贪心与分治

PKU EECS zld3794955

2017 年 7 月 18 日

前言

大家昨天过得愉快吧~

今天的内容我们会分两个课件来讲。

跟昨天数据结构不一样，今天的内容所需要的相关知识比较少，故我们以讲一些简单的例题为主。

我们先讲贪心与分治的相关内容。

贪心

OI界中，简单意义上的贪心就是追求局部最好。

比如有人要送你一张100块的钞票或者1块的钞票，你当然选择100的那张，除非那张是假的。

但是如果你拿完100块之后，接下来那个人告诉你说，如果你要1块的钞票的话我送你10000块当赠品，那么你就会后悔，这也就是贪心的局限性。

贪心不仅可以作为一种解决问题的方式，也可以用作思考分析问题的方式，这个在OI中运用也很广，但贪心法有一个重要的条件，就是你在不停以某个方式追求某个局部最好的过程中能达到全局最好，忽视这个条件会导致很多问题，这一点以后大家会慢慢了解。

贪心是什么样的

各种等价描述：

1、我们每步以XX贪心策略选择一个解，那么总解一定是最优的。

2、将XX调整为YY，结果不会变差（选择XX，还不如选择YY）

3、如果XX比YY的某个属性怎样，那么结果不会变得更差/更好。

本人水平所限，只想起来了这几种比较有用的表述。

如何贪心

1、明确每个阶段的每个决策。

2、设计一个方案，计算出每个阶段每个状态选择每个决策的收益/代价，这个收益/代价可以是题意，也可以自行设计。

3、从初态开始，以某种方式每步选一个收益/代价取最值的决策。

4、当然最好证明这个贪心是正确的，不能证明的话建议去验证一下其在小数据和特殊数据上是否正确。

这是在最优化问题中运用贪心的模式，当然贪心法在构造题中也有类似的模式，只不过这时候你需要证明的是这个构造法一定能构造出解，或者换句话说，如果以此法构造不出解，那么就无解。

当然我们不能拘泥于这个模式，但是规范化一些总是会有好处的。

一个简单的例题

小T开办了一家机器工厂，在$N(N \leq 10000)$个星期内，原材料成本和劳动力价格不断起伏。

第i周生产一台机器需要花费$C_i(1 \leq C_i \leq 5000)$ 元。若没把机器卖出去，每保养一台机器，每周需要花费$S(1 \leq S \leq 100)$元，这个费用不会发生变化。

机器工厂接到订单，在第i周需要交付$Y_i(0 \leq Y_i \leq 10^4)$台机器给委托人，第$i$周刚生产的机器，或者之前的存货，都可以进行交付。

请你计算出这n周时间内完成订单的最小代价。

The solution

明确决策——决策：每台所需要的机器在哪一周生产。

决策代价——生产费+保养费，题意中就有。

贪心过程——对任意一周的机器，都选择代价最低的决策，确定应在哪周生产这些机器。

正确性——任意两台机器之间均无关，故生产所有机器的最小代价即为生产每台机器的最小代价之和。

所以我们只需要求出每周的机器在什么时候生产花费最小，并且将结果相加即可。

线段覆盖

在一根数轴上给你n条线段（包含两端点），请选取这些线段中的若干条覆盖这些线段的并集，当然你用的线段数量越少越好。

所有线段均是包含两端点的。

$1 \leq n \leq 200000$

The solution

我们考虑这n条线段的$2n$个端点，去重之后从小到大排序记为$x_1 < x_2 < \ldots < x_p$。

不妨假设我们要覆盖的线段的并集即为$[x_1, x_p]$（否则我们可以分离成若干段这样的小区间进行处理）

我们现在考虑，对任意i，$[x_i, x_{i+1}]$这条线段应该被原线段集合中哪条线段覆盖。

The solution

定义每步决策的“代价”为：决策做完后，覆盖剩下未覆盖的部分所需的最少线段数，我们不一定能直接求出这个，但是我们知道，未覆盖的部分在某一已有基础上减少时，这个代价不会增加。

从左到右每次考虑当前未被覆盖的线段$[x_i, x_{i+1}]$中最左侧的一个，选取能覆盖该线段，且右端点最靠右的一条线段显然最优。

对线段维护一个关于右端点的优先队列，开始时将线段按照左端点排序，覆盖完$[x_i, x_{i+1}]$时，将所有左端点在x_{i+1}及之前的线段加入堆，在选择下一条线段的时候，从堆中选择右端点最靠右的一个即可，时间复杂度$O(n \log n)$。

