排序算法

主讲人 ： ACM实验室 马金昊

A sorting algorithm is an algorithm that puts elements of a list in a certain order. The most-used orders are numerical order and lexicographical order. Efficient sorting is important for optimizing the use of other algorithms (such as search and merge algorithms) which require input data to be in sorted lists.

Sorting Algorithm

排序算法概述

在计算机科学与数学中，一个排序算法是一种能将一串数据依照特定排序方式进行排列的一种算法。最常用到的排序方式是数值顺序以及字典顺序。有效的排序算法在一些算法（例如搜索算法与合并算法）中是重要的，如此这些算法才能得到正确解答。排序算法也用在处理文字数据以及产生人类可读的输出结果。

04

ACM中的应用

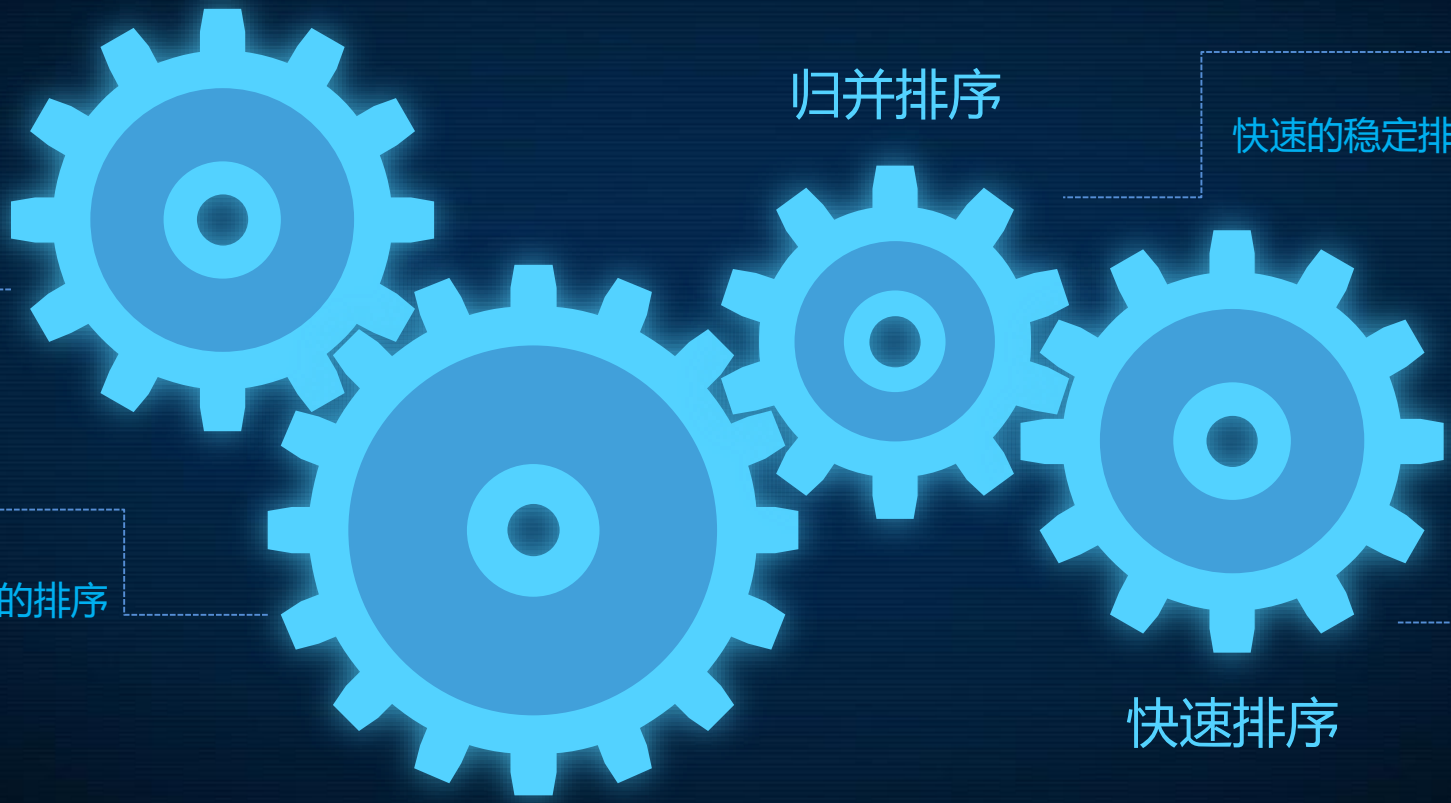
目录

Contents

04

ACM中的应用

常用排序算法



冒泡排序法

简单的排序方法

比较次数较少的排序

插入排序

归并排序

快速的稳定排序

快速排序

