第三讲：动态数据结构

王争

前 Google 工程师

什么是动态数据结构?

支持高效动态增删改查的数据结构:

- 有序数组
- 跳表
- 散列表
- 二叉树
- 堆

目录

1. 跳表

2. 散列表

3. 二叉树

4. 堆

跳表

目录

1. 基于链表实现二分查找

2. 改造链表为跳表

3. 跳表的时间复杂度分析

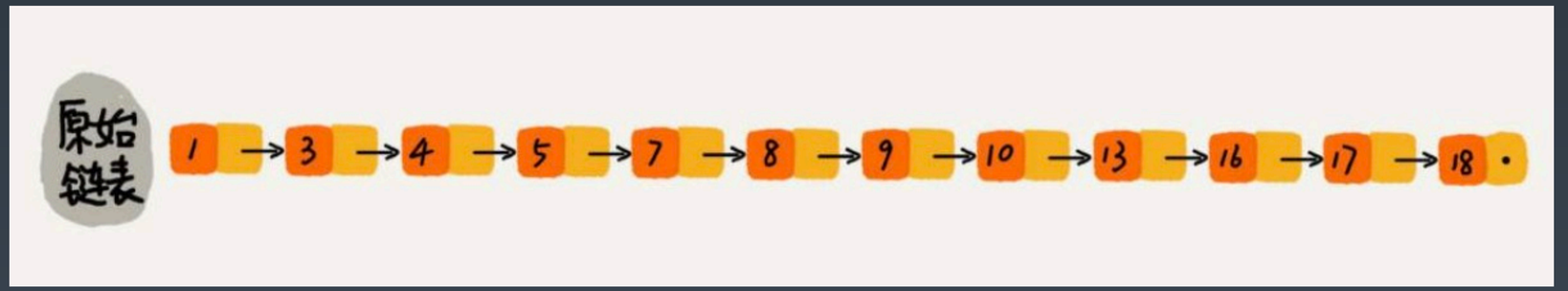
4. 跳表的空间复杂度分析

链表是否支持二分查找?

二分查找底层依赖的是数组随机访问的特性，所以只能用数组来实现。

如果数据存储在链表中，还能否用二分查找算法?

如何在链表中实现二分查找?

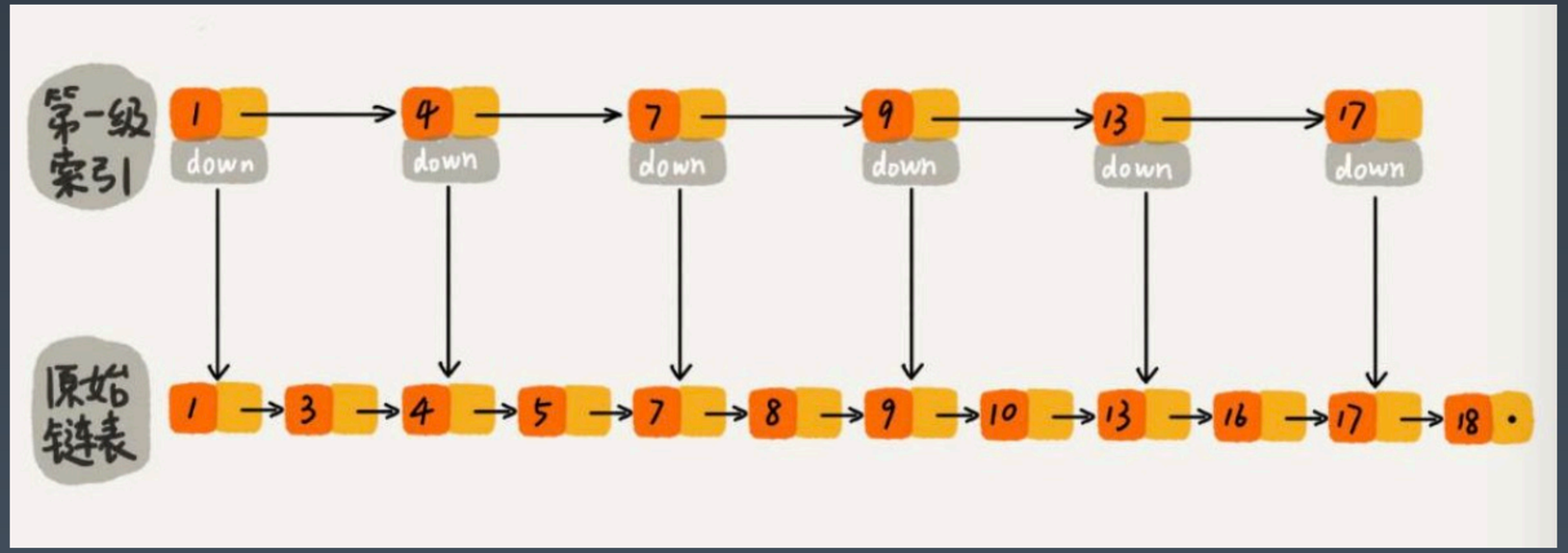


时间复杂度:

n/2 + n/4 + n/8 + … + 1 = O(n)

