11.1.2 折半查找（二分查找）

11.1.2 折半查找

有序表：如果顺序表中的记录按关键字值有序，
即：R[i].key≤R[i+1].key（或R[i].key≥R[i+1].key），
i=1,2,...,n-1，则称顺序表为有序表 。

1 3 7 10 12 124　　有序表

1 3 7 13 12 124　　无序表

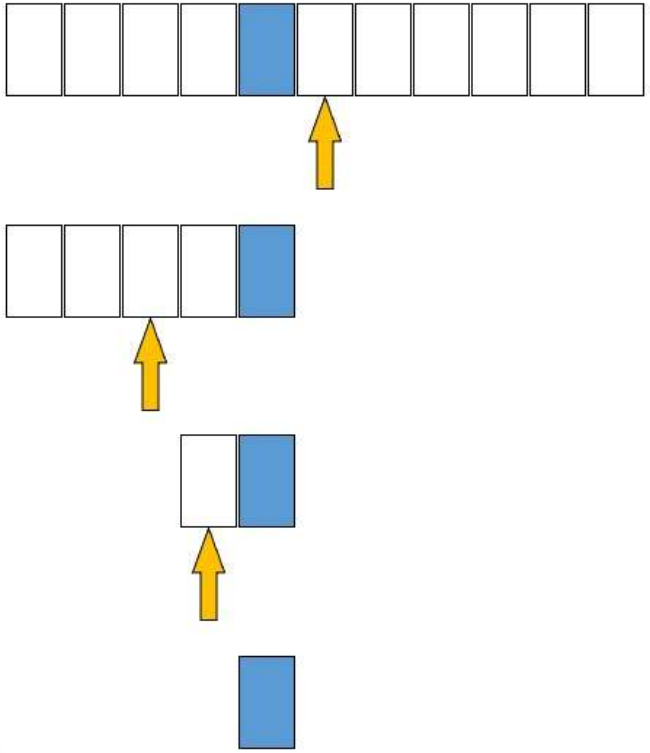
11.1.2 折半查找（二分查找）

查找过程：

将待查关键字与有序表中间位置的记录进行比较，

- 若相等，查找成功
- 若小于，则只可能在有序表的前半部分
- 若大于则只可能在有序表的后半部分

因此，经过一次比较，就将查找范围缩小一半，这样一直进行下去直到找到所需记录或记录不在查找表中。



11.1.2 二分查找的基本实现

查找过程：

将待查关键字与有序表中间位置的记录进行比较，

- 若相等，查找成功
- 若小于，则只可能在有序表的前半部分
- 若大于则只可能在有序表的后半部分

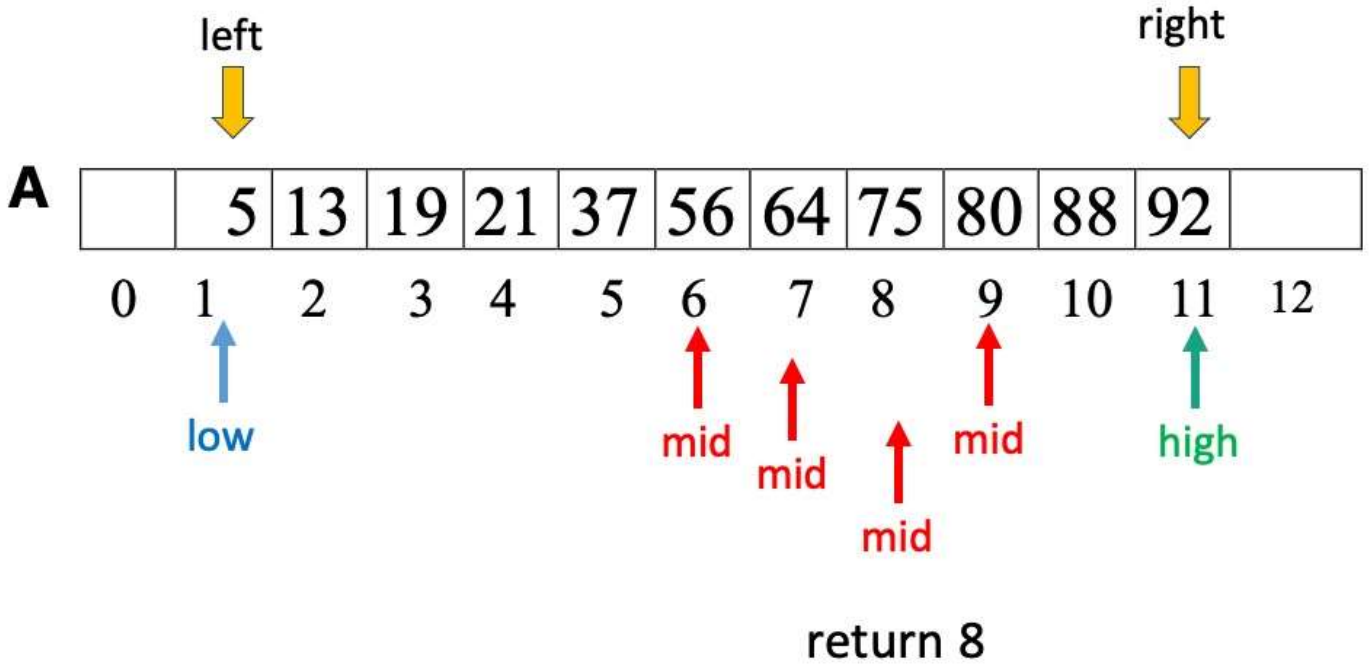
因此，经过一次比较，就将查找范围缩小一半，这样一直进行下去直到找到所需记录或记录不在查找表中。

