# 查找算法

## ASL

由于查找算法的主要运算是关键字的比较，所以通常把查找过程中对关键字的平均比较次数（平均查找长度）作为衡量一个查找算法效率的标准。ASL= ∑(n,i=1) Pi\*Ci，其中n为元素个数，Pi是查找第i个元素的概率，一般为Pi=1/n，Ci是找到第i个元素所需比较的次数。

## 顺序查找

原理是让关键字与队列中的数从最后一个开始逐个比较，直到找出与给定关键字相同的数为止，它的缺点是效率低下。**时间复杂度o(n)**。

## 折半查找

**折半查找要求线性表是有序表**。搜索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，则搜索过程结束；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。这种搜索算法每一次比较都使搜索范围缩小一半。**折半搜索每次把搜索区域减少一半，时间复杂度为O(log n)。**

* **可以借助二叉判定树求得折半查找的平均查找长度**：log2(n+1)-1。
* 折半查找在失败时所需比较的关键字个数不超过判定树的深度，n个元素的判定树的深度和n个元素的完全二叉树的深度相同log2(n)+1。

public int binarySearchStandard(int[] num, int target){
 int start = 0;
 int end = num.length - 1;
 while(start <= end){ //注意1
 int mid = start + ((end - start) >> 1);
 if(num[mid] == target)
 return mid;
 else if(num[mid] > target){
 end = mid - 1; //注意2
 }
 else{
 start = mid + 1; //注意3
 }
 }
 return -1;
}

* 如果是start < end，那么当target等于num[num.length-1]时，会找不到该值。
* 因为num[mid] > target, 所以如果有num[index] == target, index一定小于mid，能不能写成end = mid呢？举例来说：num = {1, 2, 5, 7, 9}; 如果写成end = mid，当循环到start = 0, end = 0时（即num[start] = 1, num[end] = 1时），mid将永远等于0，此时end也将永远等于0，陷入死循环。也就是说寻找target = -2时，程序将死循环。
* 因为num[mid] < target, 所以如果有num[index] == target, index一定大于mid，能不能写成start = mid呢？举例来说：num = {1, 2, 5, 7, 9}; 如果写成start = mid，当循环到start = 3, end = 4时（即num[start] = 7, num[end] = 9时），mid将永远等于3，此时start也将永远等于3，陷入死循环。也就是说寻找target = 9时，程序将死循环。

## 分块查找

分块查找又称索引顺序查找，它是一种性能介于顺序查找和折半查找之间的查找方法。**分块查找由于只要求索引表是有序的，对块内节点没有排序要求，因此特别适合于节点动态变化的情况**。