dp 选讲

Codeforces Scoreboard

一场 codeforces 比赛有 n 道题目。按照 codeforces 的计分规则，在 t 时间通过第 i 道题目可以获得 $\max(b_i - k_i \cdot t, a_i)$ 的分数。若在第 $1, 2, ..., n$ 时间分别通过一道题目，最终得分最大值。$n \leq 200000$，$a_i, b_i, k_i \leq 10^9$。

让损失的分数最小。

$$Ans = \sum_{i=1}^{n} \max\{b_i - k_i \cdot t, a_i\} = \sum_{i=1}^{n} b_i - \sum_{i=1}^{n} \min\{k_i \cdot t_i, b_i - a_i\}$$

设 $c_i = b_i - a_i$ 。如果已经钦定了一些题目取 c_i 的价值，剩下的题目应贪心按照 k_i 不增的顺序从小到大分配时间。设 $f_{i,j}$ 表示考虑前 i 道题目，有 j 道题目取 $k_i \cdot t$ 的价值的最小值。

$$f_{i,j} = \begin{cases} f_{i-1,j} + c_i & j = 0 \\ \min\{f_{i-1,j} + c_i, f_{i-1,j-1} + k_i \cdot j\} & j > 0 \end{cases}$$

这是 $O(n^2)$ 解法。

发现 $f_{i,j}$ 关于 j 是凸的。设 $g_{i,j} = f_{i,j} - f_{i,j-1}$。用 $g_{i,j}$ 重新表示状态转移方程。

$$f_{i,0} + \sum_{k=1}^{j} g_{i,k} = \min\{f_{i-1,0} + \sum_{k=1}^{j} g_{i-1,k} + c_i, f_{i-1,0} + \sum_{k=1}^{j-1} g_{i-1,k} + k_i \cdot j\}$$

$$c_i + \sum_{k=1}^{j} g_{i,k} = \sum_{k=1}^{j-1} g_{i-1,k} + \min\{g_{i-1,j} + c_i, k_i \cdot j\}$$

用 $j - 1$ 代替 j 重写后作差

$$c_i + \sum_{k=1}^{j-1} g_{i,k} = \sum_{k=1}^{j-2} g_{i-1,k} + \min\{g_{i-1,j-1} + c_i, k_i \cdot (j-1)\}$$

$$g_{i,j} = g_{i-1,j-1} + \min\{g_{i-1,j} + c_i, k_i \cdot j\} - \min\{g_{i-1,j-1} + c_i, k_i \cdot (j-1)\}$$

k_i 不增。接下来归纳论述对于 g_{i-1}，$\forall j, g_{i,j} - g_{i,j-1} \geq k_i$ 可以推出由 g_{i-1} 变换得到 g_i 的方式，再由这种转移方式推出 g_i 也有这种性质。

归纳源头：当 $i = 2$ 时，$f_{2,0} = c_1 + c_2, f_{2,1} = \min\{k_1 + c_2, k_2 + c_1\}, f_{2,2} = k_1 + 2 * k_2, g_{2,2} - g_{2,1} = f_{2,2} + f_{2,0} - 2 * f_{2,1} = k_1 + c_1 + 2 * k_2 + c_2 - 2 * \min\{k_1 + c_2, k_2 +$

$c_1\} > k_1 + c_1 + 2 * k_2 + c_2 - k_1 - c_2 - k_2 - c_1 = k_2$。可以得到 $i = 2$ 时满足 $\forall j, g_{i,j} - g_{i,j-1} \geq k_i \geq k_{i+1}$ 。

观察转移方程中 $\min\{g_{i-1,j} + c_i, k_i \cdot j\}$ 这一项。前者 $g_{i-1,j}$ 的增长速度大于后者 $k_i \cdot j$ 的增长速度。则存在某一个位置 pos ，满足 $j < pos$ 时最小值取 $g_{i-1,j} + c_i$ ， $j \geq pos$ 时最小值取 $k_i \cdot j$。对于 $\min\{g_{i-1,j-1} + c_i, k_i \cdot (j-1)\}$ 只是将这个转移位置后移了一位。