另一道简单题

已知一个n个数的正整数序列$a_1, a_2, \ldots, a_n$，你的任务就是找到一个区间，使得区间内最小值$\times$区间内元素个数的结果最大。

$1 \leq n \leq 500000$。

The solution

我们考虑任意一个解，比如区间$[l, r]$，记区间内元素最小值为x。

如果$a_{l-1} \geq x$（这里定义$a_0 = a_{n+1} = 0$），那么显然区间$[l-1, r]$比区间$[l, r]$结果更优。

所以，若$[l, r]$为最优解，那么显然$a_{l-1}, a_{r+1} < x$，这样我们就缩小了寻找最优解的范围。

那么我们如何来利用这个条件呢？

考虑右端点固定时，区间最小值随左端点的变化。

将这些变化点用位置p记下来，表示当前右端点i下，左端点从i向左移动到p时，区间最小值变成a_p。

我们在从小到大枚举右端点的过程中，可以用一个栈来实时维护变化点，在右端点从$i-1$移到i时，我们需要加入当前位置i，但在此之前，如果栈顶元素p满足$a_p \geq a_i$，那么我们就需要将p从栈中弹出并继续判断，直到$a_p < a_i$或栈为空。

知道这些信息，我们就可以对每个右端点，筛出满足左端点条件$a_{l-1} < x$的区间了。

而满足右端点条件的区间，则可以在栈弹出元素的时候进行判断，每次弹出时，利用弹出的位置和当前的位置，我们即可得到一个同时满足左右端点条件的区间并计算该区间的结果。

注意扫到最右端结束的时候还要判断一下。

由于弹出元素最多$O(n)$次，因此最多有$O(n)$个符合条件的区间，利用单调栈的信息可以做到$O(1)$求得这些区间内的最小值，因此总时间复杂度为$O(n)$。

一道构造题

有n个数$a_1, a_2, ..., a_n$，我们对任意$1 \leq i, j \leq n$均求出(a_i, a_j)，得到n^2个数。

现在我们知道这n^2个数，但是并不知道其对应哪对(i, j)的结果，请你还原出原来的n个数，数据保证这样的n个数存在。

$1 \leq n \leq 500$。

题目来源：http://codeforces.com/contest/582/problem/A

The solution

注意到$gcd(x,y) \leq min(x,y)$以及$a_1, a_2, ..., a_n$必定会在这n^2个数中出现一次这两条性质，则这n^2个数中最大的那个一定是原来的n个数之一，不妨取作a_1。

同理将a_1删去后，我们再考虑剩下的数中最大的那个，那么其一定也是原来的n个数之一，不妨取作a_2。

接下来，我们要删除两个$gcd(a_1, a_2)$以及一个a_2以排除已知结果，再考虑剩下的数中最大的那个，取作a_3。

以此类推，我们就可以还原出这n个数，如果数据不保证有解，还可以判断无解的情况（挺简单，大家自己想）。

这个用昨天讲的支持删除元素的堆可以做，也可以用C++STL模板中的multiset，总时间复杂度$O(n^2 \log n)$。

另一道题

团长做了一个梦，梦见好多Kyon，从1到n编号。

在梦里，Kyon们在团长的命令下被逼玩起类似叠罗汉的游戏。每个Kyon都要用自己的力量托起在他上面的Kyon，对于每个Kyon，有一个正整数的重量值w和力量值s。

玩游戏有危险。定义一个Kyon的危险值，等于在他上面所有Kyon的重量之和（不包括他自己的重量）减去他自己的力量值。

团长的潜意识里不希望伤害Kyon，她想安排一个Kyon的叠放顺序，使得所有Kyon的危险值中，最大的危险值尽量小。

数据范围：$1 \leq n \leq 10^5$，$1 \leq s_i, w_i \leq 10^9$。

题目来源：我已经无从得知了，知道的可以告诉我下么……谢谢

The solution

假设当前Kyon序列从上到下为$i_1, i_2, ..., i_n$，接下来我们考虑相邻的两个元素i_k与i_{k+1}。

设i_k之前（不含i_k）的Kyon的重量和为W，则i_k的危险值即为$A = W - s_{i_k}$，i_{k+1}的危险值即为$B = W + w_{i_k} - s_{i_{k+1}}$。

