数据库项目实验报告

用 B+树实现 MEDRANK 算法

01 组　　骆铭涛　童云钊　黄建武　查李吉飞

目录

一、实验目的 2

二、实验环境 2

三、实验原理 2

1、预处理 2

2、建 B+树 2

3、MedRank 算法 2

四、实现细节 3

1、预处理 3

2、建 B+树 4

3、MedRank 算法 4

五、实验结果 5

1、实验结果描述 5

2、评测指标及结果分析 6

六、实验结论 7

七、心得总结 7

八、小组成员分工 9

一、实验目的

本次实验旨在考察学生对 B+树索引和 MEDRANK 算法相关概念的理解，以及实现 B+树索引和 MEDRANK 算法相关代码的能力。同时，本实验还能训练小组成员之间的分工合作能力，提升同学们配合的默契度，为以后合作大型项目打下坚实的基础。

二、实验环境

编程语言：C++（C++11 标准）
操作系统：Linux
编译器：g++4.8.4

三、实验原理

本实验可以细分为三个步骤：预处理、建 B+树、MedRank 算法。下面详细说明各个步骤的具体逻辑：

1、预处理

首先要在 d 维空间中生成 m 个“投票者”，在本实验中，变量 d 和 m 分别设为 784 和 50，“投票者”是程序随机生成的 50 个向量。

其次，我们将数据集中的点（实验中数据集有 60000 个在 d 维空间中的点）在这 50 个“投票者”向量上作投影，然后给这些投影值排序（升序排列）。

2、建 B+树

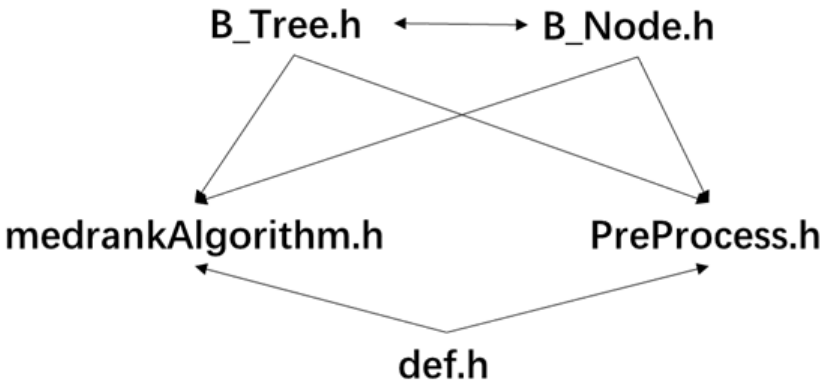
本部分的目的是给每个文件建立一个 B+树索引方便查询。首先我们规定一个页面大小为 1KB，然后将文件中存在的数据按物理顺序分成一个个页面并进行编号。我们以每个页面第一个投影的数值为基础向上一层一层建立索引（也就是说在索引页面中，一个数值为 k 的记录代表该文件中第一个投影的值是 k 的子页面）。

3、MedRank 算法

本部分首先将每个查询向量在 50 个“投票者”上作投影，根据投影值使用 B+树索引在每个“投票者”对应的文件中找到相应的页面并且初始化 h 和 l 指针。然后根据 50 个投票者中的 h 和 l 指针进行 MedRank 算法，最终在数据库里找出一个与查询点最接近的点。值得注意的是，由于 MedRank 算法找出的最近邻居只

是近似最近邻居，不一定是实际上的最优解，因此我们在实验结果部分还需要穷举所有点以找出真正最近的点，然后评测 MedRank 所找的近似最近点的精准度。

四、实现细节



以上为各头文件的关系图，箭头的方向代表被引用，B_Tree.h 和 B_Node.h 由于被互相引用，所以分别使用前置声明。以下将介绍各部分的工作和各模块设计及实现的细节。