实际应用中最快的
不稳定排序

冒泡排序

优缺点

可以原地排序，实现简单。
复杂度较高, 效率很差

复杂度

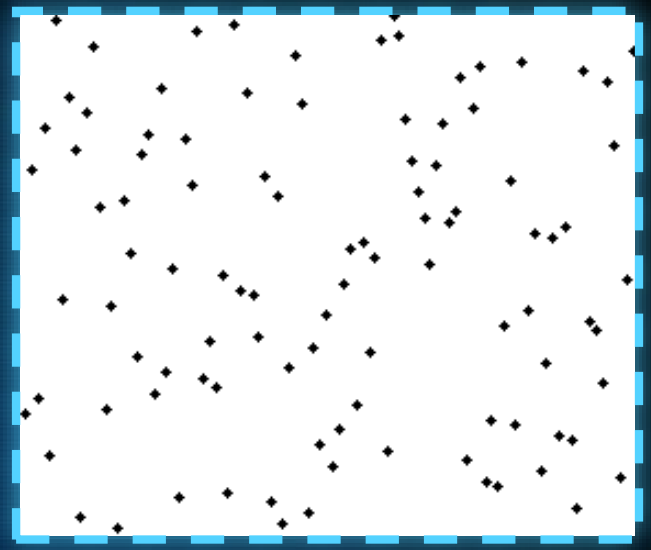
时间复杂度：O（n^2）
空间复杂度：O（n），需要辅助空间O（1）

做法

每次扫描整个序列，把序列中最大的元素移动到队尾，再对它之前的序列进行如上扫描

伪代码

```
i∈[0,N-1)          //循环N-1遍
  j∈[0,N-1-i)      //每遍循环要处理的无序部分
    swap(j,j+1)    //两两排序（升序/降序）
```



插入排序

优缺点

简单，直观，元素比较次数较少
复杂度较高, 效率很差

复杂度

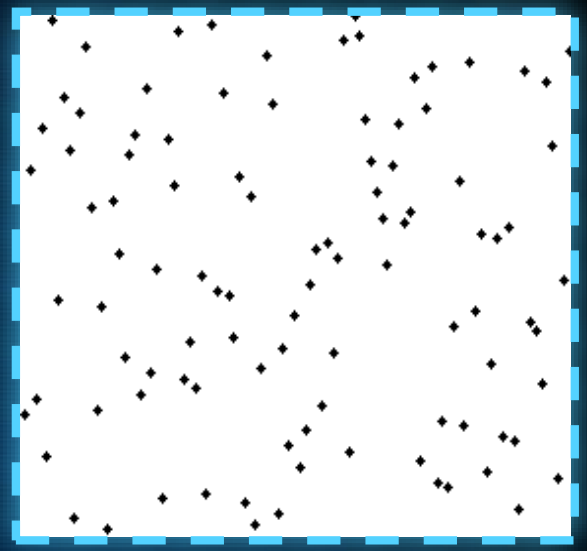
时间复杂度：O（n^2）
空间复杂度：O（n），需要辅助空间O（1）

做法

把序列分为两个部分，有序区和无序区，不断将无序区的元素插入到有序区之中。

代码

```
for (i = 1; i < len; i++)
{
        temp = arr[i]; //将较大的向后移动
        for (j = i - 1; j >= 0 && arr[j] > temp; j--)
                arr[j + 1] = arr[j];
        arr[j] = temp; //插入到对应位置
}
```



