学习目标和要求

1.理解查找的基本含义

2.写出顺序查找算法

3.写出带监督元的顺序查找算法

4.分析比较上述算法的性能

查找

- 查找(search)，给定结点的关键字值（以下简称结点值）x，查找值等于x的结点的存储地址。
  - 按关键字x查

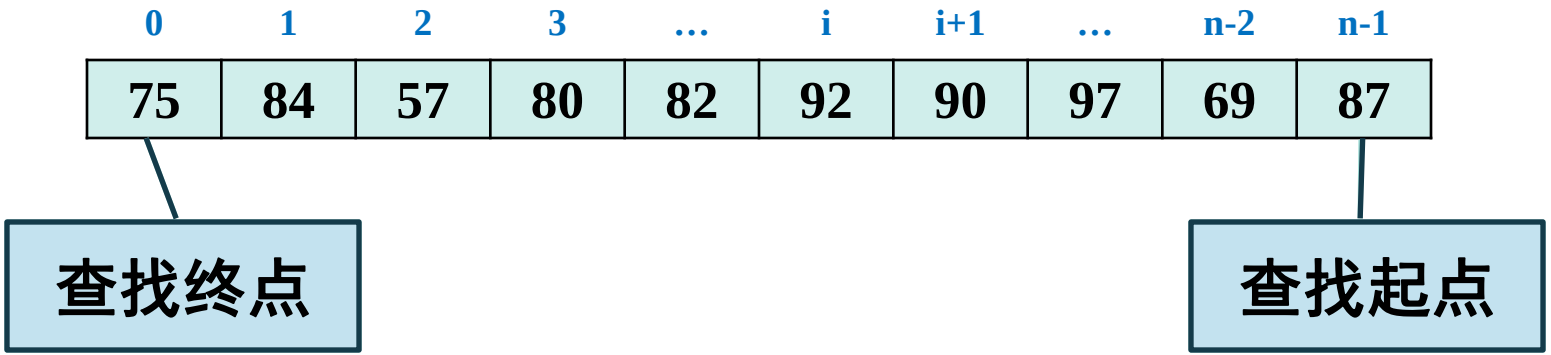
查找结果：成功，返回x的地址

不成功，返回无效地址

  - 按非关键字查

查找结果：可能找出多个符合条件的结点

顺序查找



顺序查找：从表的一端，向另一端，逐个元素查看

- 从左至右
- 从右至左

顺序查找

在a[0]至a[n-1]中，查找值为x的结点的程序段

（1）从左向右查：

```
for(i=0; i<n; i++)
  if (a[i]==x)
    return(i);
return(-1);//表示没找到
```

（2）从右向左查：

```
for (i=n-1; i>=0; i--)
  if (a[i]==x)
    return(i);
return(-1);
```

T(n)=O(n)

顺序查找

在a[0]至a[n-1]中，查找值为x的结点的程序段

（1）从左向右查：

```
for(i=0; i<n; i++)
  if (a[i]==x)
     return(i);
return(-1);//表示没找到
```

