# CS300 - Fall 2024-2025 Sabancı University Homework#2

## Introduction

A point quadtree is "a multidimensional generalization of a binary search tree" designed to store and query 2D points. For further details, refer to the attached document taken from the book *Design & Analysis of Spatial Data Structures* by Hanan Samet.

In this assignment, you will implement a point quadtree of cities. Specifically, given a 2D space and a list of cities (each represented as a 2D point), you will insert the cities into an initially empty point quadtree in the order they are provided. Given a query point $A$ in the same space, together with a radius $r$, you will find all cities within radius $r$ of $A$. For further details about point quadtrees and the implementation of the insert and find operations, refer to the attached document.

**Note:** This assignment is designed to evaluate your ability to understand and implement related data structures using the knowledge gained in the lectures. Therefore, only the textbook definition of point quadtrees is provided.

## Program Specifications

Your program should read two text files, `cities.txt` and `queries.txt`. The former specifies a 2D space and lists the cities located within it, while the latter specifies the queries to be answered.

### Input Format

**File:** `cities.txt`

The first line in `cities.txt` specifies the 2D space by giving the coordinates of the upper-right corner of the space in the form:

$$< x \text{ coordinate} > < y \text{ coordinate} >$$

The origin $(0, 0)$ corresponds to the lower-left corner of the space, and negative coordinate values are not valid.

Each subsequent line in `cities.txt` represents a city, formatted as follows:

$$< \text{city\_name} > < x \text{ coordinate of the city} > < y \text{ coordinate of the city} >$$

City names contain no spaces, and coordinates cannot be negative.

**File:** `queries.txt`

Each line in `queries.txt` corresponds to a query, formatted as follows:

$$< x \text{ coordinate} > < y \text{ coordinate} > < \text{radius} >$$

### Program Output

#### Point Quadtree Construction

Given the `cities.txt` file, your program should insert the cities into an initially empty point quadtree in the specified order. Then, it should pretty print the tree using the following recursive algorithm:

```
pretty_print(root):  // Pretty print the quadtree rooted at root
  if root != NULL:   // If the tree is not empty
    print the city name stored at the root
    pretty_print(root.SE)  // Recursively print the southeast subtree
    pretty_print(root.SW)  // Recursively print the southwest subtree
    pretty_print(root.NE)  // Recursively print the northeast subtree
    pretty_print(root.NW)  // Recursively print the northwest subtree
```

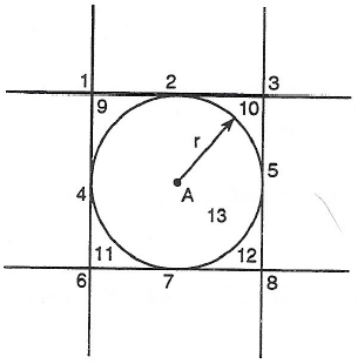
**Query Processing**

Once the point quadtree is constructed, the `queries.txt` file should be processed. For each query, your program should produce two lines of output:

1. A comma-separated list of cities within the specified radius from the given point $(x, y)$.
2. A comma-separated list of cities visited during the search operation.

**Notes:**

- Cities should be reported in the order they are found and visited, respectively.
- Subtrees (i.e., quadrants) should be visited recursively in exactly the order specified in Figure 2.17 of the attached document.
- If no cities are found within the specified radius, output `<None>`.
- Query results should be printed immediately after processing each query line.




Problem: Find all nodes within radius r of point A
Solution: If the root is in region I (I=1...13), then continue to search in the quadrant specified by I

| | | |
|---|---|---|
| 1. SE | 6. NE | 11. All but SW |
| 2. SE, SW | 7. NE, NW | 12. All but SE |
| 3. SW | 8. NW | 13. All |
| 4. SE, NE | 9. All but NW | |
| 5. SW, NW | 10. All but NE | |

Figure 2.17: Relationship between a circular search space and the regions in which a root of a point quadtree may reside.