求出这个最小的 pos 满足 $c_i + g_{i,j} > k_i \cdot j$。则由 g_{i-1} 变换到 g_i 的过程如下：$j < pos$ 的位置不变。$j \geq pos$ 的位置的值加 k_i 。中间插入 $k_i \cdot j - c_i$。

需要论证这样变换之后 g_i 仍满足 $\forall j, g_{i,j} - g_{i,j-1} \geq k_i$。只需要关注插入值相邻的两侧即可。由 pos 的定义得到

$$g_{i-1,pos} \geq k_i \cdot pos - c_i$$

$$g_{i,pos-1} = g_{i-1,pos-1} \leq k_i \cdot (pos - 1) - c_i$$

$$g_{i,pos} = k_i \cdot pos - c_i$$

$$g_{i,pos+1} = g_{i-1,pos} + k_i$$

可证明

$$g_{i,pos+1} - g_{i,pos} = g_{i-1,pos} + k_i - k_i \cdot pos + c_i \geq k_i$$

$$g_{i,pos} - g_{i,pos-1} = k_i \cdot pos - c_i - g_{i-1,pos-1} \geq k_i$$

归纳毕。关于 g_i 的性质和转移均正确。平衡树维护 g_i 。时间复杂度 $O(n \log n)$ 。这种维护原序列差分斜率的技巧也叫做 slope trick。

本文关于核心性质 $\forall j, g_{i,j} - g_{i,j-1} \geq k_i$ 的证明费尽周折，最终从性质推出转移形式再推回性质归纳而来。关于题目本身组合意义研究方向也有一番成果，可见 from_ez_lcw，从选取点集的性质方面证明了该核心性质。在此不作赘述。codeforces 官方题解对于此性质的证明一笔带过，可补充选阅。

code : from_yixiuge777

Tree Cutting

树。执行 k 次操作，每次选择一条边删掉并选择仅一个联通块保留，在纸上写下剩余联通块的大小。多少种操作方案使写下的序列等于给定序列。$n \leq 5000, k \leq 6$。

树上 dp 。但是发现最困难的一件事是如何记录树的形态，因为需要满足每一次操作都是符合条件的，而树的形态会随着操作而变化，操作到底是否符合条件依靠树的形态决定。

上面说的树的形态太抽象，具体来说，设计 $dp_{i,S}$ 表示 i 的子树里选了一些边，包括 i 和父亲的边在内，他们钦定的操作的集合为 S 。且不说儿子如何合并，如果 i 和父亲的边要操作，分为两种：1.保留子树，需要知道现在子树的大小才可以判断操作是否符合条件；2.删掉子树，需要知道现在子树的大小才可以判断切掉的子树部分的大小是否符合条件。由此一看，树的形态具体到子树来说只需要记录现在子树的大小就可以了，与子树外无关，可以直接子树 dp 。暴力方法是直接多加一维记录现在子树的大小是多少，时间上升至最少 $O(n^2 2^k)$ ，无法接受。

其实我们可以把现在子树大小通过操作的方式来推出。操作分为保留子树和切掉子树，记录下每个状态中最早的保留子树的操作，那么与父亲的边要想操作，必须要赶在最早的保留子树操作之前操作，而在这之前子树内全部是切除子树操作，由此可以根据切除操作切掉的数的大小推出现在子树还剩的大小。对于兄弟之间合并需要判断兄弟之间最多有一个保留子树操作。这样时间复杂度 $O(n(k3^k + k^2 2^k))$ 。

