

简单 DP 选讲

4182_543_731

2021 年 2 月 20 日

- 相信大家对基础难度的动态规划问题有足够的了解。

- 相信大家对基础难度的动态规划问题有足够的了解。
- 后面的一道题大概留 15+ 分钟的时间。


- 相信大家对基础难度的动态规划问题有足够的了解。
- 后面的一道题大概留 15+ 分钟的时间。
- 有想法的可以直接私发，最好不要公开说正解以免影响他人切题。

- 相信大家对基础难度的动态规划问题有足够的了解。
- 后面的一道题大概留 15+ 分钟的时间。
- 有想法的可以直接私发，最好不要公开说正解以免影响他人切题。
- 由于本人水平极低，相信大家都能切掉 ppt 里的题。

- 相信大家对基础难度的动态规划问题有足够的了解。
- 后面的一道题大概留 15+ 分钟的时间。
- 有想法的可以直接私发，最好不要公开说正解以免影响他人切题。
- 由于本人水平极低，相信大家都能切掉 ppt 里的题。
- 如果您觉得题目过于简单或者您全部会了，您可以提前离场。

- 相信大家对基础难度的动态规划问题有足够的了解。
- 后面的一道题大概留 15+ 分钟的时间。
- 有想法的可以直接私发，最好不要公开说正解以免影响他人切题。
- 由于本人水平极低，相信大家都能切掉 ppt 里的题。
- 如果您觉得题目过于简单或者您全部会了，您可以提前离场。
- 因为本人不太会用 latex 且日常手误，如果有错误请直接指出。

- 相信大家对基础难度的动态规划问题有足够的了解。
- 后面的一道题大概留 15+ 分钟的时间。
- 有想法的可以直接私发，最好不要公开说正解以免影响他人切题。
- 由于本人水平极低，相信大家都能切掉 ppt 里的题。
- 如果您觉得题目过于简单或者您全部会了，您可以提前离场。
- 因为本人不太会用 latex 且日常手误，如果有错误请直接指出。
- 不要喷讲题人。 /kel

- 相信大家对基础难度的动态规划问题有足够的了解。
- 后面的一道题大概留 15+ 分钟的时间。
- 有想法的可以直接私发，最好不要公开说正解以免影响他人切题。
- 由于本人水平极低，相信大家都能切掉 ppt 里的题。
- 如果您觉得题目过于简单或者您全部会了，您可以提前离场。
- 因为本人不太会用 latex 且日常手误，如果有错误请直接指出。
- 不要喷讲题人。 /kel
- 

决策单调性优化

相信大家都会这种东西。

决策单调性优化

相信大家都会这种东西。

称一个二维序列 f 满足四边形不等式，当且仅当：

决策单调性优化

相信大家都会这种东西。

称一个二维序列 f 满足四边形不等式，当且仅当：

$$\forall a < b \leq c < d, a, b, c, d \in \mathbb{N}^+, f_{a,d} + f_{b,c} \geq f_{a,c} + f_{b,d}$$

决策单调性优化

相信大家都会这种东西。

称一个二维序列 f 满足四边形不等式，当且仅当：

$$\forall a < b \leq c < d, a, b, c, d \in \mathbb{N}^+, f_{a,d} + f_{b,c} \geq f_{a,c} + f_{b,d}$$

如果 f 满足四边形不等式，则显然有：

决策单调性优化

相信大家都会这种东西。

称一个二维序列 f 满足四边形不等式，当且仅当：

$$\forall a < b \leq c < d, a, b, c, d \in \mathbb{N}^+, f_{a,d} + f_{b,c} \geq f_{a,c} + f_{b,d}$$

如果 f 满足四边形不等式，则显然有：

$$\forall a < b, a, b \in \mathbb{N}^+, f_{a,b+1} + f_{a+1,b} \geq f_{a,b} + f_{a+1,b+1}$$

决策单调性优化

相信大家都会这种东西。

称一个二维序列 f 满足四边形不等式，当且仅当：

$$\forall a < b \leq c < d, a, b, c, d \in \mathbb{N}^+, f_{a,d} + f_{b,c} \geq f_{a,c} + f_{b,d}$$

如果 f 满足四边形不等式，则显然有：

$$\forall a < b, a, b \in \mathbb{N}^+, f_{a,b+1} + f_{a+1,b} \geq f_{a,b} + f_{a+1,b+1}$$

若 f 满足这个条件，则有 $f_{a,b+1} - f_{a,b} \geq f_{a+1,b+1} - f_{a+1,b}$ 。

决策单调性优化

相信大家都会这种东西。

称一个二维序列 f 满足四边形不等式，当且仅当：

$$\forall a < b \leq c < d, a, b, c, d \in \mathbb{N}^+, f_{a,d} + f_{b,c} \geq f_{a,c} + f_{b,d}$$

如果 f 满足四边形不等式，则显然有：

$$\forall a < b, a, b \in \mathbb{N}^+, f_{a,b+1} + f_{a+1,b} \geq f_{a,b} + f_{a+1,b+1}$$

若 f 满足这个条件，则有 $f_{a,b+1} - f_{a,b} \geq f_{a+1,b+1} - f_{a+1,b}$ 。

因此有 $\forall i \in \mathbb{N}^+, f_{a,b+1} - f_{a,b} \geq f_{a+i,b+1} - f_{a+i,b}$ 。使用类似的方式可以得到 $\forall i, j \in \mathbb{N}^+, f_{a,b+j} - f_{a,b} \geq f_{a+i,b+j} - f_{a+i,b}$ 。这样就得到了第一个式子。

决策单调性优化

相信大家都会这种东西。

称一个二维序列 f 满足四边形不等式，当且仅当：

$$\forall a < b \leq c < d, a, b, c, d \in \mathbb{N}^+, f_{a,d} + f_{b,c} \geq f_{a,c} + f_{b,d}$$

如果 f 满足四边形不等式，则显然有：

$$\forall a < b, a, b \in \mathbb{N}^+, f_{a,b+1} + f_{a+1,b} \geq f_{a,b} + f_{a+1,b+1}$$

若 f 满足这个条件，则有 $f_{a,b+1} - f_{a,b} \geq f_{a+1,b+1} - f_{a+1,b}$ 。

因此有 $\forall i \in \mathbb{N}^+, f_{a,b+1} - f_{a,b} \geq f_{a+i,b+1} - f_{a+i,b}$ 。使用类似的方式可以得到 $\forall i, j \in \mathbb{N}^+, f_{a,b+j} - f_{a,b} \geq f_{a+i,b+j} - f_{a+i,b}$ 。这样就得到了第一个式子。因此判断四边形不等式只需要判断是否满足第二个式子。

决策单调性优化

考虑如下 dp:

