查　找

大连理工大学

刘 馨 月

二分法查找

二分法查找

- 有序表：线性表中所有数据元素按关键码值递增（或递减）的次序排列。

- 算法的基本思想：
  - 先给数据排序（例如按升序排好），形成有序表。
  - 将key与正中元素相比，若两者相等，则查找成功。若key小，则缩小至前一半有序表中查找；再取其中值比较，每次缩小1/2的范围，直到查找成功或失败为止。反之，如果key大，则缩小至后一半有序表中查找。

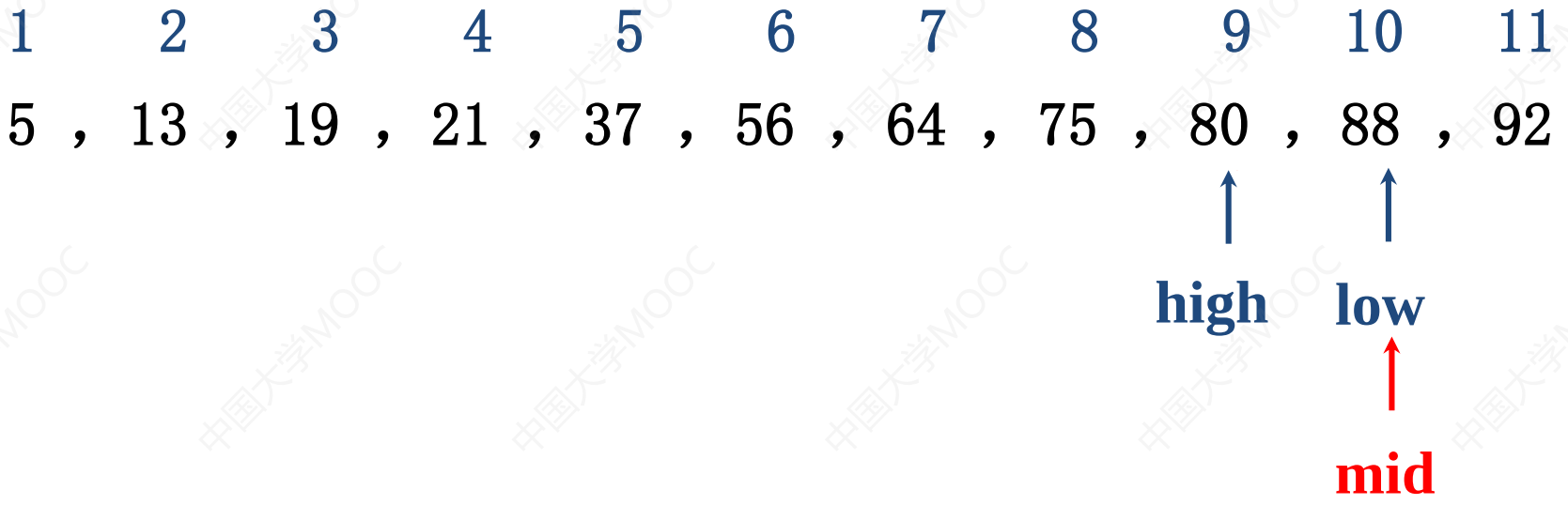
二分法查找

表中数据元素按照主关键字顺序排列。

5，13，19，21，37，56，64，75，80，88，92

二分法查找

$$mid = \lfloor (low + high)/2 \rfloor$$



查找 21

mid = 6　high = mid – 1 = 5

mid = 3　low = mid + 1 = 4

mid = 4　找到

查找 85

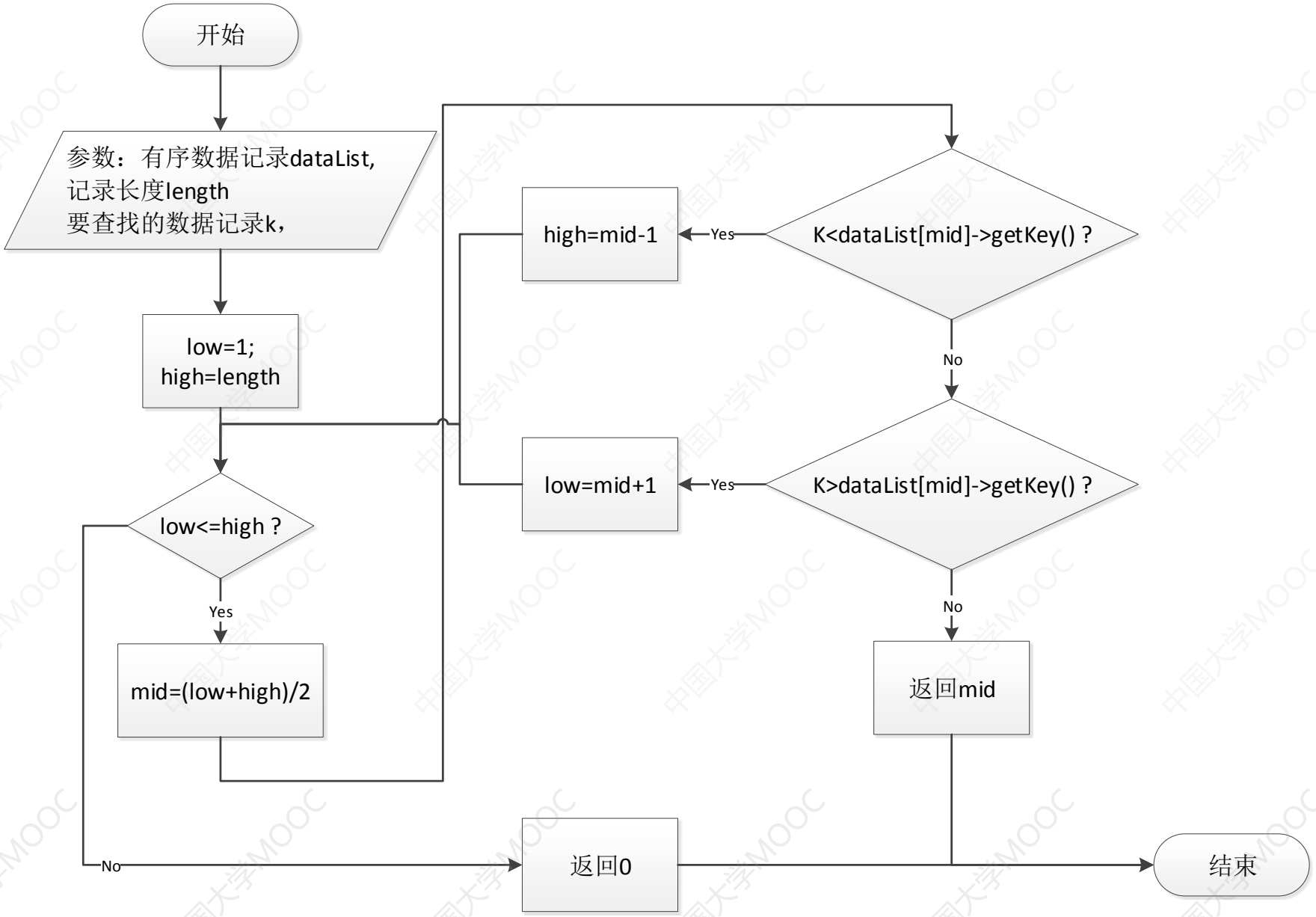
mid = 6　low = mid + 1 = 7

mid = 9　low = mid + 1 = 10

mid = 10　high = mid–1 = 9

