NOIP 2011 提高组题解

杭州外国语学校　陈立杰

November 13, 2011

Contents

1 Day I

1.1 carpet

由于只需要询问一个点，我们只需要从头到尾扫描一遍，记录最后一个覆盖该点的矩形即可。

1.2 hotel

对于50%的数据，我们可以预处理出每两个点之间最小的咖啡屋价格$minCost_{ij}$，然后n^2枚举判断即可。

首先我们可以对题目进行一个转化，令c_i在$cost_i > p$时等于1,否则等于0，那么题目实际上就等价于求出颜色一样，之间有一个c是0的咖啡屋对数。那么可以倒过来，来考虑计算颜色一样，之间的c全是1的对数，说白了这时这两个咖啡屋必须都位于一个c全是1的连续块中。我们可以扫描得到每个c全是1的连续快，对一个长度为L的块，显然可以在$O(L)$的时间算出结果。那么总时间复杂度就是$O(N)$

1.3 manya

这是一道搜索题。

首先可以注意到，一个块往左交换，跟它左边的块往右交换是等价的，而此时后者者字典序更优所以不用考虑前者，那么一次的决策最多$4 * 7 = 28$个，$n = 51/28^5 \approx 1.7 * 10^7$。

同时可以考虑优化，先是可行性剪枝，如果有一种颜色的数量在$1 \sim 2$之间，那么显然这种颜色不能被消完，则可以剪枝。

然后是最优性剪枝，注意到虽然掉落这个现象很复杂，但他并不能在横向上移动，也就是说，横向上的移动，必须要通过交换来实现。

那么考虑一个颜色$color$。如果某一列该颜色有$1 \sim 2$个，那么它自己当然是无法自我消灭的，所以肯定要通过横向移动来得到其它列的“帮助”，比如离他隔的最近的且有该颜色的列。

那么我们可以取一个列，他只有$1 \sim 2$个颜色$color$，且他离最近的可以"帮助"它的列最远，设这个距离-1为D_{color}，那么我们的操作至少要把所有颜色的D值消到0。一次操作最多让两个D减1。所以需要的最少操作次数就是

$$Max(Max_{i=1}^{10} D_i, \frac{Sum_{i=1}^{10} D_i + 1}{2}) \tag{1}$$

考虑了剪枝之后，我们还可以判重。同时由于如果只是将颜色的名字交换一下，那么两个状态本质上还是一样的，我们可以将颜色用最小表示法来表示。

那么基本上有解的状态都能在3s之内输出，如果无解的话由于节点可能很多卡时就行了。

2 Day II

2.1 factor

简单的数学计算可知答案为

$$C_k^n a^n b^m \tag{2}$$

使用递推算出C_k^n之后乘上另两项即可。

2.2 qc

首先注意到随着效验值W不断变大，所有$Sum_{i=1}^m Y_i$只会越来越小，那么我们可以二分W，求出最小的那个使得$Sum_{i=1}^m Y_i \le S$的效验值w，那么只需要检查w和$w-1$即可。

考虑确定了效验值W之后，如何计算$Sum_{i=1}^m Y_i$，我们只要预处理两个数组，cnt_i表示$1 \sim i$这些矿石中，$w_i >= W$的有几个，sum_i表示示$1 \sim i$这些矿石中，$w_i >= W$的矿石的v_i的和。就能求出所有Y_i。cnt, sum都只需简单的计算即可。

2.3 bus

首先考虑如何计算结果，我们令$maxArrive_i$表示从i点出发的人，T_i的最大值，也就是最迟的到达时间，没人从i出发则为0。再令$enter_i$表示到达i点的最早时间，那么

$$enter_i = Max(enter_{i-1}, maxArrive_{i-1}) + D_{i-1} \tag{3}$$

那么对于从A_i点到B_i点的旅客i，他用的时间为

$$enter_{B_i} - T_i \tag{4}$$

那么我们消耗的总时间就是

$$Sum_{i=1}^m (enter_{B_i} - T_i) = (Sum_{i=1}^m enter_{B_i}) - (Sum_{i=1}^m T_i) \tag{5}$$

由于减去的是一个常量，所以我们只要得到$Sum_{i=1}^m enter_{B_i}$的最小值即可。
设第i个点有cnt_i个人以该点为终点。那么答案也就是

$$Sum_{i=1}^n cnt_i * enter_i \tag{6}$$

接下来我们进一步观察$enter$的计算方法，可以发现3式等价于

$$enter_i = Max_{j=1}^{i-1}(maxArrive_j + S_i - S_j) \tag{7}$$

其中S_i表示1点到i点的距离。我们令$V_i = maxArrive_i - S_i$ 那么

$$enter_i = Max_{j=1}^{i-1}(V_j) + S_i \tag{8}$$

那么考虑减少一个D_i，会产生什么影响。

可以发现所有的$j \le i$的点的S, V都没有变化，所有$j > i$的点,$S_j = S_j - 1, V_j = V_j + 1$

由于每个点的$enter_i$值都取决于前面最大的V值，如果有多个最大的话，不妨让它被最后一个决定。那么考虑一个点的V值，他假如要决定后面一些$enter$值的话，必须满足之前的所有V值都不比他大。

我们可以令决定了后面的一些$enter$值的点分别为$a_0, a_1, a_2 ... a_{p-1}$，那么显然他们的$V$值也是单调不减的(由于条件)。同时易知每个$a_i$必然决定了连续的一个子序列。也就是讲整个序列分成了一块一块，每块都有一个v_{a_i}决定。

a_i决定了$a_i + 1 \sim a_{i+1}$这一段的$enter$值。

我们可以发现，假如我们减少$a_i \sim a_{i+1} - 1$这些点中一个的D值，那么a_i前面的块完全不受影响，a_{i+1}和之后的块的点$x, Max_{j=1}^{x-1}(V_j)+$了1，但是S_x-了1，所以$enter$值不变，同时也没有任何其它的影响。所以每个块之间是独立的。

考虑块$[l, r]$，被$[l, r]$之前一个数决定，那么我们将$D_{l-1}-1$，可以将结果减少$Sum_{i=l}^{i=r} cnt_i$，同时由于将$[l, r]$中的数的V值变大了，$[l, r]$可能被分成几块，那样之后的操作就比之前的操作烂了(因为每次只能减少一个块的$Sum_{i=l}^{i=r} cnt_i$，而块变小了)，也就是说对于每个块来说，一次操作能取得的总时间减少量只会越来越少。

那么考虑当前能节省最大时间的块，如果不取他，那么由于之后的操作都比他要劣，所以随便把之后的一个操作换成他就能更优，而那样就不是最优解了。所以一定要取他。

那么只需要开一个堆，每次从堆里取出能减少最大时间的那个块$[l, r]$，消耗k将D_{l-1}减小，那么有3种情况。

(1): 要么D_{l-1}被减成0，那么就只能减D_l了，将$[l+1, r]$放进堆

(2): 要么k被减完，结束。

(3): 要么减到一半时，[l,r]中的一个数变成了新的可以决定后面$enter$值的数，那么$[l, r]$分裂。

(1)和(3)都最多出现n次，(2)最多出现1次，每次操作复杂度O($\log_n$)。所以总复杂度

$$O(m + n \log_n) \tag{9}$$

欢迎鄙视~~~