The scope of this project is to extend the functionalities of the R-tree in order to support more complex query types (beyond range and nearest neighbor queries). You can use one of the existing implementations of the R-tree or its variants. We recommend to use the implementation by Marios Hadjieleftheriou that can be downloaded from: http://www.cs.ucr.edu/~marioh/spatialindex/index.html. Other implementations of the R-trees can be found there: http://www.rtreeportal.org/. You are free to choose any of the implementations in the web sites above.

First, you should compile and run some simple range and nearest neighbor queries using the R-tree implementation that you have chosen. Then, you have to design and implement the following query types:

1. **(Aggregation Range Query)** Given a dataset $S$ of points in a Euclidean space $D^n$, and a range query $q$, the Aggregation Range Query (ARQ) asks for the number of points from $S$ that are contained inside $q$. More formally, given $q = [(x_{1 \min}, x_{1 \max}), \ldots, (x_{n \min}, x_{n \max})]$, the answer to the query is the value: $ANS = \sum_{p \in S} (p \text{ inside } q)$. Design and implement an efficient algorithm based on R-trees to answer ARQs. State an advantage and a disadvantage of your method.

   HINT: You may need to modify the node structure of the R-tree.

2. **(Preference Query)** Let $S$ be a dataset of multi-dimensional points of dimensionality $n$. Let $p = (x_1, \ldots, x_n) \in S$ a point in the space $D^n$. Also, consider the function $S \rightarrow R : f(p) = \sum_{i=1}^{n} a_i x_i$ with $\sum_{i=1}^{n} a_i = 1$ and $a_i > 0$. A preference query asks for the point in $S$ that maximizes the function $f$, given user specified $a_i$s. You should implement an efficient search algorithm that uses an R-tree index to find efficiently the point that maximizes the function $f$, given user specified values for $a_i$s.

3. **(Dominating Set Query)** Consider again a set $S$ of points in a Euclidean space $D^n$. We define a point $p = (x_1, \ldots, x_n) \in S$ to be a dominant point, if there is no other point $q$ in $S$ that dominates $p$. A point $q$ dominates another point $p$, if $\forall i, 1 \leq i \leq n, q_i \leq p_i$, and there is at least one $j$, such that $q_j < p_j$. A dominating set query asks for all the dominant points in $S$. Design and implement an algorithm that uses an R-tree to report efficiently the dominant set of a dataset indexed by the R-tree.

**What to hand in**

- Your code and a README file describing how to run your program.
- A write-up, with your search procedures in algorithmic form and answers to questions.
- The results of using your code to run a set of queries that will be provided to you an a test dataset. The dataset and the queries will be provided to you in 2 weeks.