5. 分治法

张桄玮(gwzhang@cug.edu.cn)

郑州一中(Legacy)

2024-08-03

排序算法中的分治

归并排序

过程:

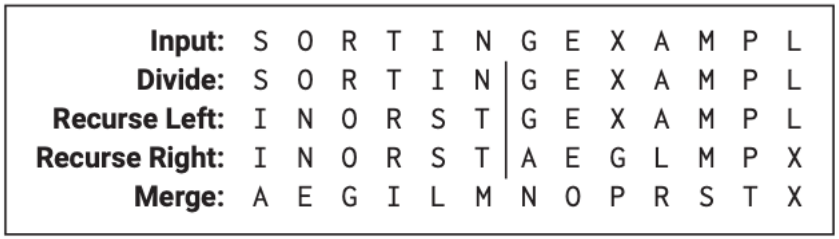
1. 分解: 将数组分成两半。
2. 递归排序: 对每一半分别进行归并排序。
3. 合并: 将两半有序的子数组合并成一个有序数组。

Base case: 只有一个数组, 本身就是有序的

Inductive case: 开两个指针, 每一次把小的那个放过来

时间复杂度: $O(n \log n)$.

- 因为解递归式$T(n) = T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + O(n)$
- 约等于$T(n) = 2T(n/2) + O(n)$, 画图可知



P1908 逆序对问题

问题: 给一列长达10^6的数, 问其中有多少对逆序对?

- 两重循环? 时间复杂度太高了!

考虑像归并排序那样分解问题

- 划分问题: 把问题划分为几乎相等的两半
- 递归求解: 求解左右两边的逆序对的个数
- 合并问题: 统计i在左边, j在右边的逆序对

关键: 如何合并?? – 分类讨论

- 对于右边的每个j, 统计左边比他大的元素个数$f(j)$, 所有元素之和就是答案.

归并排序的过程

- 合并操作从小到大进行

P1908 逆序对问题

- A[j]复制到 tmp 的时候, 左边还没有来得及复制到的元素就是所有必 A[j]大的元素.
- 直接加上$m-p$个元素即可.

```
void msort(int a[], int tmp[], int l, int r){
    if(l>=r) return ;
    int mid = l+r>>1;
    msort(a, tmp, l, mid);
    msort(a, tmp, mid+1, r);
    int i = l, j = mid+1, k=0;
    while(i<=mid && j<=r){
        if(a[i]<=a[j]){
            tmp[k++]=a[i++];
        }else{
            cnt += (mid-i+1); tmp[k++]=a[j++];
        }
    }
```

P1908 逆序对问题

```
    while(i<=mid) tmp[k++]=a[i++];
    while(j<=r) tmp[k++]=a[j++];
    for(i=l, j=0;i<=r;i++, j++) a[i]=tmp[j];
}
```

快速排序

相对于分治法速度更快, 并且不需要辅助空间

选择数组中的一个枢轴元素。 将数组划分为三个子数组（怎么划分?）

- 小于枢轴的元素;
- 枢轴元素本身;
- 大于枢轴的元素。

递归地快速排序第一个和最后一个子数组。

如何划分?

1. 指针 i 表示开始位置， 指针 j 表示结束位置
2. 继续增加 i， 直到 $a[i] > x$; 继续减少 j， 直到 $a[j] < x$
3. 交换 $A[i]$ 和 $A[j]$
4. 返回到 (2)， 直到 $i > j$

快速排序

```
void qst(int a[], int l, int r){
    if(l>=r) return;
    //(1)
    int x = a[l]; //Alterate r, (l+r)/2
    int i = l-1, j=r+1; //Move first, then compare and swap
    while(i<j){
        do i++; while(a[i]<x);
        do j--; while(a[j]>x);
        if(i<j) swap(a[i],a[j]);
    }
    qst(a, l, j); // If x:=r, then this line should be qst(a,l,i-1)
    qst(a,j+1,r); // and this line should be qst(a,i,r)
}
```

快速选择算法

问题：输入n个整数和一个正整数k, 输出这个整数从小到大排序后的第k个. $n \leq 10^7$.

朴素做法: 先排序然后再输出 $\rightarrow 10^7$对于快速排序的时间复杂度非常高.