1、预处理

（1）主要相关文件

PreProcess.h、PreProcess.cpp

（2）模块概述

文件"PreProcess.h"声明生成随机的线、在线上作投影以及对投影进行排序的相关函数，并在文件"PreProcess.cpp"里实现了这些函数。文件"main.cpp"通过引用头文件"PreProcess.h"，在主函数中调用函数 Indexing 进行预处理。

（3）主要函数解释

ZeroToOneUniformDistribution

生成（0,1）的均匀分布，返回（0,1）之间的一个值。

GenerateRandomProjectionVectors

生成随机的线（用向量表示）并对向量进行单位化。

Indexing

将 N 个点在 GenerateRandomProjectionVectors 函数返回的 M 条线上分别作投

影并将每条线上的投影进行升序排序，最后为每一条线新建 B_Tree 并调用 bulkload 将已经排序的投影写入文件中。

2、建 B+树

（1）主要相关文件

B_Node.h、B_Node.cpp、B_Tree.h、B_Tree.cpp、block_file.h、block_file.cpp

（2）模块概述

文件"B_Node.h"声明了 B_Node 类，B_Node 类包含与节点相关的基本信息，例如高度，左右兄弟节点，键值对数组；文件"B_Tree.h"声明了 B_Tree 类，B_Tree 类包含与 B+树相关的基本信息，例如该树所对应的文件，根节点以及根节点所在的块的编号还有树的高度，并且包含 bulkload 成员函数；文件"block_file.h"声明了 block_file 类，block_file 类管理文件的基本信息，例如文件名称，块的大小，块的数量等等。

（3）主要函数解释

Bulkload

Bulkload 为一边建树一边将树的节点写到文件中，以减少内存的使用。首先创建叶子节点，将点在该线上的投影值和点的 ID 添加到叶子节点中，假如当前叶子节点已满，就重新创建新的叶子节点并设置已满的叶子节点和新的叶子节点的左右兄弟节点，最后将已满的叶子节点写到文件中，重复以上步骤直至所有点写到该 B+树中。索引节点也是同样的道理，只不过是将子节点第一个投影值和子节点所在的块的编号添加到索引节点中。最后，当该 B+树建完后，更新文件中 HeaderNode 的信息，例如块的数量和根节点所在的块的编号。

SearchNode

该函数负责通过 queryData 的值在 B+树上定位 h 和 l 所在的具体页面，并将页面的数据载入内存以 B_Node 类型存储，最后返回指向节点类型的指针。

3、MedRank 算法

（1）主要相关文件

medrankAlgorithm.h、medrankAlgorithm.cpp

（2）模块概述

文件"medrankAlgorithm.h"声明了 MedRank 算法及精准度验证的相关函数，并在

文件"medrankAlgorithm.cpp"里实现了这些函数。文件"main.cpp"通过引用头文件"medrankAlgorithm.h",在主函数中调用函数TestMedrank启动MedRank算法。

（3）主要函数解释

TestMedrank

作为MedRank模块的入口，负责从查询集文件中读入查询向量，并调用medrank算法得到近似最近邻，之后调用函数计算算法精准度并打印出性能指标。

medrank

该函数负责对一个查询向量做medrank算法。函数先是调用Object2Projection函数将向量投影到M条线上，之后初始化M条线的h和l，初始化频率表，最后进入循环进行投票，同时维护每条线的h和l，直到有一条线胜出则返回该线的编号。