算法：BinarySearch(A, left, right, key)
输入：顺序表A，数据按升序排列，整数left, right，数据key
输出：如果key在子表A[left...right]中，返回位置；否则返回-1

```
1.  low ← left     //low和high分别指向查找区域左右两端
2.  high ← right       //查找区间 [low, high]
3.  while low ≤ high do
4.  |   mid ← (low + high) / 2   //中间位置
5.  |   if A[mid] = key then
6.  |   |   return mid
7.  |   else if key < A[mid] then
8.  |   |   high ← mid – 1    //查找前半区间 [low, mid-1]
9.  |   else       //key > A[mid]
10. |   |   low ← mid + 1    //查找后半区间 [mid+1, high]
11. |   end
12. end
13. return -1
```

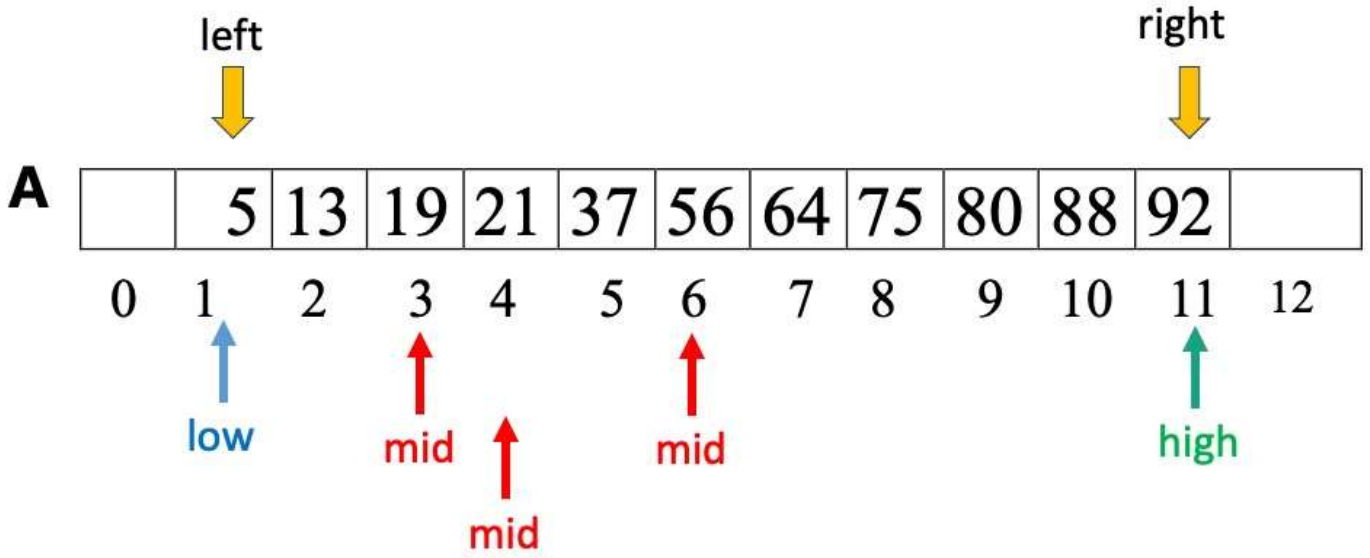
时间复杂度：O(log(n))

例如: key=75 的查找过程如下:



return 8

例如: key=20 的查找过程如下:



high < low: 查找失败!