如果我们交换i_k与i_{k+1}的次序，那么i_k的危险值即为$A' = W + w_{i_{k+1}} - s_k$，$i_{k+1}$的危险值即为$B' = W - s_{i_{k+1}}$，除此之外的Kyon的危险值显然都是不变的。

若$w_{i_k} + s_{i_k} > w_{i_{k+1}} + s_{i_{k+1}}$，那么$B - A' > 0$，又显然$B > B'$，则$B > \max(A', B')$，即有$\max(A, B) > \max(A', B')$，故交换之后，这两个Kyon的危险程度最大值减小，总最大危险值不增加。

因此，Kyon序列$i_1, i_2, ..., i_n$每个元素的$w + s$从小到大排好序的时候，最大的危险值一定最小。

贪心的一个经典模型——NOIP2004合并果子

有n堆果子，每堆有a_i个果子，你每次可以任意选择第p, q堆的果子，将它们合并为一堆果子，这一堆果子的果子数量为$a_p + a_q$，这一次合并的代价也为$a_p + a_q$。

你的任务就是将这n堆果子合并成一堆，并且使总代价最小。

$1 \leq n \leq 10^5$。

The solution

每次选择果子数量最少的两堆合并即可。

正确性证明：比较复杂，我也不是非常地清楚，但大概可以体会一下。

每次取最值的操作可以使用堆来实现，时间复杂度$O(n \log n)$。

要是不愿意写堆，那么可以将果子数量从小到大塞入一个队列$Q1$中，每次合并的时候，都将合并的结果放入另一个队列$Q2$的末尾，那么显然每次放入$Q2$的元素都是单调不减的，因此，当前局面的最小值一定来源于$Q1$和$Q2$的队头。

这样做排序$O(n \log n)$，后续的合并操作是$O(n)$的。

该模型的改造

如果我们每次可以选择至多k堆果子合并，那么怎么做？

若$k-1|n-1$，那么每次取最小的k堆果子合并即可得到最优解。

否则，我们在果子堆中补0（易知对答案无影响），使得$k-1|n-1$，再进行操作即可。

我们接下来提一下该贪心模型的原型——Huffman树。

Huffman树

已知有n种元素，每种元素出现的频率为t_i，现在要将每种元素对应一个01串，要求没有一个元素的01串是另一个元素的01串的前缀。

我们的目标是最小化$\sum t_i \times$对应01串长度。

$1 \leq n \leq 10^5$

The solution

显然该问题等价于构造一棵由01串构成，恰有n个叶结点的Trie（可以画图体会一下），使得$\sum t_i \times$对应叶节点的深度最小。

将每种元素看作一个权重为t_i的节点，我们从叶子到根构造这棵Trie。

每次在节点集中取两个权重最小的节点，将它们作为一个新的节点p_{new}的两个子节点，p_{new}的权值为取出的两个节点的权重之和，最后在节点集中删除原来的那两个节点，再加上p_{new}。

如此进行操作，直到最后剩下一个节点为止，我们就得到了所需要的Trie以及给每种元素分配01串的方案。

可以看到，这里的贪心过程和代价计算与合并果子问题完全相同。

最后一道经典题

现在有一个n个数的正整数序列$a_1, a_2, ...a_n$，你要将其划分成k个连续段，使得每个连续段的和的最大值最小。

数据范围：$1 \leq n, k \leq 10^5$。

The solution

显然这个问题可以DP，但是其时间复杂度太高。

如果直接贪心，则存在一个非常重要的问题——对什么贪心？

原题等价于找到最小的x，使得原序列可以被划分成k个连续段，并且每段的和均$\leq x$，显然，如果对于某个x可行，那么$x+1$也一定可行，即可行的答案具有单调性。

这时候我们就可以考虑二分答案，在二分完答案x之后，我们的贪心目标就非常明确了——将序列划分成数量尽可能少的连续段，使得每段的和均$\leq x$，x可行等价于最终划分的连续段个数$\leq k$。