Figure 1: Quadtree quadrant traversal order as per Figure 2.17

# Sample Run

Given the following `cities.txt` file:

```
100 100
Chicago 35 42
Mobile 52 10
```

```
Toronto 62 77
Buffalo 82 65
Denver 5 45
Omaha 27 35
Atlanta 85 15
Miami 90 5
```

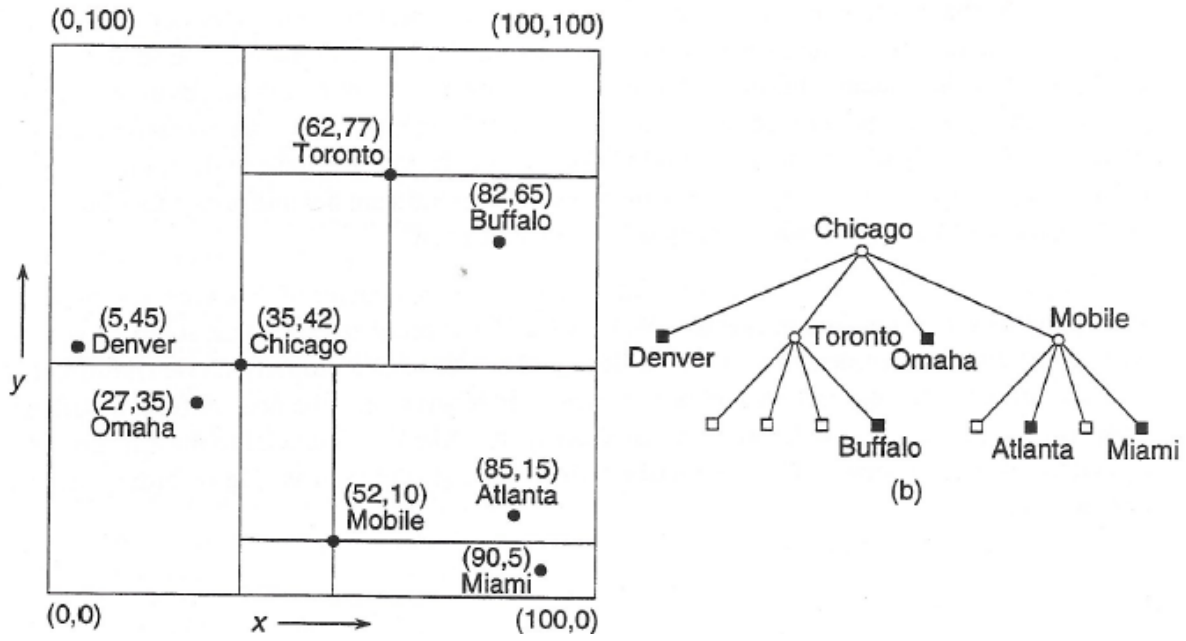



Figure 2: Quadtree as per Figure 2.4 in attached notes

The following quadtree shall be constructed (as illustrated in Figure 2.4 of the attached document)Thus, the following output should be printed out:

```
Chicago
Mobile
Miami
Atlanta
Omaha
Toronto
Buffalo
Denver
```

Then, given the following `queries.txt` file:

```
83, 10, 8
25, 33, 10
42, 83, 2
82, 35, 32
62, 77, 24
```

The output should be as follows:

```
Atlanta
Chicago, Mobile, Miami, Atlanta


Omaha
Chicago, Mobile, Omaha, Denver
```

```
<None>
Chicago, Toronto


Miami, Atlanta, Buffalo
Chicago, Mobile, Miami, Atlanta, Toronto, Buffalo

Toronto, Buffalo
Chicago, Toronto, Buffalo
```

## Submission

Your code should be submitted to SUCourse at the deadline given on the SUCourse. You should submit a source.cpp for testing the queries.txt files and Quadtree.cpp and Quadtree.h, Similarly you can have your class signatures and implementations in the same file and submit with source.cpp and Quadtree.h or Quadtree.cpp.