

COP 4720 - Information Systems and Databases II, Spring 2008, Programming Project

Due date: 04/ 18 / 2008

Goal

The goal of this project is to become familiar with the querying algorithms for multi-dimensional index structures. You will implement the kd-tree, the Vkd-tree (a variation of the kd-tree) and the range query algorithm. This project can be done individually or in groups of size two.

Programming Language & Platform

You can use either C/C++ or Java as the programming language for this project. There is no restriction on the development platform (Windows, Solaris, Linux, etc), but your final code should compile and run in a Linux or unix platform. Therefore, if you decide to develop (code and debug) on a different platform, you should allow some extra time before the actual deadline to test your code on the Linux or Unix platform mentioned above.

It is your responsibility to make sure that your code runs properly on our testing platform. You can test your code on a department computer such as “sand” or “rain”.

Specifications

You will implement range queries for high dimensional data using three strategies. The first one will scan the database sequentially without using any index structure. The second one will use the kd-tree as described in class and in book. The third one will use a variation of the kd-tree, called *Vkd-tree*. The Vkd-tree is described later.

Input: Your program should have the following command-line input arguments:

1. An integer denoting the choice for index option:
 - Zero (0) denotes sequential search.

- One (1) denotes search using the kd-tree.
 - Two (2) denotes search using the Vkd-tree.
2. The name of the database file. The database consists of multi-dimensional data points in the form of x_1, x_2, \dots, x_k (here k is the number of dimensions. Each line contains a single record. You can download the sample database from www.cise.ufl.edu/~tamer/teaching/spring2008/other/projDB.gz.
 3. The file containing all range queries. Each line of this file is a range query and has the form

$$x_{min} \quad x_{max} \quad y_{min} \quad y_{max} \quad \dots$$

In other words, we want to find the points for which $x_{min} \leq x \leq x_{max}$ and $y_{min} \leq y \leq y_{max}$. You can download the sample queries from www.cise.ufl.edu/~tamer/teaching/spring2008/other/projquery.

4. The maximum number of key-pointer pairs per block.

In short, your program should run using the following command:

```
./rangeQ searchOption database queries indexBlock
```

For example

```
./rangeQ 1 database queries 50
```

will run range query using the kd-tree with 50 key-pointer pairs per block. For sequential search the last parameter is not needed. For example

```
./rangeQ 0 database queries
```

will sequentially check all the points in the database.

Output: Your program will print the first range query from the input query file, followed by records satisfying this query (each on a separate line), the second range query followed by records satisfying this query, and so on.

Details

1. The program will simulate the database search process with the help of one or two data structures depending on the search option.
 - The first one is an array of *data points*. Each entry of this array will contain a record. This array simulates the data on the disk.

- The second one is the index structure (kd-tree or Vkd-tree depending on the search option). Each index block at the leaf level of this index will contain b key-pointer pairs, where b is a given input parameter (e.g., $b = 50$). This data structure will be used only for the index-based search and it will simulate the index in main memory.
2. Your program will read the database into the array for data point. You are free to keep the data sorted according to any of the dimensions. However this is not required.
 3. For sequential search you will search the data blocks array sequentially.
 4. For index-based search you will use the index structure to decide which data blocks you need to access.
 5. The output files created by the three search options should be EXACTLY the same. Otherwise, it means that there is something wrong with at least one of them. You can use *diff* command to compare them.
 6. **Brief description of the Vkd-tree.** Vkd-tree differs from the kd-tree in the way it selects the dimension to partition the search space. Assume that the database contains n points $\{P_1, P_2, \dots, P_n\}$. Also assume that each point has k dimensions, represented by x_1, x_2, \dots, x_k . For example, the first two dimensions of P_1 is $P_1.x_1$ and $P_1.x_2$. If n is greater than the capacity of an index block (i.e., $n > b$), then Vkd tree selects a dimension and partitions the points into two sets along this dimension as follows:

- **SELECTION OF THE DIMENSION:** Vkd tree selects the dimension for which the data points have the highest variance. The variance of a dimension (e.g., the first dimension) is computed as follows. First the mean μ of that dimension is computed:

$$\mu = \frac{1}{n} \sum_{i=1}^n P_i.x_1.$$

After that the variance is computed as

$$V = \frac{1}{n} \sum_{i=1}^n (P_i.x_1 - \mu)^2.$$

- **PARTITIONING THE DATASET:** Once the dimension with the highest variance is selected, Vkd tree partitions the dataset into roughly two equal sized sets along that dimension. In order to do this, it finds the median of the values of the dataset in that dimension. For example, assume that dimension x_1 is selected in the previous step. Vkd tree partitions the points using the value M as the separator, where $P_i.x_1 < M$ for (roughly) half of the $P_i \in \{P_1, \dots, P_n\}$.

Project Evaluation

Please follow the following instructions when submitting your code (OTHERWISE, we would not evaluate your source code and you would get a score of zero for this part; SORRY!): Every group needs to do the following for the project evaluation:

1. Each group (only one person per group) should submit all the relevant files using course tool. A link to course tool is available on the class page. This includes your source code, a Makefile, and a Readme file (NOT the dataset or the query set).

The source code should contain all the files necessary for your code to compile; DO NOT submit any binary/object files such as *.o / *.out files for C/C++ or *.class files for Java.

The Makefile should generate an executable file called *rangeQ*. The Readme file should list names and emails of the group members, as well as anything that you want us to know. One example would be what you haven't implemented, but is required. We will deduct more points if we find a bug in your program, but less if you report the bug in the Readme file.

- (1). Create a directory named after one of the group members' last name and copy all your project files into it. The directory should contain the source code, the Makefile, and the Readme file.
- (2). Tar the files by typing the following command: `tar cvf lastName.tar lastName/`. Here *lastName* is the created directory in step (1) that contains your project. DO NOT type this command inside your *lastName* directory; do it outside this directory.
- (3). Now compress your tar file by typing the following command: `gzip lastName.tar` This will produce a file called `lastName.tar.gz`. This will be your submission file.

- (4). Email the project. In case you need, you can resubmit your project (provided that it is before the deadline). Remember that we will only consider the latest submission.
2. Each group need to turn in a hard copy of a report that describes the role of each group member, if the project is done by a group of two.
 3. Each group need to turn in a hard copy that includes a graph that shows the average running the time of the two methods for increasing query range. For this, use the sample two dimensional dataset given on the class page. You will use the test query files which will be available on class page

`www.cise.ufl.edu/~tamer/teaching/spring2008/other/test2,`
`www.cise.ufl.edu/~tamer/teaching/spring2008/other/test3,`
`www.cise.ufl.edu/~tamer/teaching/spring2008/other/test4,`
`www.cise.ufl.edu/~tamer/teaching/spring2008/other/test5,`

Each test file contains 100 random queries. The range size in “test2” is 2^2 , the range size in “test3” is 2^3 , and so on. In other words the difference between these test files is the size of the query, and thus, the size of the result set.

Find the average running time for each test file. Plot the result with the range size on the x axis and running time on the y axis. Also, return a brief discussion of your results.

4. Every group needs to make a demo to the TA. The demo will be in CSE 309. It will span the TA’s office hours from April 18 to April 23. We will use your final submission to coursetools during demo. You will be asked to run some queries and/or answer questions about your source code.