École Polytechnique — CSE202 Design and Analysis of Algorithms

# Midterm Exam

## The Skyline Problem

**Guidelines.** Download `skyline.py` and `test_skyline.py`. Write your solution in `skyline.py`. To test your code, run the file `test_skyline.py`. You don't have to justify the complexity of your programs.

Given rectangular buildings in a 2-dimensional city, the Skyline Problem asks to compute the skyline of these building, that is the outer contour of the silhouette formed by the buildings when viewed from a distance. The main task is to remove all sections of the buildings that are not visible. All buildings rise from a horizontal ground and are therefore represented by a triple `(left,height,right)` where

- `left` is the x coordinate of the left side,
- `right` is the x coordinate of the right side,
- `height` is the height of the building.

A **skyline** is a list of rectangular strips, described from *left to right*. Each strip is represented by a pair `(left,height)` where `left` is the coordinate of the left side of the strip and `height` is its height. Since strips are described from left to right, the right side of a strip is the left side of the *following strip*. The last strip of a skyline must always have height 0. An example list of buildings and the corresponding skyline is given in Figure 1. Given a list $L$ of buildings and a position $x$, we define the *height of* $L$ *at* $x$ by $h_L(x) = \max(\{0\} \cup \{h : (l, h, r) \in L \text{ and } l \leqslant x < r\})$.

**Question 1** (1 point)**.** Complete the function `rects_height_at(L,x)` that computes $h_L(x)$. It must have complexity $O(n)$ where $n$ is the length of $L$. Test your function using `test_skyline.py`.

Given a skyline[1] $S = [(x_1, h_1), \ldots, (x_k, h_k)]$ and a position $x$, we define the *height* $h_S(x)$ *of* $S$ *at* $x$ as follows : if $x < x_1$ or $x \geqslant x_k$ then $h_S(x) = 0$. Otherwise, there exists a unique $i$ such that $x_i \leqslant x < x_{i+1}$ and we let $h_S(x_i) = h_i$. Formally, $S$ is a skyline of a list $L$ of building if $h_L(x) = h_S(x)$ for any $x$ ; this corresponds to the intuitive notion.

In what follows, we will always want functions to return **minimal skylines**. Intuitively, a skyline is minimal if no strip can be removed without changing the height of some position. Formally, a skyline $S = [(x_1, h_1), \ldots, (x_k, h_k)]$ is minimal if for every $i$, $h_i \neq h_{i+1}$. An example of (non-minimal) skyline is given in Figure 2.

**Question 2** (1 point)**.** Complete the function `simplify_skyline(S)` that returns the (unique) minimal skyline that has the same height as $S$ everywhere. It must have complexity $O(n)$ where $n$ is the length of $S$. Test your function using `test_skyline.py`.

We now make the following observation : if $L$ is a list of buildings, we can define the set of *interesting positions* $X(L)$ by

$$X(L) = \{\ell : (\ell, h, r) \in L \text{ for some } h, r\} \cup \{r : (\ell, h, r) \in L \text{ for some } \ell, h\}.$$

It is not hard to see (but you are not asked to verify it) that if $X(L) = \{x_1, \ldots, x_k\}$ with $x_1 < \cdots < x_k$, then the list

$$[(x_i, h_L(x_i)) : i = 1, \ldots, k]$$

is a skyline of $L$, but it might not be minimal.

