IOI 2014 解题报告

徐寅展 俞鼎力 董宏华 沈 洋

2014 年 7 月 28 日

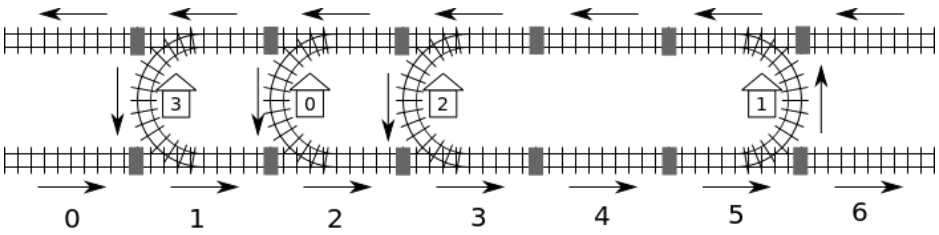
Contents

Chapter 1

Day 1

1.1 Day 1 Rail

1.1.1 题目大意



有两条平行的单向铁路（上方的从右到左，下方的从左到右），分为 m 段。有 n 个车站，每个车站为 C 类型（只能从上往下）或 D 类型（只能从下往上），分布在某些段中，每个段最多一个车站。

已知0号车站是 C 类型，并给出0号车站的位置，最多可以询问两车站之间的距离 $3(n-1)$ 次（距离指经过段与段连接处的次数，例如上图0号车站到2号车站的距离为5），要求确定每个车站的位置和类型。保证车站两两可达。

1.1.2 算法讨论

先询问得到0号车站到其他车站的距离，而最近的一个，就是0号车站右侧第一个 D 类型的（称之为 j 号车站）。

然后询问得到j号车站到其他车站的距离，其中最近的一个，可能是0号车站，也可能是其他车站（都称之为 k 号车站），显然 j 和 k 之间不会再有其他车

站，而0和 k 之间也不会有其他的 D 类型的车站，所有 k 号车站到其他车站的距离可直接算出。

有了 j 和 k 到其他车站的距离，那就可以轻松分出左右了（离 j 号近，就在 k 的左侧，否则在 j 的右侧）。

但分出左右后还是不能确定具体位置，而这时对于每个车站我们还留下一次询问的机会。接下来称当前车站为 i 号车站。

而这次询问一定是留给特殊位置的车站，假设当前车站在左侧，则考虑当前确定的最左侧的车站（称之为 L 号）。

按离 j (或 k)号车站的距离从近到远的顺序处理剩下的车站，那么只有这两种情况：

```
(   (   (   )
i   L   k   j
```

以及（注意下面这种L和i之间还会有C类型的车站）

```
(   )   (   )
L   i   k   j
```

两者都会有以下关系式： $dist(j, L) + |pos_j - pos_L| = dist(j, i) + x(x \geq 0)$

第一种情况多出来的是L到它右侧第一个D类型车站的距离 $\times 2$ ，而第二种情况多出来的是L到它右侧第一个C类型车站的距离 $\times 2$ 。

所以，算出 x 之后，只要到L右侧的 $x/2$ 的距离处看下车站的类型就可以确定位置了。这样问题就解决了。

如果当前车站在右侧，那么询问与已确定的最右侧车站的距离，类似讨论即可。

1.2 Day 1 Wall

1.2.1 题目大意

维护一个长度为n的整数序列，一开始每个元素均为0，支持以下两种操作：

- 将连续一段中小于k的元素修改为k
- 将连续一段中大于k的元素修改为k

问所有m个操作进行完之后序列各元素的值。

1.2.2 算法讨论

不难发现对某一个元素的操作是可加的，即说对于某一个元素来说，应用在其上的每一个操作可以都表示为“如果它的初值小于l，那么最终它等于l；如果它的初值大于r，那么最终它等于r；否则它最终等于初值”这样的形式，并且多个这样的形式是可以合并的。于是我们可以把每个操作都看成一个值，这样原问题就转化成“维护一个序列，每次对一段区间加上一个值，问最后每个元素的值”。这是可以用带标记的线段树直接维护的。该算法的时间复杂度为$O(n + m \log n)$。

对于“维护一个序列，每次对一段区间加上一个值，问最后每个元素的值”这个问题，我们也可以使用扫描线进行维护。但本题中的值是不可减也不满足交换律的，因此在扫描过程中我们需要使用一个线段树来维护覆盖到当前点的值并将它们按时间顺序依次求和。该算法的时间复杂度为$O(n + m \log m)$。