决策单调性优化

考虑如下 dp:

$$dp_i = \min g_j + f_{j,i}$$

决策单调性优化

考虑如下 dp:

$$dp_i = \min g_j + f_{j,i}$$

称 j' 为 i 的转移点, 当且仅当 \min 在 j' 处取到。

决策单调性优化

考虑如下 dp:

$$dp_i = \min g_j + f_{j,i}$$

称 j' 为 i 的转移点, 当且仅当 \min 在 j' 处取到。

设 c 的转移点为 b , d 的转移点为 a , $c < d$ 。若 $a < b$, 则有:

决策单调性优化

考虑如下 dp:

$$dp_i = \min g_j + f_{j,i}$$

称 j' 为 i 的转移点, 当且仅当 \min 在 j' 处取到。

设 c 的转移点为 b , d 的转移点为 a , $c < d$ 。若 $a < b$, 则有:

$$dp_c = g_b + f_{b,c}, dp_d = g_a + f_{a,d}$$

决策单调性优化

考虑如下 dp:

$$dp_i = \min g_j + f_{j,i}$$

称 j' 为 i 的转移点, 当且仅当 \min 在 j' 处取到。

设 c 的转移点为 b , d 的转移点为 a , $c < d$ 。若 $a < b$, 则有:

$$dp_c = g_b + f_{b,c}, dp_d = g_a + f_{a,d}$$

但如果设 $dp'_c = g_a + f_{a,c}$, $dp'_d = g_b + f_{b,d}$, 即交换转移点, 则根据四边形不等式显然有 $dp'_c + dp'_d \leq dp_c + dp_d$, 则至少存在一个原转移点不是最优或者两种转移方式相同。

决策单调性优化

考虑如下 dp:

$$dp_i = \min g_j + f_{j,i}$$

称 j' 为 i 的转移点, 当且仅当 \min 在 j' 处取到。

设 c 的转移点为 b , d 的转移点为 a , $c < d$ 。若 $a < b$, 则有:

$$dp_c = g_b + f_{b,c}, dp_d = g_a + f_{a,d}$$

但如果设 $dp'_c = g_a + f_{a,c}$, $dp'_d = g_b + f_{b,d}$, 即交换转移点, 则根据四边形不等式显然有 $dp'_c + dp'_d \leq dp_c + dp_d$, 则至少存在一个原转移点不是最优或者两种转移方式相同。

因此上面的 dp 中, 存在一组转移点 v_1, v_2, \dots, v_n 满足

$$v_1 \leq v_2 \leq \dots \leq v_n$$

决策单调性优化

考虑如下 dp:

$$dp_i = \min g_j + f_{j,i}$$

称 j' 为 i 的转移点, 当且仅当 \min 在 j' 处取到。

设 c 的转移点为 b , d 的转移点为 a , $c < d$ 。若 $a < b$, 则有:

$$dp_c = g_b + f_{b,c}, dp_d = g_a + f_{a,d}$$

但如果设 $dp'_c = g_a + f_{a,c}$, $dp'_d = g_b + f_{b,d}$, 即交换转移点, 则根据四边形不等式显然有 $dp'_c + dp'_d \leq dp_c + dp_d$, 则至少存在一个原转移点不是最优或者两种转移方式相同。

因此上面的 dp 中, 存在一组转移点 v_1, v_2, \dots, v_n 满足

$$v_1 \leq v_2 \leq \dots \leq v_n。$$

将四边形不等式中的 \geq 换成 \leq , dp 中换成 \max , 使用类似的方式可以得到相同的结论。

决策单调性优化

考虑如下 dp:

$$dp_i = \min g_j + f_{j,i}$$

称 j' 为 i 的转移点, 当且仅当 \min 在 j' 处取到。

设 c 的转移点为 b , d 的转移点为 a , $c < d$ 。若 $a < b$, 则有:

$$dp_c = g_b + f_{b,c}, dp_d = g_a + f_{a,d}$$

但如果设 $dp'_c = g_a + f_{a,c}$, $dp'_d = g_b + f_{b,d}$, 即交换转移点, 则根据四边形不等式显然有 $dp'_c + dp'_d \leq dp_c + dp_d$, 则至少存在一个原转移点不是最优或者两种转移方式相同。

因此上面的 dp 中, 存在一组转移点 v_1, v_2, \dots, v_n 满足

$$v_1 \leq v_2 \leq \dots \leq v_n$$

将四边形不等式中的 \geq 换成 \leq , dp 中换成 \max , 使用类似的方式可以得到相同的结论。

如果只反转一侧, 则可以得到反向的决策单调性: 存在一组转移点

$$v_1, v_2, \dots, v_n \text{ 满足 } v_1 \geq v_2 \geq \dots \geq v_n$$

决策单调性优化

有两种经典的做法：

决策单调性优化

有两种经典的做法：

考虑分治，初始需要求出 $[1, n]$ 区间的答案，可能的决策点在 $[1, n]$ 。

决策单调性优化

有两种经典的做法：

考虑分治，初始需要求出 $[1, n]$ 区间的答案，可能的决策点在 $[1, n]$ 。

假设当前需要求出 $[l, r]$ 区间的答案，可能的决策点在 $[x, y]$ 。定义这样的过程为 $solve(l, r, x, y)$

决策单调性优化

有两种经典的做法：

考虑分治，初始需要求出 $[1, n]$ 区间的答案，可能的决策点在 $[1, n]$ 。

假设当前需要求出 $[l, r]$ 区间的答案，可能的决策点在 $[x, y]$ 。定义这样的过程为 $solve(l, r, x, y)$

设 $mid = \lfloor \frac{l+r}{2} \rfloor$ ，求出 mid 对应的决策点 v_{mid} ，根据决策单调性， $> mid$ 的决策点一定 $\geq v_{mid}$ ，小于的类似。因此接下来只需要分别求 $solve(l, mid - 1, x, v_{mid})$, $solve(mid + 1, r, v_{mid}, y)$ 即可。

决策单调性优化

有两种经典的做法：

考虑分治，初始需要求出 $[1, n]$ 区间的答案，可能的决策点在 $[1, n]$ 。

假设当前需要求出 $[l, r]$ 区间的答案，可能的决策点在 $[x, y]$ 。定义这样的过程为 $solve(l, r, x, y)$

设 $mid = \lfloor \frac{l+r}{2} \rfloor$ ，求出 mid 对应的决策点 v_{mid} ，根据决策单调性， $> mid$ 的决策点一定 $\geq v_{mid}$ ，小于的类似。因此接下来只需要分别求 $solve(l, mid - 1, x, v_{mid})$, $solve(mid + 1, r, v_{mid}, y)$ 即可。

显然这个做法的复杂度为 $O(n \log n)$ 。

决策单调性优化

有两种经典的做法:

