GeeksforGeeks

A computer science portal for geeks

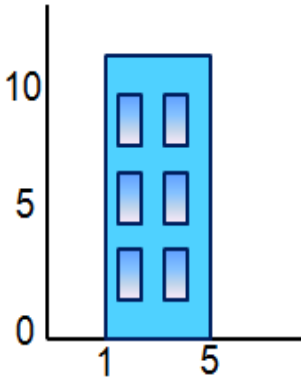
GeeksQuiz

- 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

Divide and Conquer | Set 7 (The Skyline Problem)

Given n rectangular buildings in a 2-dimensional city, computes the skyline of these buildings, eliminating hidden lines. The main task is to view buildings from a side and remove all sections that are not visible.



All buildings share common bottom and every **building** is represented by triplet (left, ht, right)

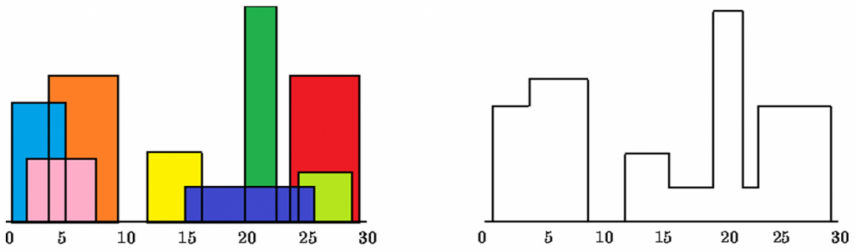
'left': is x coordinated of left side (or wall).
'right': is x coordinate of right side
'ht': is height of building.

For example, the building on right side (the figure is taken from here) is represented as (1, 11, 5)

A **skyline** is a collection of rectangular strips. A rectangular **strip** is represented as a pair (left, ht) where left is x coordinate of left side of strip and ht is height of strip.

Examples:

```
Input: Array of buildings
       { (1,11,5), (2,6,7), (3,13,9), (12,7,16), (14,3,25),
         (19,18,22), (23,13,29), (24,4,28) }
Output: Skyline (an array of rectangular strips)
        A strip has x coordinate of left side and height
        (1, 11), (3, 13), (9, 0), (12, 7), (16, 3), (19, 18),
        (22, 3), (25, 0)
The below figure (taken from here) demonstrates input and output.
The left side shows buildings and right side shows skyline.
```




```
Consider following as another example when there is only one
building
Input:  {(1, 11, 5)}
Output: (1, 11), (5, 0)
```

A **Simple Solution** is to initialize skyline or result as empty, then one by one add buildings to skyline. A building is added by first finding the overlapping strip(s). If there are no overlapping strips, the new building adds new strip(s). If overlapping strip is found, then height of the existing strip may increase. Time complexity of this solution is $O(n^2)$

We can find Skyline in Θ(nLogn) time using **Divide and Conquer**. The idea is similar to Merge Sort,

divide the given set of buildings in two subsets. Recursively construct skyline for two halves and finally merge the two skylines.