选择排序

优缺点

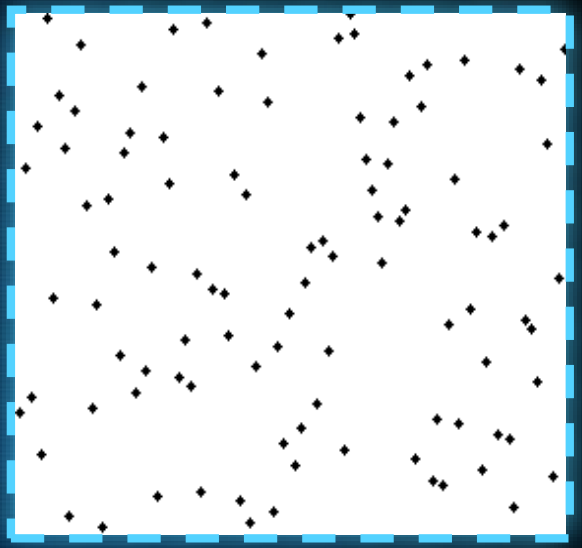
实现简单，好理解
复杂度较高, 效率很差

复杂度

时间复杂度：O（n^2）
空间复杂度：O（n），需要辅助空间O（1）

做法

把序列分为两个部分，有序区和无序区，选择无序区中最小的元素插入到有序区的末尾。



归并排序

优缺点

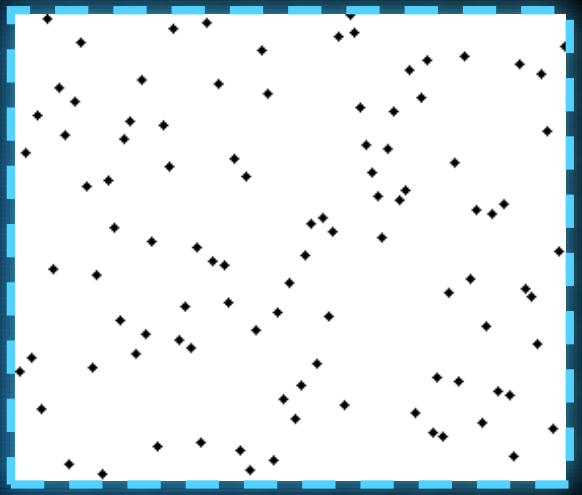
速度较快的稳定排序
复杂度常数较大

复杂度

时间复杂度：O（nlogn）
空间复杂度：O（n）

做法

利用分治将无序序列划分成多个有序序列，再将小的有序序列合并得到结果。



对于两个分别有序的序列，我们只需比较序列头最小的元素，每次把两个序列中最小的元素拿出来，合成一个序列，就能得到一个大的有序序列

合并操作

对于每个无序序列来说，我们每次把他分成2份，会得到两个小序列，再对每个小序列分成两份……，如此以往会得到多个有序的序列，因为最小的序列只有一个元素。

递归操作

快速排序

优缺点

平均速度最快的排序算法
不稳定

复杂度

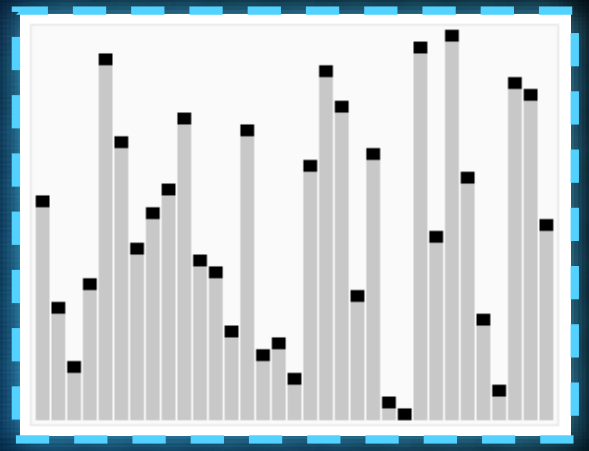
时间复杂度：O（nlogn）
空间复杂度：O（n）

做法

1.分而治之，各个击破
2.挖坑填数，快速调整

伪代码

```
function quicksort(q)
    var list less, pivotList, greater
    if length(q) ≤ 1  return q
    else {
        select a pivot value pivot from q
        for each x in q except the pivot element
            if x < pivot then add x to less
            if x ≥ pivot then add x to greater
        add pivot to pivotList
        return concatenate(quicksort(less), pivotList,
quicksort(greater))
    }
```



快速排序

01 选择一个数作为基准

在每一轮排序中，至少会把“基准数”放在正确的位置，把这个数单独拿出来，原位置留下一个“坑”

