浅谈信息学竞赛中的区间问题

华东师大二附中

周小博

【摘要】

本文对一些常用的区间问题模型做了简单介绍，包括一些算法及其正确性的证明，并从国际、国内的信息学竞赛与大学生程序设计竞赛中选了近 10 道相关例题，进行简要分析。

【关键字】

区间模型 转化 贪心 动态规划 优化

【引言】

在信息学竞赛中，有很多问题最终都能转化为区间问题：例如从若干个区间中选出一些满足一定条件的区间、将各个区间分配到一些资源中、或者将一些区间以某种顺序放置等。这类问题变化繁多，解法各异，需要用到贪心、动态规划等算法，并可以用一些数据结构优化算法。

本文将从几个方面对区间问题做一个简单的介绍，给出一些算法及其正确性的证明，具体分如下几个方面进行讨论：

1.最大区间调度问题

2.多个资源的调度问题

3.有最终期限的区间调度问题

4.最小区间覆盖问题

5.带权区间调度、覆盖问题

6.区间和点的有关问题

我们将对上述每个问题都给出基本模型、算法、证明及其实现，并从 ACM-ICPC、CEOI、CTSC 等比赛中选出了近 10 道相关例题，进行简要分析，有的例题还给出了各种不同的算法及其时间效率的分析。

本文中所讨论的问题主要由两个部分组成，一部分为近几年来各类竞赛题的归纳总结，另一部分来自于参考文献。

【正文】

1. 最大区间调度问题

数轴上有 n 个区间，选出最多的区间，使得这些区间不互相重叠。

算法：

将所有区间按右端点坐标从小到大排序，顺序处理每个区间。如果它与当前已选的所有区间都没有重叠，则选择该区间，否则不选。

证明：

显然，该算法最后选出的区间不互相重叠，下面证明所选出区间的数量是最多的。设 f_i 为该算法所接受的第 i 个区间的右端点坐标，g_i 为某最优解中的第 i 个区间的右端点坐标。

命题 1.1　当 $i \geq 1$ 时，该算法所接受的第 i 个区间的右端点坐标 f_i ≤某最优解中的第 i 个区间的右端点坐标 g_i。

该命题可以运用数学归纳法来证明。对于 $i=1$，命题显然为真，因为算法第一个选择的区间拥有最小右端点坐标。令 $i>1$，假定论断对 $i-1$ 为真，即 $f_{i-1} \leq g_{i-1}$。则最优解的第 i 个可选区间所组成的集合包含于执行该算法时第 i 个可选区间所组成的集合；而当算法选择第 i 个区间时，选的是在可选区间中右端点坐标最小的一个，所以有 $f_i \leq g_i$。证毕。

设该算法选出了 k 个区间，而最优解选出了 m 个区间。

命题 1.2　最优解选出的区间数量 m =该算法选出的区间数量 k。

假设 $m>k$，根据命题 1.1，有 $f_k \leq g_k$。由于 $m>k$，必然存在某区间，在 g_k 之后开始，故也在 f_k 之后开始。而该算法一定不会在选了第 k 个区间后停止，还

会选择更多的区间，产生矛盾。所以$m \le k$,又因为m是最优解选出区间个数，所以$m = k$。

综上所述，算法选出的区间是最优解。

实现：

在判断某个区间与当前已选的所有区间是否重叠时,可以直接判断是否和所选的最后一个区间是否重叠。时间复杂度：排序$O(n\log n)$+扫描$O(n)$=$O(n\log n)$。

例题 1：Latin America - South America 2001 Problem A

题目大意：

基因是由许多外显子（exon）组成，每个外显子都有一个起始位置和一个结束位置。两个外显子能够互相联接必须满足某个外显子的起始位置在另一个外显子的结束位置之后。给出$n(0 < n < 1000)$个外显子，要求找到一条由外显子组成的基因链，使得外显子数量最多。

分析：

将每个外显子看作一个区间，两端点坐标分别为外显子的起始、结束位置。则现在的问题和原问题基本相同，只是端点上也不能重叠。这里就不写算法了。

2.多个资源的调度问题

有n个区间和无限多的资源，每个资源上的区间之间不互相重叠。将每个区间都分配到某个资源中，使用到的资源数量最小。

定义区间集合深度d为包含任意一点的区间数量的最大值，则显然有：

命题 2.1　需要的资源数至少为d。

设区间I_1, I_2, …, I_d全部包含某一点，则必须把这些区间分配到不同资源中，故至少需要d个资源。

其实竞赛中也出现过计算区间集合深度的题目，如 North America - Northeast 2003 Problem E。下面给出计算区间集合深度的算法。

d的计算方法：

将每个区间拆成两个事件点，按坐标从小到大排序，顺序处理每个区间。记录一个值t，表示当前点被包含次数。每次遇到区间的左端点就+1，遇到右端点就-1。d的值就是在该过程中t的最大值。注意两个相同坐标不同类型的事件点的位置关系和区间是否能在端点处重叠有关，这在排序过程中应该注意。时间复杂度为排序$O(n\log n)$+扫描$O(n)$=$O(n\log n)$

由此可得出一个构造算法。

算法 1：

计算出d。将所有区间按左端点坐标从小到大排序，顺序处理每个区间。处理某个区间时，排除在它前面且与之重叠的区间所用到的资源，然后在1, 2, 3, …, d这些资源中任意分配一个未被排除的资源。

证明：

设排序后的n个区间分别为I_1, I_2, …, I_n

命题 2.2　每个区间都能分配到一个资源。

考虑任何一个区间I_j，假设存在t个区间在它前面且与之重叠。一定有$t+1\le d$，否则由于这$t+1$个区间都包含I_j的起始时刻，与d的定义矛盾。故有$t\le d-1$，从而在d个资源中至少有一个资源没被这t个区间排除。所以对于每个

区间，都至少有一个可分配资源。证毕。

命题 2.3　没有两个互相重叠的区间被分配到了同一个资源。

考虑任意两个互相重叠的区间I_{j_1}、I_{j_2}，其中$j_1 < j_2$。当算法在处理区间I_{j_2}时，区间I_{j_1}所用资源已经被排除，所以不会被分配到同一资源。

综上所述，该算法可以成功的构造出正好使用d个资源的一组解，而根据命题 2.1，所用资源的下限是d，故该解是用到资源数量最少的解。

实现：

区间分配资源时，如果直接做排除，则时间复杂度为$O(n^2)$，当然可以通过记录每个资源的最大右端点坐标以将时间复杂度降为$O(nd)$；由于可以每次找一个最大右端点坐标最小的资源，故可以用一个优先队列来储存每个资源的最大右端点坐标。

用二叉堆实现该优先队列，每次查找最小值的时间复杂度为$O(1)$，修改关键字的时间复杂度为$O(\log d)$，分配资源操作的时间复杂度也就降为$O(n\log d)$。故总时间复杂度为：d的计算$O(n\log n)$+排序$O(n\log n)$+分配资源$O(n\log n)=O(n\log(nd))$。其实这里可以不计算出d，而通过不断地增加资源数量来使所有区间都能分配到资源。

该算法同时也证明了以下命题：

命题 2.4　用到的资源数量的最小值为区间集合深度d。

基于这个命题，我们很容易想到另外一种算法。

算法 2：

计算 d 。将所有区间按右端点坐标从小到大排序，顺序处理每个区间。对于每个区间，都在 $1,2,3,\cdots,d$ 这些资源中分配一个可用的且右端点坐标最大的资源。

证明：

显然每个资源上的区间都不互相重叠，下面证明每个区间都能分配到一个可用资源。用一个元素个数为 d 的有序表 S_i（表中元素始终从小到大排序），记录处理 i 个区间后，各个资源的最大右端点坐标（当某个资源从没被用过时，可认为值为 $-\infty$ ）。同样用表 S_i' 表示处理 i 个区间后，某最优解的状态（根据命题 2.5，该表中元素个数也是 d ）。状态 S_i' 不优于 S_i 状态指 $\forall 1\le j\le d, S_i[j]\le S_i'[j]$。