考虑分治, 初始需要求出 $[1, n]$ 区间的答案, 可能的决策点在 $[1, n]$ 。
假设当前需要求出 $[l, r]$ 区间的答案, 可能的决策点在 $[x, y]$ 。定义这样的过程为 $solve(l, r, x, y)$

设 $mid = \lfloor \frac{l+r}{2} \rfloor$, 求出 mid 对应的决策点 v_{mid} , 根据决策单调性, $> mid$ 的决策点一定 $\geq mid$, 小于的类似。因此接下来只需要分别求 $solve(l, mid - 1, x, v_{mid})$, $solve(mid + 1, r, v_{mid}, y)$ 即可。

显然这个做法的复杂度为 $O(n \log n)$ 。

但在转移不分层的时候 (例如 $dp_i = \min_{j < i} dp_j + f_{j,i}$), 求出 dp_i 必须知道之前的所有 dp 值, 因此不能直接使用分治做法。

决策单调性优化

有两种经典的做法：

考虑分治，初始需要求出 $[1, n]$ 区间的答案，可能的决策点在 $[1, n]$ 。
假设当前需要求出 $[l, r]$ 区间的答案，可能的决策点在 $[x, y]$ 。定义这样的过程为 $solve(l, r, x, y)$

设 $mid = \lfloor \frac{l+r}{2} \rfloor$ ，求出 mid 对应的决策点 v_{mid} ，根据决策单调性， $> mid$ 的决策点一定 $\geq mid$ ，小于的类似。因此接下来只需要分别求 $solve(l, mid - 1, x, v_{mid})$, $solve(mid + 1, r, v_{mid}, y)$ 即可。

显然这个做法的复杂度为 $O(n \log n)$ 。

但在转移不分层的时候（例如 $dp_i = \min_{j < i} dp_j + f_{j,i}$ ），求出 dp_i 必须知道之前的所有 dp 值，因此不能直接使用分治做法。

一个做法是在分治做法之外再加一个 cdq 分治，但这样复杂度不够优秀，此时还有下一个做法：

决策单调性优化

考虑依次求出 dp_1, dp_2, \dots, dp_n 的值。

决策单调性优化

考虑依次求出 dp_1, dp_2, \dots, dp_n 的值。

设当前求出了 $dp_{1,\dots,i}$, 设只考虑 $dp_{1,\dots,i}$ 的转移时, $dp_k (k > i)$ 的转移点为 $v_{k,i}$ 。根据决策单调性, 有 $v_{i+1,i} \leq v_{i+2,i} \leq \dots \leq v_{n,i}$ 。即每个点能转移到的一定是连续的一段 dp_k 。

决策单调性优化

考虑依次求出 dp_1, dp_2, \dots, dp_n 的值。

设当前求出了 $dp_{1,\dots,i}$ ，设只考虑 $dp_{1,\dots,i}$ 的转移时， $dp_k (k > i)$ 的转移点为 $v_{k,i}$ 。根据决策单调性，有 $v_{i+1,i} \leq v_{i+2,i} \leq \dots \leq v_{n,i}$ 。即每个点能转移到的一定是连续的一段 dp_k 。

维护 $[i+1, n]$ 中转移点相同的每一段以及这一段的转移点。在求 dp_{i+1} 时，它的转移点一定在第一段中。然后将第一段的左端点由 $i+1$ 改为 $i+2$ 。考虑求出 dp_{i+1} 后转移点的变化：

决策单调性优化

考虑依次求出 dp_1, dp_2, \dots, dp_n 的值。

设当前求出了 $dp_{1,\dots,i}$ ，设只考虑 $dp_{1,\dots,i}$ 的转移时， $dp_k (k > i)$ 的转移点为 $v_{k,i}$ 。根据决策单调性，有 $v_{i+1,i} \leq v_{i+2,i} \leq \dots \leq v_{n,i}$ 。即每个点能转移到的一定是连续的一段 dp_k 。

维护 $[i+1, n]$ 中转移点相同的每一段以及这一段的转移点。在求 dp_{i+1} 时，它的转移点一定在第一段中。然后将第一段的左端点由 $i+1$ 改为 $i+2$ 。考虑求出 dp_{i+1} 后转移点的变化：

显然有 $v_{i+2,i+1} \leq v_{i+3,i+1} \leq \dots \leq v_{n,i+1}$ ，因此满足 $v_{k,i+1} = i+1$ 的 k 一定是一段后缀。

决策单调性优化

考虑依次求出 dp_1, dp_2, \dots, dp_n 的值。

设当前求出了 $dp_{1,\dots,i}$ ，设只考虑 $dp_{1,\dots,i}$ 的转移时， $dp_k (k > i)$ 的转移点为 $v_{k,i}$ 。根据决策单调性，有 $v_{i+1,i} \leq v_{i+2,i} \leq \dots \leq v_{n,i}$ 。即每个点能转移到的一定是连续的一段 dp_k 。

维护 $[i+1, n]$ 中转移点相同的每一段以及这一段的转移点。在求 dp_{i+1} 时，它的转移点一定在第一段中。然后将第一段的左端点由 $i+1$ 改为 $i+2$ 。考虑求出 dp_{i+1} 后转移点的变化：

显然有 $v_{i+2,i+1} \leq v_{i+3,i+1} \leq \dots \leq v_{n,i+1}$ ，因此满足 $v_{k,i+1} = i+1$ 的 k 一定是一段后缀。

从后往前考虑每一段，如果这一段的开头的决策点会改变，则这一段内的决策点全部会变为 $i+1$ 。这时只需删去这一段。

决策单调性优化

考虑依次求出 dp_1, dp_2, \dots, dp_n 的值。

设当前求出了 $dp_{1,\dots,i}$ ，设只考虑 $dp_{1,\dots,i}$ 的转移时， $dp_k (k > i)$ 的转移点为 $v_{k,i}$ 。根据决策单调性，有 $v_{i+1,i} \leq v_{i+2,i} \leq \dots \leq v_{n,i}$ 。即每个点能转移到的一定是连续的一段 dp_k 。

维护 $[i+1, n]$ 中转移点相同的每一段以及这一段的转移点。在求 dp_{i+1} 时，它的转移点一定在第一段中。然后将第一段的左端点由 $i+1$ 改为 $i+2$ 。考虑求出 dp_{i+1} 后转移点的变化：

显然有 $v_{i+2,i+1} \leq v_{i+3,i+1} \leq \dots \leq v_{n,i+1}$ ，因此满足 $v_{k,i+1} = i+1$ 的 k 一定是一段后缀。

从后往前考虑每一段，如果这一段的开头的决策点会改变，则这一段内的决策点全部会变为 $i+1$ 。这时只需删去这一段。

否则，可以在这一段内二分改变的后缀，这时前面的决策点都不会改变。这时就求出了决策点改变的一段。

决策单调性优化

考虑依次求出 dp_1, dp_2, \dots, dp_n 的值。

设当前求出了 $dp_{1,\dots,i}$ ，设只考虑 $dp_{1,\dots,i}$ 的转移时， $dp_k (k > i)$ 的转移点为 $v_{k,i}$ 。根据决策单调性，有 $v_{i+1,i} \leq v_{i+2,i} \leq \dots \leq v_{n,i}$ 。即每个点能转移到的一定是连续的一段 dp_k 。