1.3 Day 1 Game

1.3.1 题目大意

有一张n个点的无向图，小B每次会询问某两个点之间是否有边相连，小A每次回答yes或no。如果在小B把所有$\frac{n(n-1)}{2}$条边问完之前，小B就能确定这整张图是否联通，小A就输了。现在让你当小A，依次对每个询问回答yes或no，求一种获胜方案。$1 \leq n \leq 1500$。

1.3.2 算法讨论

考虑到，如果在一个已经联通（这里指的是通过已经回答过yes的那些边而联通）的联通块中，还存在一些没有询问的边，那么小B总可以把这些边留到最后问，小A肯定输了。如果小A 能每时每刻保证已经联通的联通块中，已经没有还没询问过的边了，那么小A就肯定获胜了。

那么每次小B询问一条边(x, y)，如果此时x和y所在的联通块之间只有一条边了，就回答yes；否则回答no。这样就可以保证上述那个性质了。维护两个联通快之间的边数可以使用并查集。时间复杂度$O(n^2)$。

Chapter 2

Day 2

2.1 Day 2 Gondola

2.1.1 题目大意

初始有一个序列，由 $1 \sim n$ 循环位移得到，即可能为 $(i, \cdots, n, 1, \cdots, i-1)$，$i$ 是 1 到 n 范围内的任意一个数字。

之后有若干操作，每次操作时，首先找到当前最小的还未用过的编号 id，将序列中的某个数字替换为 id。

定义替换序列为每次操作中被替换的数字组成的序列。

要回答三个问题：

1. 一个操作完的序列是否合法？
2. 构造一个可能的替换序列。
3. 可能的替换序列的个数 $(\text{mod } 1,000,000,009)$.

2.1.2 数据范围

前三个子任务只需回答第一个问题：

子任务	分值	n	`inputSeq`
1	5	$n \leq 100$	从 1 到 n 的数字恰好出现一次
2	5	$n \leq 100,000$	$1 \leq$ `inputSeq[i]` $\leq n$
3	10	$n \leq 100,000$	$1 \leq$ `inputSeq[i]` $\leq 250,000$

接下来的三个子任务只需回答第二个问题：

子任务	分值	n	`gondolaSeq`
4	5	$n \leq 100$	$1 \leq$ `gondolaSeq[i]` $\leq n+1$
5	10	$n \leq 1,000$	$1 \leq$ `gondolaSeq[i]` $\leq 5,000$
6	20	$n \leq 100,000$	$1 \leq$ `gondolaSeq[i]` $\leq 250,000$

接下来的四个子任务只需要回答第三个问题：

子任务	分值	n	`inputSeq`
7	5	$4 \leq n \leq 50$	$1 \leq$ `inputSeq[i]` $\leq n+3$
8	15	$4 \leq n \leq 50$	$1 \leq$ `inputSeq[i]` ≤ 100，$1 \sim n$ 中至少有 $n-3$ 个出现在操作完的序列中。
9	15	$n \leq 100,000$	$1 \leq$ `inputSeq[i]` $\leq 250,000$
10	10	$n \leq 100,000$	$1 \leq$ `inputSeq[i]` $\leq 1,000,000,000$

2.1.3 算法讨论

第一问

如果存在 $1 \sim n$ 中任意一个，那么先检查相对位置是否正确。

检查是否所有数字互不相同。

第二问

如果存在 $1 \sim n$ 中任意一个，那么可以确定最开始的序列；否则任选一个初始序列。

从小到大枚举之后放进去的数字，如果出现在最终序列中，那么放在该位置，否则放在任意一个非确定的位置。

第三问

从之前的构造可以得到计数的方法：

首先用第一问检查是否合法。

如果存在 $1 \sim n$ 中任意一个，那么可以确定最开始的序列；否则任选一个初始序列，亦即最后答案会乘以 n.

对于子任务 $7 \sim 9$，依然可以枚举放进去的数字，如果不出现在最终序列中，那么答案乘以非确定的位置的个数。

对于最后一个子任务，可以对最终序列中的数字排序，分段计算即可。

2.1.4 时空复杂度

时间复杂度：$O(n \log n)$。
空间复杂度：$O(n)$。

2.2 Day 2 Friend

2.2.1 题目大意

有一个点带权的无向图，最开始只有点0，随后点1至点$n-1$依次加入。点i加入时，会有一个已经加入的点作为它的$host$，记为$host_i$，它会在点i和其他一些已经加入的点之间连边。具体连边方式有以下三种：

