ReadMe of the Code of the P$^+$-Tree

Beta version 1.0

Rui Zhang

rui@csse.unimelb.edu.au

http://www.csse.unimelb.edu.au/∼rui

Department of Computer Science
and Software Engineering,
The University of Melbourne,
Carlton, Victoria, Australia, 3053
2007

1 Paper related to this code

Rui Zhang, Beng Chin Ooi, Kian-Lee Tan: **Making the Pyramid Technique Robust to Query Types and Workload**, *International Conference on Data Engineering (ICDE), Boston, 2004.*

2 Copyright

3 Implementation Notes

The code of the GiMP is based on a B^+-tree coded by Dai Haoyu.

4 How to Use the Code

4.1 Compilation

The code was compiled on Fedora 2 and can be compiled on most Linux and Unix systems. Use the command "make" to compile.

4.2 Follow the following steps to run the code

1. **Data domain and type:** The code assumes that data are of type float, and normalized to $[0,1]^d$. The data type is defined by the constant DATA_TYPE in file "btree.h". E.g. "typedef float DATA_TYPE;" means the data are of type float.
2. **Dimensionality:** Define the dimensionality of the data space by defining the constant D in the file "btree.h". E.g. "#define D 4" means dimensionality is 4.

3. **Other parameters:** The order of division of the space determines how much the data space is divided. It is defined as a constant "T" in the main program "pplus.c". "#define T 4" means the order of division is 4 and there will be $2^4 = 16$ subspaces after division. Suggested value for T is 6 to 8.

4. **Data:** Data points are sequentially stored one by one in a binary file called "data0". A data point is stored dimension by dimension. "data0" must be put in the directory "cluster". If there is already a "data0" in the directory "cluster", simply overwrite it with your data file.

5. **Space division:** Before building the P^+-tree, an auxiliary structure called the "space-tree" must be created first. Compile the program "dividespace.c" in the directory "cluster" to an executable, say, "ds"; then run the executable to divide space and build the space-tree. The dimensionality also needs to be defined for the program "dividespace.c", which is defined by the constant "D". When running "ds", the order of division is given as an input. The same value should be used here as the value defined for "T" in the file "pplus.c". After running "ds", the space-tree is built and stored as a binary file "SpaceTree".

6. **Queries:** In the main program ("pplus.c"), queries are read from files. The query file is referred by the pointer "fp_query". Queries are stored sequentially one by one in binary format. A window query is stored in the format of $\{l_1, u_1, l_2, u_2, ..., l_d, u_d\}$, where l_i and u_i are the lower and upper bounds of the query window in dimension i.

7. **Compile the code:** After the above settings, compile the code by "make". The executable "pplus" is generated.

8. **Run the executable:** Build the index by running "pplus b". Then perform the window queries by "pplus"

9. **Results:** Results are written to the file "result" in text format. The coordinates of each answer point are listed for each query.

5 Optimization Notes

1. Internal nodes of the B^+-tree are loaded into memory before the queries. The internal nodes are less than 0.3% of the whole index, which can fit into memory. To test the effect of buffer, code needs to be touched up.