02 从后向前找到比“基准”小的数

把这个比基准小的放入“坑”中，这个数原来的位置也留下一个坑

03 从前向后找到比“基准”大的数

把这个比基准大的放入“坑”中，这个数原来的位置也留下一个坑

04 最后把基准数放入“新坑”中

结果就是基准左边的数都比基准小，右边的数都比基准大！基准数放入了正确的位置

05

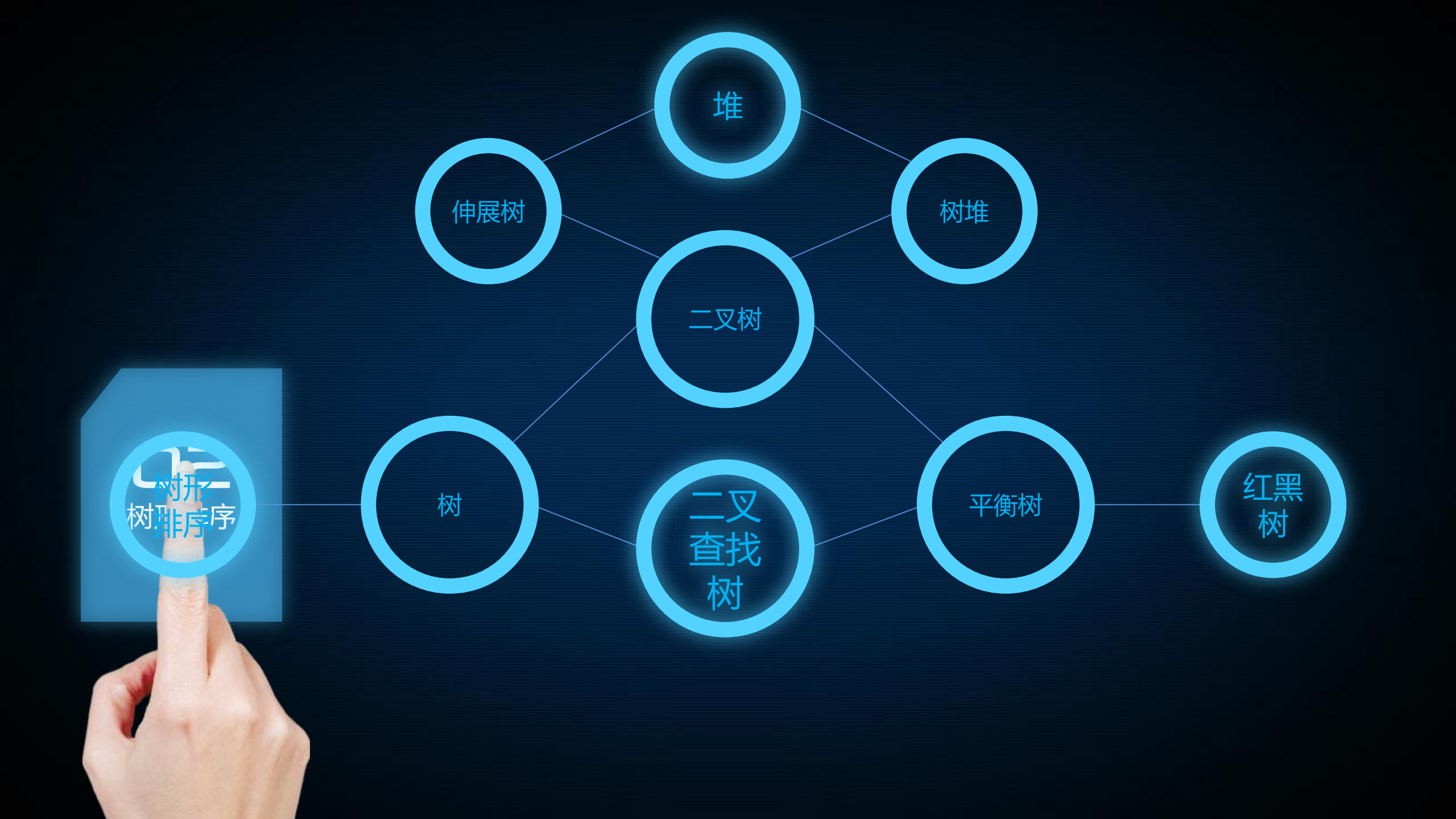
我们再递归的对基准数左右两个子序列重复1~4操作，最终将得到目标序列

目录

Contents

