|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 查找表结构 | 结构特点 | 查找方法 | 算法思想 | 步骤 | 伪代码 | 性能分析 |
| 无序线性结构 | 前后相继 | 顺序查找 | 从线性表一端开始，逐个进行元素的关键字和给定的值比较，若某个值相等，则查找成功。若查找到线性表末端，都未出现给定值，则查找不成功。 | 1、输入查找序列和待查元素  2、从线性表头遍历元素，比较当前元素与关键值  若相等：查找成功  若不等：继续遍历  3、查找到线性表末端，都未出现给定值，则查找不成功。 | Bool find(List<int>& L,int k){  Int it;  For(L.setstart();L.getvalue(it);L.next())  If(K==it)return true;  Return false  } | O(n) |
| 有序线性结构 | 储存元素间可比较且元素间按大小顺序排列的。 | 二分查找 | 对查找有序序列的范围不断折半，逐步缩小范围直到找到或找不到元素为止。 | 1、输入递增有序数组和要查询的元素值  2、定义两个指针l和r分别指示所查范围的下界和上界，mid指示待查区间中位  3、若待查元素范围大于1，则取mid与关键值比较  若相等：查找成功  小于：将范围的下界和上界设为l和mid-1  大于：将范围的下界和上界设为mid+1和r  否则待查范围没有待查元素，查找不成功 | nt binary\_search(vector<int>v,int key)  {  int left=1,right=v.size()-1,mid;  while(left<=right)  {  mid=(left+right)/2;  if(v[mid]<key) left=mid+1;  else if(v[mid]>key) right=mid-1;  else if(v[mid]==key) return mid;  }  return -1;  } | O(logn) |
| 频率有序查找表 | 将线性表中元素按访问频率排序 | 访问频率有序的查找表 | 根据估算的访问频率从高到低排列记录，执行顺序查找 | 同顺序查找 | 同顺序查找 | C(n)= Σ(pi\*I) |
| 比特位集合 |  | 集合检索 | 在关键码值范围有限的情况下，使用一个位数组，为每一个可能元素分配一个比特位位置，用来表示这个集合，利用布尔操作和位操作实现检索 | 1、将待查找集合按照查找条件转换为位数组  2、检查每个位的布尔值是否为真  3、若为真，则查找成功  若未假，不是 |  |  |