动态规划常见优化+数据结构+杂题

CF1205D

- 你会得到一个包含n节点的树。你需要给每条边赋值，使长度从1到(2*n^2)/9 的路径都至少出现一次
- 数据保证有解，边权要求非负
- n<=1000
- 特殊性质1：给定的树是一条链
- 特殊性质2：给定的树是一个菊花图

解法

- 特殊性质部分解法较多
- 我们考虑进制的思想，即把树分为两半，其中一半的边权设为1,2,3,...,L-1，另一半的边权设为L,2L,3L,(n-L+1)*L
- 下面给出两种参考
- 特殊性质1：把链分成两半，一半的边权设为1，剩下的边权设为n/2
- 特殊性质2：等价于n-1个数两两之和包含[2,(2*n^2)/9]，因此考虑用进制的思想，一半的边设为1至(n/2-1)，另一半设为（n/2）的整数倍 （相当于n/2进制）

- 正解：
- 我们考虑类似特殊性质的做法，把树分成两半，用进制思想赋值。观察题目限制我们可以发现（2*n^2/9）=（2n/3）*(n/3)。因此只要保证每一边的点数大于n/3就可以了。
- 我们可以找到树的重心，将其所有子树按结点数从小到大排序，可以发现一定能够找到一个分界点，使两堆子树点数均大于n/3
- 然后我们分两边构造边权。
- 设其中一边结点数为L。先把重心作为根，把边一条条加入树中，使新加到已在树上的点的边的边权为该点子树中以该点为起点的最长链长度加一(如果是构造大步的那一块，则为加 L+1)。

假设子树的大小分别为 $s_1, s_2, \cdots, s_m$。首先当 $m = 2$ 时显然。

由于 $\sum_{i=1}^{0} s_i = 0$，$\sum_{i=1}^{m} s_i = n - 1$，故一定存在一个分割点 k，使得 $\sum_{i=1}^{k} s_i \leq \frac{n}{3}$ 而 $\sum_{i=1}^{k+1} s_i > \frac{n}{3}$。

如果 $\sum_{i=1}^{k+1} s_i \leq \frac{2n}{3}$，则取 $S_1 = \{s_1, \cdots, s_{k+1}\}$，$S_2 = \{s_{k+2}, \cdots, s_m\}$ 即可。否则 $\frac{n}{3} < s_{k+1} \leq \frac{n}{2}$，取 $S_1 = \{s_{k+1}\}$，$S_2 = \{s_1, \cdots, s_k, s_{k+2}, \cdots, s_m\}$。

洛谷 P7244

简化题意

有一个长度为 n 的序列 a。要求将这个序列**恰好**分成**连续且非空**的 k 段，并定义第 i 段的立意值为该段的所有元素的最大值，记为 b_i。要求最大化 $\gcd\limits_{i\in[1,k]}\{b_i\}$ 并输出这个最大值。

对于 100% 的数据，$1 \le k \le n \le 10^5$，$1 \le a_i \le 10^6$。

子任务	分值	n	k	a_i
1	5	≤ 5	/	/
2	10	$\le 10^2$	/	/
3	10	/	2	/
4	15	/	3	/
5	20	$\le 3\times 10^3$	/	/
6	10	/	/	$\le 2\times 10^2$
7	30	/	/	/

最大的a_i一定会作为b_i出现，即答案一定为全局最大值的因数

我们从大到小枚举每个因数检查是否能够分成k段，使得每段的b_i都是这个因数的倍数

不难发现当我们把多个b_i满足条件的段合并时，合并成的大段依然满足b_i的限制。因此只需保证可以分成的段落数$\geq k$

我们可以设计dp计算分段数。记f_i表示以i为结束位置，在满足限制的条件下最多可以分成几段

我们将a_i分两种情况讨论：是否为它所在段的b_i。

记j表示i左边第一个比a_i大的数的位置

当a_i为所在段的最大值时 $f_i = max(f_k + 1)(j \leq k < i)$ 反之 $f_i = f_j$

最后若$f_n \geq k$则为答案

直接做是$O(n^2)$的，发现转移可以用单调栈优化。用单调栈维护当前点左边第一个比它大的数的位置j，以及弹栈的同时维护$max(f_k + 1)$，对每个因数的dp复杂度优化至$O(n)$