命题 2.5　处理完第 $i(i\ge 1)$ 个区间后，某最优解此时的状态 S_i' 不优于运行该算法后此时的状态 S_i 。

该命题可以运用数学归纳法来证明。对于 $i=1$，命题显然为真，因为第一个区间无论分配哪个资源，状态都是一样的。令 $i>1$，假定论断对 $i-1$ 为真，即 $\forall 1\le j\le d, S_{i-1}[j]\le S_{i-1}'[j]$。设处理第 i 个区间时，该算法分配的资源是 $S_{i-1}[j_1]$, 在最优解中分配的资源为 $S_{i-1}[j_2]$。设第 i 个区间的右端点坐标为 $e[i]$。

表 S_i 更新为：$(S_{i-1}[1], S_{i-1}[2], \ldots, S_{i-1}[j_1-1], S_{i-1}[j_1+1], S_{i-1}[j_1+2], \ldots, S_{i-1}[d-1], e[i])$；

表 S_i' 更新为：$(S_{i-1}'[1], S_{i-1}'[2], \ldots, S_{i-1}'[j_2-1], S_{i-1}'[j_2+1], S_{i-1}'[j_2+2], \ldots, S_{i-1}'[d-1], e[i])$。

而该算法分配的资源 $S_{i-1}[j_1]$ 在所有可分配资源中拥用最大右端点坐标，故区间 i 的左端点坐标小于 $S_{i-1}[j_1+1]$，又 $S_{i-1}[j_1+1]\le S_{i-1}'[j_1+1]$，所以 $j_2\le j_1$。

⑴当 $1\le j<j_2$ 时：$S_i[j]=S_{i-1}[j]\le S_{i-1}'[j]=S_i'[j]$；

⑵当 $j_2\le j<j_1$ 时：$S_i[j]=S_{i-1}[j]\le S_{i-1}'[j]\le S_{i-1}'[j+1]=S_i'[j]$；

⑶当 $j_1 \le j < d$ 时： $S_i[j] = S_{i-1}[j+1] \le S'_{i-1}[j+1] = S'_i[j]$；

⑷当 $j = d$ 时： $S_i[j] = S'_i[j] = e[i]$。

所以有 $\forall 1 \le j \le d, S_i[j] \le S'_i[j]$，证毕。

命题 2.6　每个区间都能分配到一个资源。

处理第 i 个区间时，状态可用 S_{i-1} 表示。最优解肯定能给该区间分配到资源，设为 $S'_{i-1}[j_0]$。根据命题 2.5，$S_{i-1}[j_0] \le S'_{i-1}[j_0]$。则运行该算法时，至少可以给该区间分配资源 $S_{i-1}[j_0]$。证毕。

综上所述，该算法可以成功的构造出正好使用 d 个资源的一组解，而根据命题 2.1 所用资源的下限是 d，故该解是用到资源数量最少的解。

实现：

考虑如何分配资源。如果直接记录每个资源的最大右端点坐标，时间复杂度为 $O(nd)$。可以用证明中所提到的有序表，用一个平衡二叉树来维护它。每次处理某个区间时，可以在 $O(\log d)$ 时间内找到合适资源，并在 $O(\log d)$ 时间内修改该资源结束时间。这样做的总时间复杂度也是 $O(n \log(nd))$。

第一个构造算法证明了最少所用资源的数量，并用编程复杂度较低的方法构造出了可行解。第二个贪心算法是基于第一个算法的结论得到的，编程复杂度也比第一种高，然而利用它的局部最优性，还可以附加更多的条件。

例题 2：2004 年广东省大学生程序竞赛试题 Problem B

题目大意：

一个港口被分成了 $m(1 \le m \le 10)$ 个区域，每个区域最多允许同时停靠

$r(r \le 10000)$艘船，每艘船都只能停靠在特定的一个区域。有$n(1 \le n \le 100000)$艘船，已知每条船的到达时刻、离开时刻和它应该进入的区域。求一个调度方案使得能有最多的船得以停靠。

分析：

显然每个区域都可以单独处理。可以把船看作一个区间，则该问题是问题 2 的一个变化：规定使用资源的数量r，问最多能选出多少区间。在算法 2 上稍加改动即可得出该题的算法。

算法：

将所有区间按右端点坐标从小到大排序，顺序处理每个区间。对于每个区间，若能找到可用资源，则将该区间安排到一个可用的且右端点坐标最大的资源；若找不到，则不去选它。

在前面的证明中我们已经证明了选择资源的局部最优性，唯一剩余的问题是区间的取舍。如果该算法选择了某区间I，而某最优解中没有选它，显然最优解选择了一个与该区间重叠且右端点坐标大于等于它的区间I'，我们用I替换I'，就构成了一个包含I的最优解；如果该算法没有选择某区间，显然最优解也不会选择该区间。故该算法得出的解就是最优解。

实现：

还是利用平衡二叉树维护资源信息并寻找合适资源，处理每个区间的时间复杂度是$O(\log)$，故总时间复杂度为：排序$O(n\log)$+处理区间$O(n)\cdot O(\log r)=O(n\log(nr))$。

3. 有最终期限的区间调度问题

有n个长度固定、但位置可变的区间，将它们全部放置在$[0,+\infty)$上。每个区

间有两个已知参数：长度 t_i 和最终期限 d_i，设 f_i 为其右端点坐标。定义 $l_i = \begin{cases} f_i - d_i & if\ f_i > d_i \\ 0 & if\ f_i \le d_i \end{cases}$，$L = \max_{1 \le i \le n} \{l_i\}$。放置所有区间，使它们不互相重叠且最大延迟 L 最小。

算法：

将所有区间按最终期限从小到大排序，顺序安排各区间，使中间没有空闲段。

证明：

由于该算法所有区间都顺序放置，因此区间之间不互相重叠。下面证明该算法求出的 L 最小。

命题 3.1　存在一个没有空闲段的最优解。

若在某最优解中，两相邻区间 i、$i+1$ 之间有空闲段，可将将 $i+1$、$i+2$、…、n 全部往左移，直至 i、$i+1$ 之间没有空闲段。显然 f_{i+1}、f_{i+2}、…、f_n 减小，即 l_{i+1}、l_{i+2}、…、l_n 减小。所以此时的最大延迟 L' 一定小于等于原先的 L，又 L 已是最小值，所以 $L' = L$。因此任何一个有空闲段的最优解不断地进行上述操作后终会变成一个没有空闲时间的最优解。证毕。

命题 3.2　任何一个既没有逆序、又没有空闲段的解有着相同的最大延迟 L。

如果解中既没有逆序、又没有空闲段，那么相同最终期限的区间只有在次序上可能不同。而这些区间的最大延迟只和最后一个区间的右端点坐标有关，区间的次序却并不影响最后一个区间的右端点坐标，所以最大延迟始终相同。证毕。

命题 3.3　既没有逆序、又没有空闲段的解是最优解。

根据命题 3.1，存在一个没有空闲段的最优解。假设某最优解中相邻的某两区间 i、$i+1$ 为逆序关系，即有 $d_i > d_{i+1}$。设区间 d_i 的左端点坐标为 t_0，若交换两

区间，则原来两区间的右端点坐标为 $f_i = t_0 + t_i$， $f_{i+1} = t_0 + t_i + t_{i+1}$，现在则分别为 $f'_{i+1} = t_0 + t_{i+1}$， $f'_i = t_0 + t_i + t_{i+1}$。设两区间在交换前后的延迟分别为$l_i$、$l_{i+1}$、$l'_i$、$l'_{i+1}$。

因为 $\left.\begin{array}{l} f'_{i+1} < f_{i+1} \Rightarrow l'_{i+1} \le l_{i+1} \\ d_i > d_{i+1} \\ f'_i < f_{i+1} \end{array}\right\} \Rightarrow l'_i \le l_{i+1} \Bigg\} \Rightarrow \max\{l'_i, l'_{i+1}\} \le l_{i+1} \le \max\{l_i, l_{i+1}\}$，所以 l_i、l_{i+1} 不优于 l'_i、l'_{i+1}。因此一个存在逆序的解交换相邻的逆序项可以使之更优，只要将一个有逆序的解不断进行上述操作后都能成为一个无逆序的解。证毕。