Object2Projection

该函数负责通过做点积，将一个784维的查询向量投影至M条线上。

InitializeTwoPointers

该函数负责初始化一条线上的h和l。通过调用searchNode函数在B+树上搜索h和l可能位置的结点，再遍历该结点，确定h和l的位置。

LeftShift

该函数负责左移h，并更新相应的变量。RightShift同理。

五、实验结果

1、实验结果描述

```
tongyunzhao@tongyunzhao-ThinkPad-X240s:~/桌面/01_骆铭涛_DBproject$ make run
rm -f *.bin
./medrank -n 60000 -d 784 -qn 100 -ds ./data/Mnist.ds -qs ./data/Mnist.q
Min Ratio = 1.000000
Max Ratio = 1.987715
Average Ratio = 1.254947
Indexing time: 12.579011 s
c-ANN query time: 2.220828 s
calculate ratio time: 17.293623 s
Average I/O Cost = 4362.690000
tongyunzhao@tongyunzhao-ThinkPad-X240s:~/桌面/01_骆铭涛_DBproject$
```

```
tongyunzhao@tongyunzhao-ThinkPad-X240s:~/桌面/01_骆铭涛_DBproject$ make run
rm -f *.bin
./medrank -n 60000 -d 784 -qn 100 -ds ./data/Mnist.ds -qs ./data/Mnist.q
Min Ratio = 1.000000
Max Ratio = 1.993062
Average Ratio = 1.260140
Indexing time: 12.850277 s
c-ANN query time: 2.023561 s
calculate ratio time: 16.870938 s
Average I/O Cost = 4477.640000
tongyunzhao@tongyunzhao-ThinkPad-X240s:~/桌面/01_骆铭涛_DBproject$
```

```
tongyunzhao@tongyunzhao-ThinkPad-X240s:~/桌面/01_骆铭涛_DBproject$ make run
rm -f *.bin
./medrank -n 60000 -d 784 -qn 100 -ds ./data/Mnist.ds -qs ./data/Mnist.q
Min Ratio = 1.000000
Max Ratio = 1.883123
Average Ratio = 1.266932
Indexing time: 12.873971 s
c-ANN query time: 1.990086 s
calculate ratio time: 16.830370 s
Average I/O Cost = 4459.960000
tongyunzhao@tongyunzhao-ThinkPad-X240s:~/桌面/01_骆铭涛_DBproject$
```

实验结果输出 100 个查询最小的 Ratio，最大的 Ratio，平均 Ratio，建立 B+树的 Indexing time，100 个查询总共的时间 c-ANN query time，暴力求最近点并计算 ratio 的时间，以及每个查询的平均 I/O 次数。另外每个查询的近似最近点及对应的 ratio 写进文本文件 output.txt。
以上为连续运行程序 3 次的截图。

2、评测指标及结果分析

Index Size and Indexing Time:

每棵 B+树的大小为 608KB，1KB 存储 100 个点，总共有 60000 个点，所以叶子节点有 600KB，非叶子节点有 8KB。

建立 50 棵 B+树，写入 50 个文件的时间为 12.5s。

Overall Ratio:

同样的算法，论文中的平均 Ratio 都是大于 3 的，而我们程序的平均 Ratio 为 1.26 左右，最小的 ratio 还能达到 1，即查询的结果就是真正的最近点。

I/O Cost:

平均每个查询的 I/O 次数是 4400 次，因为有 50 棵 B+树，即平均每个查询每棵树 80 多次 I/O，因为我们投影为了减小精度误差是用 double 存储，所以每个节点(1K)存储的数据少，因此 I/O 次数也会较多。

Running Time:

100 个查询总时间为 2s，平均每个查询 0.02 秒，证明程序的运行效率还是很不错的。

六、实验结论

这次试验将查询点和候选点放在一个多维空间，每个坐标作为投票者,排名基于到对应查询坐标的距离远近，而获胜者是那些排名最高的点。通过投影降低维度的处理，这种方法产生一个简单的，对数据库友好的算法。这个算法非常有效，经常探索不超过 20%的数据就可以获得高质量的结果。