04

ACM中的应用


堆
伸展树
树堆
二叉树
树
树形排序
二叉查找树
平衡树
红黑树

二叉查找树

对于每个节点来说，

左儿子 < 根节点

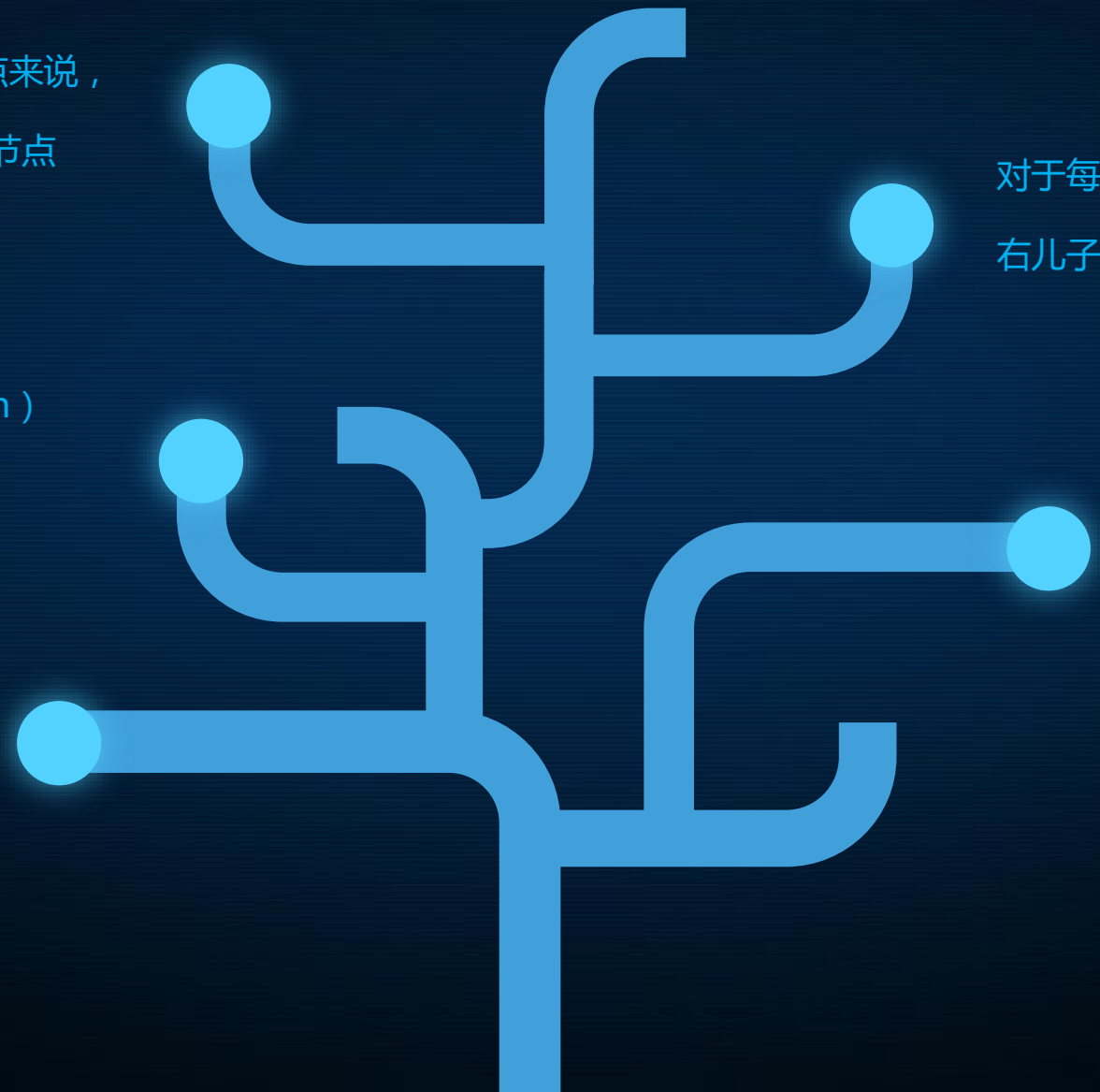
对于每个节点来说，

右儿子 > 根节点

每次查找的效率为O（lgn）

容易产生不平衡，导致退化成链状结构

不允许有值相同的节点存在



堆结构

对于每个节点来说，

左儿子 < 根节点

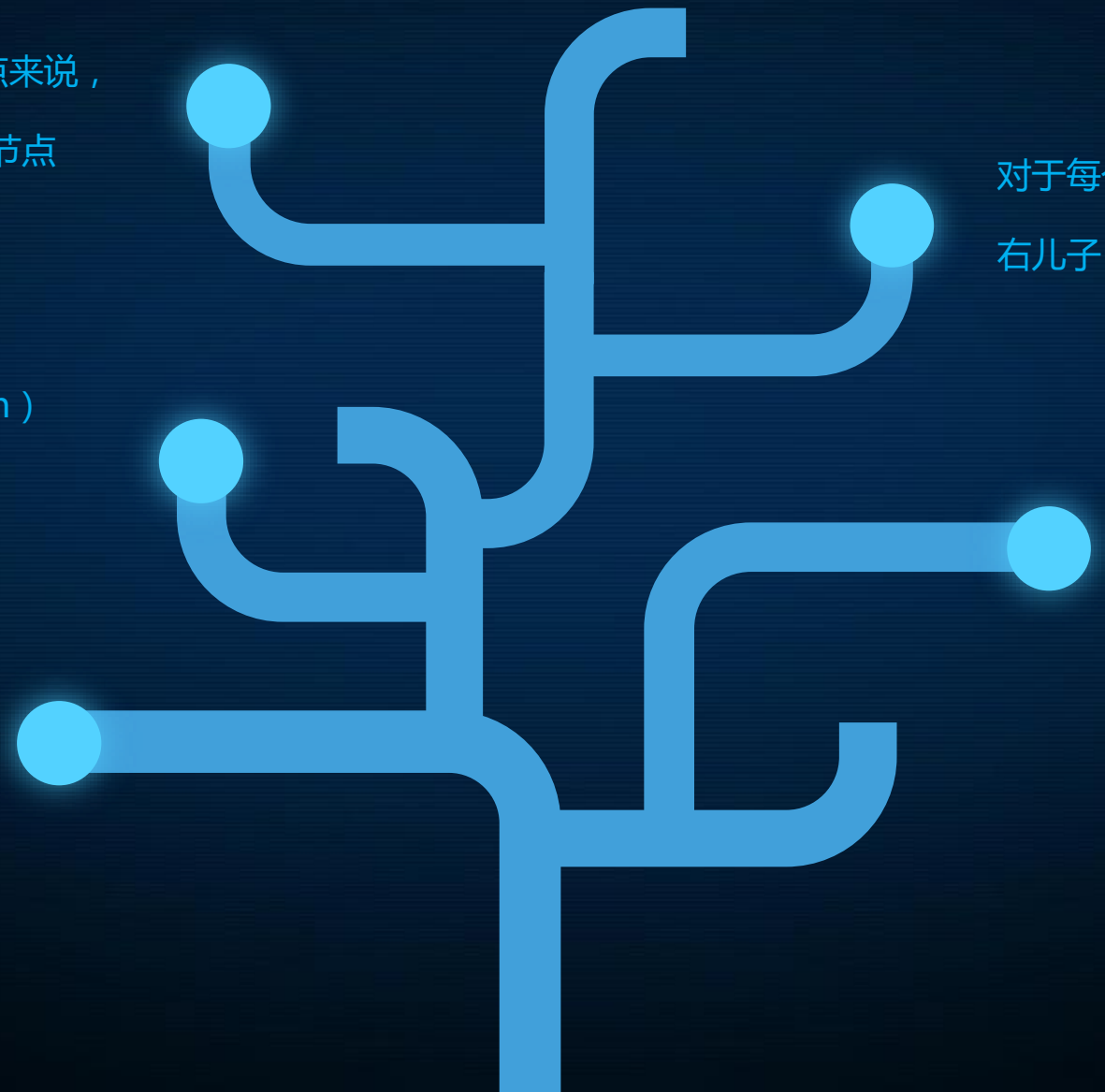
对于每个节点来说，

右儿子 < 根节点

每次查找的效率为O（lgn）

左儿子和右儿子的大小无需讨论，每个节点的左右子树都是堆

允许有值相同的节点存在



堆排序的操作

Fixup（插入）

逐次替换儿子中最大
且比父节点大的数
向上移动

```
void Fixup(int p)
{
    for(int f = fa(p) ; (f >= 0 && p) && a[f] < a[p] ; f=fa(p=f) )
        swap(a[p],a[f]);
}
```