综上所述，该算法成功构造了一个最优解。

实现：

按算法直接模拟。时间复杂度：排序 $O(n \log n)$+扫描 $O(n)$=$O(n \log n)$。

一般在区间位置可变时，最优解中各区间位置常常是按照最终期限排序，交换最终期限互为逆序的区间总能使结果更优。当然区间带权时，要另外考虑。

例题 3.1：CTSC 2007 DAY2 pendant

题目大意：

有 $N(N \le 200000)$粒缀珠，每粒缀珠有两个参数：挂钩的最大承受重量C_i和本身重量W_i。将缀珠经挂钩连接后挂起形成挂坠，要求每粒缀珠下方的缀珠重量之和不能超过它挂钩的最大承受重量。问挂坠最多能由多少个缀珠组成，并求出满足缀珠数量最多时挂坠质量的最小值。

分析：

把每粒缀珠看作一个长度为W_i的区间，这些区间的位置是可变的。则问题转化为：选出最多的区间，并将它们不互相重叠地放置在$[0,+\infty)$上，使得左端点

坐标不能超过C_i。在保证区间数量最大时，需求出最大右端点坐标的最小值。

根据原问题，容易证明下面这个结论：显然必有一个最有解，它的区间按最后期限W_i+C_i排序，连续地放置在$[0,+\infty)$上。（证明和原问题的证明方法类似）在这个结论的基础上，可以得出以下两种算法。

算法 1：

容易想到一个$O(N^2)$的动态规划。将区间按W_i+C_i的值从小到大排序，设$f[i,j]$为前i个区间，选取了j个区间后，最大右端点坐标的最小值。则对于任意一个合法的$f[i,j]$，做下面的两个动态转移：若第$i+1$个区间不选：$f[i+1,j]=\min\{f[i+1,j],f[i,j]\}$；若$f[i,j]\le C_{i+1}$，则可以选择该区间：$f[i+1,j+1]=\min\{f[i+1,j+1],f[i,j]+W_{i+1}\}$。状态数为$O(N^2)$，状态转移时间为$O(1)$，故时间复杂度为$O(N^2)$。然而仅仅$O(N^2)$是不够的，还需要优化。

优化：

设$g[i,j]=f[i,j]-f[i,j-1]$，显然i不变时，$g[i,j]$随着j的增大而呈单调递增（如果$g[i,j]<g[i,j-1]$，那么$f[i,j-1]$肯定不是最大右端点坐标的最小值）。故每次从$g[i]$递推$g[i+1]$时，可以进行如下的两个操作：⑴找到两个位置，$left=\min\{j\mid g[i,j]>W_{i+1}\}$，$right=\max\{j\mid f[i,j]\le C_{i+1}\}$；⑵删除$g[right+1]$，将$g[left+1]$，$g[left+2]$，…，$g[right]$的值向后平移到$g[left+2]$，$g[left+3]$，…，$g[right+1]$，并将$g[left+1]$的值修改为$W_{i+1}$。

时间复杂度：

用一棵平衡二叉树维护$g[i]$，则从$g[i]$递推$g[i+1]$的时间复杂度为$O(\log N)$，

所以总时间复杂度为：排序$O(N\log N)$+$g[i]$的维护$O(N)\cdot O(\log N)=O(N\log N)$。

该算法要用到平衡二叉树，编程复杂度较大。其实仔细分析算法 1，并加以改进，就可以得到一个贪心的算法。

算法 2：

维护一个按W_i+C_i排序的有序表，初始为空。将所有区间按长度W_i从小到大排序，依次处理。处理某个区间时，若它能够放入有序表，则选择该区间并放入表中，否则不选择。最后的表即是要求的状态。

实现：

算法的瓶颈在有序表的维护上。如果直接模拟该表的所有操作，时间复杂度是$O(N^2)$。用线段树来维护该表，可以在$O(\log N)$的时间里完成每个区间的取舍及插入操作(这些操作并不是非常容易实现,但与论文主题无关,这里不再细写)。因此，总时间复杂度为$O(N\log N)$。

例题 3.2：Europe - Southeastern 2007 Problem D

题目大意：

一家银行收到了$N(0\le N\le 10000)$个贷款申请，每个贷款都最晚要在$d_i(0\le d_i\le 10000)$时完成，利润为p_i。每个贷款需要一个单位时间处理，银行在同一时间内最多可以接受$L(0\le L\le 100)$个贷款。求如何安排才能获得最大利润。

分析：

将每个贷款申请看作一个单位长度、位置可变且带权的区间，则题目转化为选出一些区间，将它们不互相重叠地放在L个资源$[0,+\infty)$上，使利润最大，且区

间的左端点不得超过d_i。可以用与例题 3.1 算法 2 类似的贪心算法解决该问题。

算法：

将这些区间按照权值从大到小排序，顺序处理每个区间。设$D=\max_{1\le i\le N}\{d_i\}$。由于区间都是单位长度的，我们可以用一维数组$s[i](1\le i\le D)$来记录当前的情况，表示$(i,i+1)$被覆盖次数，根据题意有$0\le s[i]\le L$。处理第$i$个区间时，若可以选择该区间（即$\exists 0\le i\le d_i, s[i]<L$），则将该区间放置到一个$i$最大的一个位置，即$\max_{0\le i\le d_i}\{i\mid s[i]<L\}$，否则就不选择该区间。

处理每个区间时，若直接模拟，时间复杂度是$O(ND)$，最好还是做适当优化。

优化 1：

若用一棵线段树来维护数组$s[i]$，可以在$O(\log D)$的时间里处理每个区间。总时间复杂度为：排序$O(N\log N)$+处理每个区间$O(N)\cdot O(\log D)=O(N\log(ND))$。

优化 2：

每次$s[i](i>0)$的值增加后，若$s[i]=L$，则将之所属集合与前面的$s[i-1]$所属集合合并成一个新集合。处理第i个区间时，它选择的位置必是$s[d_i]$所在集合中的最前面的元素，（若最前面的元素是$s[0]$且$s[0]=L$，则表示该区间不能被选择）。如果我们用并查集来实现集合的各种操作，维护它的时间复杂度可近似地看作$O(N+D)$。总时间复杂度：排序$O(N\log N)$+处理每个区间$O(N+D)=O(D+N\log N)$。

第 1 种优化由于用到了线段树，不论在空间上还是在时间上都有着巨大的系

数；而第 2 种优化用到的空间非常小，且在时间效率和编程复杂度两方面都比第 1 种好很多。

4. 最小区间覆盖问题

有 n 个区间，选择尽量少的区间，使得这些区间完全覆盖某线段 $[s,t]$。

算法：

将所有区间按左端点坐标从小到大排序，顺序处理每个区间。每次选择覆盖点 s 的区间中右端点坐标最大的一个，并将 s 更新为该区间的右端点坐标，直到选择的区间已包含 t。

证明：

显然，该算法最后选出来的区间完全覆盖 $[s,t]$，下面证明所选出区间的数量是最少的。设 f_i 为该算法所接受的第 i 个区间的右端点坐标，g_i 为某最优解中第 i 个区间的右端点坐标。

命题 4.1　当 $i \geq 1$ 时：该算法所接受的第 i 个区间的右端点坐标 f_i ≥某最优解中的第 i 个区间的右端点坐标 g_i。

该命题可以运用数学归纳法来证明。对于 $i=1$，命题显然为真，因为算法第一个选择的区间是能覆盖点 s 的区间中右端点坐标最大的那个。令 $i>1$，假定论断对 $i-1$ 为真，即 $f_{i-1} \geq g_{i-1}$，最优解中选的第 i 个区间的右端点坐标为 g_i，设它的左端点坐标为 b_i。由于该区间覆盖 g_{i-1}，即 $b_i \leq g_{i-1} \leq g_i$。当 $g_i \leq f_{i-1}$ 时，由于 $f_{i-1} \leq f_i$，有 $g_i \leq f_i$；当 $g_i > f_{i-1}$ 时，有 $b_i \leq g_{i-1} \leq f_{i-1} < g_i$，此时最优解所选择的区间覆盖点 f_{i-1}，又算法选择的第 i 个区间是右端点坐标最大的那个，故 $f_i \geq g_i$。证毕。

设该算法选出了 k 个区间，而最优解选出了 m 个区间。

命题 4.2　最优解选出的区间数量 m =该算法选出的区间数量 k 。

假设 $m<k$，根据命题 4.1，有 $f_m \geq g_m$。根据该算法，$g_m \leq f_m < t$，因此该最优解不可能覆盖 t，产生矛盾。所以 $m \geq k$，又因为 m 是最优解中选出区间个数，所以 $m=k$ 。

