r] then if leftMax is less than height[l] update leftMax to height[l] else add leftMax-height[l] to your final answer and move the l pointer to the right i.e l++. If height[r] is less than height[l], then now we are dealing with the right block. If height[r] is greater than rightMax, then update rightMax to height[r] else add rightMax-height[r] to the final answer. Now move r to the left. Repeat these steps till l and r crosses each other.

Implementation

```
#include <bits/stdc++.h>

using namespace std;
int trap(vector<int> &arr)
{
  int n = arr.size();
  int left = 0, right = n - 1;
  int res = 0;
  int maxLeft = 0, maxRight = 0;
  while (left <= right)
  {
    if (arr[left] <= arr[right])
    {
      if (arr[left] >= maxLeft)
```

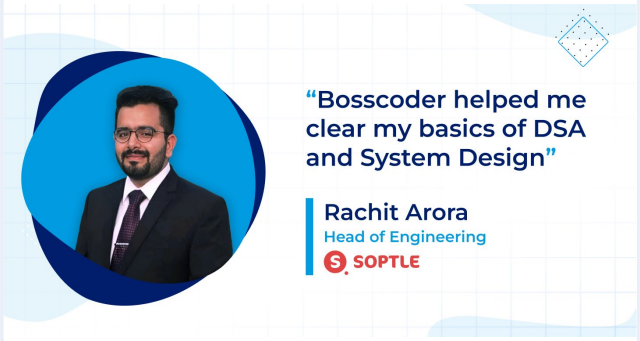
```
      }
      else
      {
        res += maxLeft - arr[left];
      }
      left++;
    }
    else
    {
      if (arr[right] >= maxRight)
      {
        maxRight = arr[right];
      }
      else
      {
        res += maxRight - arr[right];
      }
      right--;
    }
  }
  return res;
}

int main()
{

  int n;
  cin >> n;
  vector<int> arr(n);
  for (int i = 0; i < n; i++)
    cin >> arr[i];
  cout << trap(arr) << endl;
}
```



Bosscoder had everything I was looking for

Read

Date: 12th May, 2023



Bosscoder helped me clear my basics of DSA and System Design

Read

Date: 5th May, 2023

View All blogs