深圳大学研究生课程论文

题目 VP 树的创建及测试 成绩

专业 计算机科学与技术

课程名称、代码 通用搜索技术 2703060

年 级 2018 级 姓 名 杨树

学 号 1800271003 时 间 2019 年 1 月 12 日星期六

任课教师 毛睿

课程论文（截止时间：2019 年 1 月 16 日 23:59）：

实现 GH 或者 VP 树，并作性能测试和分析，内容包括：

1. 实现 GH 或者 VP 树（二选一），能够首先接收原始数据（命令行、文件等方式）建立索引树，然后能够反复接收范围查询并给出查询结果（含数据及其与查询对象的距离）（40 分）
2. 说明相关参数的设置，例如支撑点个数，选取方法，划分个数，划分算法，叶子里面数据个数上限等。（10 分）
3. 编程语言不限，提交代码。（10 分）
4. 代码应包含足够清晰详细的注释。（10 分）
5. 分别测试两种以上数据类型，例如向量数据（至少 2 维，欧几里得距离）和字符串数据（编辑距离（插入删除代价 1，替换 2））（10 分）
6. 所测试的树应至少 3 层高，可以通过分析或者测试给出证据。（10 分）
7. 两种数据分别测试至少 5 个范围查询，给出测试截图。（10 分）

提交方式：“学号-姓名.zip”作为邮件标题和附件名邮件发送给 mao@szu.edu.cn.

报告应由各人独立完成，部分报告会被随机挑选出来与往年作业和其它课程作业进行对比，如果雷同以抄袭论处。

一、实验过程

1. 程序的工程目录如下图 1 所示。其中“main.cpp”是主函数所在文件。

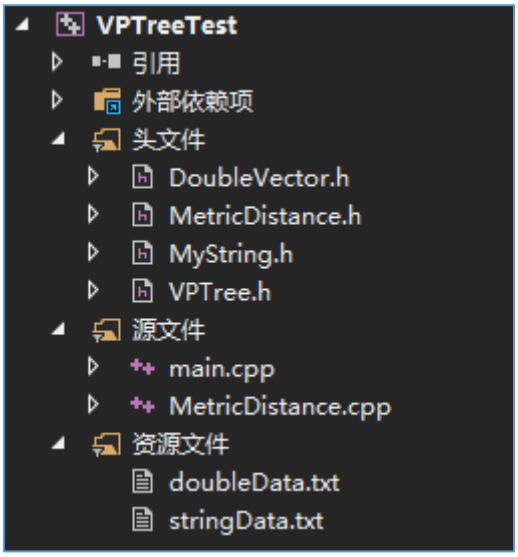

图 1　工程文件

2. 首先介绍“DoubleVector.h”和“MyString.h”，这两个文件中分别定义了向量数据和字符串数据。向量的维度可根据读取的数据进行改变，另外这两种数据也同时存储了数据在原数据文件中的行号，即“dataId”。具体定义如下表 1 所示。

表 1　向量类和字符串类的定义

```
class DoubleVector {
public:
    void setID(int id);
    int getID();
    int getDim();
    double getData(int i);
private:
    int dataId;
    int dim;
    std::vector<double> dataList;
};
```

```
class MyString {
public:
    void setID(int id);
    int getID();
    std::string getData();
private:
    int dataId;
    std::string str;
};
```

3. “MetricDistance.h”和“MetricDistance.cpp”中定义和实现了距离度量函数。向量和向量之间采用欧几里得（Euclidean）距离进行度量，字符串和字符串之间采用编辑距离（即 Levenshtein 距离）进行度量。如表 2 所示。在计算字符串之间的编辑距离时，插入和删除的代价都设置为 1，替换的代价设置为 2，在程序的具体实现中，将其设置成了可变的参数，具体的可以在“MetricDistance.cpp”的 13、14 和 15 行对插入、删除以及替换的代价进行修改。

表 2　向量和字符串的距离函数定义

```
/** 计算字符串之间的 Levenshtein 距离（即编辑距离）
* @param myStr1:  第一个字符串
* @param myStr2:  第二个字符串
* @return:  返回从第一个字符串到第二个字符串的 Levenshtein 距离
*/
double editDistance(MyString myStr1, MyString myStr2);

/** 计算向量之间的 Euclidean 距离
* @param vec1：第一个向量
* @param vec2：第二个向量
* @return：返回两个向量之间的 Euclidean 距离
*/
double euclideanDistance(DoubleVector vec1, DoubleVector vec2);
```