考虑快速排序: 划分的阶段结束后,

- $A[p..r]$按照基准元素分为了$A[p..q], A[q], A[q+1..r]$.
- 则可以根据左边的元素个数 $q-p+1$ 和 k 的大小关系只在左边或者右边递归求解。

期望的意义下时间复杂度为$O(n)$.

二分查找

例子: 猜数字游戏, 给定范围, 每次猜完之后又反馈(大了, 小了, 猜中了). 问最多情况最少多少次猜中?

第一个二分程序:

```
int bsearch(int *A, int x, int y, int v){
  int m;
  while(x<y){
    m=x+(y-x)/2;
    if(A[m]==v) return m;
    else if(A[m]>v) y=m;
    else x = m+1;
  }
  return -1;
}
```

问题: 如果数组中有多个相同的元素(3 3 3 3 3), 返回的是哪一个目标呢?

二分查找

- 中间的

希望: 能不能找到第一个出现的(lower_bound), 最后一个出现的(upper_bound)?

lower_bound:

- 当 v 存在时返回它出现的第一个位置。
- 如果不存在, 返回这样一个下标 i :在此处插入 v(原来的元素 $A[i], A[i+1], \ldots$ 全部往后移动一个位置) 后序列仍然有序。

```
int lower_bound(int *A, int x, int y, int v){
  while(x<y){
    m=x+(y-x)/2;
    if(A[m]>=v) y=m;
    else x = m+1;
  }
  return x; // <- 我们这里返回 x, 而不是-1
}
```

二分查找

这段程序可以返回那些值?

- 查找区间是左闭右开的$[x..y)$, 返回值可能是$[x..y]$.

关于正确性的分类讨论:

- $A[m] = v$: 至少已经找到一个, 而左边可能还有, 因此区间变为$[x, m]$;
- $A[m] > v$: 所求位置不可能在后面, 但有可能是 m, 因此区间变为$[x, m]$;
- $A[m] < v$: m和前面都不可行, 因此区间变为$[m + 1, y]$

警惕: 有没有可能死循环?

- 循环的范围没有变小: $(x, m) = (x, y)$或者$(m + 1, y) = (x, y)$?
- 如何进行程序分析:
  - 根据语法/语义假设某种前提条件
  - 然后模拟执行程序, 看一看有没有发现不对劲的东西

二分查找

upper_bound:

- 当 v 存在时返回它出现的最后一个位置的后面一个位置;
- 如果不存在, 返回这样一个下标 i : 在此处插入 v (原来的元素 $A[i], A[i+1], \ldots$ 全部往后移动一个位置) 后序列仍然有序。

```
int lower_bound(int *A, int x, int y, int v){
  while(x<y){
    m=x+(y-x)/2;
    if(A[m]<=v) x = m+1; // 更改这两行
    else y=m;                    // 两边互换
  }
  return x;
}
```

所以 lower_bound, upper_bound 的返回值如果是L, R, 那么v出现的子序列是$[L..R)$

注记: 左闭右开的区间

为什么总是使用左闭右开的区间?

- 好计算长度. $[L..R)$的长度为$R-L$. 而闭区间$[L..R]$的长度为$R-L+1$.
- 好进行区间的合并. $[L..R) \cup [R..T) = [L..T)$, 而$[L..R] \setminus \{R\} \cup [R..T] = [L..T]$.

为什么不使用右闭左开的区间?

- 人类阅读习惯为从左往右阅读!

注记: STL 对于排序, 二分查找的支持

- 排序: `sort` 函数
- 二分查找: `upper_bound`, `lower_bound` 类似.

练习: P2249 查找

二分查找容易写错:

- Java 标准库中一个类似的查找函数使用了类似的二分方法.
- 这个 Bug 在 Java 的数组标准库里面待了 9 年.(和整数溢出有关)

核心:

- `int mid =(low + high) / 2;` 在加两个大整数的时候会溢出;
- 更正: `int mid = low + ((high - low) / 2);`

二分答案

一般的设问: 最大值最小

- 条件: 决策单调性
- 策略: 猜一个, 问问行不行; 行了缩紧条件, 不行了放松条件.