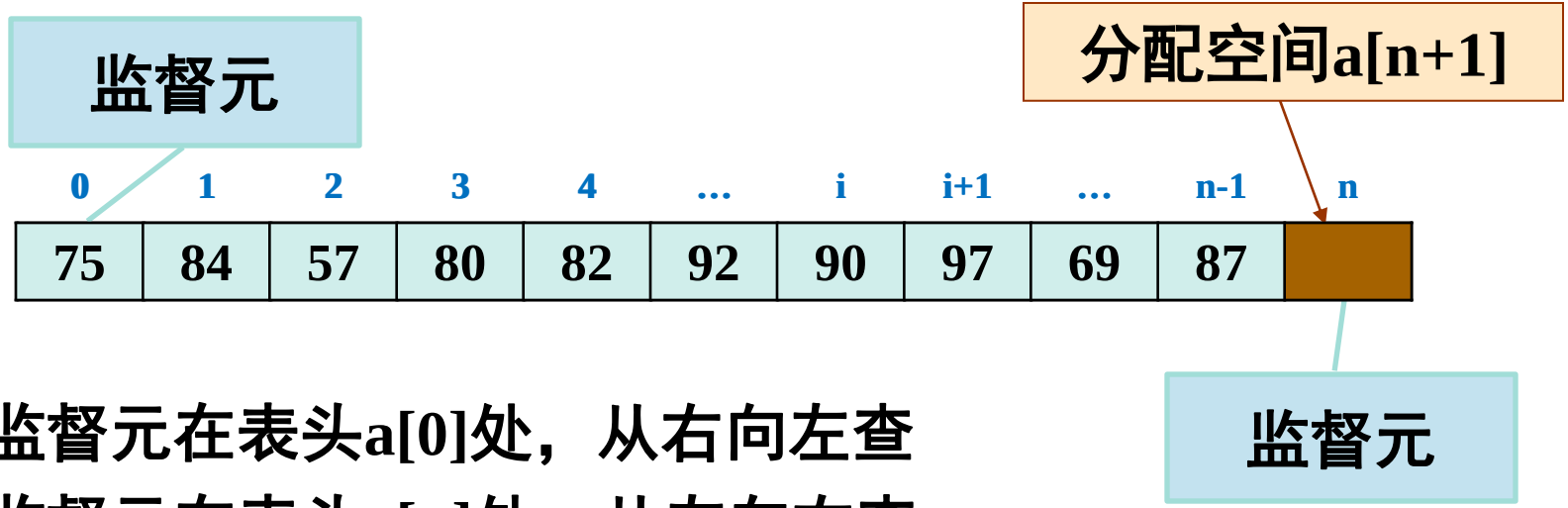
（2）从右向左查：

```
for (i=n-1; i>=0; i--)
   if (a[i]==x)
      return(i);
return(-1);
```

判断两个条件

顺序查找（带监督元）

在查找终点预留一个空白结点（监督元）



监督元在表头a[0]处，从右向左查

监督元在表头a[n]处，从左向右查

顺序查找（带监督元）

带表头监督元的顺序查找算法：

```
int SQsearch(int a[ ], int x, int n)
{      int i;
1.     i=n;   //查找起点
2.     a[0]=x;     //预置监督元
3.    while(a[i]!=x)
          i--;  //从右向左查
4.     return  i ;
}
```

顺序查找（带监督元）

- 只牺牲一个存储结点，便可将查找效率提高近一倍，这在表长**n**较大、查找频繁的情况下，是很“合算”的。这种“以空间换取时间”的做法用得好的话往往能够产生奇效。
- 需要说明的是，虽然查找速度提高了一倍，但时间复杂性的阶不变，仍是**O(n)**，只是时间复杂性函数的常系数变减小了。

教学目标和要求

1.准确描述二分查找的思想

2.写出二分查找的算法

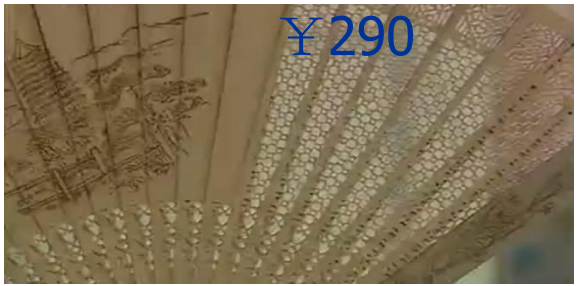
3.能够根据二分查找算法画出判定树，并分析算法性能

实例：猜价格

猜价格

游戏规则

在规定时间内，根据主持人给出关于商品价格高低的提示，快速猜出商品的准确价格。

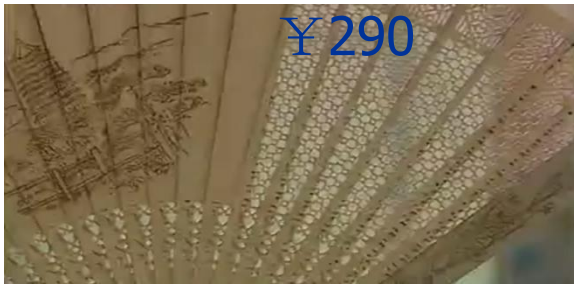


实例：猜价格

猜价格

游戏规则

在规定时间内，根据主持人给出关于商品价格高低的提示，快速猜出商品的准确价格。



竞猜次数	竞猜者价格	主持人回答
1	100	低了
2	200	低了
3	300	高了
4	250	低了
5	260	低了
6	270	低了
7	280	
8	290	恭喜你，答对了