How to Merge two Skylines?
The idea is similar to merge of merge sort, start from first strips of two skylines, compare x coordinates. Pick the strip with smaller x coordinate and add it to result. The height of added strip is considered as maximum of current heights from skyline1 and skyline2.
Example to show working of merge:

```
  Height of new Strip is always obtained by takin maximum of following
     (a) Current height from skyline1, say 'h1'.
     (b) Current height from skyline2, say 'h2'
  h1 and h2 are initialized as 0. h1 is updated when a strip from
  SkyLine1 is added to result and h2 is updated when a strip from
  SkyLine2 is added.

  Skyline1 = {(1, 11),  (3, 13),  (9, 0),  (12, 7),  (16, 0)}
  Skyline2 = {(14, 3),  (19, 18), (22, 3), (23, 13),  (29, 0)}
  Result = {}
  h1 = 0, h2 = 0

  Compare (1, 11) and (14, 3).  Since first strip has smaller left x,
  add it to result and increment index for Skyline1.
  h1 = 11, New Height  = max(11, 0)
  Result =   {(1, 11)}

  Compare (3, 13) and (14, 3). Since first strip has smaller left x,
  add it to result and increment index for Skyline1
  h1 = 13, New Height =  max(13, 0)
  Result =  {(1, 11), (3, 13)}

  Similarly (9, 0) and (12, 7) are added.
  h1 = 7, New Height =  max(7, 0) = 7
  Result =   {(1, 11), (3, 13), (9, 0), (12, 7)}

  Compare (16, 0) and (14, 3). Since second strip has smaller left x,
  it is added to result.
  h2 = 3, New Height =  max(7, 3) = 7
  Result =   {(1, 11), (3, 13), (9, 0), (12, 7), (14, 7)}

  Compare (16, 0) and (19, 18). Since first strip has smaller left x,
  it is added to result.
  h1 = 0, New Height =  max(0, 3) = 3
  Result =   {(1, 11), (3, 13), (9, 0), (12, 7), (14, 3), (16, 3)}

Since Skyline1 has no more items, all remaining items of Skyline2
are added
  Result =   {(1, 11), (3, 13), (9, 0), (12, 7), (14, 3), (16, 3),
              (19, 18), (22, 3), (23, 13), (29, 0)}

One observation about above output is, the strip (16, 3) is redundant
(There is already an strip of same height). We remove all redundant
strips.
  Result =   {(1, 11), (3, 13), (9, 0), (12, 7), (14, 3), (19, 18),
              (22, 3), (23, 13), (29, 0)}

In below code, redundancy is handled by not appending a strip if the
previous strip in result has same height.
```