七、心得总结

骆铭涛：

这次实验让我深入了解 B+树的建立过程以及如何将 B+树写入文件，B+树的建立是一个比较简单的过程，难点在于理清 block_file、B_Node 以及 B_Tree 之间的关系并清洗划分每个类的功能和接口，为建树和 MedRank 算法的实现带来极大的方便。这次实验也让我理解到系统架构师这个职位的重要性，通过这次实验我对以下问题有了更多经验和看法，例如如何规划项目的总体架构，如何定义清晰明确的接口，如何指导团队开发和对组员进行分工。虽然这是比较小的项目，但是组长的工作也显得尤为重要。

实验过程中也遇到不少 bug，例如内存泄漏和文件读取出错，用 gdb 调试大大加快我们定位 bug 的速度，以及了解 bug 的错误信息。对于代码优化，可以输出每部分花费的时间再进行优化，例如 MedRank 算法中每次都新建 B+树然后进行投票，时间明显都花在新建 B+树然后读取文件上，最后新建一个数组管理所有 B+树的指针，将投票的时间从二十分钟缩短至十秒左右，效率提升明显。

童云钊：

总体来说，这次项目考验了我们的编码能力、学习能力，更锻炼了我们的团队协作能力和软件工程能力，对我来说有很大的收获。

在团队里我主要负责了 medrank 算法的编码和调试工作。这是这个项目子模块的最后一个，模块本身实现的难度不是非常大，但需要和编码其他模块的同学进行很多沟通工作，需要比较熟练地掌握 c++的语法和原理，这样才能顺利完成任务。

虽然我负责的工作看上去完成难度并不大，但过程中还是遇到了各种各样的问题，值得我去认真地思考和总结。首先，由于团队没有先对整个项目做一个清晰的架构设计，缺少规范的软件工程开发流程，导致每个子模块的编码人员相互之间沟通困难，理解出现偏差，为后面的代码整合带来很多不必要的麻烦。其次，由于我这个学期代码打的不够多，c++知识遗忘比较严重，导致了程序运行起来出现了很多 bug，花了很多时间去调试。最后，我觉得团队的版本管理做得不够好，没有清晰的版本号，每个人手上都有一个版本，增加了整合代码的成本。

总而言之，这次项目很好地锻炼了我的工程能力，积累了宝贵的经验教训。

黄建武：

这次课程设计项目是学习 C 语言 C++以来最大的一个项目，是对 B+树以及 C++等编程知识的运用，通过这个项目，对 B+树和 MEDRANK 算法也有了更深的理解和认识，见识到 B+树进行查找的高效率。这次项目不能使用 STL，也锻炼了我的编程能力，进行团队合作和沟通的能力，学会了使用 gdb 进行调试的能力，不断进行优化，最大程度地减小内存，减小运行时间，减小 ratio，尽力达到最优。

但是通过这次项目也暴露出一些问题，如对指针的操作不够细心，出现了指针悬空，花了很长的时间才调试出来，虽然进行了分工，但是各个模块的接口没有进行说明，导致后期整合出现了如下标从 0 从 1 开始等差异，增加了整合的难度，缺失版本管理也是一个要注意的问题。

查李吉飞：

本次项目中我的任务比较轻松，主要是算法思路和逻辑上的问题，在编码实现上仅负责一块函数 searchNode 的具体设计实现和算法模块 MedRank 的调试。

在项目中，我和另一位小组成员负责实现 MedRank 算法模块。在他设计的基础框架下，我负责逻辑上的改进和漏洞补丁、更正代码错误。

由于 MedRank 算法需要调用 B_Tree 类大部分的成员函数，因此我们实现 MedRank 算法前需要理解另外两位成员的设计逻辑和具体代码实现，这很考验我们的团队合作，因为互相之间有许多必不可少的交流。

本次项目一定程度上提升了我的编码能力和团队合作能力，对于工程实现积累了一定的实践基础。

八、小组成员分工

小组成员	分工
骆铭涛	预处理，B+树的建立，进行调试优化
童云钊	MedRank 算法实现和调试，整合代码
黄建武	B+树的建立，整合代码，进行调试优化
查李吉飞	MedRank 算法设计和调试