Alice is planting n_1 flowers $f_1, \ldots, f_{n_1}$ among n_2 rectangular gardens $\mathcal{G}_1, \ldots, \mathcal{G}_{n_2}$. Bob's task is to determine which flowers belong to which gardens. Alice informs Bob that no two gardens overlap; therefore, if a flower belongs to a garden, then the flower belongs to *exactly* one garden and a garden can contain multiple flowers. If a flower f_i does not belong to any garden, then Bob returns an *undefined* garden for f_i.

Each garden $\mathcal{G}_i$ is given by a pair of points $\mathcal{G}_{BL}[i]$ and $\mathcal{G}_{TR}[i]$, representing the bottom left and top right corners of the garden respectively. Each flower is represented by a point $F[i]$ representing its location. Let $n = n_1 + n_2$.

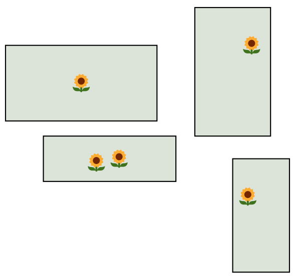

Figure 1: *A collection of $n_1 = 5$ flowers and $n_2 = 4$ gardens.*

Formally, you are given three arrays:

- $\mathcal{G}_{BL} = [(x_1, y_1), \ldots, (x_{n_2}, y_{n_2})]$, where $\mathcal{G}_{BL}[i] = (x_i, y_i)$ represents the bottom left point of garden $\mathcal{G}_i$,
- $\mathcal{G}_{TR} = [(x_1, y_1), \ldots, (x_{n_2}, y_{n_2})]$, where $\mathcal{G}_{TR}[i] = (x_i, y_i)$ represents the top right point of garden $\mathcal{G}_i$, and
- $F = [(x_1, y_1), \ldots, (x_{n_1}, y_{n_1})]$, where $F[i] = (x_i, y_i)$ represents the location of flower f_i.

You are not guaranteed that a flower belongs to a garden; it is possible that a flower is planted in none of the gardens. Your goal is to return the garden that a flower f_i belongs to (if any), for *each* f_i.

- For example, in the diagram above, if the flower ordering is sorted by its x-coordinate and the rectangular gardens are sorted by their x-coordinate of $\mathcal{G}_{BL}$, then we return

$$f_1 : \mathcal{G}_1, \quad f_2 : \mathcal{G}_2, \quad f_3 : \mathcal{G}_2, \quad f_4 : \mathcal{G}_4, \quad f_5 : \mathcal{G}_3.$$

(a) We first solve the special case where all of the gardens intersect with a horizontal line. Describe an $O(n \log n)$ algorithm to determine which flowers belong to which gardens (if such a garden exists).

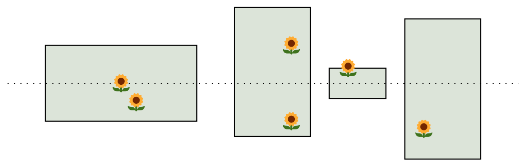

Figure 2: *A collection of $n_1 = 6$ flowers and $n_2 = 4$ gardens that intersect with a horizontal line.*

Hint. *What do you know about two adjacent gardens if they have to intersect with a horizontal line?*

(b) We now remove the assumption that every garden intersects with a horizontal line. Describe an $O(n(\log n)^2)$ algorithm to determine which flowers belong to which gardens (if such a garden exists).

Note. *This was an assignment problem from COMP3121/9101, 23T3.*

Rubric.

- Your solution should clearly outline which subpart you're answering.
- If your solution is a simple modification of a previous solution, you do not need to restate the solution; you can simply refer to the previous part in your solution.
- As usual, you should argue the correctness of the algorithm and its time complexity.
- This task will form part of the portfolio.
- Ensure that your argument is clear and keep reworking your solutions until your lab demonstrator is happy with your work.

Solution.

(a) We must only consider x coordinates as no two gardens lined in a horizontal may intersect. We will denote such intervals as $(x_i^{(b)}, x_i^{(t)})$ as the garden interval for $\mathcal{G}_i$, and for flower f_i, we will use $f_i^{(x)}$. If for any flower $f_j^{(x)} \in (x_i^{(b)}, x_i^{(t)})$, then it must be so that $f_j \in \mathcal{G}_i$. This will be our check to verify if the flower is in the garden.

We begin by sorting our gardens by their intervals, which can be done by considering their $x_i^{(t)}$ and sorting based on those values. Then, for each flower, we perform a binary search on the gardens. If $f_j^{(x)} \in (x_i^{(b)}, x_i^{(t)})$, then we stop. If $f_j^{(x)} \leq (x_i^{(b)}$, then we restrict our search space to the bottom half of the array and repeat. Similarly, if $f_j^{(x)} \geq (x_i^{(t)}$, then we restrict the searching domain to the top half. If a garden has been found, one must then simply check the y coordinates by seeing if $f_j^{(y)} \in (y_i^{(b)}, y_i^{(t)})$, where $f_j^{(y)}$ is the y coordinate of f_j, and $y_i^{(b)}$ and $y_i^{(t)}$ is the bottom-left and top-right y coordinates of $\mathcal{G}_i$ respectively.

Sorting the gardens is done in $O(n_2 \log n_2)$ time for the n_2 gardents and the binary search of the gardens for n_1 flowers would be $O(n_1 \log n_2)$, so our final process will run in $O\left((n_1 + n_2) \log n_2\right) = O\left(n \log n\right)$.

(b) Our goal is to use the solution from part (a) by identifying horizontal lines that could work. Since we are only given numeric values, we can simply consider some measurment of central tendency, i.e. the mean or median. We will call this value M. Any garden passing through the line $y = M$ will be able to use the algorithm from part (a). If a garden is not cut by this line, it must then be either above or below the line. So we can isolate the gardens above and below and rerun the algorithm recursively. For any flower that is unaccounted for in this algorithm, we can divide into set of flowers that are above and below $y = M$, making the process more efficient for checking. We stop this process if we are left with only one element (a single flower or garden).

Our algorithm must run in $O(n \log n)$ each time we run part (a)'s algorithm. So, our time complexity will be recursively called as $R_n = 2R_{\lfloor n/2 \rfloor} + O(n \log n) \leq 2R_{\lfloor n/2 \rfloor} + Cn \log n$, where C is some constant. We can solve this quite simply by verifying:

$$R_n \leq 2R_{\lfloor n/2 \rfloor} + Cn \log n \leq 2\left[2R_{\lfloor n/4 \rfloor} + Cn \log n\right] + C\frac{n}{2} \log \frac{n}{2} \leq 4R_{\lfloor n/4 \rfloor} + 2Cn \log n.$$

We must repeat this process $\log_2(n)$ times in total as that is the limiting division of the gardens and flowers. It's clear that for all layers, it must be $2R_{\lfloor n/2 \rfloor} + Cn \log n \leq 2\left[R_{\lfloor n/4 \rfloor} + Cn \log n\right]$, so repeating this $\log_2 n$ times yields:

$$R_n \leq 2^{\log_2 n} R_1 + Cn \log^2 n = O(n \log^2 n).$$