4. “VPTree.h”中实现了模板类 VPTree，在类中实现了创建 VP 树以及利用 VP 树进行查找的方法，具体的实现如下表 3 所示。

表 3　模板类 VPTree 的定义

```
template<typename T, double(*distance)(T, T)>
class VPTree {
public:
    VPTree() : _root(0), leafNumber(3){}
    ~VPTree() { delete _root; }
    void setLeafNumber(int num);  // 设置叶子节点存放数据数目
    void create(std::vector<T> &items);  /** 创建VP树 */
    void query(const T& target, std::vector<T> &results, std::vector<double> &distances);
    void knnQuery(const T& target, std::vector<T> &results, std::vector<double> &distances, size_t k);
    void rangeQuery(const T& target, double range, std::vector<T> &results, std::vector<double> &distances);

private:
    std::vector<T> _items;
    double _kth_dist;
    int leafNumber;

    struct VPLeafNode {};  // VP树的叶子节点的定义
    struct VPInternalNode {}*_root;  // VP树的内部节点的定义，以及根节点指针 _root 的创建
    struct DistItem {};  // 用于查找时存放索引和距离的结构体
    struct DistanceComparator;  // 距离比较器，用于构建VP树的时候作划分
    VPInternalNode* _buildVPTree(int lower, int upper);
    void _kquery(VPInternalNode* node, const T& target, size_t k, std::priority_queue<DistItem>& distQueue);
    void _rquery(VPInternalNode* node, const T& target, std::priority_queue<DistItem>& distQueue);
};
```

5. 如上表 1 所示，在实现 VP 树时，内部节点和叶子结点采用了不同的结构体，这有利于实现不同的功能。在内部节点中，一个节点存放一条数据；而在叶子节点中可以存放许多数据，即数据量小到一定程度便不再进行划分，而是用数组存储并放置在叶子结点中，这里默认叶子结点数据个数的上限是 3 个，但可以通过其中的“setLeafNumber”方法进行修改，例如 5 个或 10 个都可以。模板类 VPTree 中的属性“leafNumber”用来存放叶子结点数据个数上限。

6. 在创建 VPTree 时，Vantage Point 的选取采用的是随机的方法。每次创建内部节点时，先随机选取一个 Vantage Point，然后计算其余点与该点的距离，将数据均分成两部分，距离近的放到左子树，距离远的放到右子树。具体实现时，依赖求得的距离采用了类似于快速排序的方法，即枢轴左边全是小于枢轴的数据，枢轴右边全是大于枢轴的数据，两边的数据不保证有序，且被选作枢轴的距离要能够将数据均分。这样做的好处就是可以使得最终构建的 VP 树是一棵平衡二叉树，在查找时可以尽可能地节省时间。

7. 模板类 VPTree 中实现了 k-近邻查找以及 r-范围查找。“knnQuery”函数中实现的是 k-近邻查找，查到的结果通过“`std::vector<T> &results`”和“`std::vector<double> &distances`”进行返回，并且返回的 k 个结果按照距离从近到远进行排列。“rangeQuery”函数中实现的是 r-范围查找，查到的结果通过“`std::vector<T> &results`”和“`std::vector<double> &distances`”进行返回，并且返回的结果按照距离从近到远进行排列。

8. 创建 VPTree 之前，需要读取数据，通过重载“istream”的“>>”符号来自定义数据读取方法。这个重载方法在“main.cpp”中实现。

表 4　重载“istream”的“>>”

```
// 重载 >>
std::istream& operator >> (std::istream& is, std::vector<MyString> &strItems) {}
std::istream& operator >> (std::istream& is, std::vector<DoubleVector> &douItems) {}
```

9. 存放向量数据的文件名是**“doubleData.txt”**，其第一行的两个数字“30　2”分别代表该文件共有 30 条向量数据且每条向量数据都是 2 维的。下边 30 行就是具体的向量数据的数值。存放字符串数据的文件名是**“stringData.txt”**，其第一行的数字“30”代表该文件共有 30 条字符串数据。下边 30 行即为字符串数据。