有一条长度为W的直线，初始状态下你可以在直线上的任意位置。

直线上一共有N个物品，其中第i个物品在p_i位置，价值为v_i，若你在t_i秒恰好处于p_i，你可以获得它。

你每秒最多能够移动两个单位，求可以获得的最大价值总和

对于15%的数据，$n \leq 10$

对于35%的数据，$n \leq 5000$

对于所有数据，$1 \leq w_i, t_i \leq 10^8, 1 \leq n \leq 100000$

数据保证在int范围内

设计一个朴素dp

f_i表示最后一个取第i个物品的最大价值

$$f_i = max(f_j + v_i)(2(t_i - t_j) \geq |p_i - p_j|, i \geq j)$$

时间复杂度$O(n^2)$

我们把绝对值拆开，得到$2(t_i - t_j) \geq p_j - p_j \geq 2(t_j - t_i)$

即$\begin{cases} 2t_i - p_i \geq 2t_j - p_j \\ 2t_i + p_i \geq 2t_j + p_j \end{cases}$

转化为树状数组维护二维偏序问题求解

ARC037D

- 有N个格子排成一行。这些格子从左到右标号为1到N 。
- 你有两个棋子，开始分别放在A号格子和B号格子里。你将要按照接收的顺序执行Q个以下形式的请求：
- 给定一个正整数xi ，选择两个棋子中的一个，把它移动到xi 号格子里。
- 把一个棋子移动一格需要一秒钟的时间。也就是说，把一个棋子从X号格子移到Y号格子需要|X-Y|秒的时间。
- 你的目标是在尽量短的时间内执行完所有的请求。
- 你只能为了回应请求而移动棋子，并且你不能同时移动两个棋子，此外你不能更改接收请求的顺序。但是允许在某个时刻，两个棋子在同一个格子里。

子任务1：$Q \leq 15$

子任务2：$N \leq 100$

子任务3：$Q \leq 4000$

对于所有测试点 $N, Q \leq 200000, 1 \leq A, B, x_i \leq N$

设计一个动态规划，$f(k,i)$ 表示已经做完前 k 个请求，并且最后一次请求中不动的棋子停留在 i，当然，移动的棋子最终停留在 x_k。

那么 $f(1,a)=|x_1-b|$，$f(1,b)=|x_1-a|$。对于 $k\geq 2$，我们有：

$$f(k,i)=\begin{cases}\min\limits_{1\leq j\leq n}\left\{f(k-1,j)+|x_k-j|\right\}, & i=x_{k-1};\\ f(k-1,i)+|x_{k-1}-x_k|, & i\neq x_{k-1}.\end{cases}$$

注意到 $i\neq x_{k-1}$ 的情况实际上是对一个数组（除了某一个元素）整体加上一个固定的数，这只需要在外部记录一个标记即可。

至于 $i=x_{k-1}$ 的情况中，绝对值符号内部有与 j 相关的式子，我们不如将它拆为：

$$\min\left\{\min_{1\leq j<x_k}\left\{f(k-1,j)-j+x_k\right\},\min_{x_k\leq j\leq n}\left\{f(k-1,j)+j-x_k\right\}\right\}$$

所以我们就是要维护数组 $\left\{f(k,i)-i\right\}$ 的前缀最小值，以及 $\left\{f(k,i)+i\right\}$ 的后缀最小值，最后进行一次单点修改。

注意到每一次单点修改中，$f(k,x_{k-1})\leq f(k-1,x_{k-1})+|x_k-x_{k-1}|$，也就是说在与其余元素一起进行了整体增加操作之后，单点修改只会使该位减小。因此，这可以使用树状数组维护。

时间复杂度 $O(q\log n)$。

先提一下平衡树

- 先讲概念，平衡树是一种二叉树
- 多数情况下平衡树上的各种操作单次复杂度为log(n)级别，其可以维护按关键字大小排序的序列
- 在简单的实现下，平衡树可以支持树上二分查找、动态加点删点，区间修改（区间加减定值）等操作
- 在本节课中大家可以将其简单理解为功能类似线段树、但空间复杂度为O(n)且可以动态插入下标的数据结构

- 平衡树的种类：splay、treap、无旋treap、红黑树、替罪羊树等
- 如果有学习板子的兴趣推荐学习无旋treap，能够完成基本操作、代码难度低且易于理解