return -1

11.1.2 二分查找的简单实现

查找过程：

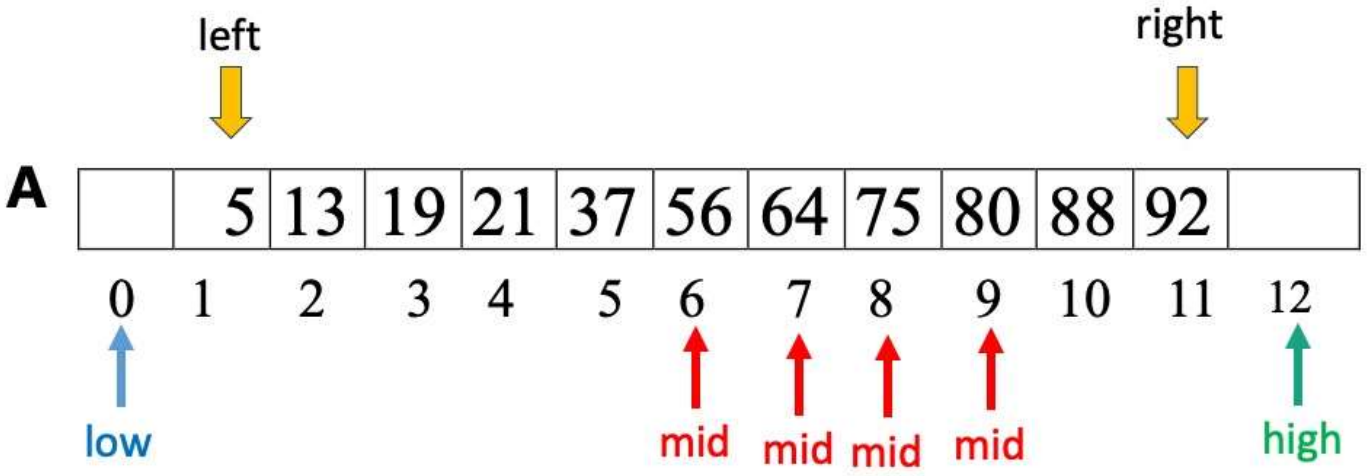
将待查关键字与有序表中间位置的记录进行比较，

- 若相等，查找成功
- 若小于，则只可能在有序表的前半部分
- 若大于则只可能在有序表的后半部分

因此，经过一次比较，就将查找范围缩小一半，这样一直进行下去直到找到所需记录或记录不在查找表中。

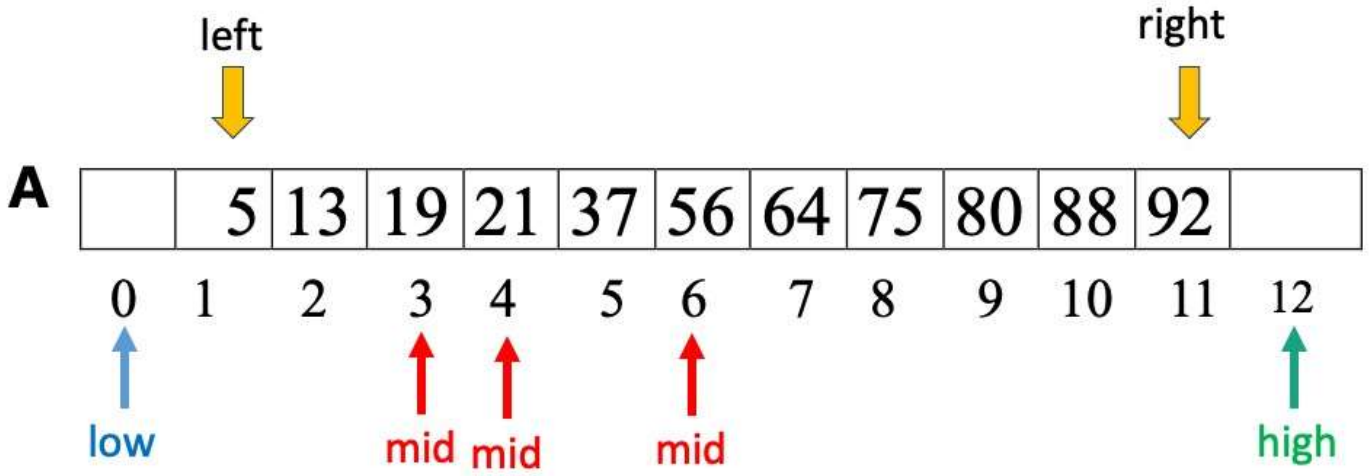
```
算法：BinarySearch(A, left, right, key)
输入：顺序表A，非负整数left, right，数据key
输出：如果key在子表A[left...right]中，返回位置；否则返回-1

1.  low ← left - 1        //low和high放在查找区域外边!
2.  high ← right +1  //查找区间 (low, high)
3.  while high - low > 1 do
4.  |   mid ← (low + high) / 2  //中间位置：low<mid<high
5.  |   if A[mid] = key then
6.  |   |   return mid
7.  |   else if key < A[mid] then
8.  |   |   high ← mid          //查找前半区间 (low, mid)
9.  |   else          //key > A[mid]
10. |   |   low ← mid          // 查找后半区间 (mid, high)
11. |   end
12. end
13. return -1                    时间复杂度：O(log(n))
```

例如: key=75 的查找过程如下:



return 8

例如: key=20 的查找过程如下:



high - low = 1 查找失败!

return -1

多选题 1分

在未排序的数组上作顺序查找和在排好序的数组上作二分查找，下面的说法中，哪些是错误的

A 顺序查找一定比二分查找慢

B 顺序查找在最好情况下只需比较1次，而二分查找做不到

C 如果查找的元素不在数组中，两个算法的时间效率基本相同

D 二分查找也可以在双向链表上执行

11.1.2 二分查找的应用：区间查询

问题描述：假设序列（顺序表）A[1...n], 满足$A[1] \le A[2] \le \cdots \le A[n]$，求A中所有大小在区间$[a, b)$中的数据。

思路：求序列A中 $\ge a$ 的最小值位置和 $< b$ 的最大值位置

关键算法：

- 假设$A[\text{left} - 1] = -\infty, A[\text{right} + 1] = +\infty$

(1) 开始时，low ← left – 1，high ← right + 1

(2) 计算区间(low, high)的中间位置mid ← (low + high) / 2

(3) 如果$A[\text{mid}] \ge \text{key}$, high ← mid; 若$A[\text{mid}] < \text{key}$, low ← mid