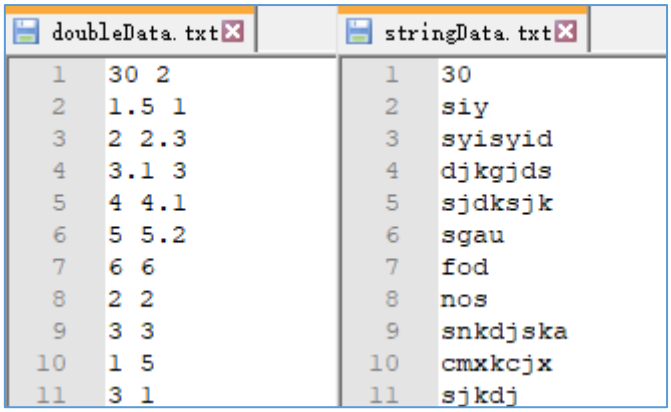

图 2　向量和字符串数据文件格式

10. 下边针对课程论文的要求作出回答。

11. 该程序实现的是**“VP 树”**，通过**文件读取**来获取数据，数据文件的格式前边已经讲述，如图 2 所示。有关参数如下：每次划分时，**随机选取一个** Vantage Point，按照距离远近将数据**均分成两部分**，距离近的放在左子树，距离远的放在右子树；对于向量数据，**使用 2 维向量进行测试，可根据数据文件的输入自动调整为更高维度**，采用**欧几里得（Euclidean）距离**进行度量，对于字符串数据，采用**编辑距离（即 Levenshtein 距离）**进行度量，插入删除代价为 1、替换代价为 2（将其设计成了可变参数，代价可修改）；VP 树的**叶子结点数据个数上限默认是 3，可修改**（模板类 VPTree 中设计了上限个数修改接口）；测试时数据文件中有三十条数据，构建的 VP 树超过了三层，如图 3 所示为 VS2015 的调试界面，通过查看 VPTree 的对象 vpt_double 的属性_root（即 VP 树的根节点）的数据**可以看到，根节点算第 0 层的话，叶子结点是在第 3 层**。

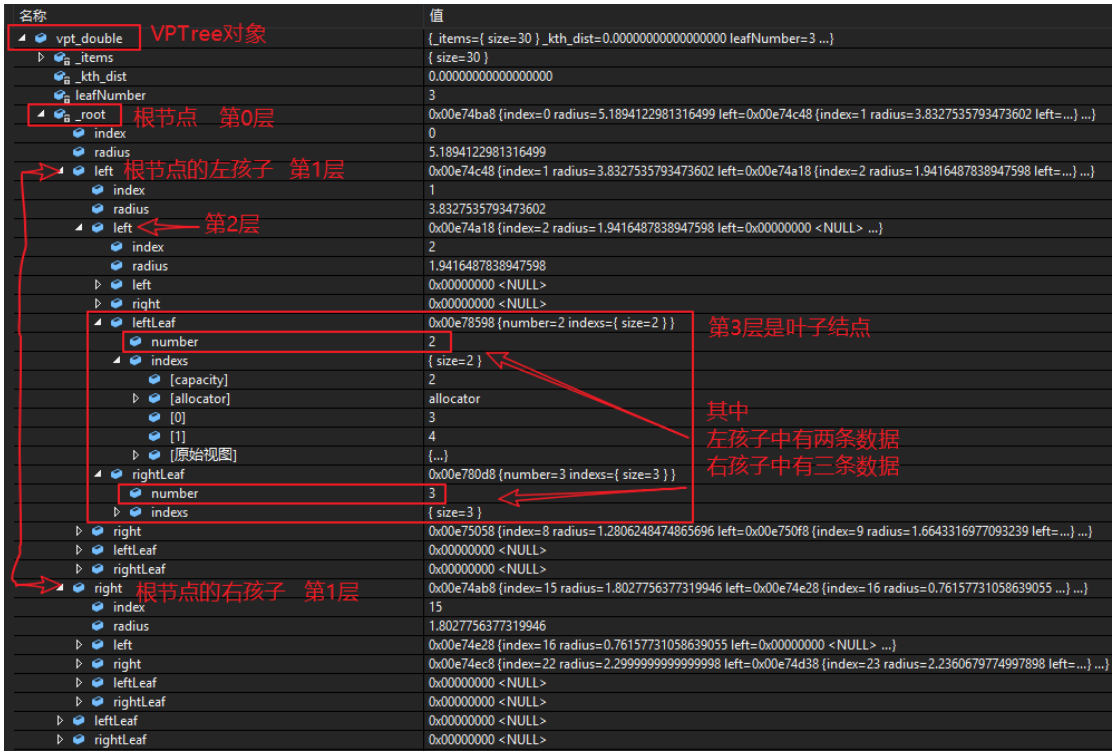

图 3　查看构建的 VP 树

二、实验结果

12. 运行程序，可以看到如图 4 所示的界面。

图 4　运行程序

13. 输入“1”查询字符串数据，之后再输入“1 sjjjk 5”，即选择 k-近邻方法查询字符串“sjjjk”，k 值设为 5，如下图 5 所示。

```
Please input a number(0: Exit   1: Query String   2: Query Vector):  1
Query String:  1 sjjjk 5
[String Query] is completed ...

 Id     字符串数据      距离
---------------------------------
 10     [sjdj]          3
 9      [sjkdj]         4
 3      [sjdksjk]       4
 14     [dsja]          5
 16     [jksd]          5
```

