苏州大学实验报告

院、系	计算机学院	年级专业	19 计算机类	姓名	张昊	学号	1927405160
课程名称	数据结构课程实践					成绩	
指导教师	孔芳	同组实验者	无		实验日期	2021 年 1 月 3 日	

实 验 名 称 查找算法的实现及性能测试与比较

一、问题描述及要求

查找算法的实现及性能测试与比较

在顺序线性表中存放 n 个整数，n 的值由用户输入确定，线性表可以是有序表或无序表。比较各查找算法在不同情况下的时间性能。

各查找算法的实测时间性能包括两个指标：算法执行的绝对时间和关键字的平均比较次数。

各查找算法要求评测查找成功与不成功的两种情形。

为了能比较出各种查找算法执行的绝对时间，需要对表中的数据进行较大量的查找，设为 m 次，m 的值也由用户输入确定。当输入 m 为 1000000 时，则对线性表作 1000000 次查找。

（1）比较在有序表和无序表中进行顺序查找时，查找成功和查找失败时的算法执行的绝对时间和关键字的平均比较次数。

（2）比较在同一有序表中进行顺序查找和二分查找时的时间性能。

（3）比较在同一有序表中进行非递归二分查找和递归二分查找的时间性能。

二、概要设计

1. 问题简析

本次实验内容是实现基本的查找算法并通过测试各个算法的绝对运行时间和平均比较次数，来比较其时间性能。

2. 系统功能列表

程序要实现如下功能：

1. 实现指定大小的有序表和无序表的生成；
2. 实现顺序查找算法、非递归二分查找算法和递归二分查找算法，并统计绝对运行时间和比较次数；
3. 分别设计查找成功和查找失败时的测试流程，并统计平均运行时间和平均查找次数。

4. 程序结构设计思路

程序在顺序表的基础上增添三个查找函数，并对构造函数作如下的约束，来满足生成有序表和无序表的需求。

为了分别评测查找成功和查找失败查找情况下的时间性能，设有序表和无序表中

存放 1..2n-1 范围的 n 个奇数。用查找 1..2n-1 中的奇数模拟成功查找，用查找 0..2n 范围中的偶数来模拟失败查找。

在测试函数中，为使得模拟尽量真实，m 次的成功查找或失败查找时查找不同关键字的概率保证相同，即查找成功时，查找 1,3,5,..2n-1 中每个数的概率是一样的；查找失败时，查找 0,2,4,6,..2n-2 中每个数的概率是一样的。

三、详细设计

1 有序表和无序表的生成

为了分别评测查找成功和查找失败查找情况下的时间性能，设有序表和无序表中存放 1..2n-1 范围的 n 个奇数。有序表的生成算法即为依次将这些奇数保存到顺序表中，而无序表只需打乱生成的有序表即可。

```
SearchList::SearchList(int size, bool unordered) : size_(size) {
  for (size_t i = 0; i < size; ++i) {
    data_[i] = 2 * i + 1;
  }
  if (unordered) {
    std::srand((unsigned) std::time(NULL));
    for (int i = 0; i < size; ++i) {
      std::swap(data_[std::rand() % size], data_[i]);
    }
  }
}
```

2 查找算法的实现

顺序查找即为从顺序表的左边开始依次比较，若相等则查找成功，返回下标索引；若完整遍历一遍仍没有相等的记录则查找失败，返回未找到信息。同时记录运行时间和比较次数。

```
int SearchList::sequenceSearch(Record target, Information &info) {
  Time timer;
  for (int i = 0; i < size_; ++i) {
    if (info.compare++, target == data_[i]) {
      info.time = timer.runtime();
      return i;
    }
  }
  info.time = timer.runtime();
  return not_found;
}
```

递归的二分查找则为取左边界 low=0，右边界 high=size-1，计算得中间位置 mid=(low + high) / 2。若 low>high 则结束递归，否则将目标记录 target 与中间位置记