Below is C++ implementation of above idea.

```
// A divide and conquer based C++ program to find skyline of given
// buildings
#include<iostream>
using namespace std;

// A structure for building
struct Building
{
    int left;  // x coordinate of left side
    int ht;    // height
    int right; // x coordinate of right side
};

// A strip in skyline
class Strip
{
    int left;  // x coordinate of left side
    int ht; // height
public:
    Strip(int l=0, int h=0)
    {
        left = l;
        ht = h;
    }
    friend class SkyLine;
};

// Skyline:  To represent Output (An array of strips)
class SkyLine
{
    Strip *arr;   // Array of strips
    int capacity; // Capacity of strip array
    int n;   // Actual number of strips in array
public:
    ~SkyLine() {  delete[] arr;  }
    int count()  { return n;   }

    // A function to merge another skyline
    // to this skyline
    SkyLine* Merge(SkyLine *other);

    // Constructor
    SkyLine(int cap)
    {
        capacity = cap;
        arr = new Strip[cap];
        n = 0;
    }

    // Function to add a strip 'st' to array
    void append(Strip *st)
    {
        // Check for redundant strip, a strip is
        // redundant if it has same height or left as previous
        if (n>0 && arr[n-1].ht == st->ht)
            return;
        if (n>0 && arr[n-1].left == st->left)
        {
```

```
            arr[n-1].ht = max(arr[n-1].ht, st->ht);
            return;
        }

        arr[n] = *st;
        n++;
    }

    // A utility function to print all strips of
    // skyline
    void print()
    {
        for (int i=0; i<n; i++)
        {
            cout << " (" << arr[i].left << ", "
                 << arr[i].ht << "), ";
        }
    }
};

// This function returns skyline for a given array of buildings
// arr[l..h].  This function is similar to mergeSort().
SkyLine *findSkyline(Building arr[], int l, int h)
{
    if (l == h)
    {
        SkyLine *res = new SkyLine(2);
        res->append(new Strip(arr[l].left, arr[l].ht));
        res->append(new Strip(arr[l].right, 0));
        return res;
    }

    int mid = (l + h)/2;

    // Recur for left and right halves and merge the two results
    SkyLine *sl = findSkyline(arr, l, mid);
    SkyLine *sr = findSkyline(arr, mid+1, h);
    SkyLine *res = sl->Merge(sr);

    // To avoid memory leak
    delete sl;
    delete sr;

    // Return merged skyline
    return res;
}

// Similar to merge() in MergeSort
// This function merges another skyline 'other' to the skyline
// for which it is called.  The function returns pointer to
// the resultant skyline
SkyLine *SkyLine::Merge(SkyLine *other)
{
    // Create a resultant skyline with capacity as sum of two
    // skylines
    SkyLine *res = new SkyLine(this->n + other->n);

    // To store current heights of two skylines
    int h1 = 0, h2 = 0;

    // Indexes of strips in two skylines
```

```
    int i = 0, j = 0;
    while (i < this->n && j < other->n)
    {
        // Compare x coordinates of left sides of two
        // skylines and put the smaller one in result
        if (this->arr[i].left < other->arr[j].left)
        {
            int x1 = this->arr[i].left;
            h1 = this->arr[i].ht;

            // Choose height as max of two heights
            int maxh = max(h1, h2);

            res->append(new Strip(x1, maxh));
            i++;
        }
        else
        {
            int x2 = other->arr[j].left;
            h2 = other->arr[j].ht;
            int maxh = max(h1, h2);
            res->append(new Strip(x2, maxh));
            j++;
        }
    }

    // If there are strips left in this skyline or other
    // skyline
    while (i < this->n)
    {
        res->append(&arr[i]);
        i++;
    }
    while (j < other->n)
    {
        res->append(&other->arr[j]);
        j++;
    }
    return res;
}

// drive program
int main()
{
    Building arr[] = {{1, 11, 5}, {2, 6, 7}, {3, 13, 9},
                      {12, 7, 16}, {14, 3, 25}, {19, 18, 22},
                      {23, 13, 29}, {24, 4, 28}};
    int n = sizeof(arr)/sizeof(arr[0]);

    // Find skyline for given buildings and print the skyline
    SkyLine *ptr = findSkyline(arr, 0, n-1);
    cout << " Skyline for given buildings is \n";
    ptr->print();
    return 0;
}

 Skyline for given buildings is
 (1, 11),  (3, 13),  (9, 0),  (12, 7),  (16, 3),  (19, 18),
 (22, 3),  (23, 13),  (29, 0),
```

Time complexity of above recursive implementation is same as Merge Sort.

$T(n) = T(n/2) + \Theta(n)$

Solution of above recurrence is $\Theta(nLogn)$

References:
http://faculty.kfupm.edu.sa/ics/darwish/stuff/ics353handouts/Ch4Ch5.pdf
www.cs.ucf.edu/~sarahb/COP3503/Lectures/DivideAndConquer.ppt

This article is contributed **Abhay Rathi**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

Related Topics:

- Find Union and Intersection of two unsorted arrays
- Pythagorean Triplet in an array
- Maximum profit by buying and selling a share at most twice
- Design a data structure that supports insert, delete, search and getRandom in constant time
- Print missing elements that lie in range 0 – 99
- Iterative Merge Sort
- Group multiple occurrence of array elements ordered by first occurrence
- Given a sorted and rotated array, find if there is a pair with a given sum

Tags: Divide and Conquer

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

Shariq Islam • a month ago
The output in example is wrong though using same building the answer is correct in the end.. Please correct typing mistake it should be (1, 11), (3, 13), (9, 0), (12, 7), (16, 3), (19, 18), (22, 3), (23, 13), (29, 0),

artem • a month ago
we can sort all events (where each event is either any side (both left and right) of the building) and then iterate over this sorted sequence of events maintaining current height
complexity would be equal to NlogN

The Internet • a month ago

I have been tinkering with this algorithm and I am not quite sure it is correct. This set of buildings appears to return the wrong skyline:
{{ 1, 10, 2 },{ 1, 20, 3 }}

I constructed this example specifically, because simply changing the height of the last building in the skyline if you append a building starting at the same left value seemed a bit odd to me. I would have expected the result to be {{1, 20}, {3, 0}}, i.e. the larger building with height 20 is completely blocking the smaller building. Switching the order of the buildings does not appear to work, either.
Is this a valid input? If so, is this code working correctly for everyone?