维护 $[i+1, n]$ 中转移点相同的每一段以及这一段的转移点。在求 dp_{i+1} 时，它的转移点一定在第一段中。然后将第一段的左端点由 $i+1$ 改为 $i+2$ 。考虑求出 dp_{i+1} 后转移点的变化：

显然有 $v_{i+2,i+1} \leq v_{i+3,i+1} \leq \dots \leq v_{n,i+1}$ ，因此满足 $v_{k,i+1} = i+1$ 的 k 一定是一段后缀。

从后往前考虑每一段，如果这一段的开头的决策点会改变，则这一段内的决策点全部会变为 $i+1$ 。这时只需删去这一段。

否则，可以在这一段内二分改变的后缀，这时前面的决策点都不会改变。这时就求出了决策点改变的一段。

这个做法的复杂度也是 $O(n \log n)$ ，可以解决不分层的问题。

决策单调性优化

考虑依次求出 dp_1, dp_2, \dots, dp_n 的值。

设当前求出了 $dp_{1,\dots,i}$ ，设只考虑 $dp_{1,\dots,i}$ 的转移时， $dp_k (k > i)$ 的转移点为 $v_{k,i}$ 。根据决策单调性，有 $v_{i+1,i} \leq v_{i+2,i} \leq \dots \leq v_{n,i}$ 。即每个点能转移到的一定是连续的一段 dp_k 。

维护 $[i+1, n]$ 中转移点相同的每一段以及这一段的转移点。在求 dp_{i+1} 时，它的转移点一定在第一段中。然后将第一段的左端点由 $i+1$ 改为 $i+2$ 。考虑求出 dp_{i+1} 后转移点的变化：

显然有 $v_{i+2,i+1} \leq v_{i+3,i+1} \leq \dots \leq v_{n,i+1}$ ，因此满足 $v_{k,i+1} = i+1$ 的 k 一定是一段后缀。

从后往前考虑每一段，如果这一段的开头的决策点会改变，则这一段内的决策点全部会变为 $i+1$ 。这时只需删去这一段。

否则，可以在这一段内二分改变的后缀，这时前面的决策点都不会改变。这时就求出了决策点改变的一段。

这个做法的复杂度也是 $O(n \log n)$ ，可以解决不分层的问题。

在处理反向的情况时（即 $v_1 > v_2 > \dots > v_n$ ）， $i+1$ 覆盖的是一段前缀，因此只需要改为在左侧依次删除即可。

决策单调性优化

单调栈/单调队列做法的优势是可以顺序处理出所有 DP 值。

决策单调性优化

单调栈/单调队列做法的优势是可以顺序处理出所有 DP 值。
分治做法的一个优势是，在分治做法中，会一次需要若干个 $f_{l,r}$ 的值，
设上一次求的值为 $f_{l',r'}$ ，定义这一次求 $f_{l,r}$ 的操作中 l, r 的移动距离为
 $|l - l'| + |r - r'|$ 。则可以得到整个过程中的移动距离和为 $O(n \log n)$ 级别。

决策单调性优化

单调栈/单调队列做法的优势是可以顺序处理出所有 DP 值。
分治做法的一个优势是，在分治做法中，会一次需要若干个 $f_{l,r}$ 的值，
设上一次求的值为 $f_{l',r'}$ ，定义这一次求 $f_{l,r}$ 的操作中 l,r 的移动距离为
 $|l-l'| + |r-r'|$ 。则可以得到整个过程中的移动距离和为 $O(n \log n)$ 级别。
因此分治做法可以解决一部分不能直接求出 $f_{l,r}$ 但可以在 l,r 改变 1 时
快速求出新的 $f_{l,r}$ 的 DP 题（例如 CF868F 和例题）。

决策单调性优化

单调栈/单调队列做法的优势是可以顺序处理出所有 DP 值。
分治做法的一个优势是，在分治做法中，会一次需要若干个 $f_{l,r}$ 的值，
设上一次求的值为 $f_{l',r'}$ ，定义这一次求 $f_{l,r}$ 的操作中 l,r 的移动距离为
 $|l-l'| + |r-r'|$ 。则可以得到整个过程中的移动距离和为 $O(n \log n)$ 级别。
因此分治做法可以解决一部分不能直接求出 $f_{l,r}$ 但可以在 l,r 改变 1 时
快速求出新的 $f_{l,r}$ 的 DP 题（例如 CF868F 和例题）。
在部分问题中，存在每个点只能从一个区间转移来的情况，此时可以套
一个线段树分治解决。（例如 [USACO 19 Feb] Mowing Mischief）。

经典例题

你有一个长度为 n 的排列，你可以将这个排列划分成若干段，每一段的代价为这一段内的逆序对数加上 k 。
求所有划分方案中每一段代价和的最小值。
 $n \leq 3 \times 10^5$ ，时间限制 5s。

经典例题

设 $f_{l,r}$ 表示 $[l, r]$ 区间逆序对数加上 k 的值。显然它满足四边形不等式。

经典例题

设 $f_{l,r}$ 表示 $[l, r]$ 区间逆序对数加上 k 的值。显然它满足四边形不等式。由于强制在线区间逆序对不好做，因此考虑分治做法。

经典例题

设 $f_{l,r}$ 表示 $[l, r]$ 区间逆序对数加上 k 的值。显然它满足四边形不等式。
由于强制在线区间逆序对不好做，因此考虑分治做法。
又因为转移不分层，所以先 cdq 分治，在 cdq 分治的每一步做决策单调性的分治做法。

经典例题

设 $f_{l,r}$ 表示 $[l, r]$ 区间逆序对数加上 k 的值。显然它满足四边形不等式。由于强制在线区间逆序对不好做，因此考虑分治做法。又因为转移不分层，所以先 cdq 分治，在 cdq 分治的每一步做决策单调性的分治做法。cdq 分治进行 $O(\log n)$ 层，因此 cdq 分治内的 l, r 移动次数为 $O(n \log^2 n)$ ，不同步之间的移动距离是 $O(n \log n)$ 的，因此总移动次数为 $O(n \log^2 n)$ 。

经典例题

设 $f_{l,r}$ 表示 $[l, r]$ 区间逆序对数加上 k 的值。显然它满足四边形不等式。由于强制在线区间逆序对不好做，因此考虑分治做法。又因为转移不分层，所以先 cdq 分治，在 cdq 分治的每一步做决策单调性的分治做法。

cdq 分治进行 $O(\log n)$ 层，因此 cdq 分治内的 l, r 移动次数为 $O(n \log^2 n)$ ，不同步之间的移动距离是 $O(n \log n)$ 的，因此总移动次数为 $O(n \log^2 n)$ 。

