BestCoder Blog

BestCoder 官方博客

2016 Multi-University Training Contest 3 solutions BY 绍兴一中

1001 Sqrt Bo

由于有5次的这个限制，所以尝试寻找分界点。

很容易发现是2^{32}，所以我们先比较输入的数字是否比这个大，然后再暴力开根。

复杂度是$O(\log \log n)$。

注意特判$n = 0$的情况。

1002 Permutation Bo

根据期望的线性性，我们可以分开考虑每个位置对答案的贡献。

可以发现当i不在两边的时候和两端有六种大小关系，其中有两种是对答案有贡献的。

那么对答案的贡献就是$\frac{c_i}{3}$。

在两端的话有两种大小关系，其中有一种对答案有贡献。

那么对答案的贡献就是$\frac{c_i}{2}$。

复杂度是$O(n)$。

注意特判$n = 1$的情况。

1003 Life Winner Bo

我们依次分析每一种棋子。

①王。

首先注意一个3*3的棋盘，开始在(1,1)，问走到(3,3)谁有必胜策略。

穷举所有情况，容易发现这是后手赢。

对于N和M更大的情况，我们把横坐标每隔3、纵坐标每隔3的点都画出来，这些点都是符合后手胜的。
（因为无论先手怎么移动，后手都能重新移动到这些格子，直到到了终点）

如果初始点不在这些点上，就必然是先手胜。因为先手可以立刻移动到上述的点。

②车。

注意到，如果目前的位置距离终点的x和y坐标差相等，一定是后手胜。
（因为先手只能向下或者向右走一段路；无论他往哪里走，后手往另一维走相同的步数，依然保持这一样一种状态。）

反之，先手必然能走到一处相等的位置，转化为上述问题，所以一定是先手胜。

③马。

同样还是画图可以得到规律。

在大多数情况下都是平局。在模3域下，某些地方会存在先后手赢。

④皇后。

画画图后，我们可以将问题转化为：

“有两堆石子，每次可以在一堆里取任意（非空）颗（相当于是车的走法），或者在两堆里取相同（非空）颗（相当于是象的走法），取到最后一颗石子的人获胜，问先后手谁有必胜策略。”

此题中$N \leq 1000$，可以直接用DP的方法解决。
设f[x][y]为横坐标距离终点x步，纵坐标距离终点y步时，必胜的是先手还是后手。

直接转移的话，可以枚举先手的下一步决策进行转移，这样是$O(N^3)$的。

注意到转移只是一行、一列或者斜着一列，这些都可以通过前缀和，做到最终$O(N^2)$。

其实对于更大的N也是可以做的。

由于叙述起来比较麻烦，具体的结论和证明可以参见：

https://en.wikipedia.org/wiki/Wythoff\%27s_game

1004 Gambler Bo

问题可以转化为一个模3域下的方程，对每个位置可以列出一个方程，即覆盖到这个位置的操作造成的影响和要使得这个位置变为0。

可以模仿解xor方程组时的线性基算法，维护线性无关组来计算，并方便地构造方案。

时间复杂度$O(N^3M^3)$，当然直接解方程复杂度也是一样的。

1005 Boss Bo

我们可以用主席树对于所有点，维护出他到所有点的距离。

大致过程就是，对于一号点，他到所有点的距离为该点的深度减一。
接着我们~DFS~整棵树，每到一个节点，就将子树中的点距离减一，不是子树的点距离加一。
我们可以得到每一个点的~DFS~序，这样只需要用主席树支持区间加减，维护区间和、区间最小值、区间最大值即可。

对于每次询问，所给的~A_i~都对应一些~DFS~序区间，我们把这些区间取并，然后取补集。
这样整个~DFS~序会有~K~段区间是合法的。
我们对于每一段区间进行询问即可。