Fixdown（构造）

将父节点与左右儿子比较
把左右儿子中最大且比父节点大的数
将较小的父节点下沉到叶子上

```
void Fixdown(int p)
 {
     for(int ch =last(p);ch < a.size() && a[ch] < a[p] ; ch=last(p=ch))
         swap(a[p],a[ch]);
 }
```

Delete（删除）

交换根节点与最后一个叶子结点
删除当前的最后一个节点
从新根节点开始重新调整整个堆

```
void Delete()
 {
     swap(a[0],a[a.size()-1]);
     bin.push_back(a[a.size()-1]);
     a.pop_back();
     Fixdown(0);
 }
```

目录

Contents

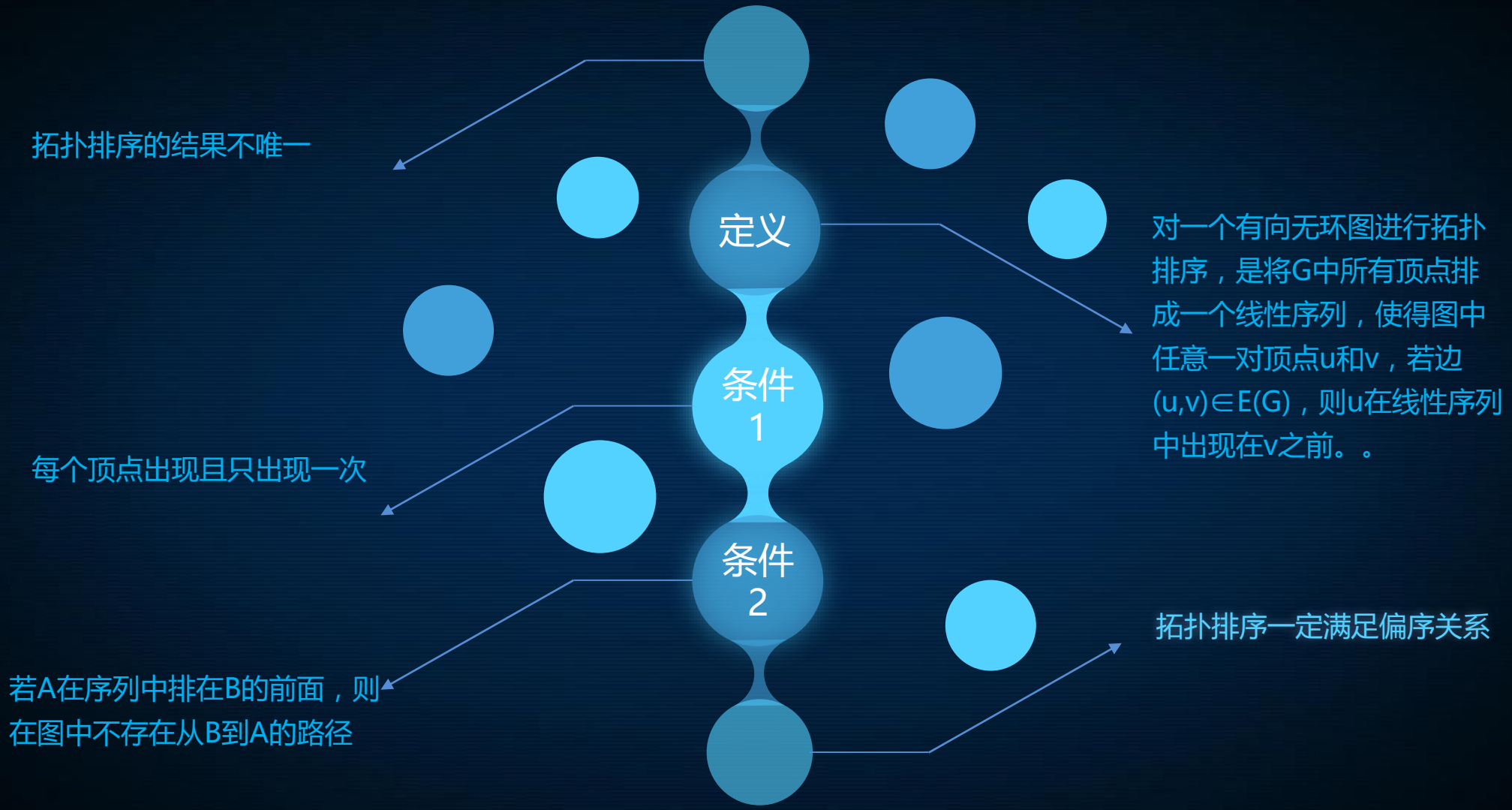
04

ACM中的应用

图状排序


SMS
@
du
淘

拓扑排序


拓扑排序的结果不唯一
定义
对一个有向无环图进行拓扑排序，是将G中所有顶点排成一个线性序列，使得图中任意一对顶点u和v，若边(u,v)∈E(G)，则u在线性序列中出现在v之前。。
条件1
每个顶点出现且只出现一次
条件2
若A在序列中排在B的前面，则在图中不存在从B到A的路径
拓扑排序一定满足偏序关系

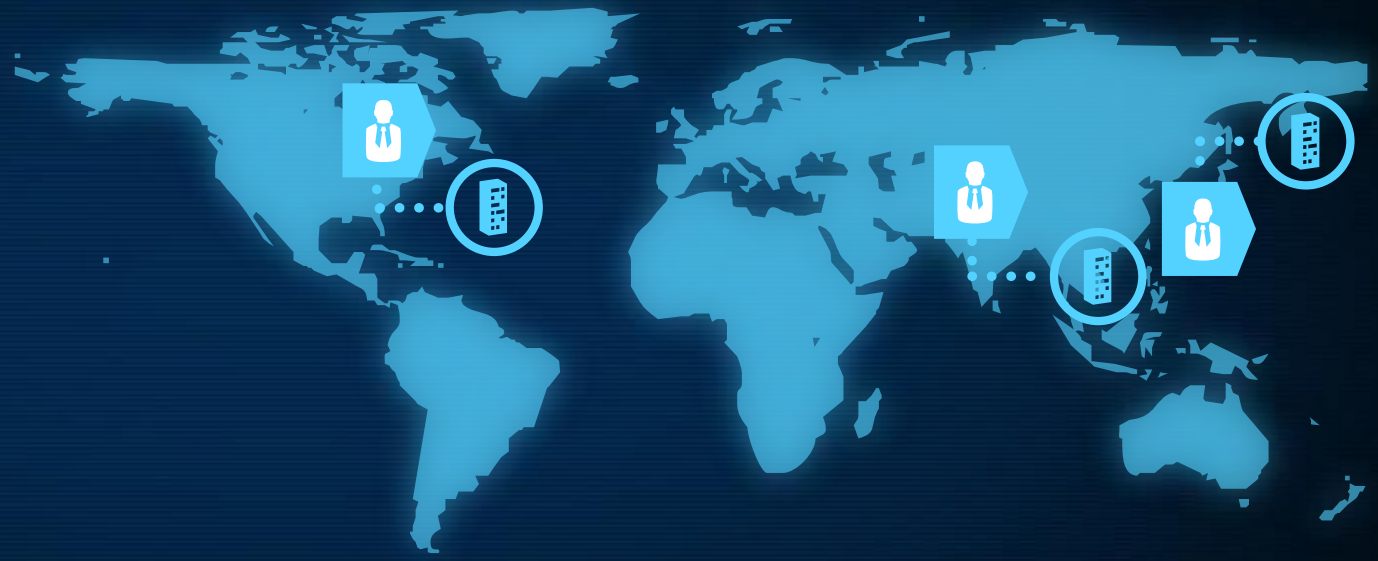
目录

Contents

04

ACM中应用

排序算法是ACM的基石



谢谢大家

祝你有个美好的一天

作者：马金昊