Reply • Share ›

The Internet • a month ago

Hey, Geeks4Geeks! I really appreciate your site and this is another great post. I would just have one sincerely request of you: please, please make use of some more of the object oriented facilities when programming C++. Do you have some kind of aversion against using vectors? Why manually manage memory and capacity if this code could be written so much shorter and simpler by using some of the STL containers and functions that have been around for decades? Why use pointers all over the place if C++11 r-value references avoid having to copy objects passed to and returned from methods?

I have done some refactoring of your code, as I often do, and here is the result. I have really only done the very basic and it the code length has been reduced significantly and I feel the readability is much higher: http://ideone.com/Fax2XY

Reply • Share ›

The Internet ➚ The Internet • a month ago

*sincere request

Ah, also: I don't quite understand why internal methods like "merge" and "append" are made explicitly public. IMHO, there should only be one public, static method: findSkyline. Its method signature is a bit awkward, too, once you use STL containers, so I added a simpler public method and hid away the rest.

Reply • Share ›

Srikar ED • a month ago

Is the example correct. I think it should be (1, 11), (3, 13), (9, 0), (12, 7), (16, 3), (19, 18), (22, 3), (23,13), (29,0).

1 Reply • Share ›

Srikar ED ➚ Srikar ED • a month ago

Just saw that the final result is having this answer. Please correct the example

description. I was stuck at the top of page for a lot of time without scrolling.

Reply • Share ›

sai • 2 months ago

c implementation of above code plz

Reply • Share ›

paralieze • 2 months ago

Can't we do it using the overlapping interval routine?
ie
for each interval that exists between (leftcoordinate[0] and rightcoordinate[n]) store max height in stack!

Reply • Share ›

zd • 2 months ago

If merge is creating a new skyline it shouldn't be the member of class skyline (at least should be a static member of that class)

Memory leaks here:
res->append(new Strip(arr[l].left, arr[l].ht));
res->append(new Strip(arr[l].right, 0));
and in merge also...

Reply • Share ›

Dhananjayan • 2 months ago

Is the example right? Should it be (1, 11), (3, 13), (9, 0), (12, 7), (16, 3), (19, 18), (22, 3), (29, 0)

Reply • Share ›

Mohit Dhuper • 2 months ago

What is n here ? Is it number of buildings ?

Reply • Share ›

ganglu • 2 months ago

i dont think append is checking correctly for redundant stripes

e.g. {{1, 13, 3}, {1, 11, 5} };

will have output
(1, 11), (1, 13), (3, 11), (5, 0),

need to put away 1, 11

so it should also check if prev 'x' same as current 'x' with smaller height has been entered

1 Reply • Share ›

GeeksforGeeks Mod ganglu • 2 months ago

Thanks for pointing this out. We have added a check for same x as well.

• Reply • Share ›

hseran GeeksforGeeks • 5 days ago

if you use max height in case of two stripes with same x value, {{2 6 7}, {3 13 7}} will result in (2, 6), (3, 13), (7, 6). Isn't it?

• 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

- Follow @GeeksforGeeks

Recent Comments

- Ashish Aggarwal

  Try Data Structures and Algorithms Made Easy -...

  Algorithms · 17 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) · 1 hour ago

- saeid aslami

  thanks

  Greedy Algorithms | Set 7 (Dijkstra’s shortest path algorithm) · 1 hour ago

- Cracker

  Implementation:...

  Implement Stack using Queues · 2 hours ago