二分查找法（查找元素21）

8	43	199	12	52	5	70	64	29	81	87	21

从小到大排序

5	8	12	21	29	43	52	64	70	81	87	199

0	1	2	3	4	5	6	7	8	9	10	11
5	8	12	21	29	43	52	64	70	81	87	199

二分查找法（查找元素21）

0	1	2	3	4	5	6	7	8	9	10	11
5	8	12	21	29	43	52	64	70	81	87	199

1、找中间位置（中值点）

mid=(0+11)/2=5

二分查找法（查找元素21）

0	1	2	3	4	5	6	7	8	9	10	11
5	8	12	21	29	43	52	64	70	81	87	199

2、与中值点比较

21<43

缩小查找范围至左边序列

二分查找法（查找元素21）

0	1	2	3	4	5	6	7	8	9	10	11
5	8	12	21	29	43	52	64	70	81	87	199

继续找中值点

mid=(0+4)/2=2

二分查找法（查找元素21）

0	1	2	3	4	5	6	7	8	9	10	11
5	8	12	21	29	43	52	64	70	81	87	199

继续与中值点比较

21>12

缩小查找范围至右边序列

二分查找法（查找元素21）

0	1	2	3	4	5	6	7	8	9	10	11
5	8	12	21	29	43	52	64	70	81	87	199

继续找中值点

mid=(3+4)/2=3

二分查找法（查找元素21）

0	1	2	3	4	5	6	7	8	9	10	11
5	8	12	21	29	43	52	64	70	81	87	199

与中值点比较

21=21

查找到，结束

二分查找思想

二分查找核心思想

1、计算中值位置

2、缩小查找区间

二分查找思想

left表示起点，right表示终点，mid表示中值点，
数组a存放元素，x为带查找元素。

1. 计算中值点：
   mid=(left+right)/2；
2. 若x=a[mid],查找成功,返回mid,结束；
2.1 若x<a[mid],则往左缩小查找区间，
   重复上述过程；
2.2 若x>a[mid],则往右缩小查找区间
   重复上述过程

二分查找思想

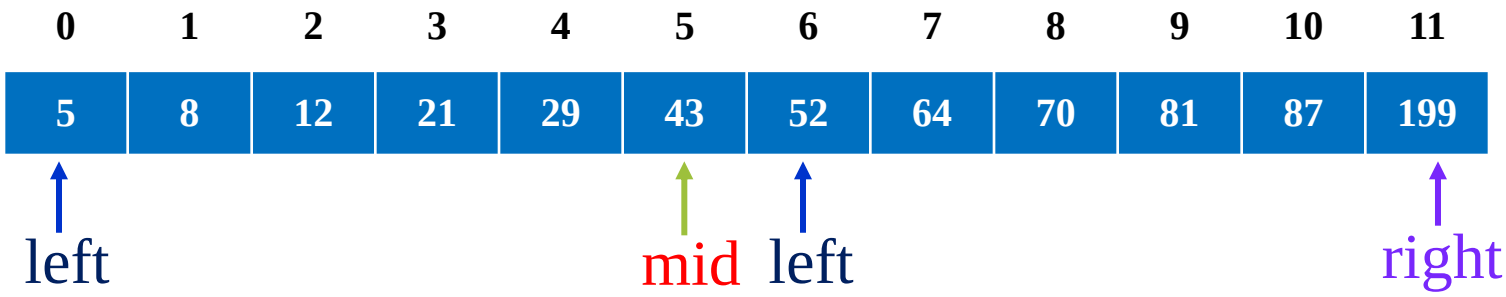
1.　计算中值点；
2. 若x=a[mid],查找成功,返回mid,结束；
2.1 若x<a[mid],则往左缩小查找区间，
　　重复上述过程；
否则
2.2 若x>a[mid],则往右缩小查找区间
　　重复上述过程