[FJWC2017]最大价值

A君有n个物品，每个物品有两个属性a_i, b_i，现在A君想从所有物品中挑选k个，并按一定顺序摆放好

一个方案中，若被摆在第j个位置（位置从1开始标号）的物品为i，它对这个方案产生的价值贡献为

$a_i * (j-1) + b_i$，一个方案的价值和为它所含的所有物品的贡献之和

现在A君想知道对于所有可能的$k(1 \le k \le n)$，在最优的选取以及摆放情况下能得到的方案价值和最大是多少

$30\% : n \le 20$

$60\% : n \le 3000$

$100\% : n \le 300000, 0 \le a_i \le 10^6, 0 \le b_i \le 10^{12}$

我们先观察性质，易证选择的物品按a_i从小到大摆放一定最优，我们将物品按a_i排序后处理

30%：直接搜索

60%：设计一个朴素dp。设$f_{i,j}$表示当前做到第i个物品，共取了j个物品可获得的最大贡献和

转移有两种：1、$f_{i,j} = f_{i-1,j-1} + a_i * (j-1) + b_i$；2、$f_{i,j} = fi-1,j$；取最大值

时间复杂度$O(n^2)$

100%:我们考虑优化60分dp，打表或观察性质可以发现，由f_{i-1}转移到f_i的过程中，存在一个下标k，使得$f_{i-1,1\sim k} \to f_{i,1\sim k}$按上述2方式转移，$f_{i-1,k+1\sim i-1} \to f_{i,k+2\sim i}$按上述1方式转移（具体证明在下页）

我们发现这样对于$f_{i-1} \to f_i$ 相当于k左侧的dp值保持不变，k右侧的dp值下标整体右移一位并加上一个等差数列$(a_i * (j-1) + b_i)$，并在$k+1$的位置插入一个新的dp值

这一部分可以平衡树维护，每次转移时树上二分找到对应的k并插入一个新的结点，区间加等差数列可以通过差分转化成区间加常数，时间复杂度$O(nlog(n))$

结论. 令 $S_0 = \emptyset$, $c_k \in \arg\max\{Q(S_{k-1}, i) \mid i \notin S_{k-1}\}$, $S_k = S_{k-1} \cup \{c_k\}$, 则 S_k 对应了恰选择了 k 个时的最大价值。

证明. 以下把“对应了恰选择了 k 个时的最大价值的集合”叫做 **k–最大集**。

对任意 $1 \le k \le n$, 对 $0 \le t \le k$ 施归纳证明一个子结论：S_t 是某个 k–最大集的子集。

当 $t = 0$ 时，$S_0 = \emptyset$, 子结论显然成立。

假设当 $t = m - 1$ 时成立，当 $t = m \le k$ 时：

任取一 k–最大集 $T \supset S_{m-1}$. 若 $S_m \subseteq T$, 子结论已经成立。以下假设 $S_m \nsubseteq T$.

方便起见，下记 $A = S_{m-1}$, $B = T \setminus A$, $i = c_m$.

1. $\min B < i$, 那么设 $j = \max B \cap (0, i)$. 我们知道 $Q(A, j) \le Q(A, i)$, $a_j \le a_i$.
   下面考虑将 $B \setminus \{j\}$ 中的元素依次加入 A. 当加入 x 时，若 $x > i$, 则它对 $Q(A, j)$ 与 $Q(A, i)$ 的增量均为 a_x; 若 $x < j$, 则它对 $Q(A, j)$ 的增量为 a_j, 对 $Q(A, i)$ 的增量为 a_i; 根据我们选取的方式，不存在 $j < x < i$ 的情况。因此在加入的过程中，任意时刻均有 $Q(A, j) \le Q(A, i)$. 最后 $A = T \setminus \{j\}$, 因此可以在这时把 j 替换为 i, 即 $(T \setminus \{j\}) \cup \{i\} = S_m \cup (B \setminus \{j\})$ 也是 k–最大集，子结论成立。
2. $\min B > i$, 那么设 $j = \min B$. 我们知道 $Q(A, j) \le Q(A, i)$.
   下面考虑将 $B \setminus \{j\}$ 中的元素依次加入 A. 当加入 x 时，若 $x > j$, 则它对 $Q(A, j)$ 与 $Q(A, i)$ 的增量均为 a_x; 根据我们选取的方式，不存在 $x < j$ 的情况。因此在加入的过程中，任意时刻均有 $Q(A, j) \le Q(A, i)$. 最后 $A = T \setminus \{j\}$, 因此可以在这时把 j 替换为 i, 即 $(T \setminus \{j\}) \cup \{i\} = S_m \cup (B \setminus \{j\})$ 也是 k–最大集，子结论成立。