1. Recall that a skyline is always described from left to right, so $x_1 < \cdots < x_k$.

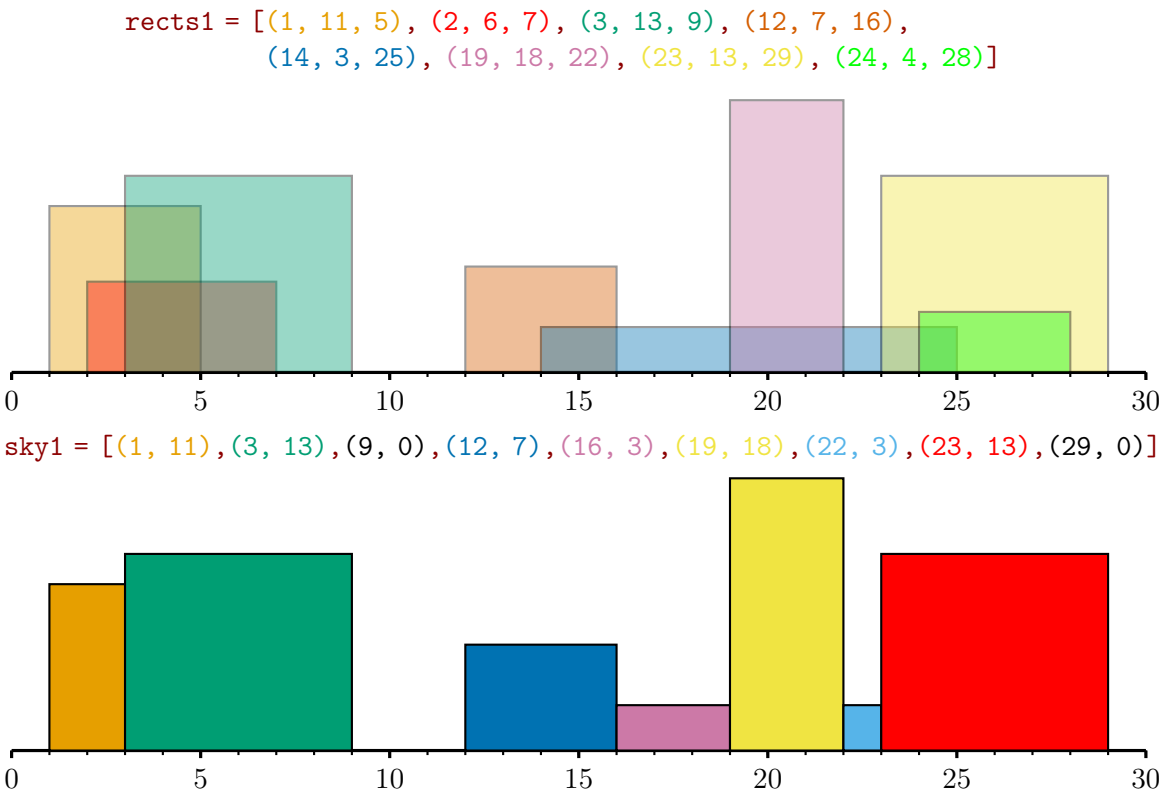


FIGURE 1 – Illustration of the buildings corresponding to `rects1` at the top, and the corresponding skyline `sky1` at the bottom. The colors match the colors in the descriptions.

```
sky1_red = [(1,11),(2,11),(3,13),(5,13),(7,13),(9, 0),(12,7),(14,7),(16, 3),
            (19,18),(22,3),(23,13),(24,13),(25,13),(28,13),(29,0)]
sky1 = [(1,11),(3,13),(9,0),(12,7),(16,3),(19,18),(22,3),(23,13),(29,0)]
```

FIGURE 2 – Example of non-minimal skyline (`sky1_red`) and an equivalent minimal skyline (`sky1`). Here `sky1` is the same skyline as in Figure 1.

```
S3 = [(1, 11), (5, 6), (7, 0), (12, 7), (16, 0), (19, 18), (22, 0), (24, 4), (28, 0)]
S4 = [(3, 13), (9, 0), (14, 3), (23, 13), (29, 0)]
S = [(1,11), (3,13), (9,0), (12,7), (16,3), (19,18), (22,3), (23,13), (29,0)]
```

FIGURE 3 – Example of two minimal skylines `S3` and `S4` and the resulting (minimal) skyline `S` of the merge. Here `S` is the same skyline as in Figure 1.

```
S5 = [(1, 10), (3, 5), (8, 0)]
S6 = [(1,20), (3, 2), (10,0)]
T = [(1, 20), (3, 5), (8, 2), (10, 0)]
```

FIGURE 4 – Example of two minimal skylines `S5` and `S6` and the resulting (minimal) skyline `T` of the merge. Here some coordinates of the two lists are the same, special care must be taken when merging.

**Question 3** (2 points)**.** Using the function `rects_height_at` of question 1, complete the function `skyline_naive(L)` that returns the (unique) minimal skyline of $L$. It must have complexity $O(n^2)$ where $n$ is the length of $L$. Test your function using `test_skyline.py`.

We will now see how to compute the skyline in time $O(n \log n)$. The approach is to compute the skyline recursively by dividing the list into two parts, and merging the resulting skylines together. This is very similar to the merge sort algorithm. Suppose $L = L_1 + L_2$ where $L_1, L_2$ are two lists. Let $S_1$ (resp. $S_2$) be the minimal skyline for $L_1$ (resp. $L_2$). Write

$$S_1 = [(x_1, h_1), \cdots, (x_k, h_k)]$$
$$S_2 = [(x'_1, h'_1), \cdots, (x'_\ell, h'_\ell)].$$

The goal is now to merge $S_1$ and $S_2$ into a skyline $S$ for $L$. The idea is similar to the merge of merge sort : start from the first strip of each list, compare the coordinate, pick the smallest one and remove it from the list. There are two details one must be careful about :

— if the coordinate of the two strips are equal then we must remove both, otherwise the resulting skyline would have two strips at the same position which is invalid,

— when picking a strip $(x, h)$ from $S_i$, the strip added to $S$ is $(x, H)$ where $H$ is not necessarily $h$.

The first point can be observed in the example in Figure 4. The second point can be observed in the example in Figure 3. Therefore when merging, one has to maintain the *current height* in $S_1$ and $S_2$. Whenever we pick a strip from a list, we update the current height of the list and the height $H$ added to the skyline $S$ will the maximum of the two current heights.

**Question 4** (4 points)**.** Complete the function `merge_skylines(S1, S2)` that returns the (unique) minimal skyline that is the result of the merge of `S1` and `S2`. It must have complexity $O(n+m)$ where $n$ (resp. $m$) is the length of `S1` (resp. `S2`). Test your function using `test_skyline.py`.

**Question 5** (2 points)**.** Complete the function `skyline_dac(L)` that returns the (unique) minimal skyline of $L$. It must have complexity $O(n \log n)$ where $n$ is the length of $L$. Test your function using `test_skyline.py`.