效率：$O((N+Q+\sum K)\log N)$。

因为空间和时间的限制，线段树的时候不能写~$Down$~，需要打静态标记。

1006 Product Bo

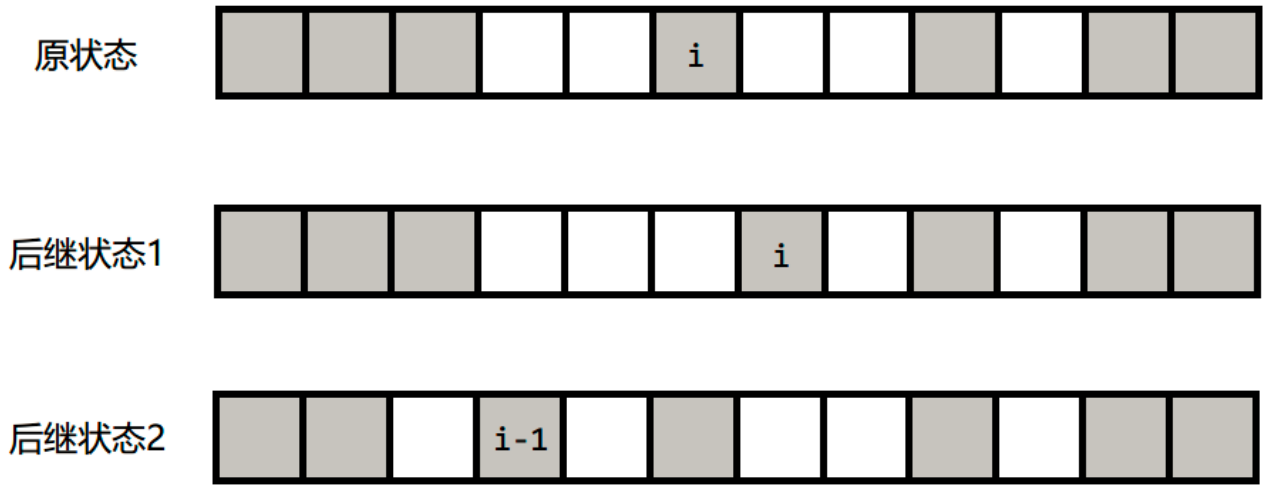
先考虑没有负数的情况。这种情况下，我们不妨把所有数从大到小排序，这样不影响最终结果。我们还可以把0当成是$-\infty$的正数。
对于第K大问题，可以想到的一种方法是：逐步确定子序列，用优先队列维护估价。具体地来说是这样的：（你可以选择跳过这段关于这种做法的描述）

对于每个子序列，我们依次逐步确定它的每个数的位置，即先确定第1个数的位置，再确定第2个数的位置……直到确定了第M个数的位置，就得到了一个子序列。不妨将逐步确定过程中的状态称为中间状态。对于一个中间状态s，不妨设它已经确定了前i个数的位置，我们可以定义它的估价$H(s)$为把这个中间状态按最优的方式补完整之后

所有数的乘积。最优的补完整方式显然就是取第i个位置之后连续的$M-i$个数。现在，我们把所有中间状态按估价H的大小放到一个优先队列中。每次取出优先队列中估价最大的状态。如果这是一个完整的状态，就表示是下一个次优的状态；如果这是一个中间状态，就按某种方式选择下一个位置，扩展出几个后继状态放到优先队列里。状态只要这样扩展：每次扩展出2个后继状态，一个是把当前第i个位置往后移一位，另一个是固定下第i个位置，从后面一位开始考虑第$i+1$个位置。当然这里还有另一种扩展方法，就是枚举i后面第一个与前面不连续的位置，这样每一个中间状态同时也是一个完整的状态，只是每次扩展的后继状态数是$O(M)$的，最终复杂度是一样的，这里暂且不论。

至于求连续的若干个数的乘积，可以预处理前缀积以及前缀积的逆，在这里也就是对数的前缀和，然后$O(1)$计算。
让我们来计算这个做法的时间复杂度。每个中间状态只有$O(1)$个后继状态，因此取出一个状态并放入后继状态的复杂度是$O(\log$状态数$)$的。由于估价是准确的，获得一个完整的状态最多需要确定M步，也就是说总共有$O(KM)$个状态。因此总复杂度为$O(KM\log(KM))$。 很遗憾不能通过这题的数据。

我们发现，之所以每获得一个完整的状态需要确定M步，是因为从i变成$i+1$时状态本质没有变化（或者说对于另一种扩展方法来说，需要扩展太多无用的后继状态）。有没有什么办法能使得优先队列中每个状态都是完整的状态而且扩展的后继状态只有$O(1)$个呢？确实是有的。我们可以这样来确定一个子序列：先从第1大的状态开始，也就是从位置1开始连续的M个数。对每个状态记录从位置1开始连续有多少个数，不妨设为$i-1$。那么对于某个状态，有两个后继状态：第一个是把第i个数的位置往后挪一位（注意如果往后挪碰到后面一个数了就不能挪，所以要记一下后面一个数的位置）；第二个是把第$i-1$个数的位置往后挪一位。至于这样做的正确性，你可以想象每个状态都用这种方法逐步确定，显然确定的方法是存在且唯一的，并且每个后继状态都不比当前状态更优。至于这些状态估价的计算，显然可以预处理每个数的逆（这里就是直接取对数的相反数）来解决。这样复杂度就做到了$O(K\log K)$，看起来不错。