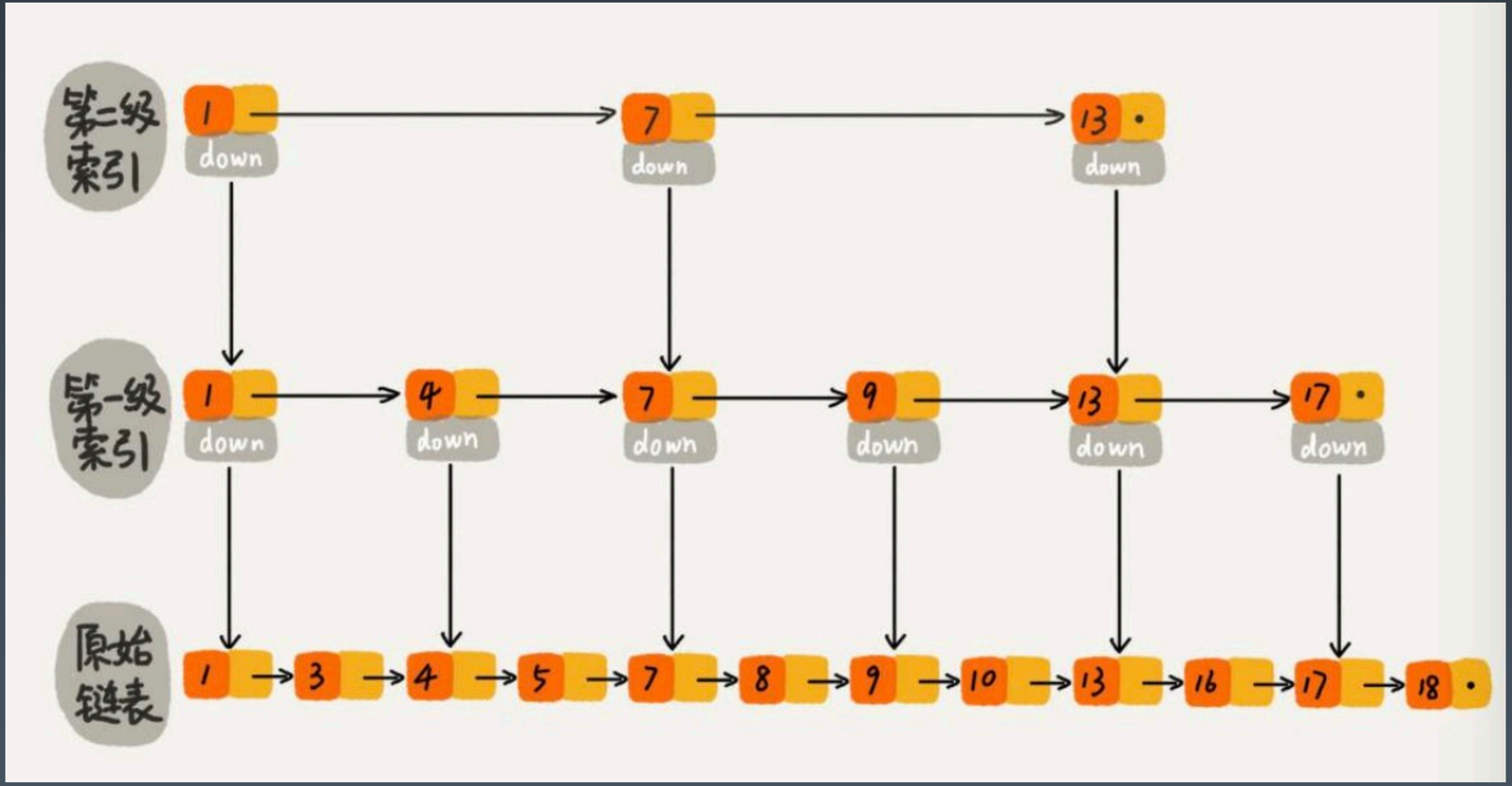
将链表改造成跳表：添加第一级索引

如何提高链表线性查找的效率?



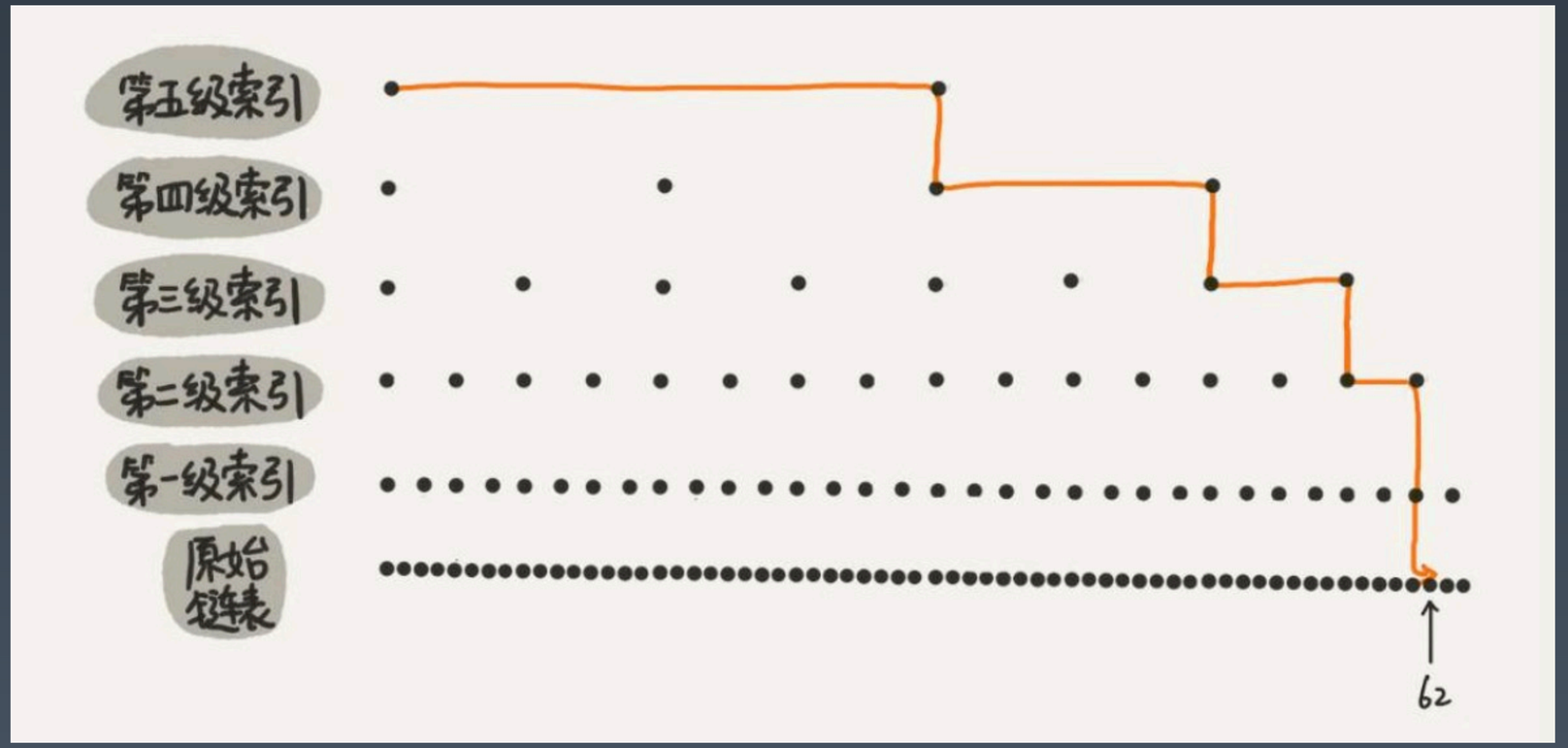
将链表改造成跳表：添加第二级索引

如何进一步提高链表查找的效率?