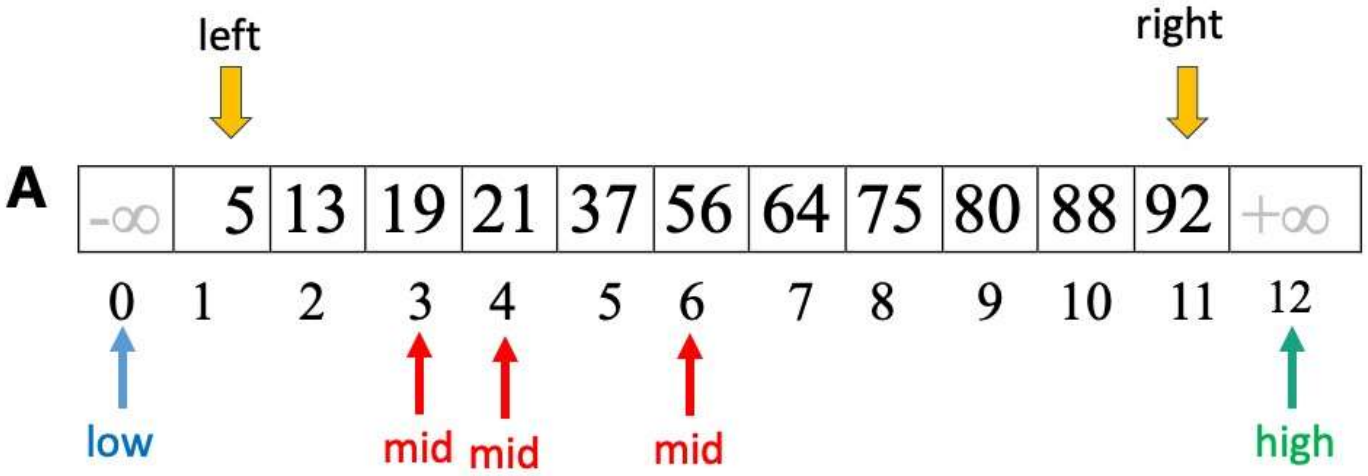
(4) 如果high – low > 1, 回到(2)，继续查找；否则结束操作

- A[high] ≥ *key*
- A[low] < key

保持

思考：当high = low + 1时，A[high]和A[low]与key是什么关系?

例如: key=20 的查找过程如下:



high - low = 1 : 19是小于20的最大值，21是大于等于20的最小值!

11.1.2 二分查找的应用：区间查询

问题描述：假设序列（顺序表）A[1...n], 满足$A[1] \leq A[2] \leq \cdots \leq A[n]$，求A中所有大小在区间$[a, b)$中的数据。

思路：求序列A中 $\geq a$ 的最小值位置和 $< b$ 的最大值位置

思考：如何求A中大小在区间 (a, b], (a, b), [a, b] 内的数据?

关键算法

```
1.  low ← left - 1
2.  high ← right +1
3.  while high - low > 1 do
4.  |   mid ← (low + high) / 2
5.  |   if key ≤ A[mid] then
6.  |   |   high ← mid
7.  |   else        //key > A[mid]
8.  |   |   low ← mid
9.  |   end
10. end
11. return high  // ≥ key的最小值位置
                 或
                 low   // < key的最大值位置
```

11.1.2 二分查找的应用：快速求幂

---不似“二分”，恰是二分

问题描述：给定正整数a和n，求 a^n 的值　　例如：求$3^{1000000000}$

- 直接迭代：$a^n = a^{n-1} * a$

  时间复杂度（相乘次数）$O(n)$

- 二分递归：$a^n = a^{\frac{n}{2}} * a^{\frac{n}{2}} * a^{n\%2}$

  时间复杂度: $O(\log(n))$

```
算法：  Power(a, n)
输入：正整数a, n
输出：a^n
---------------------------------------
1. if n = 1 then
2. |   return a
3. end
4. pow ← Power(a, n/2)  //递归计算a^(n/2) (二分)
5. pow ← pow * pow
6. if n % 2 = 1 then  //n是奇数
7. |   pow ← pow * a
8. end
9. return pow
```

11.1.2 二分查找的应用：快速查找 ---不似“二分”，恰是二分

问题描述：查找未排序序列$< a_1, a_2, \ldots, a_n >$中的第k小元素 $(1 \le k \le n)$。

算法1：循环k次选择排序或冒泡排序 ---时间复杂度: O(kn)

思考：插入排序是否可用?

算法2：快速或递归排序 ---时间复杂度: O(nlog(n)) + O(1)

排序 找第k小元素

算法3：快速建最小堆+k次出堆（调整）

---时间复杂度: O(n) + O(klog(n))

思考：能否实现O(n)时间的快速查找?

11.1.2 二分查找的应用：快速查找 ---不似“二分”，恰是二分

问题描述：查找未排序序列< $a_1, a_2, \ldots, a_n$ >中的第k小元素 $(1 \le k \le n)$。

思考：能否实现O(n)时间的快速查找?

思路：

- 对序列< $a_1, a_2, \ldots, a_n$ >快速排序时，首先选择一个基准值pivot对序列进行划分
- **划分结果**：≤ pivot 的所有数据排在它前面，>pivot 的数据全部排在后面，然后返回pivot在序列中的排位m
- 如果m = k，说明pivot就是第k小元素
- 如果m < k, 第k小元素在pivot的后半段
- 相反，若m > k, 第k小元素在pivot的前半段

} 二分原则!

11.1.2 二分查找的应用：快速查找

---不似“二分”，恰是二分

问题描述：查找未排序序列$< a_1, a_2, \ldots, a_n >$中的第k小元素 $(1 \le k \le n)$。

算法：

时间复杂度：

- 最好情况：O(1)
- 最坏情况：$O(n^2)$
- 平均情况：？？

算法：QuickSearch(A, l, r, k)
输入：序列A, 整数l, r , k，满足$1 \le l \le k \le r$
输出：在子串A[l...r]中查找A的第k小元素

```
1. if l < r then  //子串A[l...r]含多个元素
2. |  m ← Partition(A, l, r)  //选A[r]为pivot进行划分，
3. |                          //返回基准值的排位 (l ≤ m ≤ r)
4. |  if m = k then
5. |  |  return A[m]
6. |  end
7. |  if m < k then     // m < k ≤ r 成立
8. |  |  return QuickSearch(A, m+1, r, k)  //A[m+1...r]中查找
9. |  else    // l ≤ k < m
10.|  |  return QuichSearch(A, l, m-1, k)  //A[l...m-1]中查找
11.|  end
12.end
13.return A[l]  //子串只含一个元素，即 l = r = k
```