现在来考虑有负数的情况。这时要考虑符号的问题。最大的几个应该是正数，接着是一些0，最后是一些负数。如果我们确定了要选择的负数的个数，并且保证扩展出去的负数个数不变，那么也就确定了所有扩展出去的状态的符号。所以，我们可以把正数和负数分成两部分分别排序，然后固定好每部分要选的数的个数，然后先用之前只有正数的情况下的方法来选出正数的方案，固定下正数的方案之后再按照之前只有正数的情况下的方法来选出负数的方案。为此，初始时我们可以枚举要选择的负数的个数，然后按照正负性找出这种情况下最大的选法（如果答案为负就要选择负数中连续的绝对值最小的那几个数），作为一种初始状态放入优先队列中。当然为了方便，你也可以先处理完答案为正的情况，如果还没有取到第K个，就再处理答案为负的情况。注意到0还是可以作为正数处理，因为0的正负性对答案没有影响。这样复杂度就是$O((M+K)\log(M+K))$，可以通过本题。

小拓展：如果这题不是以对数的形式给出所有数，然后要求答案模一个数的值怎么办？如果可以求逆元，那只要解决比较大小的问题。比较大小可以比对数，不过当答案特别大时对数会有精度问题，这也是本题给出对数的原因。如果不能求逆元，那对于复杂度较劣的那种方法，求一个区间里的数的乘积可以用线段树；对于最终能通过的那种方法，可以记前$i-1$个数的乘积以及第i个数以后（不包括第i个数）的乘积，这样就可以避免求逆元了。

1007 Explorer Bo

对于最小链覆盖的问题,答案是$(sum_{leaf}+1)/2$下取整,即除了多出来的一个叶子,剩下的链都是从叶子开始,到叶子结束.

然后考虑分叶子数奇偶情况考虑一下.

对于叶子数是偶数的情况,可以考虑一个暴力的dp,令$f_{i,j}$表示做完i号点的子树还有j条链向上的最小值.然后树形dp合并的时候再枚举一下几条链并掉,复杂度是$O(N^3)$.然而这个题有一些性质,即从一个子树里连上来的树不超过2.考虑最浅一个三个点交汇的位置,那么通过调整可以发现两条链并起来,一条链向上的答案比原来更优.所以j那一维至多到2.