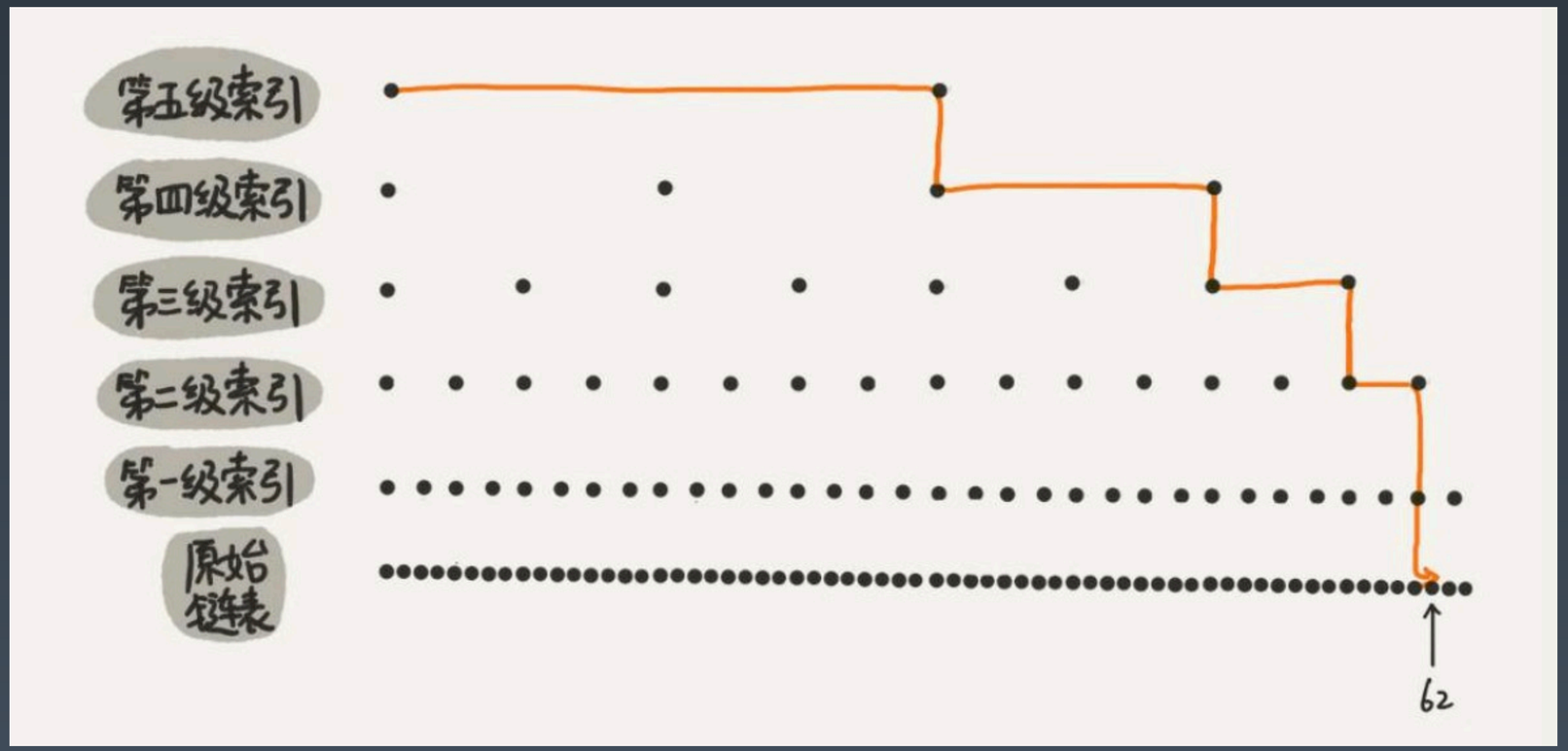


将链表改造成跳表：添加多级索引

如何再进一步提高链表查找的效率?



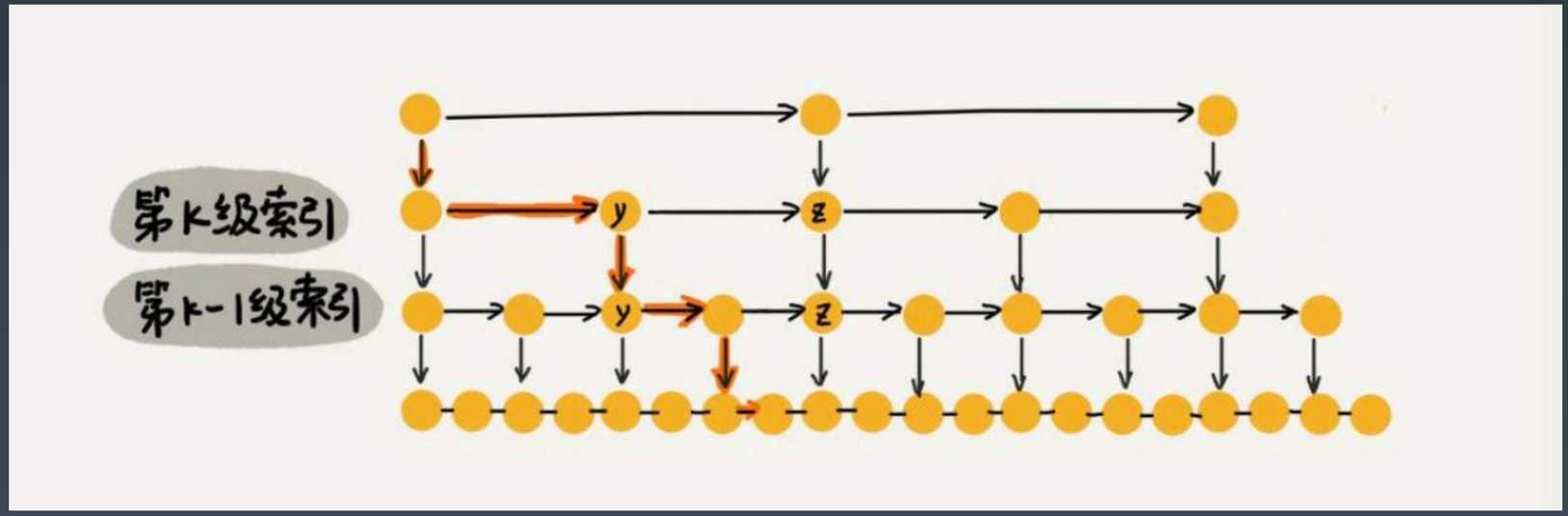
跳表查询的时间复杂度分析



n/2、n/4、n/8、第 k 级索引结点的个数就是 n/(2^k)

假设索引有 h 级，最高级的索引有 2 个结点。n/(2^h) = 2，从而求得 h = log2(n)-1

跳表查询的时间复杂度分析



索引的高度：logn，每层索引遍历的结点个数：3

在跳表中查询任意数据的时间复杂度就是 O(logn)