上述做法已经初步尝试用记录操作达到记录子树大小的目的，但还不够深入。其实对于一个操作，进行操作之后子树或子树外就会被扔掉，也就是说剩余的操作只能在其中一部分之内进行了。具体的，如果与父亲的操作是保留子树操作，剩余的操作必须全部在子树内进行。如果与父亲的操作是切除子树操作，剩余的操作必须全部不在子树内进行。如果不满足上述条件，后续也一定不会满足，可以直接将 dp 值清零。那么再来看最关键的对于自己和父亲这条边的取舍，如果子树内有第一个保留子树的操作，其 dp 值上推的时候，后续操作一定在这个子树内，因此不可能再次在这个保留子树操作之后操作。换句话说，当前钦定的父亲与自己要在哪里操作，这个位置之前一定都是切除子树操作，只需要放心把前面的操作全部切除即可，直接计算出现在的子树大小。如果这个操作是保留子树操作，需要满足这个操作一定是最后一个可操作的位置，也就是后面不能再有可操作的空位。

重新回过头来看我们到底是优化了什么。思考一下第二个方法中为什么要特殊记录一下第一个保留子树操作是什么位置，就是为了限定与父边的操作必须在此之前，而在此之前子树内的操作全是切除子树操作，可以通过操作切掉的数量推出子树大小。其实无需记录这一维度，因为如果保留子树操作之后的操作必须都在子树内进行，如果不满足之后也一定不会满足，可以直接去掉。额外记录第一个保留操作在什么时候进行，实际上是没有及时清理非法状态，导致必须记录一个冗余状态才可以维护正确性。只要及时清理了非法状态，与父边的操作之前的子树内的操作一定是切除子树操作，直接可以计算。

最后是合并兄弟的方法。因为保留子树后的操作都在子树内进行，所以只需要保证两兄弟操作不交就能贡献，这样也一定不存在两兄弟同时进行了保留子树操作的情况。时间复杂度 $O(n(3^k + 2^k k))$ ，瓶颈在于合并兄弟时的枚举子集，可以使用集合幂级数科技优化至 $O(n2^k k)$ 。

code : from_yixiuge777

Decinc Dividing

能将一个区间中的数划分为两个子序列，一个单增一个单减，则区间是好的。求一个排列有多少好区间。$n \leq 200000$。

设 $dp_{l,i}$ 表示区间左端点是 l ，右端点是 i ，且 i 在上升序列里，下降序列最后一个的最大值；

设 $pd_{l,i}$ 表示区间左端点是 l ，右端点是 i ，且 i 在下降序列里，上升序列最后一个的最小值；

从大到小枚举 l ，每次暴力更新，可以做到平方复杂度。

观察一下，对于一个 i ,且 i 在上升序列里，j 为 i 左边最大的一个，最靠近 i 的一个位置，满足 $a_j > a_{j+1}$ 。这样，j 和 $j+1$ 不可能同时在上升序列里，必然有一个在下降序列。而 $[j+2, i-1]$ 里面的所有数是单调递增的，也都可以放到上升序列里，所以下降序列的末尾只有可能是 $\{a_j, a_{j+1},, inf, -inf\}$。对于每一个 i ，不管 l 怎么变化，更新的次数都是 $O(1)$ 的，最多只有4种可能。

对于 i 在下降序列里同样，pd_i 的更新也是 $O(1)$ 的。每次找到一个 j 暴力往后找 i 即可。时间复杂度 $O(n)$ 。

对于一个序列，能划分为一个单增一个单减子序列的充要条件是不存在 3412 和 2143 的相对顺序。可以从小到大加入数归纳构造证明。由此对于一个 l 求出最小的 r 出现了 3412 或 2143。处理 3412 ，对于 l 找到后面最近的一个大于他的数字，对于 r 找到前面最近的一个小于他的数字。限制变成二位偏序，离线扫描线解决。时间复杂度 $O(n \log n)$。

code(dp) : from_yixiuge777

Outermost Maximums

长为 $n+2$ 的序列，$a_0 = a_{n+1} = 0$。每次操作可以选择序列中最左侧出现的最大值将其变成前一位置的前缀最大值，或者选择序列中最右侧出现的最大值将其变成后一位置的后缀最大值。序列变成全 0 的最小操作次数是多少。$a_i \le n \le 200000$ 。

