GeeksforGeeks

A computer science portal for geeks

GeeksQuiz

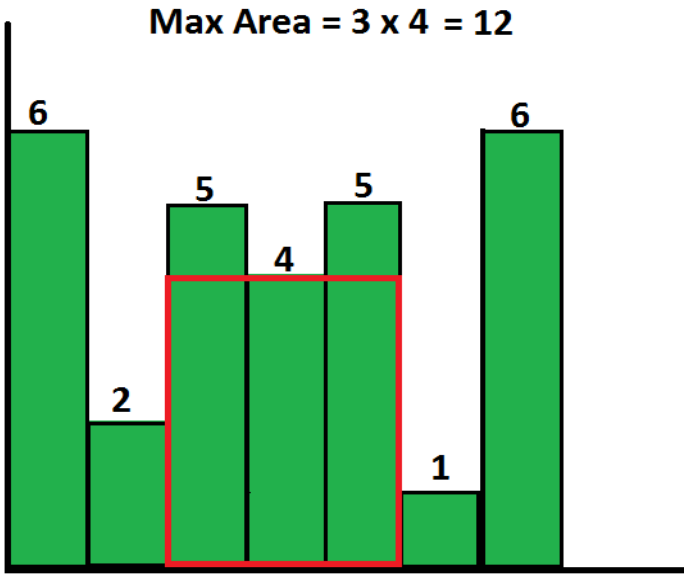
- Home
- Algorithms
- DS
- GATE
- Interview Corner
- Q&A
- C
- C++
- Java
- Books
- Contribute
- Ask a Q
- About

Array
Bit Magic
C/C++
Articles
GFacts
Linked List
MCQ
Misc
Output
String
Tree
Graph

Largest Rectangular Area in a Histogram | Set 1

Find the largest rectangular area possible in a given histogram where the largest rectangle can be made of a number of contiguous bars. For simplicity, assume that all bars have same width and the width is 1 unit.

For example, consider the following histogram with 7 bars of heights {6, 2, 5, 4, 5, 2, 6}. The largest possible rectangle possible is 12 (see the below figure, the max area rectangle is highlighted in red)



A **simple solution** is to one by one consider all bars as starting points and calculate area of all rectangles starting with every bar. Finally return maximum of all possible areas. Time complexity of this solution would be O(n^2).

We can use **Divide and Conquer** to solve this in O(nLogn) time. The idea is to find the minimum value in the given array. Once we have index of the minimum value, the max area is maximum of following three values.
a) Maximum area in left side of minimum value (Not including the min value)
b) Maximum area in right side of minimum value (Not including the min value)
c) Number of bars multiplied by minimum value.
The areas in left and right of minimum value bar can be calculated recursively. If we use linear search to find the minimum value, then the worst case time complexity of this algorithm becomes O(n^2). In worst case, we always have (n-1) elements in one side and 0 elements in other side and if the finding minimum takes O(n) time, we get the recurrence similar to worst case of Quick Sort.
How to find the minimum efficiently? Range Minimum Query using Segment Tree can be used for this. We build segment tree of the given histogram heights. Once the segment tree is built, all range minimum queries take O(Logn) time. So over all complexity of the algorithm becomes.

Overall Time = Time to build Segment Tree + Time to recursively find maximum area

Time to build segment tree is O(n). Let the time to recursively find max area be T(n). It can be written as following.
T(n) = O(Logn) + T(n-1)
The solution of above recurrence is O(nLogn). So overall time is O(n) + O(nLogn) which is O(nLogn).