二分查找思想

```
int binary_search(int a[],int x,int left,int right)
{ int  mid;
  mid=(left+right)/2;
  if(x==a[mid])  return  mid;
  if(x<a[mid])
     return binary_search(a,x,  left,mid-1  );
  else
    return  binary_search(a,x, mid+1,right );

}
```

二分查找法（查找元素83）



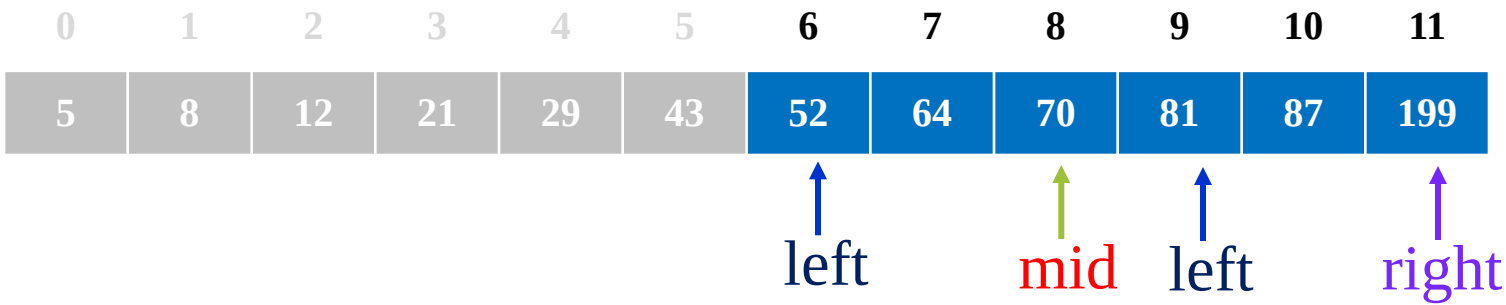
第一步：left=0，right=11

计算中值点：mid=(0+11)/2=5

第一次比较结果：x>a[5]（83>43）

置left=mid+1=6

二分查找法（查找元素83）



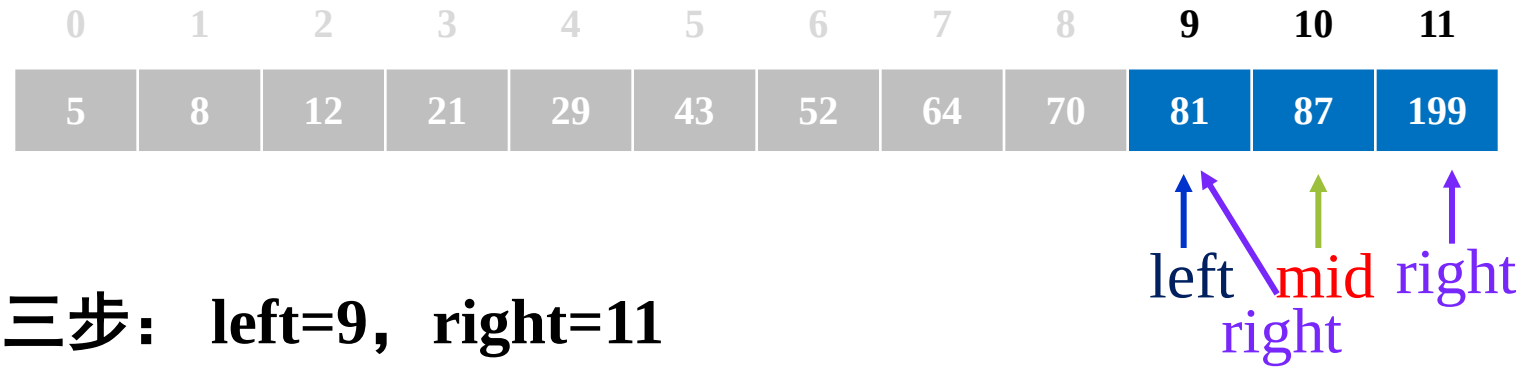
第二步： left=6，right=11

计算中值点： mid=(6+11)/2=8

第二次比较结果：x>a[8]（83>70）

置left=mid+1=9

二分查找法（查找元素83）



第三步： left=9，right=11

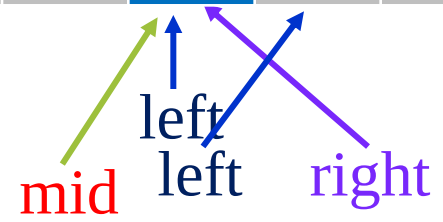
计算中值点： mid=(9+11)/2=10

第三次比较结果：x<a[10]（83<87）

置right=mid-1=9

二分查找法（查找元素83）

0	1	2	3	4	5	6	7	8	9	10	11
5	8	12	21	29	43	52	64	70	81	87	199



第四步： left=9，right=9

计算中值点： mid=(9+9)/2=9