- I方式：只将i与$host_i$连边。
- M方式：只将i与$host_i$的所有邻居连边（$host_i$本身不与i连边）。
- W方式：将i与$host_i$及其所有邻居连边。

现在已知每个点的点权，$host$以及连边方式，求该图的最大点权独立集。

2.2.2 算法讨论

注意到如果将$host_i$当做点i的父亲，我们就能得到一棵以点0为根的树，其中每个孩子节点根据相应的点的连边方式不同可以分为I、M和W三种类型。考虑树形DP。令$f(i)$表示点i可以被选的情况下以i为根的子树的最大权值，$g(i)$表示点i不能被选的情况下以i为根的子树的最大权值（这里的能否被选是在不考虑以i为根的子树的情况下）。

首先考虑$g(i)$的计算。因为点i是不能选的，所以点i的所有M类和W类孩子都是不能选的，I类孩子是可以选的。因此

$$g(i) = \sum_u g(u) + \sum_v f(v)$$

其中u是i的M类或W类儿子，v是i的I类孩子。

$f(i)$的计算分两种情况。第一种情况，我们选择了点i，那么点i的所有I类和W类孩子就不能选了，而M类孩子仍是可以选的，所以在这种情况下

$$f(i) = \sum_u g(u) + \sum_v f(v)$$

其中u是i的I类或W类孩子，v是i的M类孩子。第二种情况，也就是不选择点i的情况稍微复杂一些，我们需要决定i的每个孩子u是否能够被选择，唯一的限制是，一旦我们决定了一个I类或W类儿子是可以选择的，那么在它之后加入（编号比它大）的M类孩子和W类孩子就是不能选择的了。我们可以对i的所有孩子进行一个简单的DP来作出最优决定，再加上i点本身的权值即可得到这种情况下的$f(i)$。最后，在两种情况下得到的$f(i)$中取较大的一个作为最后的$f(i)$即可。

最终，$f(0)$即为答案。该算法的时间复杂度为$O(n)$。

2.3 Day 2 Holiday

2.3.1 题目大意

n个城市依次排开，编号为0到$n-1$。i号城市与$i-1$号城市和$i+1$号城市相邻。一开始你在$start$号城市。每一个时刻，你可以选择移动到相邻的一个城市，或者游玩当前城市，并获得a_i的娱乐值，其中i是你现在所在的城市编号。你不能游玩同一个城市多次，但是能多次经过同一个城市。问d天你能获得的最大愉悦值是多少。$1 \leq n \leq 100000$。

2.3.2 算法讨论

起点在0号城市

可以发现，在这种情况下，只会一直往右移动，而不会回头。因此可以枚举往右走到的最右边那个城市，然后再从剩余的天数中，选择愉悦值最大的若干个城市进行游览。后者可以用可持久化线段树实现。

只往右走，对每个d求得答案

令f_d为可以游览d天，最优的那个右端点。我们有如果$x < y$，那么$f_x \leq f_y$.

考虑f_x与f_{x+1}，假设$f_{x+1} < f_x$。先把能游览$x+1$天的右端点与游览x天的右端点放在f_x处。右端点每次向左移动一格，游览x天的与游览$x+1$天的，都能游览一个新的城市（当最右端的那个城市本来也被选时，是两个新的城市）。但由于$x+1$天的本来就比x天的游览了更多的城市，所以这个x天的新城市的愉悦值要大于等于$x+1$ 天的新城市的愉悦值。当它们移动到f_{x+1}时，x天的增加的愉悦值大于等于$x+1$天的增加的愉悦值。于是就矛盾了。因此有$f_x \leq f_{x+1}$。

接下去考虑分治，要求d在$[l,r]$中的所有f_d,并且知道这些f_d在$[a,b]$中。令$mid = \lfloor \frac{l+r}{2} \rfloor$, 首先暴力找到$f_{mid}$,接下去递归分别找$f_l \dots f_{mid-1}$,$f_{mid+1} \dots f_r$。其中，$f_l \dots f_{mid-1}$都在$[a, f_{mid}]$之间；$f_{mid+1} \dots f_r$都在$[f_{mid}, b]$之间。这样总的时间复杂度是$O(n \log^2 n)$的。

原问题

最优策略肯定是往右走再折回左边，或者往左走再折回右边。既然可以对每一个d都求出只在左边或者只在右边的答案，也可以求出对于每一个d往一个方向走，再折回$start$的答案（具有类似上文中的单调性），那么只要求出两边分别对于所有d的答案后，就能求出整个问题的最优解了。