跳表的空间复杂度分析

原始链表大小为 n，每 2 个结点抽 1 个，每层索引的结点数：

$$\frac{n}{2}, \frac{n}{4}, \frac{n}{8}, \cdots, 8, 4, 2$$

原始链表大小为 n，每 3 个结点抽 1 个，每层索引的结点数：

$$\frac{n}{3}, \frac{n}{9}, \frac{n}{27}, \cdots, 9, 3, 1$$

空间复杂度是 O(n)

散列表

散列表+跳表实战

假设猎聘网有10万名猎头，每个猎头都可以通过做任务（比如发布职位）来积累积分，然后通过积分来下载简历。
假设你是猎聘网的一名工程师，如何在内存中存储这10万个猎头ID和积分信息，让它能够支持这样几个操作：

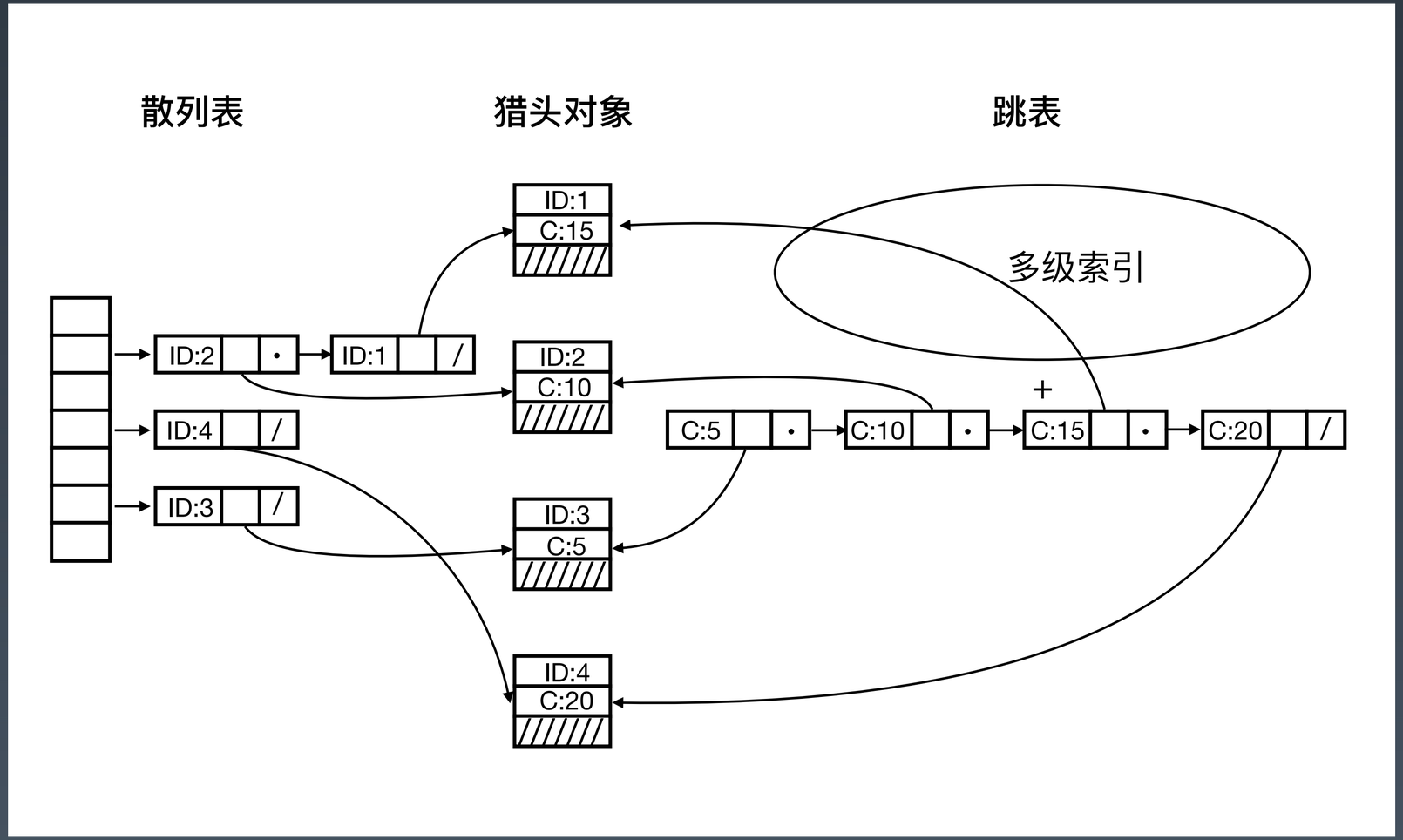
- 根据猎头的ID快速查找、删除、更新这个猎头的积分信息；
- 查找积分在某个区间的猎头ID列表；

解题思路：

这个问题既要通过ID来查询，又要通过积分来查询，所以，对于猎头这样一个对象，我们需要将其组织成两种数据结构，才能支持这两类操作。

按照ID，将猎头信息组织成散列表。
按照积分，将猎头信息组织成跳表，按照积分来查找猎头信息，就非常高效，时间复杂度是O(logn)。

散列表+跳表实战


散列表
猎头对象
跳表
ID:1
C:15
多级索引
ID:2
ID:1
ID:2
C:10
ID:4
ID:3
C:5
C:10
C:15
C:20
ID:3
C:5
ID:4
C:20

引申思考

大部分数据结构和算法书籍中，在讲某种数据结构和算法的时候，都会拿整数、字符串这些基本数据类型作为要处理的数据的类型。

实际上，在真实的软件开发中，数据结构中存储的数据、算法要处理的数据，往往都不是简单的整数，而是“对象”。这里的“对象”很好理解，就是编程语言的中的“类与对象”中的对象。