第四次比较结果：x>a[9] （83>81）

置left=mid+1=10

当left>right未查找到

二分查找算法

```
int binary_search(int a[],int x,int left,int right)
{ int  mid;
  if(left>right) return –1;   //查找失败
  mid=(left+right)/2;
  if(x==a[mid])  return  mid;//查找到，返回
  if(x<a[mid])
    return binary_search(a,x,left,mid-1);//左端查找
  return  binary_search(a,x,mid+1,right);//右端查找
}
```

二分查找算法

```
int binary_search(int a[],int n,int x)
{ int  left,right,mid;
  left=0;right=n-1;  //确定查找段的起点和终点
  while(left<=right)
  {
     mid=(left+right)/2;
     if(x==a[mid])  return  mid; //查找到，返回
     if(x<a[mid])   right=mid-1; //左端查找
     else   left=mid+1; //右端查找
   }
   return -1;
}
```

二分查找算法

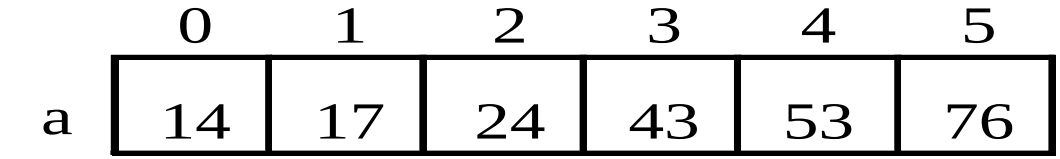
分而治之——分治法

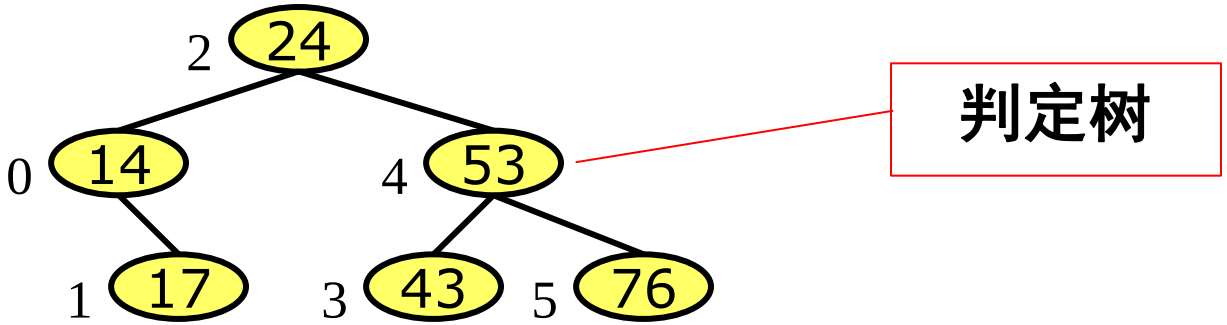
将一个难以直接解决的大问题，分割成一些规模较小的、性质相同的子问题，以便各个击破，分而治之。

