**实验报告**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **课程名称** | 《算法分析与设计》 | **实验日期** | 2020年 3月10 日 至2020 年 3月 17 日 | | | | |
| **学生姓名** | 陶玮 | **所在班级** | 计算机184 | | | **学号** | 2018212212019 |
| **实验名称** |  | | | | | | |
| **实验地点** |  | | | **同组人员** | 无 | | |

# 问题

 写出两种检索算法：在一个排好序的数组T[1..n]中查找x，如果x在T中，输出x在T的下标j；如果x不在T中，输出j=0.

# 解析

**顺序查找**：从第一个元素m开始逐个与需要查找的元素x进行比较，当比较到元素值相同(即m=x)时返回元素m的下标，如果比较到最后都没有找到，则返回0。

**二分查找**：查找过程从数组的中间元素开始，如果中间元素正好是要查找的元素，则查找过程结束；如果某一特定元素大于或者小于中间元素，则在数组大于或小于中间元素的那一半中查找，而且跟开始一样从中间元素开始比较。如果在某一步骤数组为空，则代表找不到。

# 设计

**二分算法：**

int BinarySearch(int T[], int x) {

int l = 1, r = n;

while (1 <= r)

{

int m = floor((1 + r) / 2.0);

if (T[m] == x)

return m;

else if (T[m] > x)

r = m - 1;

else

l = m + 1;

}

return 0;

}

顺序查找：

#include<stdio.h>

#include<math.h>

#define n 10

int Search(int T[], int x) {

int i = 0;

for (i = 0; i < n; i++)

{

if (x == T[i]) {

printf("%d", i + 1);

break;

}

if (i == n - 1)

printf("0");

}

return 0;

}

# 分析

二分查找的时间复杂度是O(logn)，顺序查找的时间复杂度是O(n)。

# 源码

https://github.com/kekekeQWQ/3.git