11.1.2 二分查找的应用：快速查找 ---不似“二分”，恰是二分

问题描述：查找未排序序列$< a_1, a_2, \ldots, a_n >$中的第k小元素 $(1 \le k \le n)$。

算法：

时间复杂度：

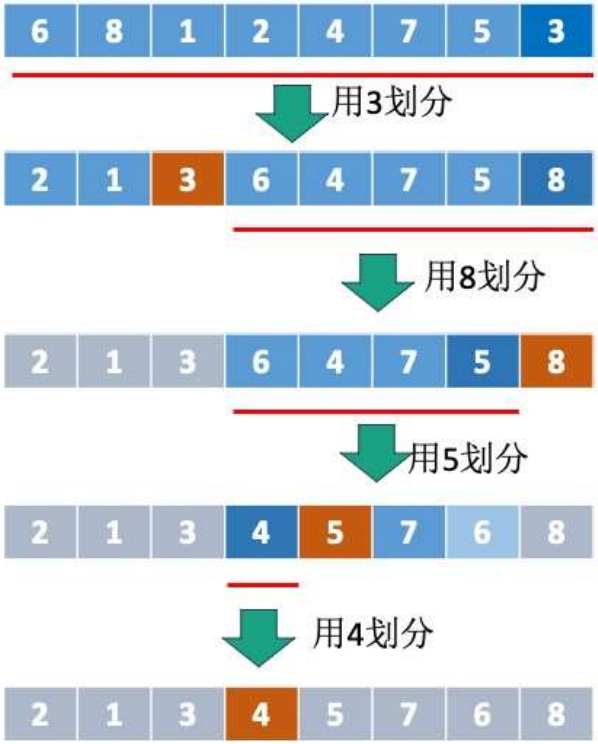
- 最好情况：O(n)
- 最坏情况：$O(n^2)$
- 平均情况：？？

- 随机选择基准值对序列划分，两个子序列的平均长度比为1 ： 3
- 并且90%以上的概率，子序列的长度比不低于1 ： 19!
- 因此，90%以上的概率，查找的时间T(n)满足

$$
\begin{aligned}
T(n) &\le T\left(\frac{19}{20}n\right) + O(n) \\
&\le T\left(\frac{19^2}{20^2}n\right) + O\left(\frac{19}{20}n + n\right) \\
&\le T\left(\frac{19^k}{20^k}n\right) + O\left(\frac{19^{k-1}}{20^{k-1}}n + \cdots + \frac{19}{20}n + n\right) \\
&\le T(1) + O\left(\ldots + \frac{19^k}{20^k}n + \cdots + \frac{19}{20}n + n\right) \\
&= O(n)
\end{aligned}
$$

线性时间

例如: 查找第4小元素的过程如下:



查找成功，返回4

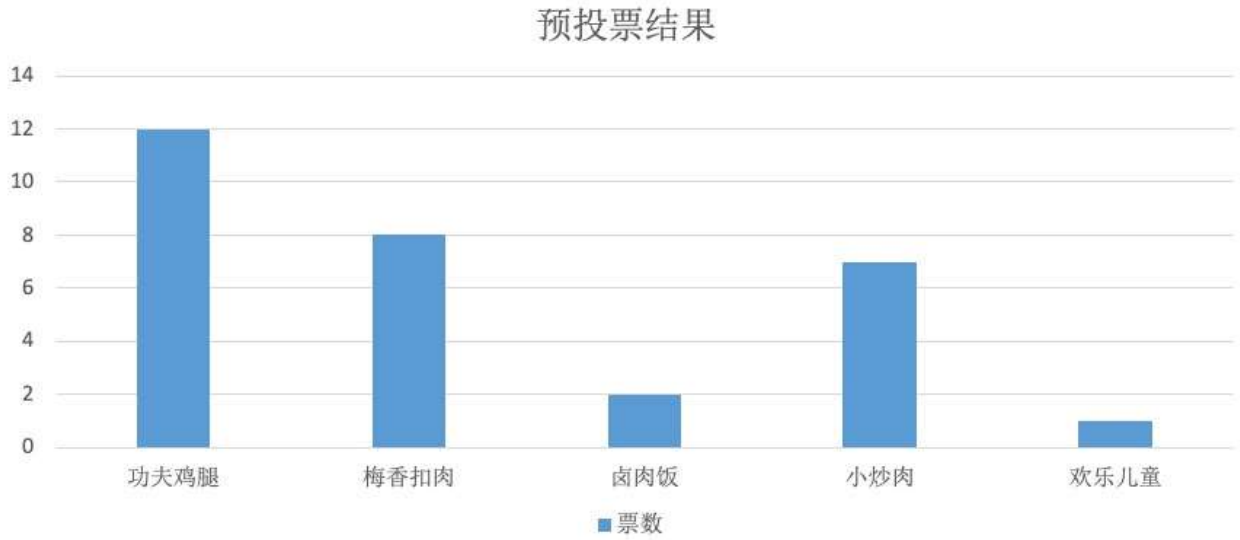
顺序查找的应用：一道趣题*

问题描述：

- Z.Y.班n个人在实验室学习，到了中午，大家决定一起从乡村基点外卖。
- 按照惯例，大家一人一票投票表决，然后一起点获票数最多的套餐！
- 但今天，班长突然“萌发奇想”，想尝尝**儿童套餐**，怎么办呢？
- 为“梦想成真”，精明的班长先做了一次预投票，提前了解到每个人想投的套餐。
- 要让儿童套餐获票最多，必须“以理服人”，让足够多的人改选儿童套餐，但思想工作难做（ZY班没有省油的灯！！！）
- 因此，班长让你设计一个拉票方案，使儿童套餐的获票最多（不允许票数并列），同时“说服”的人数最少。