P1676 进击的奶牛

问题：农夫约翰建造了一座有 n 间牛舍的小屋，牛舍排在一条直线上，第 i 间牛舍在 x_i 的位置，但是约翰的 m 头牛对小屋很不满意，因此经常互相攻击。约翰为了防止牛之间互相伤害，因此决定把每头牛都放在离其它牛尽可能远的牛舍。也就是要最大化最近的两头牛之间的距离。

牛们并不喜欢这种布局，而且几头牛放在一个隔间里，它们就要发生争斗。为了不让牛互相伤害。约翰决定自己给牛分配隔间，使任意两头牛之间的最小距离尽可能的大，那么，这个最大的最小距离是多少呢？

- 首先猜一个答案，然后去施展我们应该有的构造，最后来看一看这个是不是太小了.

P1676 进击的奶牛

策略:

- 假设牛棚都是空的, check 时如果当前牛棚与上一个住上牛的牛棚之间的 距离 dis>=mid, 我们就可以让这个牛棚里住上牛, 反之向更远的距离寻找牛棚.
  - 如果最后能安排的牛总数小于总的牛数, 那么就可以扩大需求.
  - 反之, 就要缩小;

贪心策略的正确性(留给大家自己验证)

更多的练习:

- P2878 跳石头
- P3853 路标设置

P1314 聪明的质检员

问题：矿产共有 n 个矿石,从 1 到 n 逐一编号, 每个矿石都有自己的重量 w_i 以及价值 v_i 。流程是:

1. 给定 m 个区间 $[l_i, r_i]$;
2. 选出一个参数 W ;
3. 对于一个区间 $[l_i, r_i]$, 计算矿石在这个区间上的检验值 y_i :$y_i = \sum_{j=l_i}^{r_i} [w_j \geq W] \times \sum_{j=l_i}^{r_i} [w_j \geq W] v_j$

其中 j 为矿石编号。 这批矿产的检验结果 y 为各个区间的检验值之和。即: $\sum_{i=1}^{m} y_i$ 若这批矿产的检验结果与所给标准值 s 相差太多, 就需要再去检验另一批矿产。他想通过调整参数 W 的值, 让检验结果尽可能的靠近标准值 s, 即使得 $|s - y|$ 最小。请你帮忙求出这个最小值。

在这之前先介绍一下 Iverson 的括号:

定义 01：假设P是一个命题，定义

$$[P] = \begin{cases} 1, & \text{命题} P \text{ 为真} \\ 0, & \text{命题} P \text{ 为假} \end{cases}$$

- 便于优化很复杂的求和操作，并且可以把求和符号的下标转换为命题之间的操作.

  例子：求和式 $\sum_{0 \le k \le n} k$ 可被改为 $\sum_k k[0 \le k \le n]$.

  如果 k 未指定限定条件，我们认为 $k \in \mathbb{Z}$. 也就是说上述式子

P1314 聪明的质检员

$$\sum_k k[0 \le k \le n]$$

$$= (0 \cdot 1) + (1 \cdot 1) + \cdots + (n \cdot 1)$$

例子：如果K与K'是两个整数集合，那么$\forall k$,

$$[k \in K] + [k \in K'] = [k \in (K \cap K')] + [k \in (K \cup K')]$$

由此可以导出对应的和式

$$\sum_{k \in k} a_k + \sum_{k \in k'} a_k = \sum_{k \in k \cap k'} a_k + \sum_{k \in k \cup k'} a_k$$

例子：$[k \in K] + [k \in K'] = [k \in (K \cap K')] + [k \in (K \cup K')]$对 k, k' 为可数集，$\forall k$.

回到正题：

P1314 聪明的质检员

- W越大, y越小. 于是考虑猜测W.
  - 当 $y < s$ 时, 我们就要减小 W, 使得$|y - s|$变大;
  - 当 $y > s$ 时, 我们就要增大 W, 使得$|y - s|$变小.
  - 当 $y = s$ 时, 找到啦! $|y - s| = 0$.

然后, 要求区间的和$\rightarrow$区间前缀和!

递归与分治

自相似

- 第一节已经介绍过相当多的例子了. 下面再看几个例子.