从大到小考虑每一个数，贪心，会换成左右边最大值较小的那个。设在当前的一些最大值当中，l_i 为 i 左边的不包括自己的值的最大值， r_i 为 i 右边的不包括自己值的最大值。那么当最左边的最大值变成 $min(l, r)$ 之后，他不会再立刻成为前缀或后缀的最大值，也就是说 l_i, r_i 不会受到其他数字变化的干扰，最初就可以知道。

由此我们知道，一个位置上的数字会变成什么，只与更小的数字有关，与比他大的数字变成了什么没有关系。这使得可以每个数字单独计算贡献。这是本题的核心思想。

暴力模拟上述过程是 $O(n^2)$ 的，如何优化上述过程。从值域大到小考虑每个值，刚刚碰到这个值的时候并不知道每个位置会变成什么更小的数字，要继续往下看。相反碰到这个值会让比他更大的值确定下来要变成什么。设在区间里出现最左边这个值的位置是 l ，最右边出现这个值的位置是 r （这里的 l, r 与上文无关）。比他更大的数字，如果是“未确定”并且在 l 的左侧会变成“左”，因为此时在他的右侧出现了一个大的数字，只能换成左边的数；同理是“未确定”并且在 r 的右侧会变成“右”，在 $[l, r]$ 内的“未确定”还是会变成“未确定”，但是注意他们已经完成了变换，相当于变换完了之后又要进行下一次变换。

对值域操作后相当于对整个序列上的比他大的数操作了一波标记，但是在整个过程中“左”一直占领一个前缀，“右”一直占领一个后缀，“未确定”占领中间的位置。考虑新的值只会影响这三种标记的分界点，不会将他们打乱。于是我们只用维护这三个标记的两个分界点 L, R 即可维护住标记集合。

计算答案的时候，刚刚考虑到一个值的时候先不去计算他的答案，因为此时比他小的数字还没有考虑到，未知他会变成什么数字。等到一个数字操作完了，标记重新变化为“未确定”，相当于进行了一次操作，此时我们计算答案。具体的，考虑到了一个值，最左侧出现的位置是 l ，最右侧出现的位置是 r ，那么 $[l, r]$ 之内的所有带标记的数字都会贡献答案，因为不管是什么标记都相当于操作了一次，“未确定”还是变成“未确定”，但是已经操作过了，仍然统计答案。剩下的如果 $L > r$ 会对一部分左标记影响，如果 $l > R$ 会对一部分右标记影响。对于一个值操作后的 L, R 的变化都可以通过讨论解决，剩下的只有求在特定的一段区间里有多少数字有标记，也就是有多少数字比当前值大。树状数组维护单点加区间查询即可。时间复杂度 $O(n \log n)$ 。

code : from_yixiuge777

Edge Elimination

树上每次删去一条与偶数条边相邻的边。求一个方案或无解。$n \leq 200000$。

删边的时候相连两点的度数的奇偶性相同，设两点度数奇数的时候删掉的边为奇边，相反为偶边。观察，一个点每删去一条边奇偶性发生变化，问题有解的必要条件是奇点的奇边比偶边大一，偶点的奇边与偶边相同。

这也是充分条件。对于每一个点，将边按照任意合法删边顺序偏序，即奇边偶边相间删除。不管怎么安排，边的删除顺序形成有向无环图，一定有一个构造解。

确定每条边的奇偶性。从叶子结点开始，他连向父亲的边是奇边。之后每一个点都知道了他连向儿子的边的奇偶性，就能唯一确定自己连向父亲的边的奇偶性，矛盾就一定无解。确定好之后对于每个点随便确定一个删边顺序，拓扑排序求解。时间复杂度 $O(n)$ 。

code : from_yixiuge777

Tenzing and Random Real Numbers

n 个值域是 $[0, 1]$ 的随机实数 $x_1, x_2, ..., x_n$ 。m 个限制形如 $x_i + x_j \leq 1$ 或 $x_i + x_j \geq 1$。求概率。$n \leq 20$。