Following is C++ implementation of the above algorithm.

```
// A Divide and Conquer Program to find maximum rectangular area in a histogr
#include <math.h>
#include <limits.h>
#include <iostream>
using namespace std;

// A utility function to find minimum of three integers
```

```
int max(int x, int y, int z)
{  return max(max(x, y), z); }

// A utility function to get minimum of two numbers in hist[]
int minVal(int *hist, int i, int j)
{
    if (i == -1) return j;
    if (j == -1) return i;
    return (hist[i] < hist[j])? i : j;
}

// A utility function to get the middle index from corner indexes.
int getMid(int s, int e)
{   return s + (e -s)/2; }

/*  A recursive function to get the index of minimum value in a given range o
    indexes. The following are parameters for this function.

    hist   --> Input array for which segment tree is built
    st    --> Pointer to segment tree
    index --> Index of current node in the segment tree. Initially 0 is
              passed as root is always at index 0
    ss & se  --> Starting and ending indexes of the segment represented by
                 current node, i.e., st[index]
    qs & qe  --> Starting and ending indexes of query range */
int RMQUtil(int *hist, int *st, int ss, int se, int qs, int qe, int index)
{
    // If segment of this node is a part of given range, then return the
    // min of the segment
    if (qs <= ss && qe >= se)
        return st[index];

    // If segment of this node is outside the given range
    if (se < qs || ss > qe)
        return -1;

    // If a part of this segment overlaps with the given range
    int mid = getMid(ss, se);
    return minVal(hist, RMQUtil(hist, st, ss, mid, qs, qe, 2*index+1),
                  RMQUtil(hist, st, mid+1, se, qs, qe, 2*index+2));
}

// Return index of minimum element in range from index qs (quey start) to
// qe (query end).  It mainly uses RMQUtil()
int RMQ(int *hist, int *st, int n, int qs, int qe)
{
    // Check for erroneous input values
    if (qs < 0 || qe > n-1 || qs > qe)
    {
        cout << "Invalid Input";
        return -1;
    }
```

```
    return RMQUtil(hist, st, 0, n-1, qs, qe, 0);
}

// A recursive function that constructs Segment Tree for hist[ss..se].
// si is index of current node in segment tree st
int constructSTUtil(int hist[], int ss, int se, int *st, int si)
{
    // If there is one element in array, store it in current node of
    // segment tree and return
    if (ss == se)
       return (st[si] = ss);

    // If there are more than one elements, then recur for left and
    // right subtrees and store the minimum of two values in this node
    int mid = getMid(ss, se);
    st[si] =  minVal(hist, constructSTUtil(hist, ss, mid, st, si*2+1),
                      constructSTUtil(hist, mid+1, se, st, si*2+2));
    return st[si];
}

/* Function to construct segment tree from given array. This function
   allocates memory for segment tree and calls constructSTUtil() to
   fill the allocated memory */
int *constructST(int hist[], int n)
{
    // Allocate memory for segment tree
    int x = (int)(ceil(log2(n))); //Height of segment tree
    int max_size = 2*(int)pow(2, x) - 1; //Maximum size of segment tree
    int *st = new int[max_size];

    // Fill the allocated memory st
    constructSTUtil(hist, 0, n-1, st, 0);

    // Return the constructed segment tree
    return st;
}

// A recursive function to find the maximum rectangular area.
// It uses segment tree 'st' to find the minimum value in hist[l..r]
int getMaxAreaRec(int *hist, int *st, int n, int l, int r)
{
    // Base cases
    if (l > r)  return INT_MIN;
    if (l == r)  return hist[l];

    // Find index of the minimum value in given range
    // This takes O(Logn)time
    int m = RMQ(hist, st, n, l, r);

    /* Return maximum of following three possible cases
       a) Maximum area in Left of min value (not including the min)
       a) Maximum area in right of min value (not including the min)
       c) Maximum area including min */
```

```
    return max(getMaxAreaRec(hist, st, n, l, m-1),
               getMaxAreaRec(hist, st, n, m+1, r),
               (r-l+1)*(hist[m]) );
}

// The main function to find max area
int getMaxArea(int hist[], int n)
{
    // Build segment tree from given array. This takes
    // O(n) time
    int *st = constructST(hist, n);

    // Use recursive utility function to find the
    // maximum area
    return getMaxAreaRec(hist, st, n, 0, n-1);
}

// Driver program to test above functions
int main()
{
    int hist[] =  {6, 1, 5, 4, 5, 2, 6};
    int n = sizeof(hist)/sizeof(hist[0]);
    cout << "Maximum area is " << getMaxArea(hist, n);
    return 0;
}
```

Output:

```
Maximum area is 12
```

This problem can be solved in linear time. We will soon be publishing the linear time solution as next set.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Related Topics:

- Linearity of Expectation
- Iterative Tower of Hanoi
- Count possible ways to construct buildings
- Build Lowest Number by Removing n digits from a given number
- Set Cover Problem | Set 1 (Greedy Approximate Algorithm)
- Find number of days between two given dates
- How to print maximum number of A's using given four keys
- Write an iterative O(Log y) function for pow(x, y)

Tags: Divide and Conquer

Writing code in comment? Please use ideone.com and share the link here.

26 Comments GeeksforGeeks 1 Login

Recommend 1 Share Sort by Newest

Join the discussion…

GOPI GOPINATH • 4 days ago

An O(n^2) solution.

http://ideone.com/frcKjc

Reply • Share ›

zenlotus • 8 months ago

there is typo of the example in the graph, the array should be {6, 2, 5, 4, 5, 1, 6} , the 2nd to the last one is 1 not 2.