high < low　查找不成功

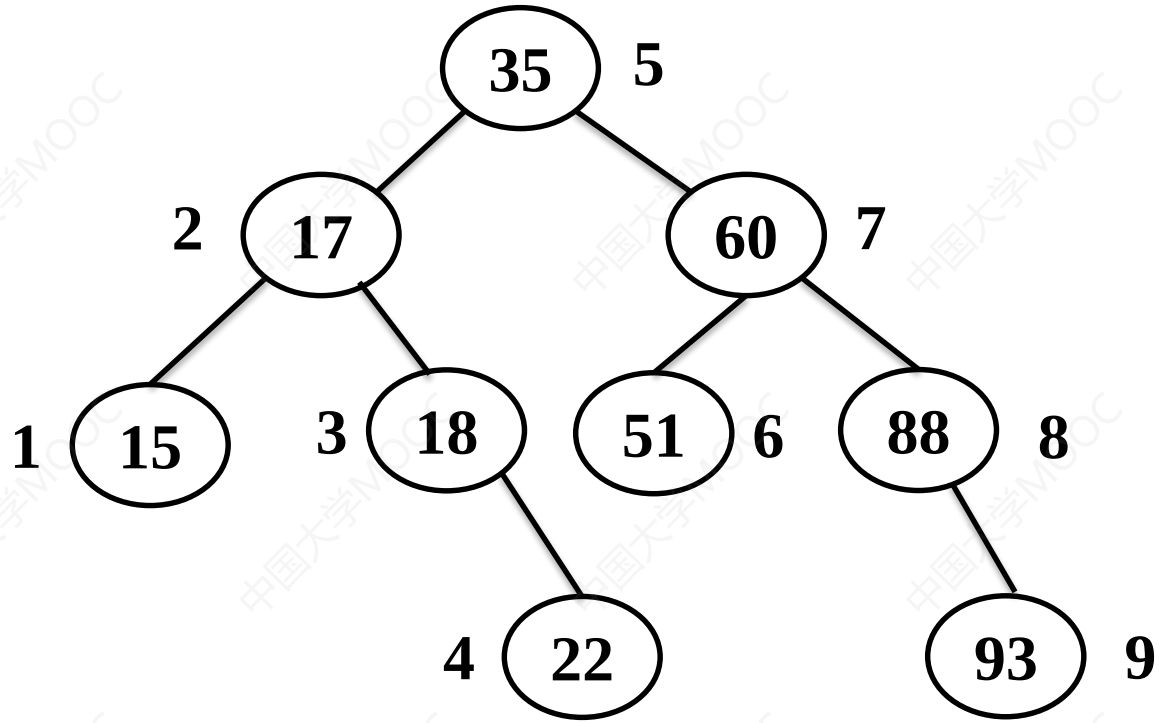
二分法查找


开始
参数：有序数据记录dataList,
记录长度length
要查找的数据记录k，
low=1;
high=length
low<=high ?
Yes
mid=(low+high)/2
K<dataList[mid]->getKey() ?
Yes
high=mid-1
No
K>dataList[mid]->getKey() ?
Yes
low=mid+1
No
返回mid
No
返回0
结束

二分法查找

```
template  <class Type> int BinSearch (vector<Item<Type>*>& dataList,int length,Type k)
    {
    //low, high分别记录数组首尾位置
    int low=1, high=length, mid;
    while (low<=high){
        mid=(low+high)/2;
        if (k<dataList[mid]->getKey())
                high = mid-1;    //向左缩检索区间
        else if (k>dataList[mid]->getKey())
                low = mid+1;     //向右缩检索区间
        else
            return mid;    //检索成功，返回元素位置
     }
       return 0;     //检索失败，返回0
}
```

二分法查找



- 最大检索长度为 $\lceil \log_2 (n+1) \rceil$
- 失败的检索长度是$\lceil \log_2 (n+1) \rceil$ 或$\lfloor \log_2 (n+1) \rfloor$

二分法查找

- 查找成功的平均检索长度为：

$$\begin{aligned} \mathrm{ASL} &= \frac{1}{n}\left(\sum_{i=1}^{j} i \cdot 2^{i-1}\right) \\ &= \frac{n+1}{n}\log_2(n+1)-1 \\ &\approx \log_2(n+1)-1 \end{aligned}$$

(n > 50)

- 优缺点
  - 优点：平均检索长度与最大检索长度相近，检索速度快
  - 缺点：要排序、顺序存储，不易更新(插/删)

查　找

大连理工大学

刘 馨 月