综上所述，算法选出的区间是最优解。

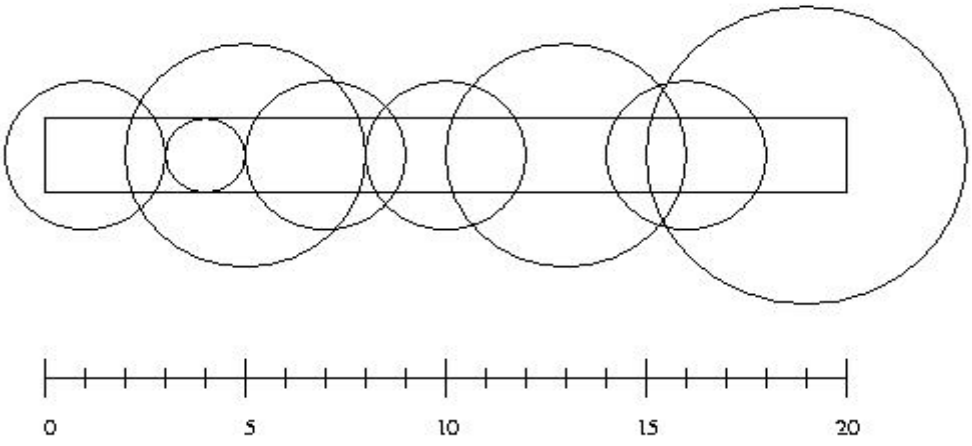
实现：

选择区间可以通过线性扫描来实现。时间复杂度：排序 $O(n\log n)$+扫描 $O(n)=O(n\log n)$。

例题 4：ACM/ICPC Regional Warm-up Contest 2002 Problem E

题目大意：

有一块草坪，长为 l，宽为 w，在它的中心线的位置处装有 $n(n\leq 10000)$ 个点状的喷水装置。每个喷水装置 i 喷水的效果是以它为中心半径为 r_i 的圆都被润湿。请选择尽量少的喷水装置，把整个草坪全部润湿。如图所示：



分析：

显然覆盖整个草坪的充要条件是要能覆盖上边界。将每个喷水设置能覆盖的一段上边界看成一个区间，转化后的问题就是原问题。

5.带权区间调度、覆盖问题

若区间带权，如何解决调度、覆盖一类问题呢？这时，刚才一直用的贪心算法已经不再普遍适用，大部分情况下，只能用更一般性的方法——动态规划。

例题 5.1：Europe - Northeastern Europe 2004 Problem J

题目大意：

有 $N(N \le 1000)$ 只可爱的小海龟在赛跑。询问每只小海龟它是第几名，它会回答你两个数：a_i, b_i，分别表示在它前面的小海龟数和在它后面的小海龟数。接着你发现其中有些小海龟对你撒了谎，因为根据他们的说法你根本没法给他们排队！但是你是善良的，你不希望有很多小海龟在撒谎，想找出最少有哪几只小海龟在撒谎。（注意：小海龟的名次可能是并列的！）

分析：

若一只海龟说了真话，那么该海龟的位置一定是在区间 $[a_i+1, N-b_i]$ 上。若有 K 只海龟选择了相同的区间 $[a_i+1, N-b_i]$，则根据并列关系，该区间最多能同时拥有 $\min\{N-a_i-b_i, K\}$ 只海龟。可以计算出每个区间最多能有多少只海龟，把数值看做区间的“权”。则问题转化为，在一些带权区间中，选出权和最大的区间，使它们之间不能互相重叠。

算法：

算出每个出现过的区间的“权” $v[i]$，接下来的算法就是动态规划了。先按

右端点坐标从小到大排序，令 $p[i]$为在区间 i 左边的且与之无公共点的最大区间编号，设状态 $f[i]$为在前 i 个区间中可选出区间的最大权和，则状态转移方程为 $f[i]=\max\{f[i-1], f[p[i]]+v[i]\}$，说真话海龟的最大数量就是最后一个区间的 f 值。

实现：

由于上述所有操作的排序关键字都在 $0,1,...,N$ 中，所以可以使用时间复杂度为 $O(N)$的基数排序，其它各操作也均能在 $O(N)$时间内完成（具体细节留给读者思考）。所以，总时间复杂度为 $O(N)$。

例题 5.2：USACO 2005 dec silver

题目大意：

仓库从第 M 秒到第 E 秒的任意时刻都需要有人打扫，包括 M 和 E，$0\le M\le E\le 86399$。有 $N(1\le N\le 10000)$个工人前来应聘打扫仓库的工作，每人给出自己可以工作的时间段：从第 $T1$ 秒到第 $T2$ 秒（包括第 $T1$ 秒和第 $T2$ 秒），需要支付工资 S 元。 $M\le T1\le T2\le E$， $0\le S\le 500000$。

每个应聘者，你能够录用或者不录用，但不能只雇佣他提出的一部分时间。一旦录用，你就得支付他提出的 S 元给他。现在要保证从 M 秒到第 E 秒的任意时刻都得有人打扫，问最少要付多少工资。

分析：

将每个人的打扫时间看作一个左右端点分别为 $T1_i$ 和 $T2_i$，且权为 S_i 区间。则问题转化为：选出一些区间，使它们覆盖 $[M,E]$上的所有整数点，求权和最小值。

算法：

这道题很容易想到一个$O(N^2)$的动态规划算法。将所有区间按右端点坐标从小到大排序，顺序处理每个区间。用$f[i]$表示在前i个区间中选择，且第i个区间必须选时，覆盖$[M,T2_i]$的权和最小值。

则状态转移方程为：$f[i]=\begin{cases}\underset{1\le j<i}{Min}\{f[j]\mid T2_j+1\ge T1_i\}+S_i & if\ M\notin[T1_i,T2_i]\\ S_i & if\ M\in[T1_i,T2_i]\end{cases}$，

最后的结果就是$\underset{1\le i\le N}{Min}\{f[i]\mid E\in[T1_i,T2_i]\}$。

第一种优化：

对于这个状态转移方程，很容易想到用线段树优化的方法。建立一棵范围是$[M,E]$的线段树，叶子节点是一个个整点坐标。每得到一个$f[i]$的值，就将它插入在$T2_i$处，并更新最小值。计算$f[i]$的值时只要选取区间$[T1_i-1,T2_i-1]$中f的最小值进行状态转移即可。算法时间复杂度为：排序$O(N\log N)$+维护线段树$O(N)\cdot O(\log(E-M))=O(N\log(E-M))$，故总时间复杂度是$O(N\log(N(E-M)))$。

这样做编程复杂度较高，且时间效率亦不是很好（虽然可以通过离散化稍微降低一点时间复杂度，但仍然系数巨大）。仔细分析，可以找到更好的优化方法。

第二种优化：

建立一个栈$stack$，$stack[i]$表示处在栈中第i个位置的区间。假设栈中已经有k个区间，保持栈中区间f值的单调性：$\forall 1\le i<j\le k, f[stack[i]]<f[stack[j]]$。每次要将第$i$个区间压入栈中时，做这样的操作：

```
while f[stack[k]] ≥ f[i]
  {pop}
push i
```

计算 $f[i]$的值时，可以借助栈找到一个区间 j，使$T2_j+1\geq T1_i$且 $f[j]$最小，进行状态转移。根据栈中元素的单调性，这里可以用二分查找。由于一共 N 个区间，每个区间进栈、出栈最多一次，故栈的维护的时间复杂度为$O(N)$。在任何时间内，栈内区间个数不超过 N，故一次二分查找复杂度为$O(\log N)$。综上所述，该方法的总时间复杂度为$O(N)+O(N)\cdot O(\log N)=O(N\log N)$。

由于栈、二分查找的系数比线段树小很多，且栈内区间个数很少有达到 N 的情况，故这种优化比前面的要好很多。

第三种优化：

以左端点为关键字从小到大排序，按顺序依次处理每个区间。维护一个以 $f[i]$为关键字的优先队列。处理区间 i 时，只要最小关键字区间的右端点坐标小于$T1_i-1$，就删除。重复上述操作，直到当前区间能和最小关键字区间进行状态转移。状态转移后，将区间 i 插入优先队列。

