CPS

ASSIGNMENT

DECEMBER 16, 2022

MUHAMMAD SAMEER SOHIAL
15000

The Skyline Problem

A city's skyline is the outer shape of the outline formed by all the buildings in that city when viewed from a distance. Given the locations and heights of all the buildings, return the skyline formed by these buildings collectively.

You task is to:

1. **Understand the problem and explain in your words. (200-300 words)**
   There are several rectangular strips that make up a skyline. A pair (left, ht) that represents a rectangular strip has left as the x coordinate for the left side and ht as the strip's height. All structures have a common bottom, and each is represented by a triplet (left, ht, right). If the locations and heights of all the buildings are provided to you, create a software that outputs the skyline created by all of these structures taken together. Location of building is specified using tuple [A1,A2,h] where 'A1' is starting point of a building, 'A2' is where building ends and 'h' is height of the building.

   For example:

   if the input is an array of building co-ordinates: [[a,b,c], [x,y,z]] then the output should be the skyline specified as [[a, b], [c, o], [X, y], [z, o]]

2. **Can this problem be solved using Divide and Conquer approach or Dynamic Programming approach? State the reason of your selection.**

   Yes this problem is solved using both Divide & Conquer approach and the Dynamic programming approach,

   - An easy approach is to begin the skyline or outcome as empty.
   - then gradually add buildings to it.
   - By initially identifying the overlapping strip, a building is then added (s).
   - In the lack of any overlap, the new structure inserts a new strip (s).

   Steps Involved:

   - ✓ Sort the array of buildings in order.
   - ✓ Find the critical points among all the buildings.
   - ✓ Run a loop from the smallest critical point to the largest.
   - ✓ Add all the buildings, whose starting point is smaller than the critical value and whose ending point is larger than the critical value, to a set.

3. Write the code and explain it. (You can copy code from internet but explanation is mandatory

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

// A structure for building
struct Building {
        // x coordinate of left side
        int left;
```

```
        // height
        int ht;

        // x coordinate of right side
        int right;
};

// A strip in skyline
class Strip {
        // x coordinate of left side
        int left;

        // height
        int ht;

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 {
        // Array of strips
        Strip* arr;

        // Capacity of strip array
        int capacity;

        // Actual number of strips in array
        int n;

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;
}

// Driver Function
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;
}
```

Output:

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

Explaination:

1. Height of the new Strip is always obtained btakingin a maximum of following
   (a) Current height from skyline1, say 'h1'.
   (b) Current height from skyline2, say 'h2'
2. h1 and h2 are initialized as 0.
3. h1 is updated when a strip from SkyLine1 is added to the sult and h2 is updated when a strip from SkyLine2 is added.
4. Compare (1, 11) and (14, 3). Since the first strip has a smaller left x.
5. add it to the result and increment index for Skyline1.
6. Do this comparison for all remaining values.
7. Since Skyline1 has no more items, all remaining items of Skyline2 are added

REF:

https://www.geeksforgeeks.org
https://www.ideserve.co.
https://leetcode.com/problems/the-skyline-problem/