顺序查找的应用：一道趣题*

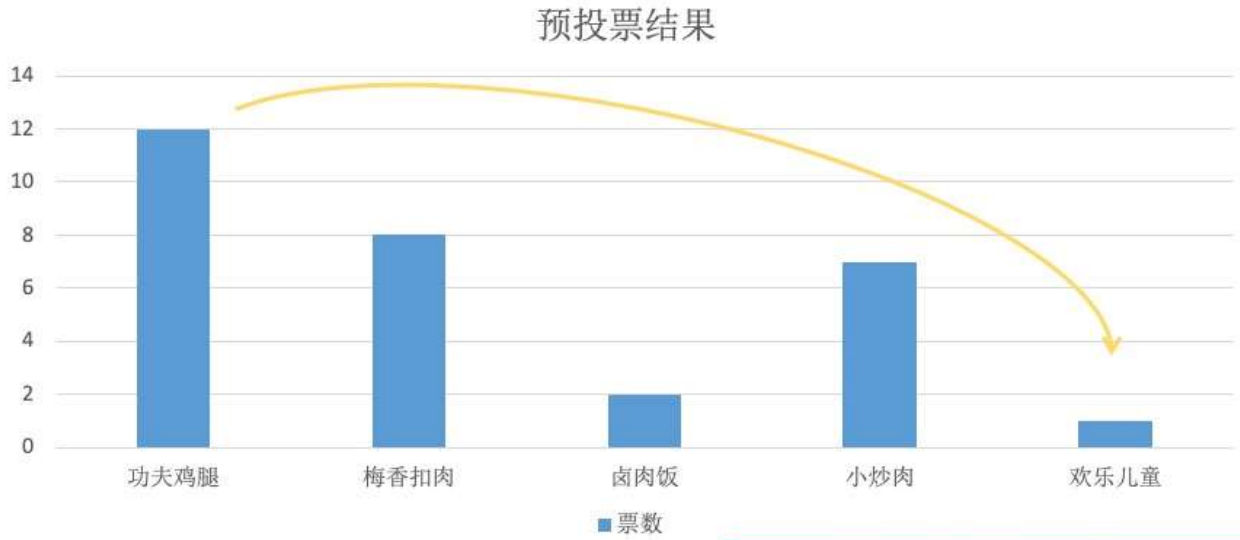
问题描述：



Q：要使儿童套餐获得最多的票，最少需要拉多少票？应该拉哪些套餐的投票？

顺序查找的应用：一道趣题*

问题描述：



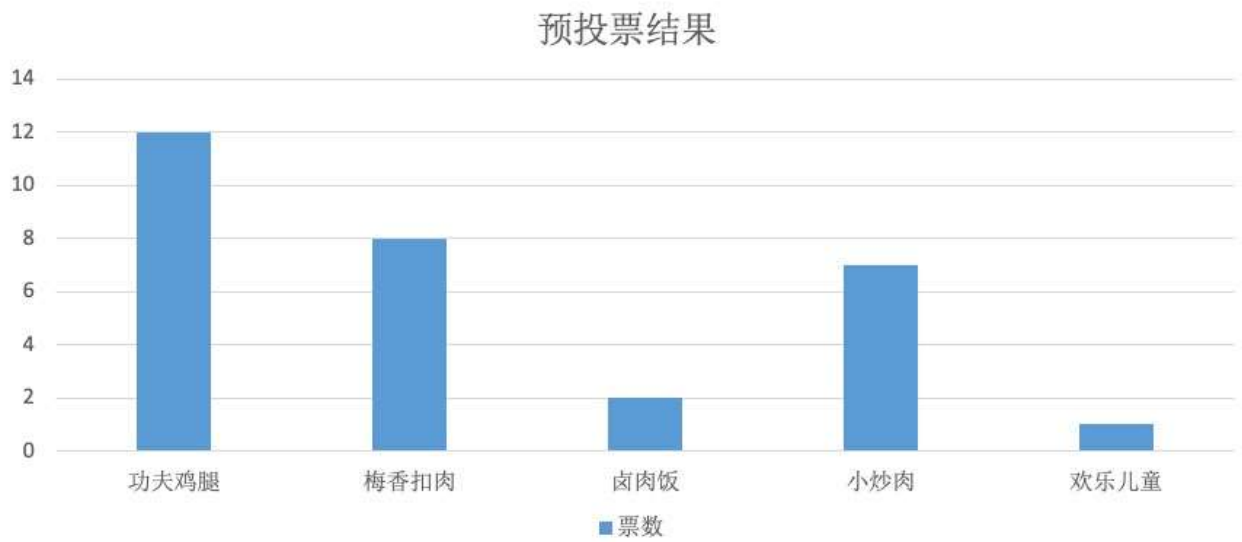
- **贪心法**：每次选择当前获票最多的套餐，从中拉1票过来，重复该过程，直到儿童套餐的得票超过其它套餐