考虑在序列两端插入/删除一个元素并维护逆序对数，只要求出当前区间内比被操作数小的数即可，可以直接维护一个 BIT，这样复杂度为 $O(n \log^3 n)$ ，这东西只跑了 3.5s。

经典例题

设 $f_{l,r}$ 表示 $[l, r]$ 区间逆序对数加上 k 的值。显然它满足四边形不等式。由于强制在线区间逆序对不好做，因此考虑分治做法。又因为转移不分层，所以先 cdq 分治，在 cdq 分治的每一步做决策单调性的分治做法。

cdq 分治进行 $O(\log n)$ 层，因此 cdq 分治内的 l, r 移动次数为 $O(n \log^2 n)$ ，不同步之间的移动距离是 $O(n \log n)$ 的，因此总移动次数为 $O(n \log^2 n)$ 。

考虑在序列两端插入/删除一个元素并维护逆序对数，只要求出当前区间内比被操作数小的数即可，可以直接维护一个 BIT，这样复杂度为 $O(n \log^3 n)$ ，这东西只跑了 3.5s。

询问的东西可以拆成两个询问序列的一个前缀 $[1, \dots, x]$ 中小于 y 的数的个数。通过分块，可以得到一个 $O(\sqrt{n})$ 单点修改 $O(1)$ 区间求和的做法。将这个做法可持久化，即可做到 $O(n\sqrt{n})$ 预处理， $O(1)$ 询问上面的问题。

经典例题

设 $f_{l,r}$ 表示 $[l, r]$ 区间逆序对数加上 k 的值。显然它满足四边形不等式。由于强制在线区间逆序对不好做，因此考虑分治做法。又因为转移不分层，所以先 cdq 分治，在 cdq 分治的每一步做决策单调性的分治做法。

cdq 分治进行 $O(\log n)$ 层，因此 cdq 分治内的 l, r 移动次数为 $O(n \log^2 n)$ ，不同步之间的移动距离是 $O(n \log n)$ 的，因此总移动次数为 $O(n \log^2 n)$ 。

考虑在序列两端插入/删除一个元素并维护逆序对数，只要求出当前区间内比被操作数小的数即可，可以直接维护一个 BIT，这样复杂度为 $O(n \log^3 n)$ ，这东西只跑了 3.5s。

询问的东西可以拆成两个询问序列的一个前缀 $[1, \dots, x]$ 中小于 y 的数的个数。通过分块，可以得到一个 $O(\sqrt{n})$ 单点修改 $O(1)$ 区间求和的做法。将这个做法可持久化，即可做到 $O(n\sqrt{n})$ 预处理， $O(1)$ 询问上面的问题。这样的复杂度为 $O(n\sqrt{n} + n \log^2 n)$ 。

凸序列

定义一个序列 v_0, \dots, v_n 是上凸的, 当且仅当 $\forall i, v_i - v_{i-1} \geq v_{i+1} - v_i$ 。即这个序列差分后不增。类似的, 定义一个序列是下凸的当且仅当 $\forall i, v_i - v_{i-1} \leq v_{i+1} - v_i$ 。

凸序列

定义一个序列 v_0, \dots, v_n 是上凸的, 当且仅当 $\forall i, v_i - v_{i-1} \geq v_{i+1} - v_i$ 。即这个序列差分后不增。类似的, 定义一个序列是下凸的当且仅当

$\forall i, v_i - v_{i-1} \leq v_{i+1} - v_i$ 。

对于一个费用流问题, 设 f_i 表示流量为 i 时的最大费用流, 则根据费用流的做法可以证明 f 是上凸的。

凸序列

定义一个序列 v_0, \dots, v_n 是上凸的, 当且仅当 $\forall i, v_i - v_{i-1} \geq v_{i+1} - v_i$ 。即这个序列差分后不增。类似的, 定义一个序列是下凸的当且仅当

$\forall i, v_i - v_{i-1} \leq v_{i+1} - v_i$ 。

对于一个费用流问题, 设 f_i 表示流量为 i 时的最大费用流, 则根据费用流的做法可以证明 f 是上凸的。

存在一部分 DP 的状态可以看成费用流的模型, 这时的 DP 的某个状态即为上凸序列。而凸序列在转移时可能存在更优的复杂度。

凸序列

例如如下转移 ($\max + \text{卷积}$):

凸序列

例如如下转移 (max + 卷积):

$$f_i = \max_{j+k=i} g_j + h_k$$

凸序列

例如如下转移 (max + 卷积):

$$f_i = \max_{j+k=i} g_j + h_k$$

设序列长度为 n , 正常的做法是 $O(n^2)$ 的。但如果 g, h 均为上凸序列, 则存在更优的做法。

凸序列

例如如下转移 (max + 卷积):

$$f_i = \max_{j+k=i} g_j + h_k$$

设序列长度为 n , 正常的做法是 $O(n^2)$ 的。但如果 g, h 均为上凸序列, 则存在更优的做法。

初始 $f_0 = g_0 + h_0$, 然后可以看成有两个序列 $g_1 - g_0, g_2 - g_1, \dots$ 和 $h_1 - h_0, h_2 - h_1, \dots$ 。求 f_k 相当于求出在两个序列开头分别拿若干个数, 得到的总和的最大值加上 f_0 。

凸序列

例如如下转移 (max + 卷积):

$$f_i = \max_{j+k=i} g_j + h_k$$

设序列长度为 n , 正常的做法是 $O(n^2)$ 的。但如果 g, h 均为上凸序列, 则存在更优的做法。

初始 $f_0 = g_0 + h_0$, 然后可以看成有两个序列 $g_1 - g_0, g_2 - g_1, \dots$ 和 $h_1 - h_0, h_2 - h_1, \dots$ 。求 f_k 相当于求出在两个序列开头分别拿若干个数, 得到的总和的最大值加上 f_0 。

但因为 g, h 上凸, 这两个序列都是不增的。因此每次贪心拿出最大的开头即可。

凸序列

例如如下转移 (max + 卷积):

$$f_i = \max_{j+k=i} g_j + h_k$$

设序列长度为 n , 正常的做法是 $O(n^2)$ 的。但如果 g, h 均为上凸序列, 则存在更优的做法。

初始 $f_0 = g_0 + h_0$, 然后可以看成有两个序列 $g_1 - g_0, g_2 - g_1, \dots$ 和 $h_1 - h_0, h_2 - h_1, \dots$ 。求 f_k 相当于求出在两个序列开头分别拿若干个数, 得到的总和的最大值加上 f_0 。