由于每个区间最多只进出队列一次，故一共要进行 $O(N)$ 次插入操作和 $O(N)$ 次删除操作。用二叉堆实现该优先队列，插入和删除操作的时间复杂度都是 $O(\log N)$，因此总时间复杂度为：排序 $O(N\log N)$+动态规划 $O(N)$+维护二叉堆 $O(N)\cdot O(\log N)$=$O(N\log N)$。

该优化时间复杂度的系数介于前两种优化之间，然而对于实现而言，使用 Pascal 的编程量和用第一种优化差不多，而使用 C/C++就可以大大减少编程量。

6. 区间和点的有关问题

有 n 个区间，m 个点。若某区间包含了某点，则构成一对匹配关系。选出最

多的区间和相同数量的点，使对应的区间和点构成匹配关系。

该问题虽然可以建立一个二分图匹配模型，但时间复杂度显然太高，应该充分利用区间的性质。

算法：

将所有的点按坐标从小到大排序，顺序处理每个点。每次都在剩余区间中取出包含该点且右端点坐标最小的区间与之配对，若没有则不选择该点。

证明：

设某最优解的匹配集合为S，该算法求出的一对匹配是第P个点和第I个区间。

命题 6.1　P和I至少有一个出现在S里。

假设它们都不出现在S里，则S中还可以加上一对新的匹配(P,I)，与S为最优解矛盾。

命题 6.2　该算法求出的所有匹配都可以出现在另一最优解S'中。

仍然考虑该算法求出的任意匹配(P,I)，若$(P,I)\notin S$，则可分以下三种情况讨论：⑴若P出现在S里、I未出现在S里，设S中P与I'匹配。在S中可删除(P,I')，并添加(P,I)。⑵若I出现在S里、P未出现在S里，设S中I与P'匹配。在S中可删除(P',I)，并添加(P,I)。⑶若P和I都出现在S里，设S中P与I'匹配、I与P'匹配。根据该算法，显然有$P'\in I'$，所以可将S的这两组匹配改为(P,I)和(P',I')。因此解S可经过一系列的转化变成解S'，且由于转化时匹配数始终保持不变，所以S'也为最优解，因此该算法中的所有匹配都在其中。

根据该算法，显然不可能有匹配在S'中却不在该算法所求出的匹配中，故

该算法求出的匹配集就是S'，为一最优解。

实现：

选点时直接模拟的时间复杂度为$O(nm)$，可用与例题 5.2 的优化 3 相似的方法来对该算法进行优化。做预处理，以区间左端点为关键字，对它们排序，得到一个有序表。维护一个优先队列，以区间的右端点作为关键字。将所有点按坐标从小到大排序，依次处理各点。处理某点时，先插入左端点小于等于该点坐标且从未进入过队列的区间（插入区间顺序和有序表一致，故每次可在$O(1)$时间内判断是否需要插入），再删除右端点小于该点坐标的区间，将优先队列里的区间中右端点坐标最小的与该点匹配并删除。

由于每个区间只进出队列一次，故一共要进行$O(n)$次插入操作和$O(n)$次删除操作。用二叉堆实现该优先队列，则每次操作的时间复杂度降为$O(\log n)$。因此，该算法总的时间复杂度为：计算有序表$O(n\log n)$+点排序$O(m\log m)$+维护优先队列$O(n)\cdot O(\log n)$+处理每个点$O(n)=O(m\log m)+O(n\log n)$。

例题 6.1：CEOI 2004 trips

题目大意：

有n个团队和m条旅行线路，每个团队最多只能选一条旅行线路，每条旅行线路至多只能供一个团队选择。第i个团队有人数s_i，第j条旅行线路有人数下限l_j和上限u_j，只有当$l_j \le s_i \le u_j$时，第i个团队才能选择第j条旅行线路。求团队和旅行线路的最大配对数。

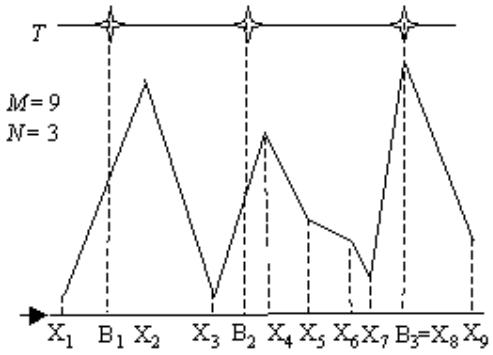
分析：

将团队能去的旅行线路看作区间，将每条旅行线路看做点。转化后的问题和原问题完全一样，这里就不再写算法了。

例题 6.2：CEOI 2000 Enlightened Landscape

题目大意：

在一片山的上空，高度为T处有$N(1 \le N \le 200)$个处于不同水平位置的灯泡（如图）。如果山的边界上某一点与灯i的连线不经过山上的其他点，我们称灯i可以照亮该点。开尽量少的灯，使得整个山景都被照亮。山景是一条折线，折点数$M \le 200$。



算法：

容易看出，照亮整个山景的充分必要条件是照亮所有的折点。我们可以在$O(NM^2)$的时间内算出某灯泡是否能照亮某折点。由于每个灯泡照射到的折点是非连续的，较难处理，我们不妨考虑每个折点能被那些灯泡照到，发现都是连续的段。于是题目转化为：给定M个区间，以及N个点，区间的两端点都是这N个点之一。选出最少的点，使每个区间都至少包含一个所选的点。

显然那些包含其它区间的区间已不必考虑，可以删去。这一步可以这么实现：将所有区间按右端点从小到大，若右端点相同则按左端点从大到小排序，从前往后依次处理。维护一个值t，记录当前已处理区间左端点坐标的最大值，初始值为$-\infty$。处理某个区间时，若该区间的左端点小于等于t，则删除该区间，否则更新t。将剩余区间按位置重新排序，顺序处理各区间。若区间已包含某点，则不去考虑，否则选择区间的右端点。

优化：

若确定了区间，则接下来选点的复杂度显然为$O(M\log M)$,时间效率的优劣取决于预处理每个折点能被哪段灯泡照到的时间。如果直接模拟，时间复杂度为$O\left(NM^2\right)$，对该题的数据规模而言，这个算法已经可以 AC 了。其实可以用栈将预处理的时间复杂度进一步降低：

先从左往右扫描，确定每个折点能被最左边的哪个灯泡照到。对于第i折点，进行这样的操作（设栈中有k个元素）：

```
while 折点 i 的高度大于等于折点 stack[k] 的高度
or 线段（折点 stack[k-1]，折点 i ）在折点 stack[k] 上方
  {pop}
算出射线（折点 i ，折点 stack[k]）与直线 y=T 的交点 (x0,T)
利用二分查找找到横坐标大于等于 x0 的最左边的灯泡
（该灯泡就是能照到折点 i 的最左边的灯泡）
push i
```

同样可以通过从左往右扫描求出每个折点能被最右边的哪个灯泡照到。由于每个折点最多进出栈一次，栈的维护需要的时间为$O\left(M\right)$，故预处理时间复杂度为：栈操作$O\left(M\right)$+二分查找$O\left(M\right)\cdot O\left(\log N\right)=O\left(M\log N\right)$，总时间复杂度也就降为$O\left(M\log\left(NM\right)\right)$。

【总结】

区间问题通常要保持有序性，排序关键字的选择非常重要。只有选择了合适的关键字，才更容易解决这类问题。

区间问题的解决方法主要有两种：①贪心（构造）；②动态规划。虽然后者能解决更多的问题，但往往前者拥有更好的时间效率，实现或优化起来更加容易。

区间问题的优化大多可以用到线段树，极少数要用到平衡二叉树，但往往这并不是最佳的优化方法。要学会在解题过程中发现更好的规律，用编程复杂度和时间效率俱佳的方法进行优化，如利用决策的单调性，或运用一些简单数据结构维护等。

【参考文献】

1. 《算法艺术与信息学竞赛》 刘汝佳 黄亮 著 清华大学出版社
2. 《计算机算法设计与分析》 王晓东编著 电子工业出版社
3. 《算法设计》 Jon Kleinberg，éva Tardos 著 清华大学出版社
4. 《广东省大学生程序设计竞赛试题（2003-2005 年）》
   郭嵩山 黎俊瑜 林祺颖 著 电子工业出版社
5. CTSC 2007 Reports, By Guo Huayang
6. 汪汀《〈turtle〉解题报告》2005 集训队作业
7. 朱晨光《基本数据结构在信息学竞赛中的应用》 2006 集训队论文

【附录】

Latin America - South America 2001 Problem A

Gene Assembly

With the large amount of genomic DNA sequence data being made available, it is becoming more important to find genes (parts of the genomic DNA which are responsible for the synthesis of proteins) in these sequences. It is known that for eukaryotes (in contrast to prokaryotes) the process is more complicated, because of the presence of junk DNA that interrupts the coding regions of genes in the genomic sequence. That is, a gene is composed by several pieces (called exons) of coding regions. It is known that the order of the exons is maintained in the protein synthesis process, but the number of exons and their lengths can be arbitrary.