然后考虑奇数的情况,那么有一条链不会到另一个叶子,但是同时可以通过调整证明,这条链可以只是从叶子到一个祖先,那么就需要多记下这么一条链是否选过。

1008 Gardener Bo

令$size(u)$表示u的子树大小，$sum(u)$表示u的子树和，容易改写$f(u)$的计算公式：

$$f(u)=w_u+size(u)*sum(u)-\sum_{fa[v]=u}size(v)*sum(v)$$

设法直接维护每个点的答案。

假设某次操作是对u的子孙三代权值都加上了x，那么答案会发生改变的就只有u的子孙三代以及u的祖先。

1、对u的孙辈节点v，v的答案加上了$x*(size(v)+1)$。

2、对u的子代节点v，令$sons_1(u)$表示u的儿子个数，v的答案加上了$x*(2+size(v)sons_1(v))$。

3、对于u，令$sons_2(u)$表示u的儿子和孙子的个数，u的答案加上

$$x*\left(2+size(u)sons_2(u)-\sum_{fa[v]=u}size(v)sons_1(v)\right)$$

4、对u的祖先v，设w是链$[u,v)$上深度最浅的点，v的答案加上$x*(sons_2(u)+1)(size(v)-size(w))$。

因为我们询问的是单点的值，而且这些修改每次加的东西都是独立的，我们只要对于上面的4种情况分别采取高效的维护方式就可以了。

对于1、2，我们直接对树的bfs序开一棵线段树维护一下就好了。

对于3，对单点开个变量维护一下就好了。

对于4，树剖，条轻边的时候直接修改单点的变量，然后在重链上也用一个线段树维护一下。

注意到重链上每次加的权值基数是一样的，而询问的又是单点，我们可以只在跳轻边的时候修改单点的值，在询问的时候再对dfs序开一棵线段树收集子树中对重链的贡献。

这样一来最后的复杂度就变成了$O(Q \log n)$。

1009 Palindrome Bo

首先可以离散化把权值范围降到$O(n)$。

考虑假设已知一个BoBo序列seq，它出现在原串中的很多位置，我们要计数就需要把它唯一对应上某一个位置。

一种对应方法是：假设第一个和最后一个字符是c，我们找到原串中第一个c和最后一个c，接着考虑第二个和倒数第二个字符，以此类推。

我们根据这个进行DP，设$dp[l][r]$为一个二元组，表示$a[l]$和$a[r]$已经配对的情况下，$a[l,r]$的最长BoBo序列长度以及个数。

设$nxt[i][j]$表示i向右第一个字符是j的位置，设$pre[i][j]$表示i向左第一个字符是j的位置。

对于$dp[l][r]$的转移就是枚举一个字符c，然后从$dp[nxt[l][c]][pre[r][c]]$转移过来。暴力实现是$O(n^3)$的。

考虑固定l，从左到右枚举r，在这个过程中$nxt[l]$数组是不变的，r每移动一次$pre[r]$数组只会修改一个值。

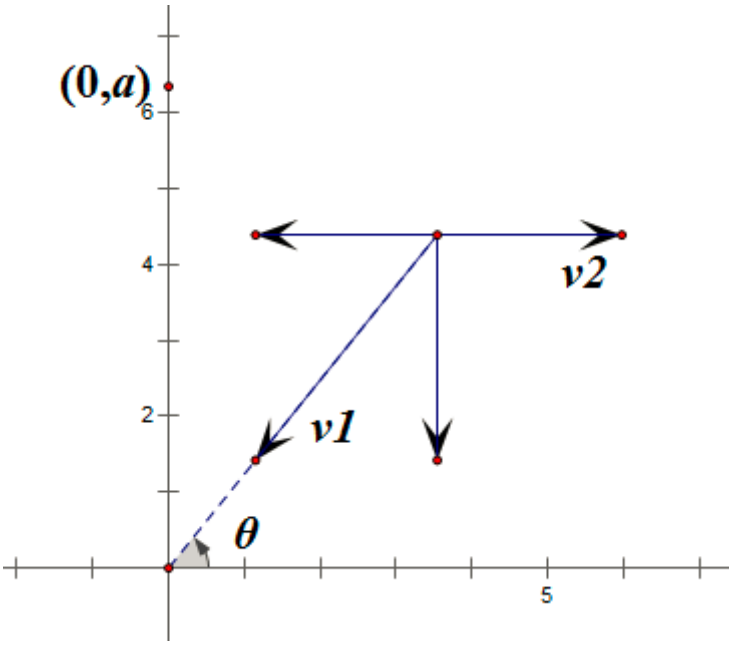
并且有效状态中要求$a[l]=a[r]$，因此我们每次询问的是$\le a[l]$的c对应的DP状态。随着r的右移，对于每个c的DP值都是越来越优的，我们只要开一个变量来维护一下当前的答案就可以了。

最后的答案就是枚举一个c，合并$dp[nxt[0][c]][pre[n+1][c]]$这些状态。复杂度是$O(n^2)$的。

1010 Rower Bo

首先这个题微分方程强解显然是可以的，但是可以发现如果设参比较巧妙就能得到很方便的做法。

先分解v_1，



设船到原点的距离是r，容易列出方程

$\frac{dr}{dt} = v_2 \cos\theta - v_1$

$\frac{dx}{dt} = v_2 - v_1 \cos\theta$

上下界都是清晰的，定积分一下：

$0 - a = v_2 \int_0^T \cos\theta dt - v_1 T$

$0 - 0 = v_2 T - v_1 \int_0^T \cos\theta dt$

直接把第一个式子代到第二个里面

$v_2 T = \frac{v1}{v2}(-a + v_1 T)$

$T = \frac{v1\,a}{{v1}^2 - {v2}^2}$

这样就很Simple地解完了，到达不了的情况就是$v_1 < v_2$（或者$a > 0$且$v_1 = v_2$）。

1011 Teacher Bo

考虑一种暴力,每次枚举两两点对之间的曼哈顿距离,并开一个桶记录每种距离是否出现过,如果某次枚举出现了以前出现的距离就输 YES ,否则就输 NO .

注意到曼哈顿距离只有 $O(M)$ 种,根据鸽笼原理,上面的算法在 $O(M)$ 步之内一定会停止.所以是可以过得.

一组数据的时间复杂度 $O(\min\{N^2, M\})$.

本条目发布于2016年7月26日 [http://bestcoder.hdu.edu.cn/blog/2016-multi-university-training-contest-3-solutions-by-%e7%bb%8d%e5%85%b4%e4%b8%80%e4%b8%ad/] 。属于多校题解分类。作者是wange。