但因为 g, h 上凸, 这两个序列都是不增的。因此每次贪心拿出最大的开头即可。

因此可以在 $O(n)$ 的时间内求出这个。这个做法相当于在凸壳上求类似闵可夫斯基和。

一个经典例题

给一棵 n 个点的树，边有边权，对于每个 k 求出大小为 k 的匹配的最大权值。

$n \leq 2 \times 10^5$ ，时间限制 $2s$ 。

一个经典例题

记 $f_{u,i}$ 表示 u 子树内匹配 i 对的最大权值, $g_{u,i}$ 表示 u 子树内, 不匹配 u , 匹配 i 对的最大权值。

一个经典例题

记 $f_{u,i}$ 表示 u 子树内匹配 i 对的最大权值, $g_{u,i}$ 表示 u 子树内, 不匹配 u , 匹配 i 对的最大权值。

因为这是一个二分图匹配问题, 显然 f_u, g_u 都是上凸序列。

一个经典例题

记 $f_{u,i}$ 表示 u 子树内匹配 i 对的最大权值, $g_{u,i}$ 表示 u 子树内, 不匹配 u , 匹配 i 对的最大权值。

因为这是一个二分图匹配问题, 显然 f_u, g_u 都是上凸序列。

考虑通过树形 dp 求出所有的 f, g , 将两个子树的 f 合并的过程可以看成做上面的 $\max +$ 卷积。因此可以线性做。

一个经典例题

记 $f_{u,i}$ 表示 u 子树内匹配 i 对的最大权值, $g_{u,i}$ 表示 u 子树内, 不匹配 u , 匹配 i 对的最大权值。

因为这是一个二分图匹配问题, 显然 f_u, g_u 都是上凸序列。

考虑通过树形 dp 求出所有的 f, g , 将两个子树的 f 合并的过程可以看成做上面的 $\max +$ 卷积。因此可以线性做。

但这样的复杂度是 $O(n^2)$ 。

一个经典例题

因为 f, g 中有意义的项数不超过子树大小的一半，可以考虑使用树链剖分优化这个过程。

一个经典例题

因为 f, g 中有意义的项数不超过子树大小的一半，可以考虑使用树链剖分优化这个过程。

只求出每一条重链顶部的点对应的 f, g 值。考虑求出一条重链顶部的 f, g 的过程（此时子树内的所有需要的 f, g 都已求出）：

一个经典例题

因为 f, g 中有意义的项数不超过子树大小的一半，可以考虑使用树链剖分优化这个过程。

只求出每一条重链顶部的点对应的 f, g 值。考虑求出一条重链顶部的 f, g 的过程（此时子树内的所有需要的 f, g 都已求出）：

首先考虑对于重链上的每个点 u ，求出只考虑它的所有轻儿子，可以选 u / 不能选 u 时这部分匹配 $1, 2, \dots$ 对的最大权值 f'_u, g'_u 。

一个经典例题

因为 f, g 中有意义的项数不超过子树大小的一半, 可以考虑使用树链剖分优化这个过程。

只求出每一条重链顶部的点对应的 f, g 值。考虑求出一条重链顶部的 f, g 的过程 (此时子树内的所有需要的 f, g 都已求出):

首先考虑对于重链上的每个点 u , 求出只考虑它的所有轻儿子, 可以选 u / 不能选 u 时这部分匹配 $1, 2, \dots$ 对的最大权值 f'_u, g'_u 。

使用分治 FFT 的思路, 对它的所有轻儿子分治, 每次分成两半, 分别求出只考虑一部分时的 f', g' , 再将这两个部分合并。

一个经典例题

因为 f, g 中有意义的项数不超过子树大小的一半, 可以考虑使用树链剖分优化这个过程。

只求出每一条重链顶部的点对应的 f, g 值。考虑求出一条重链顶部的 f, g 的过程 (此时子树内的所有需要的 f, g 都已求出):

首先考虑对于重链上的每个点 u , 求出只考虑它的所有轻儿子, 可以选 u / 不能选 u 时这部分匹配 $1, 2, \dots$ 对的最大权值 f'_u, g'_u 。

使用分治 FFT 的思路, 对它的所有轻儿子分治, 每次分成两半, 分别求出只考虑一部分时的 f', g' , 再将这两个部分合并。

设有 n 个轻儿子, 它们的子树大小和为 m , 则这样的复杂度为 $O(m \log n)$ 。

一个经典例题

因为 f, g 中有意义的项数不超过子树大小的一半，可以考虑使用树链剖分优化这个过程。

只求出每一条重链顶部的点对应的 f, g 值。考虑求出一条重链顶部的 f, g 的过程（此时子树内的所有需要的 f, g 都已求出）：

首先考虑对于重链上的每个点 u ，求出只考虑它的所有轻儿子，可以选 u / 不能选 u 时这部分匹配 $1, 2, \dots$ 对的最大权值 f'_u, g'_u 。

使用分治 FFT 的思路，对它的所有轻儿子分治，每次分成两半，分别求出只考虑一部分时的 f', g' ，再将这两个部分合并。

设有 n 个轻儿子，它们的子树大小和为 m ，则这样的复杂度为 $O(m \log n)$ 。

一个经典例题

然后考虑链上的转移。

一个经典例题

然后考虑链上的转移。

同样考虑分治，设 $h_{l,r,0/1,0/1}$ 表示只考虑重链上 $[l, r]$ 这一段以及它们的轻儿子，且链上最左侧的点是否不能选，链上最右侧的点是否不能选时，这一段内的匹配情况构成的上凸序列。显然 $h_{l,r}$ 的项数不超过这一段内的所有点的轻儿子子树大小和。

一个经典例题

然后考虑链上的转移。

同样考虑分治，设 $h_{l,r,0/1,0/1}$ 表示只考虑重链上 $[l, r]$ 这一段以及它们的轻儿子，且链上最左侧的点是否不能选，链上最右侧的点是否不能选时，这一段内的匹配情况构成的上凸序列。显然 $h_{l,r}$ 的项数不超过这一段内的所有点的轻儿子子树大小和。

显然使用相同的方式容易将两段 h 合并，因此也可以 $O(m \log n)$ 求出。最后链顶的 f, g 就是 $h_{1,k,1,1}, h_{1,k,0,1}$ 。

一个经典例题

然后考虑链上的转移。

同样考虑分治，设 $h_{l,r,0/1,0/1}$ 表示只考虑重链上 $[l, r]$ 这一段以及它们的轻儿子，且链上最左侧的点是否不能选，链上最右侧的点是否不能选时，这一段内的匹配情况构成的上凸序列。显然 $h_{l,r}$ 的项数不超过这一段内的所有点的轻儿子子树大小和。