- 可以用最大堆维护当前得票最多的套餐
- **时间复杂度**：O(m logn)，其中m是总票数（人数），n是套餐数

高等教育出版社

顺序查找的应用：一道趣题*

问题描述：



思路：

(1)设置票数上限k

(2) 如果套餐的得票超过k，将多余的票全部转给儿童套餐

(3) 设置儿童套餐的目标票数为k+1，如果加上(2)转来的票后票数超过k+1，超出的票成为废票；相反，如果票数不够k+1，还需要从其它套餐拉不足的票！

顺序查找的应用：一道趣题*

问题描述：

思路：

(1)设置票数上限k

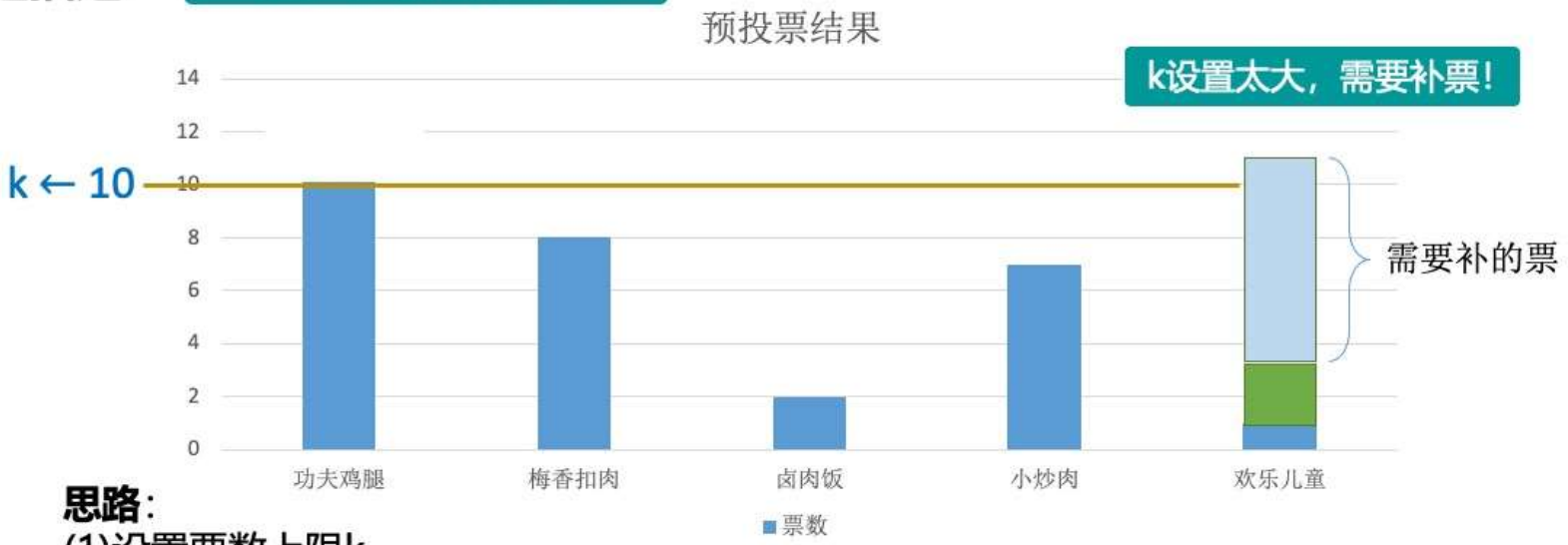
(2) 如果套餐的得票超过k，将多余的票全部转给儿童套餐

(3) 设置儿童套餐的目标票数为k+1，如果加上(2)转来的票后票数超过k+1，超出的票成为废票；相反，如果票数不够k+1，还需要从其它套餐补齐不足的票！

顺序查找的应用：一道趣题*

问题描述：

- 思考：如何求最优的k值?



思路:

(1)设置票数上限k

(2) 如果套餐的得票超过k, 将多余的票全部转给儿童套餐

(3) 设置儿童套餐的目标票数为k+1, 如果加上(2)转来的票后票数超过k+1，超出的票成为废票；相反，如果票数不够k+1，还需要从其它套餐补齐不足的票!

投票 最多可选1项

对最优的k值，哪种票是绝对不能有的?

A 废票

B 补票

C 两种票都不能有

11.1.2 二分查找的应用：一道趣题* ---不似“二分”，恰是二分

- **思考**：如何求最优的k值?