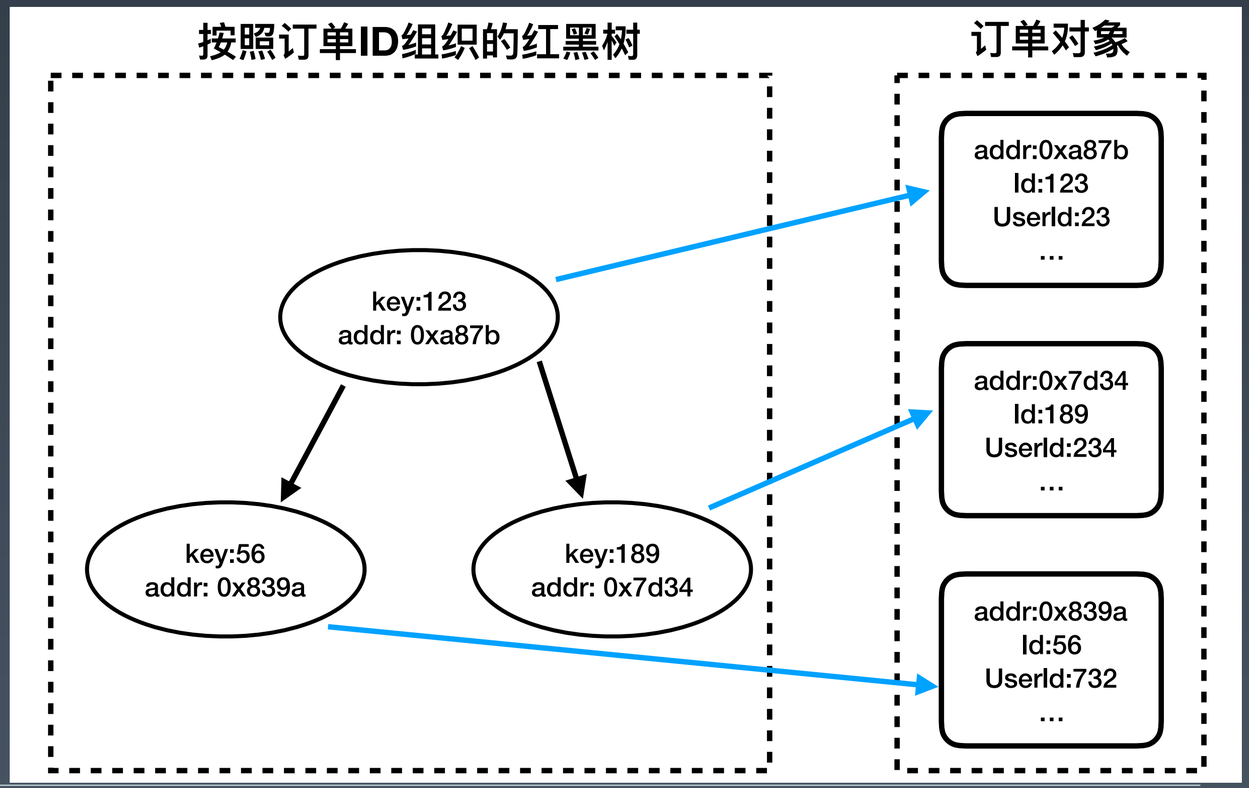
对象中的数据分为两个部分：

1. 键值：参与构建数据结构和算法
2. 卫星数据：不参与构建数据结构，只作为附属信息

引申思考

下面这几行 Java 代码，我们用红黑树来存储 Order 订单对象。

```
// key是订单ID，value是订单对象
private TreeMap<Long, Order> id2OrderMap = new TreeMap<>();

public void add(Order order) { // 添加一个订单
  id2OrderMap.put(order.id, order);
}

private class Order { // 订单类
public long id;
public long userId;
public long createTime;
  // ...more fields...
}
```



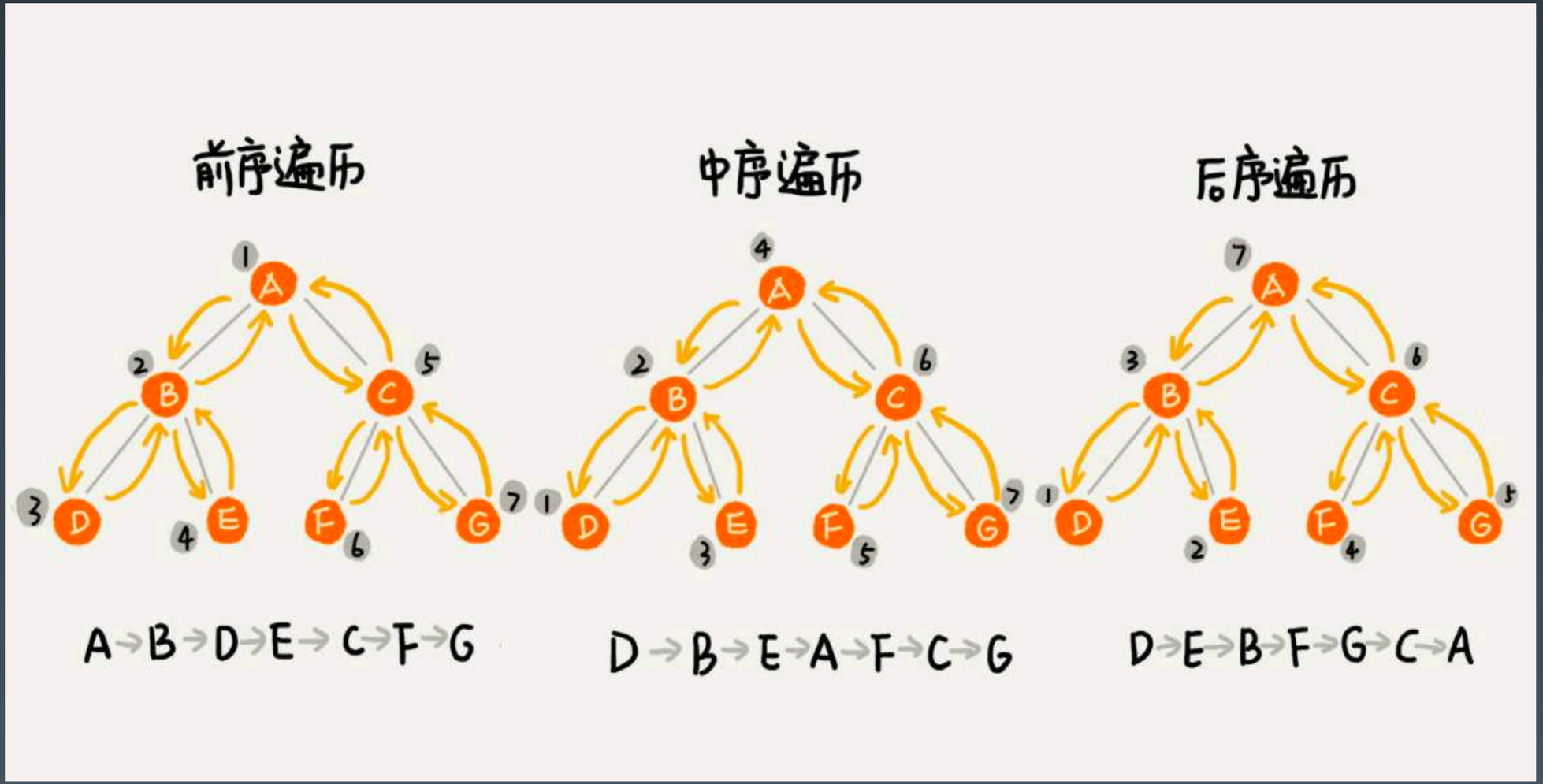
二叉树

目录

1. 前、中、后序遍历二叉树

2. 求二叉树的高度

遍历二叉树



二叉树的前序遍历

```
class TreeNode {
  public int data;
  public TreeNode left;
  public TreeNode right;
}

public void preOrder(TreeNode root) {
  if (root == null) return;
  System.out.println(root.data);
  preOrder(root.left);
  preOrder(root.right);
}
```