显然使用相同的方式容易将两段 h 合并，因此也可以 $O(m \log n)$ 求出。

最后链顶的 f, g 就是 $h_{1,k,1,1}, h_{1,k,0,1}$ 。

因为轻儿子子树大小和不超过 $O(n \log n)$ ，总复杂度为 $O(n \log^2 n)$ 。

设 f 为上凸序列，在某些情况下，求出 f 的复杂度较大，但可以以较小的复杂度求出 $\max f_i + i * x$ 以及取到最大值的 i ，其中 x 为任意实数。

设 f 为上凸序列，在某些情况下，求出 f 的复杂度较大，但可以以较小的复杂度求出 $\max f_i + i * x$ 以及取到最大值的 i ，其中 x 为任意实数。因为 f 为上凸序列，这样求出来的相当于考虑所有点 (i, f_i) 组成的上凸壳，这个凸壳的斜率为 $-x$ 的切线的切点。

设 f 为上凸序列，在某些情况下，求出 f 的复杂度较大，但可以以较小的复杂度求出 $\max f_i + i * x$ 以及取到最大值的 i ，其中 x 为任意实数。因为 f 为上凸序列，这样求出来的相当于考虑所有点 (i, f_i) 组成的上凸壳，这个凸壳的斜率为 $-x$ 的切线的切点。并且根据凸壳的性质容易发现，斜率非常小时切点一定是最右侧的点，随着斜率增大，切点的横坐标逐渐变小，最后变为最左侧的点。

设 f 为上凸序列，在某些情况下，求出 f 的复杂度较大，但可以以较小的复杂度求出 $\max f_i + i * x$ 以及取到最大值的 i ，其中 x 为任意实数。因为 f 为上凸序列，这样求出来的相当于考虑所有点 (i, f_i) 组成的上凸壳，这个凸壳的斜率为 $-x$ 的切线的切点。

并且根据凸壳的性质容易发现，斜率非常小时切点一定是最右侧的点，随着斜率增大，切点的横坐标逐渐变小，最后变为最左侧的点。

考虑求出一个 f_k 的值，二分斜率，如果当前斜率切线的切点横坐标小于它就减小斜率，否则增大斜率。这样可以找到横坐标大于等于 k 的第一个切点。

设 f 为上凸序列，在某些情况下，求出 f 的复杂度较大，但可以以较小的复杂度求出 $\max f_i + i * x$ 以及取到最大值的 i ，其中 x 为任意实数。因为 f 为上凸序列，这样求出来的相当于考虑所有点 (i, f_i) 组成的上凸壳，这个凸壳的斜率为 $-x$ 的切线的切点。

并且根据凸壳的性质容易发现，斜率非常小时切点一定是最右侧的点，随着斜率增大，切点的横坐标逐渐变小，最后变为最左侧的点。

考虑求出一个 f_k 的值，二分斜率，如果当前斜率切线的切点横坐标小于它就减小斜率，否则增大斜率。这样可以找到横坐标大于等于 k 的第一个切点。

如果这个切点就是 k 就做完了，否则 k 一定在左右两个点的连线上，可以算出这个点的 f_k 。

设 f 为上凸序列，在某些情况下，求出 f 的复杂度较大，但可以以较小的复杂度求出 $\max f_i + i * x$ 以及取到最大值的 i ，其中 x 为任意实数。因为 f 为上凸序列，这样求出来的相当于考虑所有点 (i, f_i) 组成的上凸壳，这个凸壳的斜率为 $-x$ 的切线的切点。

并且根据凸壳的性质容易发现，斜率非常小时切点一定是最右侧的点，随着斜率增大，切点的横坐标逐渐变小，最后变为最左侧的点。

考虑求出一个 f_k 的值，二分斜率，如果当前斜率切线的切点横坐标小于它就减小斜率，否则增大斜率。这样可以找到横坐标大于等于 k 的第一个切点。

如果这个切点就是 k 就做完了，否则 k 一定在左右两个点的连线上，可以算出这个点的 f_k 。

复杂度 $O(n \log v)$ 。

一个经典例题

有一个 $n \times m$ 的网格图，边有边权，求大小为 k 的最小权匹配的权值。
 $n \leq 5 \times 10^4, m \leq 4$ 。

一个经典例题

这也是二分图匹配, 因此大小为 $1, 2, \dots, \lfloor \frac{nm}{2} \rfloor$ 的最小权匹配看成序列显然是一个下凸函数。

一个经典例题

这也是二分图匹配，因此大小为 $1, 2, \dots, \lfloor \frac{nm}{2} \rfloor$ 的最小权匹配看成序列显然是一个下凸函数。

然后直接 wqs 二分，一次二分相当于将所有边权加上一个实数，再求出最小权匹配以及匹配的边数。

一个经典例题

这也是二分图匹配，因此大小为 $1, 2, \dots, \lfloor \frac{nm}{2} \rfloor$ 的最小权匹配看成序列显然是一个下凸函数。

然后直接 wqs 二分，一次二分相当于将所有边权加上一个实数，再求出最小权匹配以及匹配的边数。

因为 m 很小，沿着 m 这一维做轮廓线 dp，状态中记录轮廓线上的点有没有被选即可。

一个经典例题

这也是二分图匹配，因此大小为 $1, 2, \dots, \lfloor \frac{nm}{2} \rfloor$ 的最小权匹配看成序列显然是一个下凸函数。

然后直接 wqs 二分，一次二分相当于将所有边权加上一个实数，再求出最小权匹配以及匹配的边数。

因为 m 很小，沿着 m 这一维做轮廓线 dp，状态中记录轮廓线上的点有没有被选即可。

复杂度 $O(nm2^m \log v)$

wqs 二分

wqs 二分不太能输出方案。

wqs 二分

wqs 二分不太能输出方案。

因为不一定能找到 k 处的切点，所以不能和普通的 dp 一样，直接在 dp 时记录转移以得到方案。

wqs 二分

wqs 二分不太能输出方案。

因为不一定能找到 k 处的切点，所以不能和普通的 dp 一样，直接在 dp 时记录转移以得到方案。

但对于一部分情况，可以通过进行调整得到方案。

wqs 二分

wqs 二分不太能输出方案。

因为不一定能找到 k 处的切点，所以不能和普通的 dp 一样，直接在 dp 时记录转移以得到方案。

但对于一部分情况，可以通过进行调整得到方案。

以下这部分是从 dls 讲课抄的。

wqs 二分

考虑这样一类 dp:

考虑这样一类 dp:

$$dp_{n,k} = \min_{i=0}^{n-1} dp_{i,k-1} + f_{i,n}$$

这里 f 满足四边形不等式。

考虑这样一类 dp:

$$dp_{n,k} = \min_{i=0}^{n-1} dp_{i,k-1} + f_{i,n}$$

这里 f 满足四边形不等式。

根据一个结论, 对于任意的 dp_i , dp_i 都是一个下凸函数, 即

$dp_{i,j+1} - dp_{i,j} \geq dp_{i,j} - dp_{i,j-1}$, 且显然给所有 $f_{i,j}$ 加上同一个值不会改变四边形不等式。