二分法思路：

(1) 设置low_k ← 0, high_k ← m (总票数) //k=0表示所有票都给儿童套餐

(2) 计算mid_k = (low_k + high_k) / 2 //设置mid_k为当前的k值（上限值）

(3) 查找得票数超过mid_k的套餐，并统计多出的票的总和extra_sum

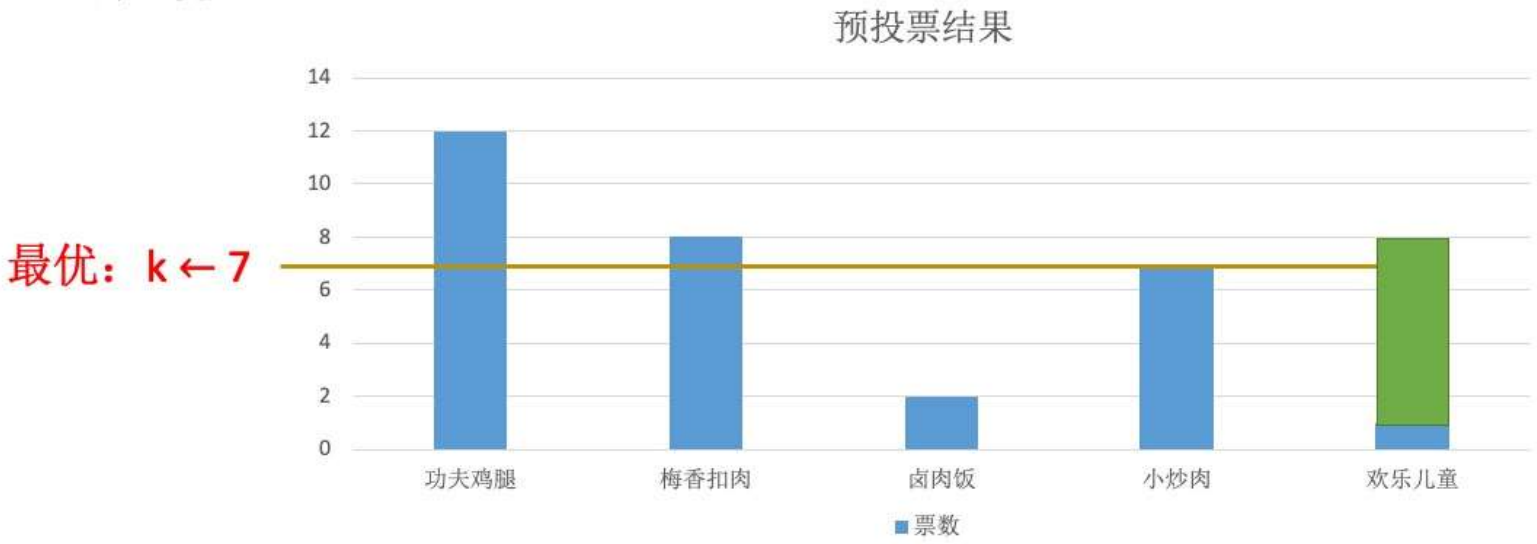
- 如果 extra_sum + #儿童 ≤ mid_k + 1，high_k ← mid_k
  //k太大，可能需要补票，可以减小k
- 如果 extra_sum + #儿童 > mid_k + 1，low_k ← mid_k
  //k太小，有废票，继续测试更大的k

(4) 如果 low_k + 1 = high_k, 结束查找，返回high_k（最优k）；否则，回到(2)继续查找

- **时间复杂度**：n log(m) **或** log(m)*log(n) + n log(n)

顺序查找的应用：一道趣题*

问题描述：

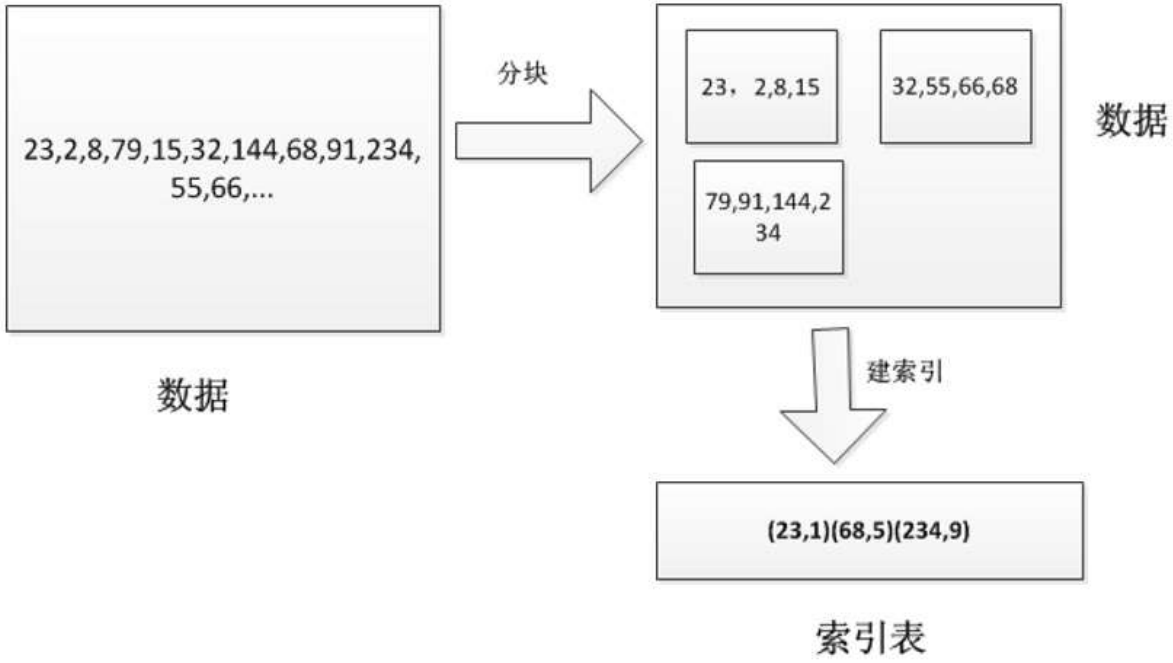


11.1.3 索引表查找

数据太多，杂乱无章，查找困难！

第一个块地址	第一个块最大关键字	第二个块地址	第二个块最大关键字	.....	.....	第n个块地址	第n个块最大关键字

11.1.3 索引查找


23,2,8,79,15,32,144,68,91,234,
55,66,...
数据
分块
23，2,8,15
32,55,66,68
79,91,144,2
34
数据
建索引
(23,1)(68,5)(234,9)
索引表


索引表有序
索引表的查找
查找表的查找
索引表的查找

例子：

关键字	22	48	86	
指针	1	7	13	19(n+1)

22, 12, 13, 8, 9, 20, 33, 42, 44, 38, 24, 48, 60, 58, 74, 49, 86, 53

例子：

查找关键字k=38

关键字	22	48	86	
指针	1	7	13	19(n+1)

22, 12, 13, 8, 9, 20, 33, 42, 44, 38, 24, 48, 60, 58, 74, 49, 86, 53