综上所述，当 $t = m$ 时子结论成立。

由上述归纳可得，S_k 是某个 k–最大集的子集。结合 $|S_k| = k$, 可知 S_k 就是 k–最大集，结论证毕。

对前 i 个物品的 j–最大集应用上述定理可知动态规划的“决策单调性”结论正确。

决策单调性优化（IOI2014 holiday）

一条直线上有n个格子，每个格子有一个价值，你初始在第s个格子。

你有d单位的时间，每个单位的时间里，你可以选择移动到相邻的格子，或获得当前格子的价值，价值不能重复获得，求可获得的最大价值

$n \leq 100000, d \leq 2.5n$，输入在int以内，时限$2s$

不难发现最优方案一定是向一边跑到底，再掉头回来往另一边跑，在途中选价值最大的几个格子，以下先讨论先往左再往右的情况（另一边同理）

观察发现一个性质：对指定的路径左端点l，设其最优解对应的右端点为r_l，则$r_{l-1} \leq r_l$（具有单调性）

证明：假设$r_l < r_{l-1}$，那么区间$[l, r_{l-1}]$的前k_l大均在区间$[l, r_l]$中，

设左端点在$l-1$时在$[l, r_l - 1]$参观的城市有k_{l-1}个，且$k_{l-1} \leq k_l$

则区间$[l, r_{l-1}]$的前k_l大也在区间$[l, r_l]$中，所以取$r_{l-1} \leq r_l$时最优解不会更劣

所以假设不成立，证毕

由以上性质，我们对s左侧的区间分治

我们取出分治区间中点Mid作为左端点，在最优解所在区间内暴力枚举右端点找最优解

由于最优解有单调性，假设此时以Mid为左端点对应的最优解为p，那么分知道下一层时Mid左边只需要枚举到p，右边只要从p开始枚举

均摊复杂度$O(nlog(n))$，用主席树维护区间k大，总复杂度$O(nlog^2(n))$

斜率优化（FJOI2019D2T1超级计算机）

给定n个作业，第i个作业需要t_i的时间完成，且有f_i的费用系数

你需要将这些作业分成若干批，每批包含相邻的若干作业，并从时刻0开始分批处理这些作业。每批作业开始前需要S的时间启动程序，处理这批作业的时间是其中各个作业需要时间的总和，且其中所有作业的完成时刻相同，批处理的费用是每个作业费用系数f_i乘上其完成时间之和，你需要计算最小的总费用

$S \leq 50, t_i, f_i \leq 100$

$n \leq 10000$

方便起见，我们约定本题解中 $t[i], f[i]$ 分别为题面中 t_i 和 f_i 的前缀和。

首先考虑最暴力的想法。由于状态转移与完成时刻有关，我们记 $g[i][j]$ 表示当前已经处理完了前 i 个任务且处理完的时刻为 j 的最小代价。容易得到状态转移

$$g[j][k] + (f[i] - f[j]) * (k + s + t[i] - t[j]) \to g[i][k + s + t[i] - t[j]]$$

时间复杂度为 $O(n^2m)$，其中 $m = n * s + \sum t[i]$，可以通过 30% 的数据。

由于 m 这一维非常大，我们考虑优化。发现其实处理完前 i 个任务的完成时刻仅取决于我们把这些任务分成了几批完成，于是考虑记 $g[i][j]$ 表示当前已经处理完了前 i 个任务，并且在完成这些任务时把它们分成了 j 批的最小代价。于是状态转移变为

$$g[j][k] + (f[i] - f[j]) * ((k + 1) * s + t[i]) \to g[i][k + 1]$$

时间复杂度为 $O(n^3)$，可以通过 50% 的数据。

现在我们考虑时间复杂度瓶颈在于需要枚举 k，我们考虑把这一维优化掉，让 $g[i]$ 直接表示处理完前 i 个任务的最优代价。这是一个有后效性的表示，我们考虑如何消除后效性。这个表示法的后效性在于，我们把前 i 个任务分成几段，会对后面的状态转移代来影响，但分成几段则和这些任务所需运行时间无关。于是我们考虑把程序启动时间和程序运行时间分开考虑。