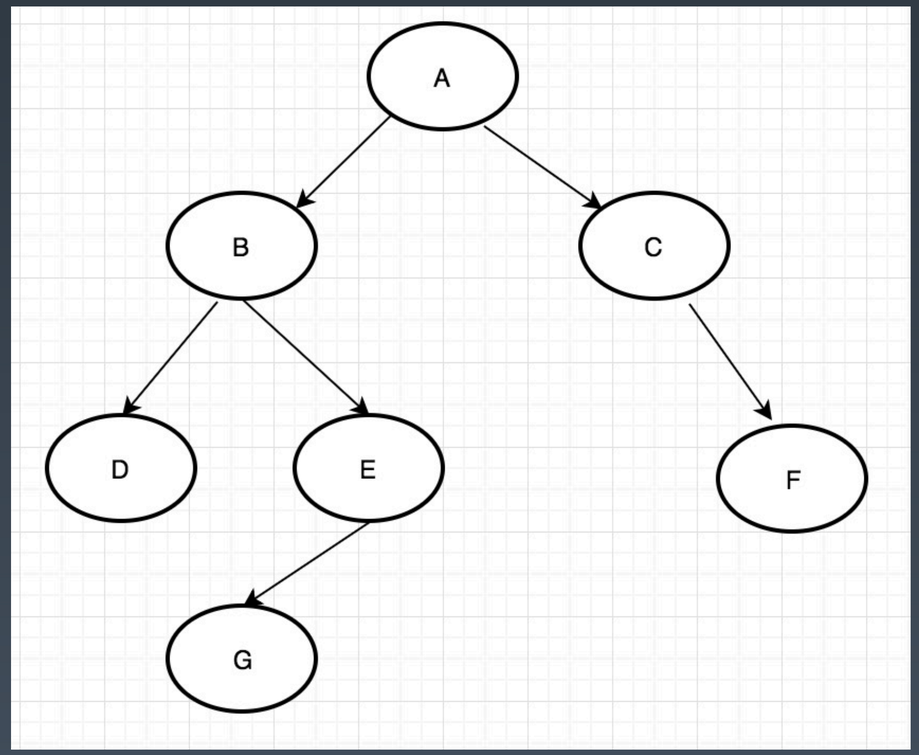
二叉树的中序遍历

```
class TreeNode {
  public int data;
  public TreeNode left;
  public TreeNode right;
}

public void inOrder(TreeNode root) {
  if (root == null) return;
  inOrder(root.left);
  System.out.println(root.data);
  inOrder(root.right);
}
```

二叉树的后序遍历

```
class TreeNode {
  public int data;
  public TreeNode left;
  public TreeNode right;
}

public void postOrder(TreeNode root) {
  if (root == null) return;
  postOrder(root.left);
  postOrder(root.right);
  System.out.println(root.data);
}
```

求二叉树的高度




```
height(root) = max(height(root.let),height(root.right)) +1
```

求二叉树的高度

```
public int getTreeHeight(TreeNode root) {
  if (root == null) return -1;
  if (root.left == null && root.right == null) {
    return 0;
  }

  int leftTreeHeight = 0;
  int rightTreeHeight = 0;
  if (root.left != null) {
    leftTreeHeight = getTreeHeight(root.left);
  }
  if (root.right != null) {
    rightTreeHeight = getTreeHeight(root.right);
  }
    return Math.max(leftTreeHeight, rightTreeHeight) + 1;
}
```