Most gene finding algorithms have two steps: in the first they search for possible exons; in the second they try to assemble a largest possible gene, by finding a chain with the largest possible number of exons. This chain must obey the order in which the exons appear in the genomic sequence. We say that exon appears before exon j if the end of i precedes the beginning of j. The objective of this problem is, given a set of possible exons, to find the chain with the largest possible number of exons that could be assembled to generate a gene.

Input

Several input instances are given. Each instance begins with the number $0 < n < 1000$ of possible exons in the sequence. Then, each of the next n lines contains a pair of integer numbers that represent the position in which the exon starts and ends in the genomic sequence. You can suppose that the genomic sequence has at most 50000 basis. The input ends with a line with a single `0'.

Output

For each input instance your program should print in one line the chain

with the largest possible number of exons, by enumerating the exons in the chain. The exons must follow the order of appearance (as defined in the statement of the problem). If there is more than one chain with the same number of exons, your program can print anyone of them.

Sample Input

```
6
340 500
220 470
100 300
880 943
525 556
612 776
3
705 773
124 337
453 665
0
2
```

Sample Output

```
3 1 5 6 4
2 3 1
```

（例题 1）

North America - Northeast 2003 Problem E

Who is Still Alive?

At various points in history a number of presidents (former and the current president) have been alive. For example, currently there are six living presidents: Ford, Carter, Reagan, George the Second (George H. Bush), Clinton, and George the Third (George W. Bush). Note: George

Washington was George the First.

Write a program that takes as input the name, inauguration date, and date of death (if appropriate) of several presidents and produces as output the periods of time when the maximum number of current or former presidents are alive, together with the names of those presidents in alphabetical order. Note, there may be several such intervals and all must be listed in chronological order.

Input

The input to your program will start with a line that contains a single integer, N, that specifies the number of presidents to be considered. Each of the following N lines of input describes a single president. An input line for a president will consist of an identifier consisting of letters only. The identifier will be followed by the inauguration date, and by an optional date of death. Dates will be expressed in the form yyyy - mm - dd where yyyy represents a year, mm represents a month, and dd represents a day. If the president is still alive, no death date will be given. One or more white space characters will separate tokens on the input line. Note that if a president was inaugurated more than once, they may appear in the input more than once; remember, however, that once a president, always a president.

Output

The output from the program will specify the period of time when the most presidents were alive, followed by the names in alphabetical order. If there are multiple intervals, the intervals will be listed in chronological order, separated by a blank line.

Sample Input

```
3
Nixon 1969-01-20 1994-04-22
Ford 1974-01-20
```

GWBush 2001-01-20

Sample Output

1969-1-20 to 1994-4-22

Ford

Nixon

1974-1-20 to 2003-11-7

Ford

GWBush

（问题 2 中曾提到）

2004 年广东省大学生程序竞赛试题 Problem B

Berth Allocation

(Input File: berth.in/Output File: berth.out)

Singapore port is one of the busiest ports in the world. In 1994, Singapore maintained its position as the world's busiest port in terms of shipping tonnage of ship arrivals. In that year, there were 101,107 ship arrivals with shipping tonnage of 678.6 million gross tons. One of the planning problems encountered at the post is to decide whether a given set of ships can be berthed in a section of the port with certain berthing constraints.

The port is divided into m sections and no ship can berthed across a section. Ships arrive at the port at different times to be berthed. Every ship has an expected duration of stay which may be different from another ship. You can berth a ship or not when it arrives. To berth a ship is to place the ship along the wharf line of a section. Once a ship if berthed, it will not be moved until its departure, that is, if two ships are in the same section and have a time overlap, they cannot share any part of

the section. What's more, the berthing of the ships cannot exceed the capacity of the section.

To simplify the problem, we just consider the section as a rectangle with a length of L hundred meters and a width of W hundred meters. When a ship arrives, you can berth it at any available position in the section at that time, or you can reject berthing it according to your berthing plan. Now the problem becomes a packing model of 3 dimensions (length, width, time). Since the width of a section if the same as ships, we can just consider the problem in 2 dimensions (length, time).

Here is the explanation of the last data set in the sample input. The port has one section, which is 2 hundred meters long. There are four ships arriving at time 1, 5, 2, 4 and departing at time 3, 6, 8, 10 respectively. They are all berth in section one. Figure 2.2.1 and Figure 2.2.2 are two legal berthing plans, from which we can know the maximal number of berthed ships is 3.

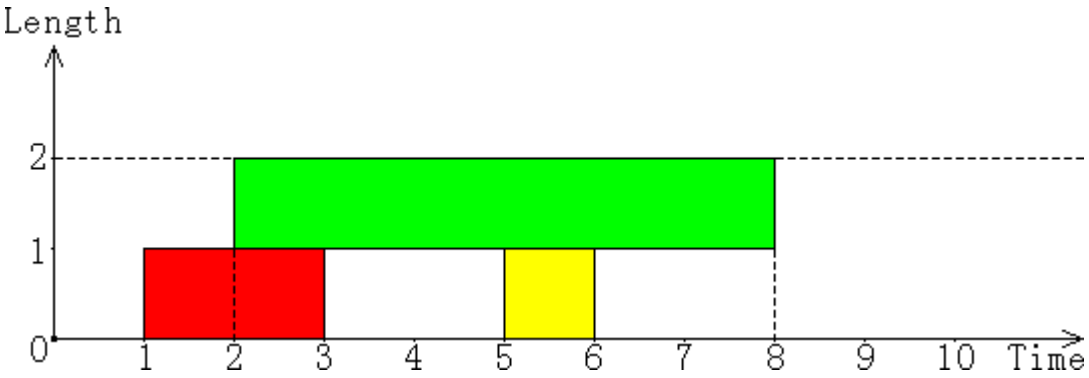


Figure 2.2.1 The Berthing Plan of 3 ships

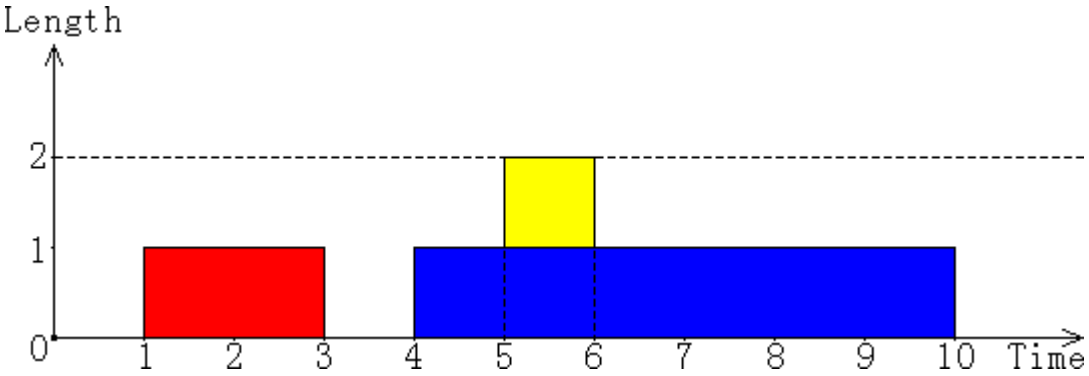


Figure 2.2.2 The other Berthing Plan of 3 ships

Max, the consultant of the harbor bureau, thinks that it is a ZSU(Zappy Ships Unallocation) problem. As his name, Max wants to make a plan to maximize the total number of berthed ships, in other words, minimize the number of unberthed ships. Just like other packing problems, it is difficult for Max to solve. But he believes that the best team of the contest is able to solve it both correctly and effectively. Do your guys want to be the best? Just try it.