录进行识别相等的比较，相同则返回 mid 位置，小于则在左侧序列 data[low.. mid – 1]中执行递归的二分查找，否则在右侧序列 data[mid + 1..high]中执行递归的二分查找。同时记录运行时间和比较次数。

```
int SearchList::binarySearch(Record target, Information &info) {
  Time timer;
  int res = binarySearchRecursive(0, size_ - 1, target, info);
  info.time = timer.runtime();
  return res;
}

int SearchList::binarySearchRecursive(int low,
                                      int high,
                                      Record target,
                                      Information &info) {
  if (low > high) { return not_found; }
  int mid = (low + high) / 2;
  if (info.compare++, target < data_[mid]) {
    return binarySearchRecursive(low, mid - 1, target, info);
  } else if (info.compare++, target > data_[mid]) {
    return binarySearchRecursive(mid + 1, high, target, info);
  }
  return mid;
}
```

非递归的二分查找与递归的思想相同，只是在实现上将递归过程改为循环过程，通过修改 low 和 high 的值来调整查找区间，具体如下所示。同时记录运行时间和比较次数。

```
int SearchList::binarySearchNonRecursive(Record target, Information &info) {
  Time timer;
  int mid, low = 0, high = size_ - 1;
  while (low <= high) {
    mid = (low + high) / 2;
    if (info.compare++, target < data_[mid]) high = mid - 1;
    else if (info.compare++, target > data_[mid]) low = mid + 1;
    else {
      info.time = timer.runtime();
      return mid;
    }
  }
  info.time = timer.runtime();
  return not_found;
}
```

3 测试流程的设计

测试中用查找 1..2n-1 中的奇数模拟成功查找，用查找 0..2n 范围中的偶数来模拟失败查找。在测试函数中，为使得模拟尽量真实，m 次的成功查找或失败查找时查找不同关键字的概率保证相同，即查找成功时，查找 1,3,5,..2n-1 中每个数的概率是一样的；查找失败时，查找 0,2,4,6,..2n-2 中每个数的概率是一样的。考虑到 m 可能不等于 n，故为查询结果均匀分布，对于测试数据的生成方法设计为：

成功查找	Record record = 2 * int(i * n / m) + 1;
失败查找	Record record = 2 * int(i * n / m);

其中，i 的范围是 0..m-1。

同时，设计了结构体 Information 来储存运行时间和比较次数。在测试函数中获取这些结构体对象，并求平均值，输出到屏幕上。

四、实验结果测试与分析

给定输入 n=100000, m=100000，运行结果如下：

```
List size n = 100000
Search times m = 100000
Sequence Search, unordered list, success
time: 0.000190
compare: 70827
Sequence Search, unordered list, fail
time: 0.000277
compare: 100000
Sequence Search, ordered list, success
time: 0.000132
compare: 48554
Sequence Search, ordered list, fail
time: 0.000279
compare: 100000
Binary Search, ordered list, success
time: 0.000000
compare: 19
Binary Search, ordered list, fail
time: 0.000000
compare: 20
Binary Search non-recursive, ordered list, success
time: 0.000000
```

```
compare: 19
Binary Search non-recursive, ordered list, fail
time: 0.000000
compare: 20
```

对于有序表，顺序查找在绝对运行时间和平均比较次数上要略逊于无序表，但是差距不是很明显，原因可能在于无序表的生成过程中对有序表的打乱不是绝对的随机，而导致的误差。

对于有序表，顺序查找的复杂度为$O(n)$，二分查找的复杂度为$O(\log_2 n)$，平均比较次数分别是$\frac{n}{2}$和$\log_2 n + 1$。可见二分查找的时间性能要优于顺序查找，这是因为二分查找极大地减少了记录与目标之间的比较次数。

对于有序表的非递归二分查找和递归二分查找，可以看出非递归算法有着微弱的优势，原因在于递归的过程需要通过栈来实现，导致时间上变慢。

五、小结

通过这次实验，我对查找技术有了更加深刻的了解，了解了二分查找非递归与递归的具体实现方法，以及查找算法的时间复杂度计算方法。

六、附录

1. **源代码路径**：C++源代码位于附件中 src 目录下。
2. **文件编码**：UTF-8；**行分隔符**：CRLF。
3. **实验环境**：Linux 操作系统，g++编译器，基于 cmake 构建；在 CLion 集成开发环境中调试运行通过。手动编译运行方法为（在 Linux/Unix shell 中）：

```
$ mkdir build  # pwd: src/
$ cd build
$ cmake ..
$ make
$ ./SearchTester
```