堆

目录

1. 堆的应用一：优先级队列

2. 堆的应用二：求 TOP K

3. 堆的应用三：求中位数

堆的应用一：优先级队列

假设我们有大小为 10GB 的文件。

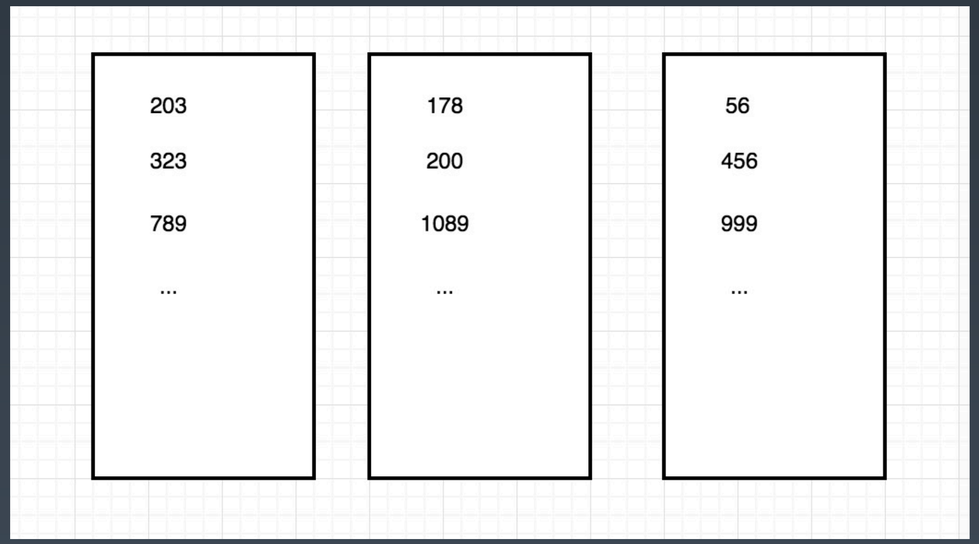
如果计算机只有大约 512MB 大小的可用内存，

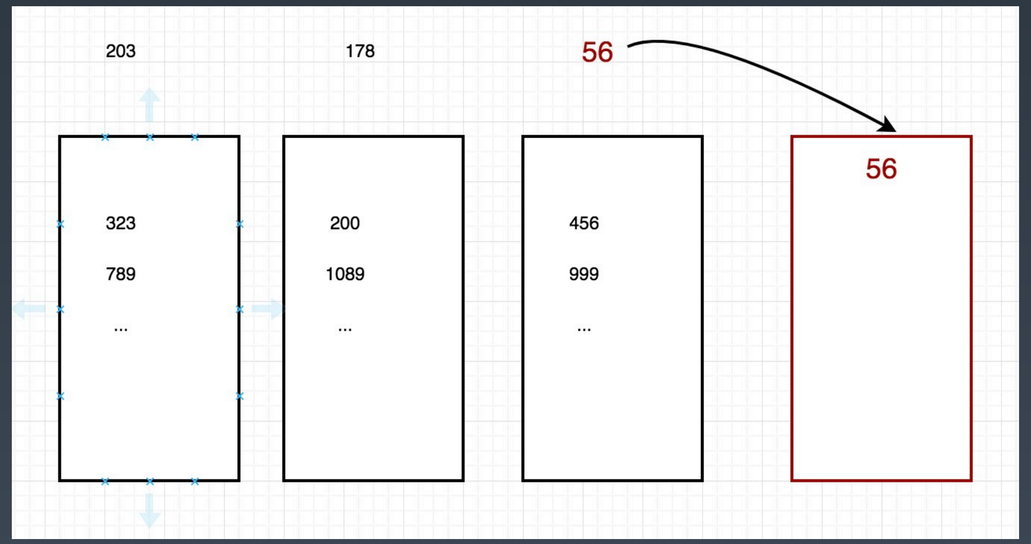
如何按照字符串大小，给这个 10GB 的大文件排序?