Input:

Input contains many test sets. A test data set is defined as follow: The first line of test data set is two integers: m and n, separated by one or more spaces(1≤m≤10, 1≤n≤100000). m is the number of sections in the port, and n is the number of ships.

The next m lines, one positive integer r (that means the length of the section is r hundred meters long) per line. The length of each section does not exceed 10000 hundred meters.

The next n lines gives one ship information on each line, with three non-negative integers s, e, sec (0≤s≤e,1≤sec≤m) per line, s is the arrival time of the ship, e is the departure time of the ship, and sec is the section which the ship should be berthed in.

Input is ended by EOF. You can assume that all the input data are legal.

Output:

For each test data set you should output one integer, the maximal number of the berthed ships, per line.

Sample Input:

```
2 6
3
3
1 2 1
1 2 1
```

1 2 1

1 2 1

1 2 2

1 2 2

1 3

2

1 3 1

2 6 1

2 8 1

1 4

2

1 3 1

5 6 1

2 8 1

4 10 1

Sample Output:

5

2

3

（例题 2）

CTSC 2007 DAY2 pendant

挂缀

【问题描述】

“珠缀花蕊，人间几多酸泪”……

挂缀在很早就被人们作为一种装饰品，垂坠的风韵，华丽摇曳的摆动，展现出一种与众不同的优雅与高贵。而我们的主人公小Q，正想买一条漂亮的挂缀放

在寝室里作为装饰。

挂坠的构成，是由若干粒缀珠相互连接而成。每一个缀珠由三部分组成：分别是珠子、珠子上方的连接环与珠子下方的挂钩（如下图）。我们可以简单的认为从上往下数的第i 个缀珠是将它的连接环套在其上方（也就是第i-1 个）缀珠的挂钩之上（第一个除外）。小Q 想买一根足够长的挂缀，这样显得更有韵味☺

然而商店的老板告诉小Q，挂缀是不可能做到任意长的，因为每一个珠子都受到重力作用，对其上方的挂钩有一定的拉力，而挂钩的承受能力是有限的。老板还告诉小Q，他一共拥有N 个珠缀（假设每一个珠缀都很漂亮，小Q 都很喜欢），每个珠缀都有其各自的重量与承受能力。一个挂缀是稳定的，当且仅当对于其上的每一个珠缀，它下方所有珠缀的重量和（不包含自身）不超过其挂钩的承受能力。

小Q 希望她的挂缀尽量长，你能帮她计算出最长可能的稳定挂缀么？当然，如果有多个可选解，小Q 希望总重量最小的。

【输入文件】

输入文件pendant.in 第一行包含一个正整数N，表示商店拥有的珠缀数目。

接下来N 行，每行两个整数(C_i , W_i)，分别表示第i 个珠缀的承受能力与重量。

【输出文件】

输出文件pendant.out 包行两行。第一行包含一个整数L，表示可以找到的最长稳定挂缀长度。第二行包含一个整数W，表示可以找到的长度为L 的稳定挂缀中的最小重量和。

【样例输入】

4

3 5

5 1

3 2

4 6

【样例输出】

3

8

【评分标准】

每一个测试点单独评分，对于每一个测试点：

如果挂缀长度L 与答案一致，且最小重量和W 也正确，则得10 分。

如果挂缀长度L 与答案一致，而最小重量和W 与答案不一致，该测试点则得4 分。

否则得0 分。

【数据规模】

对于30%的数据，N ≤ 10000；

对于100%的数据，N ≤ 200000；

对于所有的数据，W_i, C_i 不超过 2^{31}。

（例题 3.1）

Europe - Southeastern 2007 Problem D

Loan Scheduling

The North Pole Beach Bank has to decide upon a set App of mortgage applications. Each application a ∈App has an acceptance deadline d_a , ie. the required loan must be paid at a time t_a , $0 \leqslant t_a \leqslant d_a$. If the application is accepted the Bank gets a profit p_a . Time is measured in integral units starting from the conventional time origin 0, when the Bank decides upon all the App applications. Moreover, the Bank can pay a maximum number of L loans at any given time. The Bank policy if focussed solely on profit: it accepts a subset S ∈App of applications that maximizes the profit $profit(S) = \sum_{a \in S} p_a$. The problem is to compute the maximum profit the Bank can get from the given set App of mortgage applications.

For example, consider that L = 1 , App = {a, b, c, d} , (p_a, d_a) = (4, 2) , (p_b, d_b) = (1, 0) , (p_c, d_c) = (2, 0) , and (p_d, d_d) = (3, 1) . The table below shows all possible sets of accepted mortgage applications and the

scheduling of the loan payments. The highest profit is 9 and corresponds to the set {c, d, a} . The loan requested by the application c is paid at time 0, the loan corresponding to d is paid at time 1, and, finally, the loan of a is paid at time 2.

Time	Sets of accepted applications and loan scheduling																	
0	a			b	c	d		b	c	b	B	c	c	d	d		a	b	c
1		a					d	d	d	a		a		a		d	d	d	d
2			a								A		a		a	a		a	a
Profit	4	4	4	1	2	3	3	4	5	5	5	6	6	7	7	7	7	8	9

Write a program that reads sets of data from the standard input.

Input

Each data set corresponds to a set of mortgage applications and starts with two integers: $0 \leqslant N \leqslant 10000$ that shows the number of applications in the set, and $0 \leqslant L \leqslant 100$ which shows the maximum number of loans the Bank can pay at any given time. Follow N pairs of integers p_i d_i , i = 1, N , that specify the profit $0 \leqslant p_i \leqslant 10000$ and the deadline $0 \leqslant d_i \leqslant 10000$ of the application i.

Input data are separated by white spaces, are correct, and terminate with an end of file.

Output

For each data set the program computes the maximum profit the Bank can get from the accepted mortgage applications corresponding to that data set. The result is printed on standard output from the beginning of a line. There must be no empty lines on output. An example of input/output is shown below.

Sample Input

4 1 4 2 1 0 2 0 3 1

7 2

200 1 200 1 100 0 1000 2 80 1

50 20 500 1

0 100

1 0 4 1000

Sample Output

9

20 50

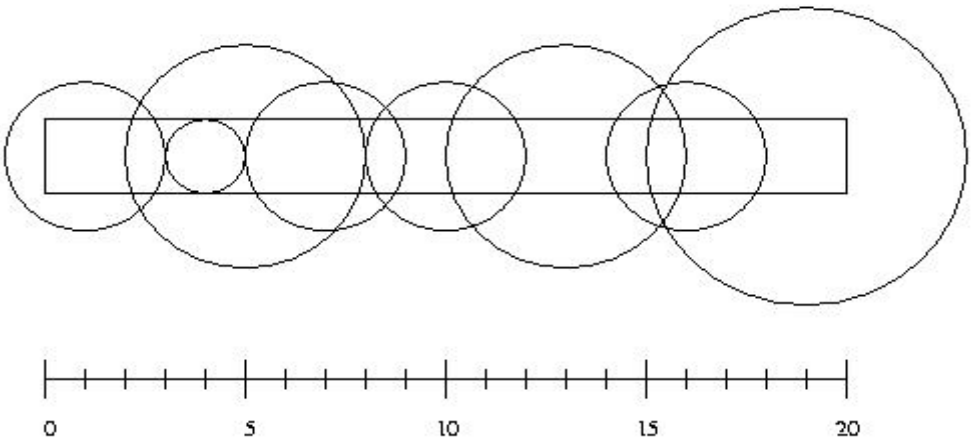
0

0

(例题 3.2)

ACM/ICPC Regional Warm-up Contest 2002 Problem E

n sprinklers are installed in a horizontal strip of grass l meters long and w meters wide. Each sprinkler is installed at the horizontal center line of the strip. For each sprinkler we are given its position as the distance from the left end of the center line and its radius of operation.



What is the minimum number of sprinklers to turn on in order to water

the entire strip of grass?

Input

Input consists of a number of cases. The first line for each case contains integer numbers n, l and w with n <= 10000. The next n lines contain two integers giving the position of a sprinkler and its radius of operation. (The picture above illustrates the first case from the sample input.)

Output

For each test case output the minimum number of sprinklers needed to water the entire strip of grass. If it is impossible to water the entire strip output -1.