图 5　k-近邻查询字符串“sjjjk”

14. 输入“1”查询字符串数据，之后再输入“2 sjjjk 4”，即选择 r-范围查找方法来查询字符串“sjjjk”，r 值设为 4，如下图 6 所示。

```
Please input a number(0: Exit   1: Query String   2: Query Vector):  1
Query String:  2 sjjjk 4
[String Query] is completed ...

 Id     字符串数据      距离
---------------------------------
 10     [sjdj]          3
 9      [sjkdj]         4
 3      [sjdksjk]       4
```

图 6　r-范围查询字符串“sjjjk”

15. 输入“2”查询向量数据，之后再输入“1　3.6　4.1　5”，即选择 k-近邻方法查询向量“[3.6, 4.1]”，k 值设为 5，如下图 7 所示。

```
Please input a number(0: Exit   1: Query String   2: Query Vector):  2
Query Vector:  1 3.6 4.1 5
[Vector Query] is completed ...

 Id     向量数据        距离
---------------------------------
 28     [3.6, 4.1]      0
 3      [4, 4.1]        0.4
 18     [4, 4]          0.412311
 14     [3, 3.5]        0.848528
 29     [2.9, 3.3]      1.06301
```

图 7　k-近邻查询向量[3.6, 4.1]

16. 输入“2”查询向量数据，之后再输入“2　3.6　4.1　1”，即选择 r-范围查找方法查询向量“[3.6, 4.1]”，r 值设为 1，如下图 8 所示。

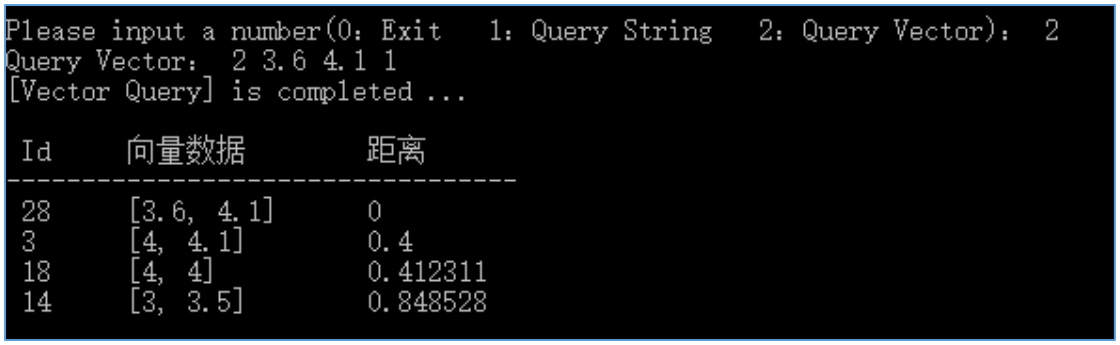

图 8　r-范围查询向量[3.6, 4.1]

17. 程序可以**反复接收字符串数据和向量数据的查询**。输入“0”可以退出程序，如图 9 所示。

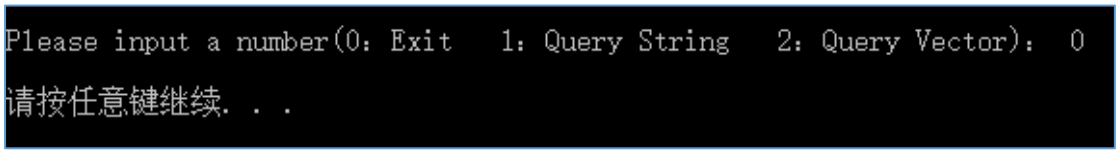

图 9　退出程序

18. 更多的查询的图随附件放在文件夹“runImages”中，请老师查阅。

三、总结和结论

对于本次项目的程序编写，难点主要在 VP 树的各种方法的实现上，其余程序实现部分没什么难度。

选课之初，对通用搜索技术完全没有概念，不清楚这是做什么用的，通过阅读老师在第一节课上推荐的参考文献学到了很多，对通用搜索技术算是有了一定的了解。随着不断的接触，许多概念也越发清晰。由于有前边的学习做铺垫，这才得以顺利的完成这次大作业。

总的来说，通过这门课的学习，除了对一些具体的知识有了学习之外，例如度量空间、支撑点空间、超平面划分、球形划分等等，毛老师在第一节课关于“通用”和“专用”的讲解使我印象非常深刻，让我思考了很多。总之，这门课让我受益匪浅。