前提：（1）顺序存储
（2）有序表

二分查找法性能分析

有序数组a[6]的查找流程：



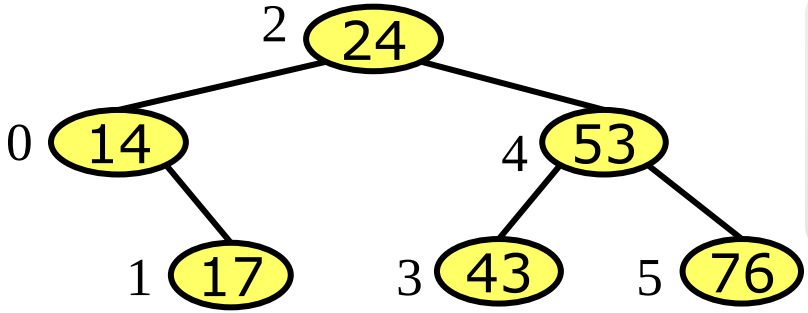


查找：对应于判定树的根到某个结点的查找路径。路径长度(结点个数)等于查找长度。

二分查找法性能分析

有序数组a[6]的查找流程：

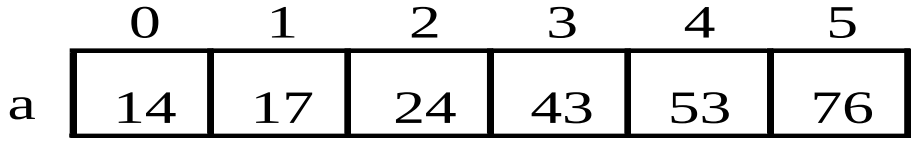
	0	1	2	3	4	5
a	14	17	24	43	53	76

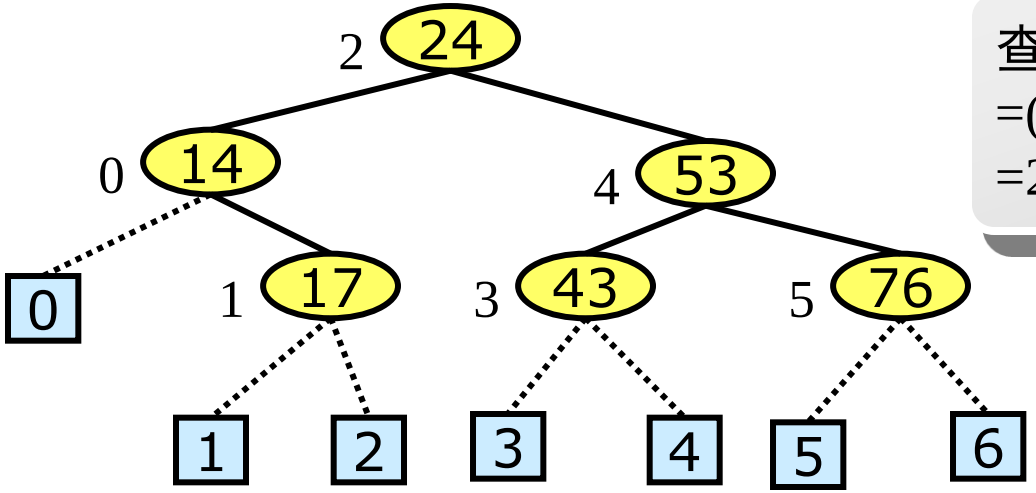


查找成功的平均查找长度
=(1+2+2+3+3+3)/6
=14/6

成功的查找：查找路径终止于结点i，查找长度等于结点i的层数。

二分查找法性能分析





查找不成功的平均查找长度
=(2+3+3+3+3+3+3)/7
=20/7

不成功的查找：查找路径终止于外部结点i，查找长度等于外结点i之父的层数。

二分查找法性能分析

- 设T(n)是在长度为n的有序表中二分查找元素x的查找长度
- 每查找一次，查找范围缩小一半
- T(n)的递推公式：

$$\begin{cases} T(1)=1 & (n=1) \\ T(n)<=1+T(n/2) & (n>1) \end{cases}$$

$$<=1+[\log_2 n]=O(\log_2 n)$$

顺序查找与二分查找性能对比

滴水不漏—穷举法

顺序查找

无序顺序表
逐个查找

一次查找 O(n)