Sample input

```
8 20 2
5 3
4 1
1 2
7 2
10 2
13 3
16 2
19 4
3 10 1
3 5
9 3
6 1
3 10 1
5 3
1 1
9 1
```

Sample Output

6

2

-1

(例题 4)

Europe - Northeastern Europe 2004 Problem J

Joke with Turtles

There is a famous joke-riddle for children:

Three turtles are crawling along a road. One turtle says: “There are two turtles ahead of me.” The other turtle says: “There are two turtles behind me.” The third turtle says: “There are two turtles ahead of me and two turtles behind me.” How could this have happened? The answer is -- the third turtle is lying!

Now in this problem you have n turtles crawling along a road. Some of them are crawling in a group, so that they do not see members of their group neither ahead nor behind them. Each turtle makes a statement of the form: “There are a_i turtles crawling ahead of me and b_i turtles crawling behind me.” Your task is to find the minimal number of turtles that must be lying. Let us formalize this task. Turtle i has x_i coordinate. Some turtles may have the same coordinate. Turtle i tells the truth if and only if a_i is the number of turtles such that $x_j > x_i$ and b_i is the number of turtles such that $x_j < x_i$. Otherwise, turtle i is lying.

Input

The input contains several test cases. The first line of each case consists of a integer number n ($1 \leqslant n \leqslant 1000$). It is followed by n lines containing numbers a_i and b_i ($0 \leqslant a_i, b_i \leqslant 1000$) that describe statements of each turtle for i from 1 to n.

Output

For each input case, print one output line containing an integer number m -- the minimal number of turtles that must be lying, followed by m integers -- turtles that are lying. Turtles can be printed in any order. If there are different sets of m lying turtles, then print any of them.

Sample Input

```
3
2 0
0 2
2 2
5
0 2
0 3
2 1
1 2
4 0
```

Sample Output

```
1 3
2 1 4
```

(例题 5.1)

USACO 2005 dec silver

Cleaning Shifts [Coaches, 2004]

Farmer John's cows, pampered since birth, have reached new heights of fastidiousness. They now require their barn to be immaculate. Farmer John, the most obliging of farmers, has no choice but hire some of the cows to clean the barn.

Farmer John has N (1 <= N <= 10,000) cows who are willing to do some cleaning. Because dust falls continuously, the cows require that the farm be continuously cleaned during the workday, which runs from second number

M to second number E during the day (0 <= M <= E <= 86,399). Note that the total number of seconds during which cleaning is to take place is E-M+1. During any given second M..E, at least one cow must be cleaning.

Each cow has submitted a job application indicating her willingness to work during a certain interval T1..T2 (where M <= T1 <= T2 <= E) for a certain salary of S (where 0 <= S <= 500,000). Note that a cow who indicated the interval 10..20 would work for 11 seconds, not 10. Farmer John must either accept or reject each individual application; he may NOT ask a cow to work only a fraction of the time it indicated and receive a corresponding fraction of the salary.

Find a schedule in which every second of the workday is covered by at least one cow and which minimizes the total salary that goes to the cows.

TIME LIMIT: 0.2 seconds

PROBLEM NAME: clean

INPUT FORMAT:

Line 1: Three space-separated integers: N, M, and E.

Lines 2..N+1: Line i+1 describes cow i's schedule with three space-separated integers: T1, T2, and S.

SAMPLE INPUT (file clean.in):

```
3 0 4
0 2 3
3 4 2
0 0 1
```

INPUT DETAILS:

FJ has three cows, and the barn needs to be cleaned from second 0 to second 4. The first cow is willing to work during seconds 0, 1, and 2 for a total salary of 3, etc.

OUTPUT FORMAT:

Line 1: a single integer that is either the minimum total salary to get the barn cleaned or else -1 if it is impossible to clean the barn.

SAMPLE OUTPUT (file clean.out):

5

OUTPUT DETAILS:

Farmer John can hire the first two cows.

(例题 5.2)

CEOI 2004 trips

Trips

Description

In the forthcoming holiday season, a lot of people would like to go for an unforgettable travel. To mostly enjoy their journey, everyone wants to go with a group of friends. A travel agency offers several trips. A travel agency offers group trips, but for each trip, the size of the group is limited: the minimum and maximum number of persons are given. Every group can choose only one trip. Moreover, each trip can be chosen by only one group. The travel agency has asked you for help. They would like to organize as many trips as possible. Your task is to match groups of people and trips in such a way, that the maximum number of trips can be organized.

Task

Write a program TRIPS, that:

reads the description of the groups and the trips from the standard input matches the groups and trips in such a way, that the maximum number of arranged trips is reached writes the result to standard output. If there are several possible solutions, your program should output anyone of them.

Characteristics

Available memory: 64Mb

Maximum running time: 3s

Input file

The first line of input file TRIPS.IN contains two integers: n and m separated by single space, 1<=n<=400000, 1<=m<=400000; n is the number of groups and m is the number of trips. The groups are numbered from 1 to n, and the trips are numbered from 1 to m.

The following n lines contain group sizes, one per line. Line i+1 contains integer si - the size of the i-th group, $1<=s<=10^9$. The following m lines contain trip descriptions, one trip per line. Line n+j+1 contains two integers: lj and uj, separated by single space. lj is the minimum, and uj is the maximum size of a group for which the trip can be arranged ($1<=l_j<=u_j<=10^9$)

Output file

The first line of output file TRIPS.OUT should contain one integer k>=0 - the maximum number of trips that can be arranged. The following k lines should contain the description of the matching. Each of these lines should contain a pair of integers separated by single space: the number of a group and the number of a trip. There can be many answers and your program may print anyone of them.

Example

TRIPS.IN

```
5  4
54
6
9
42
15
6  6
20  50
2  8
```

7 20

TRIPS.OUT

3

2 1

3 4

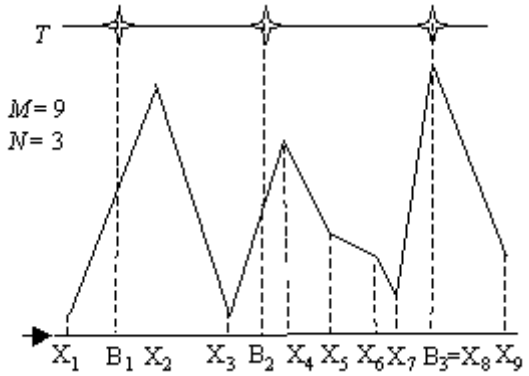
4 2

(例题 6.1)

CEOI 2000 Enlightened Landscape

Enlightened landscape

Consider a landscape composed of connected line segments:



Above the landscape, N light bulbs are hang at the same height T in various horizontal positions. The purpose of these light bulbs is to light up the entire landscape. A landscape point is considered lit if it can "see" a light bulb directly, that is, if the line segment which links the point with a bulb does not contain any other landscape segments point. Write a program that determines the minimum number of light bulbs that must be switched on in order to illuminate the entire landscape.

Input

Input file name: LIGHT.IN

Line 1: M (an integer, the number of landscape height specifications, including the first and the last point of the landscape)

Lines 2..M+1: X_i H_i (two integers, separated by a space: the landscape height H_i at horizontal position X_i, $1<=i<=M$; for $1<=i<=M-1$ we have $X_{i+1}>X_i$; any two consecutive specified points identify a segment of line in the landscape)

Line M+2: N T (two integers, separated by a space, the number of light bulbs and their height coordinate (altitude); the bulbs are numbered from 1 to N)

Line M+3: B_1 B_2 ... B_N (N integers, separated by spaces: the horizontal coordinates of the light bulbs $B_{i+1}>B_i$, $1<=i<=N-1$)

Output

File name: LIGHT.OUT

Line 1: K (an integer: the minimum number of light bulbs to be switched on)

Line 2: L_1 L_2 ... L_K (K integers, separated by spaces: the labels of the light bulbs to be switched on specified in increasing order of their horizontal coordinates)

Limits

$1<=M<=200$

$1<=N<=200$

$1<=X_i<=10000$ for $1<=i<=M$

$1<T<=10000$

$1<=H_i<=10000$ for $1<=i<=M$

$T>H_i$ for any $1<=i<=M$

$X_1<=B_1$ and $B_N<=X_M$

The task always has a solution for the test data. If there are multiple solutions, only one is required.

Example

LIGHT.IN

6

1 1

3 3

4 1

7 1

8 3

11 1

4 5

1 5 6 10

LIGHT.OUT

2

1 4

(例题 6.2)