每个数减去常数 0.5 。这样限制变成了两数相加是正是负。由绝对值大的数字符号决定。从小到大绝对值做 dp ，每次枚举符号。每个按绝对值大小形成的排列概率相同，最后除以 $n!2^n$ 即可。时间复杂度 $O(n2^n)$。

code : from_yixiuge777

Tenzing and Random Operations

序列。m 次随机一个前缀给每个数加固定值 v 。求最终乘积期望。$n \leq 5000, m, v \leq 10^9$。

$\prod_{i=1}^{n}(a_i + v + v + 0 + v + ...)$ 拆开，每一项要么选择 a_i ，要么选择一个随机变量。如果选择的这个随机变量之前选择过，那么现在其一定为 v 。如果没选择过，其在此处为 v 有一个概率。发现所有随机变量只有之前选过和没有选过的转移系数的差异。只需要额外记录之前选过的操作的数量即可转移。时间复杂度 $O(n^2)$。

通过细化转移辨别出本质不同的转移数量缩减状态。极大可能程度上合并状态。

code : from_yixiuge777

LuoTianyi and XOR-Tree

修改点权使得每个点到根路径异或和为0，最少修改多少点。$n \leq 100000$。

只让所有点到根异或和相同。贪心合并两个子树，使其操作次数越少越好。节约操作次数一定更优，因为可以通过修改根的权值随意调控异或和。启发式合并。时间复杂度 $O(n \log^2 n)$。

code : from_yixiuge777

Mex Tree

给树上节点 01 染色。权值是所有点对之间路径 mex 的和。点对包含自己一对。求最大值。$n \leq 200000$。

$f_{i,j,0/1}$ 表示 i 子树中和 i 相连的同色连通块大小是 j ，颜色是 $0/1$ 。树上背包转移。

考虑一个特殊的染色方案：二分图式染色。所有长度大于等于 2 的路径的答案都是 2 。这样损失的答案只有 $2 \cdot n$ 级别的。

如果有一个大小为 k 的同色连通块，其损失的答案为 $\frac{k(k+1)}{2}$。不能大于 $2n$ ，则 k 只有根号大小级别。时间复杂度 $O(n\sqrt{n})$ 。使用重链剖分技巧线性空间。详见代码。

code : from_yixiuge777

Wonderful Jump

序列，一开始在 1 。走一步的花费是 $\min(a_i, a_{i+1}, \ldots, a_j) \cdot (j-i)^2$。对于每个位置求走到的最小花费。$a_i \leq n \leq 400000$。

用一步一步跳跃操作缩减转移范围。A 为值域上限。跳跃一定需要满足 $\min(a_i, a_{i+1}, \ldots, a_j) \cdot (j-i)^2 < A \cdot (j-i)$ 。则 $j - i > \frac{A}{\min(a_i \ldots a_j)}$。对于 $\min(a_i \ldots a_j) > \sqrt{A}$ 的情况，暴力转移最近的

$\sqrt{A}$ 个决策即可。对于 $\min(a_i \dots a_j) <> \sqrt{A}$ 的情况，可以发现最小值一定是区间一个端点时最优，这样可以将大区间划分为小区间。维护 $[1, \sqrt{A}]$ 中的最靠右的出现位置作为区间最小值在左端点处的决策。若 $a_i < \sqrt{A}$ ，暴力向前找决策，直到遇到比他小的数字。对于 $[1, \sqrt{A}]$ 的每个值复杂度不超 $O(n)$ 。总体时间复杂度 $O(n\sqrt{A})$。

code : from_yixiuge777

Count Voting

每个人可以给不同阵营的一个人投一张票。求一种投票结果的投票方式数量。$n \leq 200$。

容斥。钦定 k 个人给自己阵营投了票。式子容易 dp。时间复杂度 $O(n^2)$。

code : from_yixiuge777