循环赛日程表

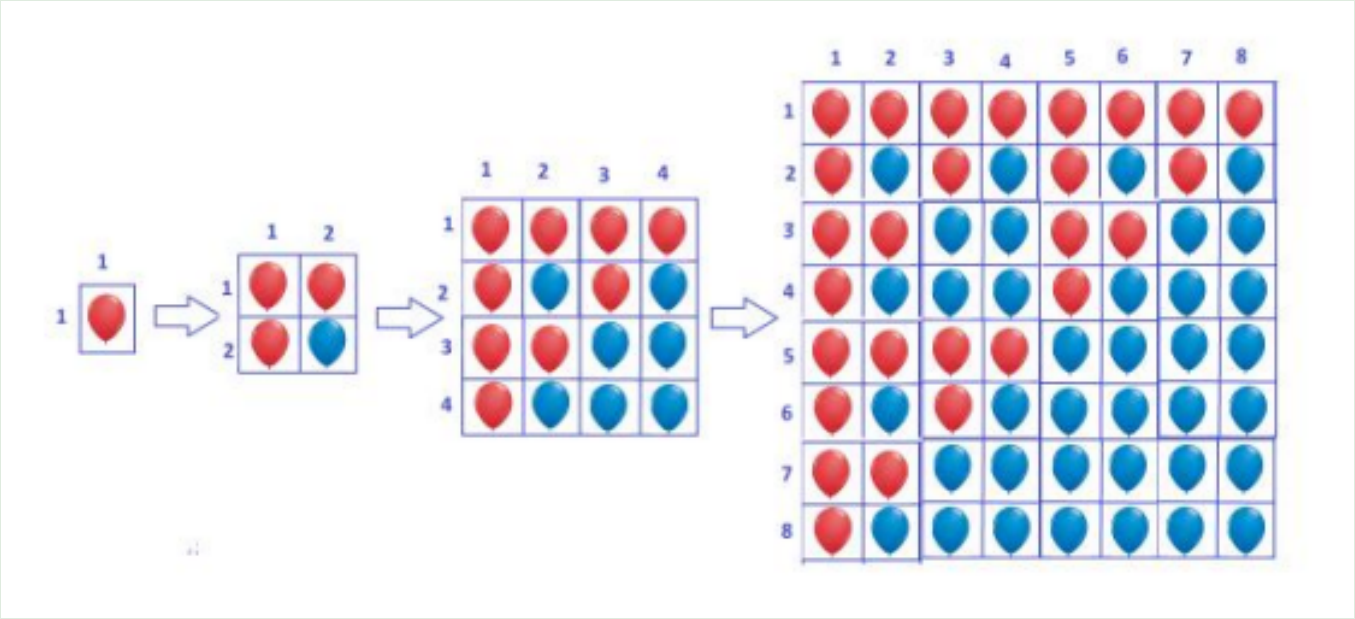
问题: $n = 2^k$ 个运动员进行网球循环赛, 需要设计比赛日程表。每个选手必须与其他 $n-1$ 个选手各赛一次; 每个选手一天只能赛一次; 循环赛一共进行 $n-1$ 天。按此要求设计一张比赛日程表, 该表有 n 行和 $n-1$ 列, 第 i 行 j 列为第 i 个选手第 j 天遇到的选手。

- 考虑$k = 1 \to k = 2$, 如何 “复制粘贴”?
- 根据中心的四块.
  - 左上角: 原来的解答; 左下角=左上角每个数+x; 右边的是把左边的倒过来.

自相似

Uva 12627 奇怪的气球膨胀

问题：一开始有一个红气球。每小时后，一个红气球会变成 3 个红气球和一个蓝气球，而一个蓝气球会变成 4 个蓝气球，经过 k 小时后，第 A~B 行一共有多少个红气球？例如，$k=3, A=3, B=7$，答案为 14 。



自相似

观察: 右下角全是蓝色气球, 其余的三个部分中, 是由上一次的左上角拼起来的.

但是询问的并没有覆盖整个子问题...

- 像循环日程表一样打印出来? ($\times$, 2^{30} 吃不消了)

另一种方法:

- $s[k][i] := k$小时$[1..i]$的红气球数
- $r[k] := k$小时红气球的总数

归纳的情况:

- 在上半部分, $i \le 2^{k-1}$, $s[k][i] = 2s[k-1][i]$(直接复制)
- 在下半部分, $s[k][i] = s[k-1]\left[i-2^{k-1}\right] + 2r[k-1]$

最后的答案: $s[k][B] - s[k][A-1]$(类似前缀和)