n次查找$O(n^2)$

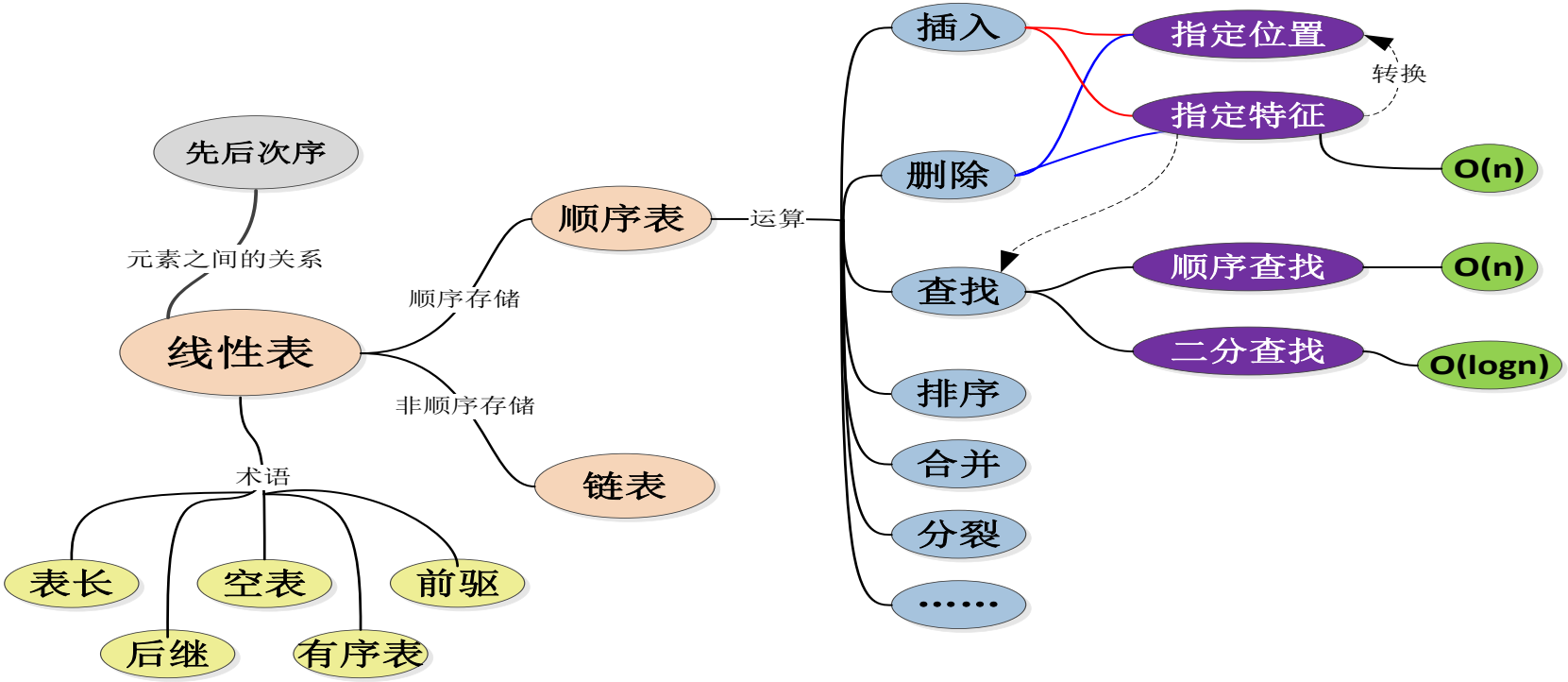
二分查找

有序顺序表
折半查找

一次查找O(logn)
排序时间O(nlogn)
n次查找O(nlogn)

分而治之—分治法

小结


先后次序
元素之间的关系
线性表
顺序存储
非顺序存储
顺序表
链表
运算
术语
表长
空表
前驱
后继
有序表
插入
删除
查找
排序
合并
分裂
……
指定位置
指定特征
转换
O(n)
顺序查找
O(n)
二分查找
O(logn)