在这里，二分答案帮助我们确定了贪心的目标，每次贪心均是线性的，总时间复杂度为$O(n \log ans)$。

分治

分治，即分而治之，中国自古以来就有这个思想——想办法使敌方分裂，然后分别解决。

分治的英文翻译即divide and conquer，这个意思也相当地明确。

其一般指将一个规模较大的问题分解成若干个小的子问题，核心在于缩减问题的规模，以换取计算上的方便或者时空复杂度的降低等等好处。

涉及分治思想一般都是比较套路的东西，比如排序、二分查找、优化的矩阵乘法什么的，当然小学奥数里的找假币问题应该也算吧。

目前阶段，分治能讲的东西不是很多，我们讲讲分治的大致过程，再以几个简单的例子带过这一部分。

分治的步骤

1、确定大问题，如果这个问题足够简单而不需要再分解，那么可不对该问题应用下面的步骤，而可以直接解决之。

2、确定将大问题分解为若干个类似的子问题的方式。

3、先解决这些子问题。

4、将这些子问题合并成原大问题的解（不一定必要）

一个入门例子——找第K大的数

这里我们讲的是基于快速排序的分治求解的方法。

1、确定大问题——在一个序列中找从小到大第K个数。

2、划分方式——类似于快排，选取一个基准，将大于这个基准的数划为一部分，将小于这个基准的数划为另一部分。

3、递归解决子问题——假设小于的部分有x个数，如果$x+1<K$，那么我们所寻找的数就是大于的部分中第$K-x-1$大的数，如果$x+1=K$，那么基准即为答案，如果$x+1>K$，那么原大问题的解就是小于的那部分中第K大的数这个子问题的解。

可以看到，我们划分后子问题仅仅在划分后的某一部分，用数学知识可以证明该算法期望复杂度为$O(n)$。

另一个入门例子——二分快速幂

假如说我们现在要求$a^b \mod p$的结果，记为$f(a,b)$，其中a,b均为非负整数。

若$b=0$或$b=1$，则显然。

否则，对于$f(a,b)$，则我们只要求出$f(a,b/2)$的结果x，若b是偶数，则$f(a,b)=x^2$，否则$f(a,b)=a*x^2$。

类似地，我们可以做二分快速乘，这个东西一般用于处理模数在10^{18}级别的乘法问题。

纯分治的例题——安排比赛

已知k是非负整数，设有两支分别有2^k名队员的队伍，队员队内编号分别为$1, 2, 3, ...2^k$。

要求安排一个2^k天的日程，每一天内，某一队的任意一名队员都会与另一队的某位队员进行比赛，且任意一名队员一天都不能参加两场比赛。

要求日程完成后，每名队员都要与另一队的所有队员分别比赛过一次，并且假如有一天队伍A的编号为x的队员与队伍B的编号为y的队员比赛，那么这一天，队伍B编号为x的队员也要与队伍A编号为y的队员进行比赛。

使用分治模式

1、确定大问题——给两个队伍（2^k名选手）安排比赛，如果$k = 0$，那么这个问题已经足够简单。

2、划分方式——将比赛日程分为前后两半，将每个队伍根据编号大小也等分为两半，在比赛日程的前一半，让两个队伍的小编号组和大编号组分别比赛，在比赛日程的后一半，让两个队伍的小编号组与大编号组交叉比赛，那么问题就被划分成了4个规模较小的子问题。

3、递归解决这4个子问题。

4、合并——注意到我们处理日程后半段的两个子问题的方式是相同的，那么得到的方案一定满足题设中的对称性，即若子问题已得到解，则直接将这4 个子问题合并即可得到原问题的一个合法解。

一个稍微复杂一点的问题

已知k是非负整数，且$n = 2^k$，有$n * n$的正方形网格，并指定一个不可覆盖格子X。

你的任务就是用3格的L形地毯无重叠地覆盖除了X之外的所有格子。

运用分治模式

1、确定大问题——覆盖原正方形网格除X之外的所有格子，如果正方形网格边长为2，那么问题就非常简单了。

2、划分——将原正方形网格等分成4块，那么一定有3块是完整的，可以放置一个*L*形地毯，使得这3块也出现一个（人造的）不可覆盖格子，将问题划分成4个相同的，规模较小的子问题。

3、递归解决这4个子问题。

4、合并时，将这4块的方案，与拆分子问题时为使子问题相同所加的那块地毯进行合并，即可得到原问题的一个方案。

于是这个问题就被轻松地解决了。

这张PPT讲完啦

课间休息一会儿~。

Questions are welcomed!