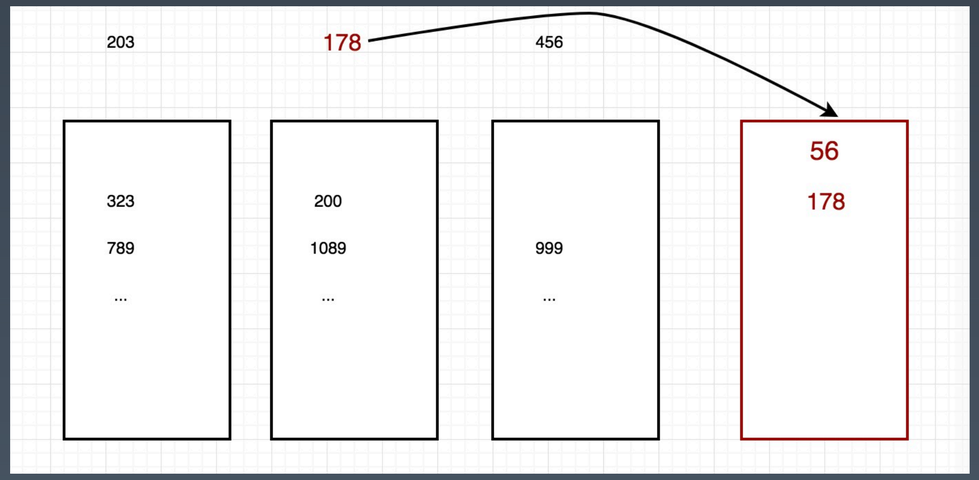
解决思路：

1. 借助桶排序
2. 借助归并排序

堆的应用一：优先级队列







堆的应用二：求TOP K

1. 针对静态数据（查询 TOP K 操作）

2. 针对动态数据（只包含添加数据操作和查询 TOP K 操作）

堆的应用二：求 TOP K

2. 针对动态数据

限制：

只允许添加数据，不允许删除、修改数据操作。

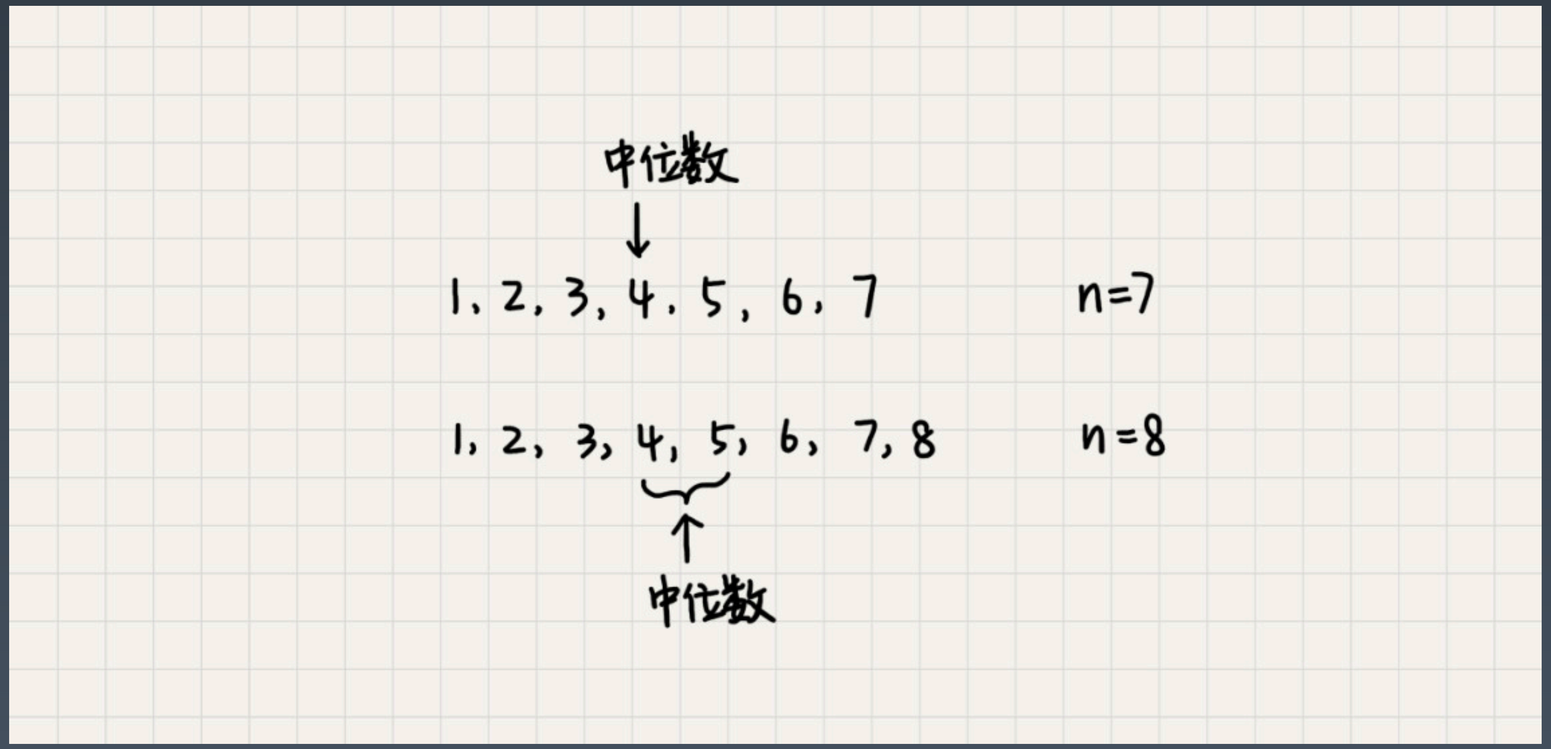
处理思路：

维护大小为 K 的小顶堆，当有数据被添加到集合中时，拿它与堆顶的元素对比：

- 如果比堆顶元素大，把堆顶元素删除，将这个元素插入到堆中；
- 如果比堆顶元素小，则不做处理。

堆的应用三：求中位数

什么是中位数?



堆的应用三：求中位数

分两种情况：

1. 静态数据求中位数

2. 动态数据求中位数

堆的应用三：静态数据求中位数

解决思路：

对于一组静态数据，中位数是固定的，我们可以先排序，第 n/2 个数据就是中位数。每次询问中位数的时候，我们直接返回这个固定的值就好了。所以，尽管排序的代价比较大，但是边际成本会很小。

堆的应用三：动态数据求中位数

效率比较低的解决方案：每次先排序，再取中间元素

效率高的解决思路：维护两个堆，一个大顶堆，一个小顶堆

限制：只支持添加数据，不支持删除、修改数据

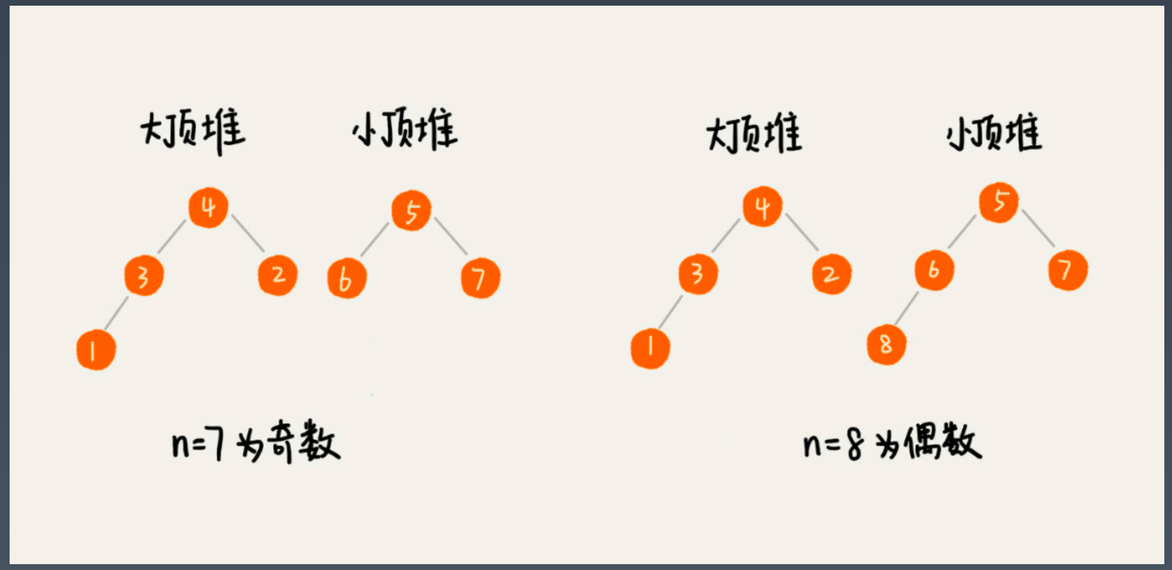
堆的应用三：动态数据求中位数

维护两个堆，一个大顶堆，一个小顶堆

每个堆中元素个数接近 n/2。
（如果 n 是偶数，两个堆中的数据个数都是 n/2；如果 n 是奇数，大顶堆有 n/2+1 个数据，小顶堆有 n/2 个数据。）

大顶堆中的元素都小于等于小顶堆中的元素

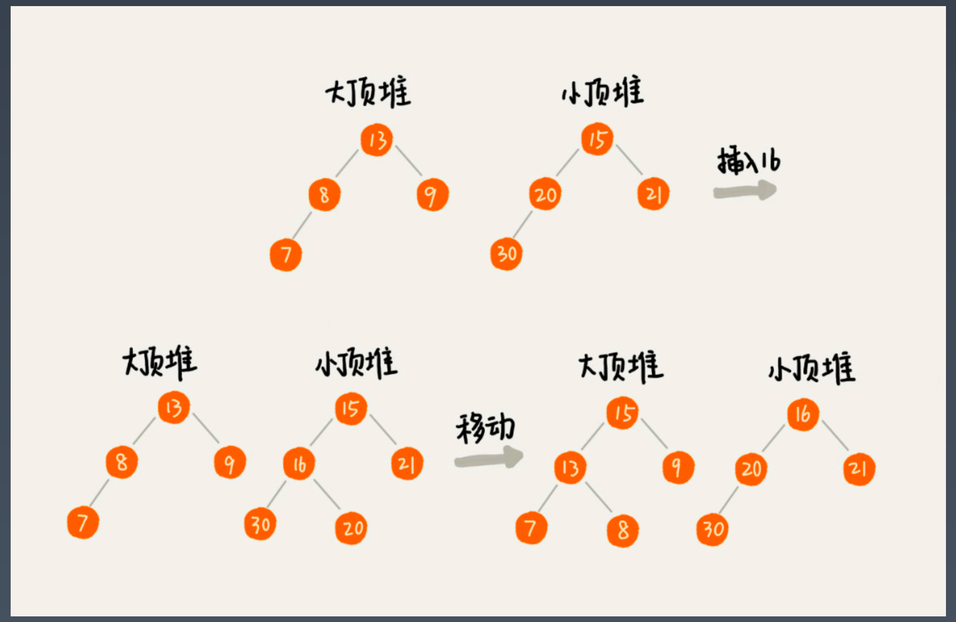
大顶堆中堆顶的元素就是中位数



堆的应用三：动态数据求中位数

数据是动态变化的，当新添加一个数据的时候，我们如何调整两个堆，让大顶堆中的堆顶元素继续是中位数呢？

如果新加入的数据小于等于大顶堆的堆顶元素，我们就将这个新数据插入到大顶堆。



THANKS!