1 Reply • Share ›

AlienOnEarth • a year ago

Why do we need to use Segmentation tree for min value? why cant we just use Min Heap? Can anyone please explain this?

1 Reply • Share ›

Nilendu Das ➚ AlienOnEarth • 10 months ago

The range over which the min-heap will be made changes in every segment. So you have to build as many segment trees as the number of segments. So, segment tree is best fit for this purpose. :)

1 Reply • Share ›

zealfire • a year ago

//would be great if someone verifies this solution

```
#include<stdio.h>

#include<algorithm>

using namespace std;

int find1(int arr[],int l,int mid,int h)

{

int k1=0,key,l1;

if((h-l)%2!=0)
```

mid=mid+1;

key=min(min(arr[mid],arr[mid-1]),arr[mid+1]);

l1=mid;

see more

1 • Reply • Share ›

A Kumar • 2 years ago

A very simple iterativer approach without stack and recursion . Please correct me if I am wrong.

```
 #include "stdio.h"
int getMaxArea(int hist[], int n )
{
        int height;
        int hist_index =0;
        int width =0;
        int maxRect =0;
        int rect ;

        for(int i = 0 ;i< n ; i++)
        {
                width = 0;
                height = hist[i];
                hist_index = i;
                //move left  till start
```

see more

2 • Reply • Share ›

pefullarton ➚ A Kumar • 2 years ago

Yes this is correct, but the complexity is O(n^2).

3 • Reply • Share ›

innosam • 2 years ago

A straight forward Recursive solution based on the first approach:
http://wp.me/3bJAF

1 • Reply • Share ›

Sachin Jain • 2 years ago

Yes, my approach is same as above one. Thanks for optimized solution.

1 • Reply • Share ›

Rohit • 2 years ago

```
public static int LargestHist(int[] arr)
                {
                        int curSide,curArea;
                        curSide=arr[0];curArea=arr[0];
                        int maxArea=arr[0],n=1;
                        for(int i=1;i<arr.length;i++)
                        {
                                n++;
                                curSide=Math.min(curSide,arr[i]);
                                curArea=curSide*n;
                                System.out.println(curArea);
                                if(curArea>maxArea)
                                {
                                        maxArea=curArea;

                                }
                                else if(n>1)
                                {
```

see more

1 • Reply • Share ›

GeeksforGeeks • 2 years ago

Sachin: Please see http://www.geeksforgeeks.org/l... . Let us know if your approach is different from the link.

• Reply • Share ›

Sachin Jain • 2 years ago

[sourcecode language="C++"]

```
#include <iostream>
#include <vector>
#include <stack>

int getMaxRectArea(std::vector<int>& histogram) ;

int main()
{
int input[] = {6, 2, 5, 4, 5, 1, 6} ;
// int input[] = {6, 2, 5, 4, 5, 2, 6} ;
// int input[] = {6, 1, 5, 4, 5, 2, 6} ;
// int input[] = {3, 4, 5, 6, 5, 4, 3} ;
std::vector<int> histogram(input, input + sizeof(input) / sizeof(int)) ;
```

```
std::cout << getMaxRectArea(histogram) << std::endl ;

return 0 ;
}
```

see more

• Reply • Share ›

Sachin Jain • 2 years ago

```
#include <iostream>
#include <vector>
#include <stack>

int getMaxRectArea(std::vector<int>& histogram) ;.

int main()
{
int input[] = {6, 2, 5, 4, 5, 1, 6} ;.
// int input[] = {6, 2, 5, 4, 5, 2, 6} ;.
// int input[] = {6, 1, 5, 4, 5, 2, 6} ;.
// int input[] = {3, 4, 5, 6, 5, 4, 3} ;.
std::vector<int> histogram(input, input + sizeof(input) / sizeof(int)) ;.

std::cout << getMaxRectArea(histogram) << std::endl ;.

return 0 ;.
}
```

see more

• Reply • Share ›

Sachin Jain • 2 years ago

Following is my linear time complexity algorithm:
Input: histogram
Output: largest rectangular area possible in a given histogram where the largest rectangle can be made of a number of contiguous bars.

Algorithm:
push index of first element in histogram array on stack.
scan through rest of the array.
while current element is less than element on top of stack.
maximum rectangular area to right of element on top of stack containing itself is height of bar * number of bars between current element and element on top of stack.
pop stack

push current element on stack.
while stack is not empty.
maximum rectangular area to right of element on top of stack containing itself is height of bar * number of bars between current element and past last element in histogram.
pop stack

see more

• Reply • Share ›

Liu Zhaoliang • 2 years ago

http://poj.org/problem?id=2559

1 • Reply • Share ›

Liu Zhaoliang • 2 years ago

?????????