发现我们每把任务分割成一段，在这之后的每一段任务（包括现在这一段任务）$(a, b]$ 都会产生一个 $(f[b] - f[a]) * s$ 的代价，即我们每把任务分成新的一段，程序启动时间带来的代价为 $\sum (f[b] - f[a]) * s$，其中 a, b 首尾相连，即 $b_i = a_{i+1}$，且最后一个 b 为 n。这就是说，把任务 $(a, b]$分成新的一段，在程序启动时间上带来的代价为 $(f[n] - f[a]) * s$。那么我们把这段代价直接算进 $g[b]$ 的代价，而不是等到之后每次计算，即可消除后效性。

至此，得到状态转移

$$g[j] + (f[n] - f[j]) * s + (f[i] - f[j]) * t[i] \to g[i]$$

时间复杂度 $O(n^2)$，可以通过全部数据。

HNOI2008 玩具装箱

P 教授要去看奥运，但是他舍不下他的玩具，于是他决定把所有的玩具运到北京。他使用自己的压缩器进行压缩，其可以将任意物品变成一堆，再放到一种特殊的一维容器中。

P 教授有编号为 $1\cdots n$ 的 n 件玩具，第 i 件玩具经过压缩后的一维长度为 C_i。

为了方便整理，P教授要求：

- 在一个一维容器中的玩具编号是连续的。
- 同时如果一个一维容器中有多个玩具，那么两件玩具之间要加入一个单位长度的填充物。形式地说，如果将第 i 件玩具到第 j 个玩具放到一个容器中，那么容器的长度将为 $x=j-i+\sum_{k=i}^{j}C_k$。

制作容器的费用与容器的长度有关，根据教授研究，如果容器长度为 x，其制作费用为 $(x-L)^2$。其中 L 是一个常量。P 教授不关心容器的数目，他可以制作出任意长度的容器，甚至超过 L。但他希望所有容器的总费用最小。

对于全部的测试点，$1\leq n\leq 5\times 10^4$，$1\leq L\leq 10^7$，$1\leq C_i\leq 10^7$。

P3195 [HNOI2008] 玩具装箱 - 洛谷题解

数据结构优化查找

- 数组、链表：基础运用跳过
- 堆：查询最大/最小值查询O(1)，修改O(log(n))
- 单调栈、单调队列：在与单调性有关的题中，可以维护出单调的序列，避免重复遍历查找
- 单调栈优化查找的实例见前面洛谷P7244
- 单调队列优化查找例题见下页

单调队列例题

给你一个长度为N的数组，一个长为K的滑动的窗体从最左移至最右端，你只能见到窗口的K个数，每次窗体向右移动一位，如下表：

Window position	Min value	Max value
[1 3 -1] -3 5 3 6 7	-1	3
1 [3 -1 -3] 5 3 6 7	-3	3
1 3 [-1 -3 5] 3 6 7	-3	5
1 3 -1 [-3 5 3] 6 7	-3	5
1 3 -1 -3 [5 3 6] 7	3	6
1 3 -1 -3 5 [3 6 7]	3	7

你的任务是找出窗口在各位置时的max value,min value.

20%:n<=500;

50%:n<=100000;

100%:n<=1000000;

线段树

- 除了区间修改、区间查询，线段树可以树上二分查第k小值
- 以此为例

```
int query(int k,int l,int r,int k){
    if(l==r) return tr[k];
    int mid=(l+r)/2;
    if(k>sz[k<<1]) return query(k<<1|1,mid+1,r,k-sz[k<<1]);
    else return query(k<<1,l,mid,k);
}
```

平衡树

- 平衡树也可以区间加减、区间查询、树上二分

你需要维护一个序列：

开始时序列为空，之后进行n步操作，每一步在某一个位置插进一个数，使得序列长度+1

最后，从前到后输出这个序列

时间限制$500MS$

对于100%的数据，保证$n \leq 100000$

我们发现如果按操作顺序从前往后做，那么后半段的下标一直在变化不好维护

把操作离线后从后往前考虑，建立一个长为n的空序列，首先可以确定最后一个操作的实际位置就是pos_n

然后考虑前一个操作，可以发现第i个操作的实际位置是当前序列从前往后的第pos_i个空位

我们用线段树上二分来查找每个数的位置

时间复杂度$O(nlog(n))$

也可以作为平衡树板子题写

杂题部分

[2016省队集训]coldwar

N个结点M次操作，操作分为两种：

1：将结点u, v用双向边连接起来

2：查询结点u, v最早在第几次操作后联通，当前未联通输出0

操作强制在线，$1 \leq N, M \leq 500000$

我们可以用并查集按时间顺序合并连通块（不进行路径压缩）。

在并查集中加边时，我们将一个点连向其父亲的边权设为这条边加入的时间

每次询问u, v时从u到v路径上边权最大值即为答案

时间复杂度最坏是$O(n^2)$的，按秩合并优化到$O(nlog(n))$

[FJSC2017D8T1]潜入苏拉玛

给你一张N个点M条边的无向图，你从S点出发要到T点

图中有P个站点，你从起点或站点出发，在不经过下一个站点的情况下最多只能走K条边

求K的最小值

时限$2s$,$1 \leq P, S, T \leq n \leq 100000, 1 \leq M \leq 150000$.测试点数不超过5

我们把S和每个站点同时作为起点跑bfs，当以一个起点出发搜到其他起点访问过的点时用并查集合并连通块

当S和T联通时当前bfs的距离即为答案

- 具体实现需要把边拆成点

一道简单题

给两个长为n的序列a和b

输出前$a_i + b_j$的前k小值

$n \leq 10000$ 时限$200ms$

- 解法多样，给出一种做法

对每个a_i，初始把$a_i + b_1$扔到堆里，被取出时将b的下标+1再放进堆里

NOI2010day1 超级钢琴

给出一个长度为N的整数序列a，求所有长度在$[L, R]$之间的连续子序列的总和的前k大值之和

时限$2s$，$N, k \leq 500000$

先用前缀和把子段和转化为端点处的值

用RMQ维护区间最小值，这样我们可以对每个右端点求出区使子段和最大的左端点。

可以考虑用堆维护每个右端点的当前最大子段和，每次取出子段和最大的右端点并更新该端点的下一最大值

对于一个右端点i，开始的时候可选的左端点是一个区间$[i-r,i-l]$，我们假设当前取出$(k,r]$这个子段，则原区间变为$[i-r,k-1]$和$[k+1,i-l]$，重新放入堆中即可

这样做堆中元素是$O(n+k)$的，可以通过

洛谷P4135

由于时间紧迫，SHY 作完诗之后还要虐 OI，于是 SHY 找来一篇长度为 n 的文章，阅读 m 次，每次只阅读其中连续的一段 $[l,r]$，从这一段中选出一些汉字构成诗。因为 SHY 喜欢对偶，所以 SHY 规定最后选出的每个汉字都必须在 $[l,r]$ 里出现了正偶数次。而且 SHY 认为选出的汉字的种类数（两个一样的汉字称为同一种）越多越好（为了拿到更多的素材！）。于是 SHY 请 LYD 安排选法。

LYD 这种傻×当然不会了，于是向你请教……

问题简述：给定 n 个不大于 c 的正整数 $a_1 \dots a_n$ 和 m 组询问，每次问 $[l,r]$ 中有多少个数出现正偶数次。

对于 100% 的数据，$1 \le n,c,m \le 10^5$，$0 \le a_i \le c$，$1 \le l,r \le n$。

本题强制在线。

对于正偶数次，可以用bitset记录某一段区间内每个数出现次数的奇偶性，不同的区间合并只需要将bitset异或起来

如果直接使用bitset通过RMQ维护每个区间的答案，空间复杂度$O(\text{爆炸})$。我们进行分块，对每个块用一个bitset维护，查询的时候整块之间异或，散块暴力查询

以上算法时间复杂度$O(\frac{c}{w} \times \frac{n}{S} \times m)$ 实测不能通过

我们发现空间有剩余，用st表维护整块的bitset使整块查询优化至单次$O(\frac{c}{w})$，可以通过