• Reply • Share ›

Lin Guo • 2 years ago

```
public class Solution {
public int largestRectangleArea(int[] height) {
if(height.length==0) return 0;
int i=0;
int max=0;
Stack<Integer> stack=new Stack<Integer>();
stack.push(0);.
while(i<height.length ||! stack.isEmpty())
{
if(i<height.length &&( stack.isEmpty() || height[i]>=height[stack.peek()])).
{
stack.push(i); i++;
}else
{
int top=stack.pop();
max=Math.max(max, height[top]*(stack.isEmpty()? i: i-stack.peek()-1));
}
}
return max;
}
}
```

• Reply • Share ›

Pathikrit • 2 years ago

Here is a short 2-liner in Scala:

[sourcecode language="Scala"]
def maxRectangleInHistogram(heights: Seq[Int]): Int = if (heights isEmpty) 0 else {
val (left, smallest :: right) = heights splitAt (heights indexOf heights.min)
Seq(maxRectangleInHistogram(left), smallest * heights.length,
maxRectangleInHistogram(right)).max
}

Rax • 2 years ago

Java Solution:

public class MaxArea {

/**
* @param args
*/
public static void main(String[] args) {
// TODO Auto-generated method stub
int[] a = {6, 2, 5, 4, 5, 2, 6};
System.out.println(new MaxArea().findMaxArea(a));
}

public int findMaxArea(int[] A){
int max_Area = Integer.MIN_VALUE;
int n = A.length;
if(n==0)return 0;
LLStack stack = new LLStack();
int continuosSubArray [] = new int[n];

anonymous • 2 years ago

Please correct me If I am wrong

```
 #include<stdio.h>
#include<conio.h>
int max(int a,int b){
    return a>b?a:b;
}
int max(int a,int b,int c){
     return max(max(a, b), c);
}


int findMinIndex(int a[],int start,int end){
```

```
int findMinIndex(int a[],int start,int end){
    int i;
    int min = a[start];
    int min_index = start;
    for(i=start+1;i<end;i++){
                            if(a[i]<min){
                                        min = a[i];
```

see more

• Reply • Share ›

Mohammad Samiul Islam • 2 years ago

Using stack? How?

• Reply • Share ›

?????? ????? • 2 years ago

eita toh mone hoy stack dia aro easily kora jay..

• Reply • Share ›

Mohammad Samiul Islam • 2 years ago

The updated article makes a lot more sense now :)

• Reply • Share ›

kk.nitrkl • 2 years ago

This doesn't seem correct. try data {3, 4, 5, 6, 5, 4, 3}

• Reply • Share ›

GeeksforGeeks ➚ kk.nitrkl • 2 years ago

Thanks for pointing this out. There was a flaw in the previous approach. We have updated the post with a new approach and code.

• Reply • Share ›

Guest ➚ GeeksforGeeks • a year ago

can this code work if we have a histogram array with zero entry?????thanks ...

• Reply • Share ›

- Interview Experiences
- Advanced Data Structures
- Dynamic Programming
- Greedy Algorithms
- Backtracking
- Pattern Searching
- Divide & Conquer
- Mathematical Algorithms
- Recursion
- Geometric Algorithms

Popular Posts

- All permutations of a given string
- Memory Layout of C Programs
- Understanding “extern” keyword in C
- Median of two sorted arrays
- Tree traversal without recursion and without stack!
- Structure Member Alignment, Padding and Data Packing
- Intersection point of two Linked Lists
- Lowest Common Ancestor in a BST.
- Check if a binary tree is BST or not
- Sorted Linked List to Balanced BST

Recent Comments

- Nikhil kumar

  public class...

  Print missing elements that lie in range 0 – 99 · 5 minutes ago

- Ashish Aggarwal

  Try Data Structures and Algorithms Made Easy -...

  Algorithms · 27 minutes ago

- Vlad

  Thanks. Very interesting lectures.

  Expected Number of Trials until Success · 1 hour ago

- cfh

  My implementation which prints the index of the...

  Longest Even Length Substring such that Sum of First and Second Half is same · 1 hour ago

- Gaurav pruthi

  forgot to see that part ;)

  Bloomberg Interview | Set 1 (Phone Interview) · 2 hours ago

- saeid aslami

  thanks

  Greedy Algorithms | Set 7 (Dijkstra's shortest path algorithm) · 2 hours ago