考虑这样一类 dp:

$$dp_{n,k} = \min_{i=0}^{n-1} dp_{i,k-1} + f_{i,n}$$

这里 f 满足四边形不等式。

根据一个结论, 对于任意的 dp_i , dp_i 都是一个下凸函数, 即 $dp_{i,j+1} - dp_{i,j} \geq dp_{i,j} - dp_{i,j-1}$, 且显然给所有 $f_{i,j}$ 加上同一个值不会改变四边形不等式。

因此, 对于这样的一类问题, 可以用 wqs 二分 + 单调队列在 $O(n \log n \log v)$ 的复杂度内求出一个 $dp_{n,k}$ 。

考虑这样一类 dp:

$$dp_{n,k} = \min_{i=0}^{n-1} dp_{i,k-1} + f_{i,n}$$

这里 f 满足四边形不等式。

根据一个结论, 对于任意的 dp_i , dp_i 都是一个下凸函数, 即 $dp_{i,j+1} - dp_{i,j} \geq dp_{i,j} - dp_{i,j-1}$, 且显然给所有 $f_{i,j}$ 加上同一个值不会改变四边形不等式。

因此, 对于这样的一类问题, 可以用 wqs 二分 + 单调队列在 $O(n \log n \log v)$ 的复杂度内求出一个 $dp_{n,k}$ 。

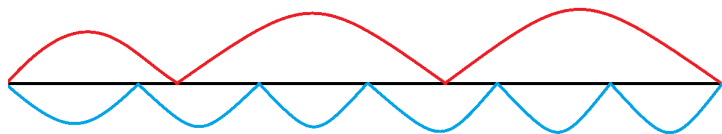
然后考虑求出方案, 显然只需要考虑 k 对应的点在同一条边上的情况。

问题可以看成在一个序列上走 k 步。

问题可以看成在一个序列上走 k 步。
设这时的斜率为 k ，考虑分别求出转移时尽量选走的步数最少的路径，
和转移时尽量选走的步数最多的路径，设这两种方案分别走了 a, b 步。

问题可以看成在一个序列上走 k 步。
设这时的斜率为 k ，考虑分别求出转移时尽量选走的步数最少的路径，
和转移时尽量选走的步数最多的路径，设这两种方案分别走了 a, b 步。
假设两种方案的路径依次如下：

问题可以看成在一个序列上走 k 步。
设这时的斜率为 k ，考虑分别求出转移时尽量选走的步数最少的路径，
和转移时尽量选走的步数最多的路径，设这两种方案分别走了 a, b 步。
假设两种方案的路径依次如下：

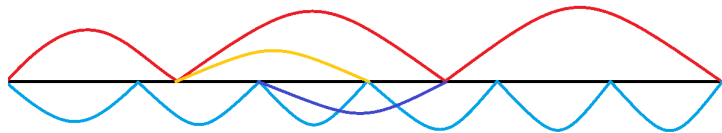


wqs 二分

这时考虑这两条路径：

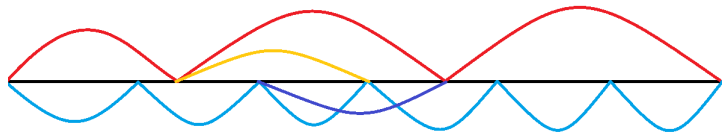
wqs 二分

这时考虑这两条路径：



wqs 二分

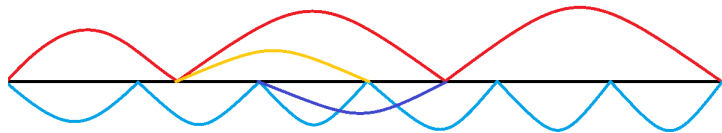
这时考虑这两条路径：



在这个完全被包含的位置调整，之后的路径互相交换。
根据四边形不等式，新加入的两条边权和不大于原来这里的两条边权和。
因此新的两条路径权值和不大于之前的。

wqs 二分

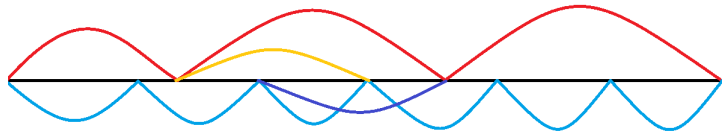
这时考虑这两条路径：



在这个完全被包含的位置调整，之后的路径互相交换。
根据四边形不等式，新加入的两条边权和不大于原来这里的两条边权和。
因此新的两条路径权值和不大于之前的。
因为原来的路径都是最短路，所以这两条都是最短路。因此这两条路径都是合法方案。
注意到在任意一个被完全包含的蓝边处都可以进行调整，在这个调整的位置左侧，每有一个被一条蓝边完全包含的红边，调整后蓝-红路径的长度会 -1 。每有一个被一条红边完全包含的蓝边，调整后蓝-红路径的长度会 $+1$ 。

wqs 二分

这时考虑这两条路径：



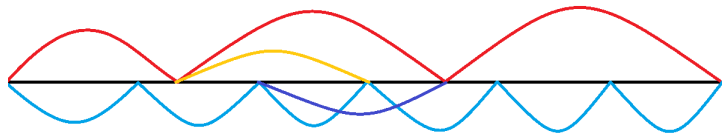
在这个完全被包含的位置调整，之后的路径互相交换。
根据四边形不等式，新加入的两条边权和不大于原来这里的两条边权和。
因此新的两条路径权值和不大于之前的。

因为原来的路径都是最短路，所以这两条都是最短路。因此这两条路径都是合法方案。

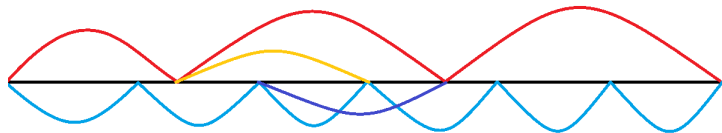
注意到在任意一个被完全包含的蓝边处都可以进行调整，在这个调整的位置左侧，每有一个被一条蓝边完全包含的红边，调整后蓝-红路径的长度会 -1 。每有一个被一条红边完全包含的蓝边，调整后蓝-红路径的长度会 $+1$ 。

易证一定可以找到一条这样的蓝边进行调整，使得得到的一条路径边数为 $k!$!1。

wqs 二分

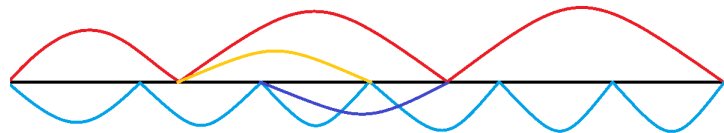


wqs 二分



取蓝-红的这一条路径。那么一定存在一个蓝色路径上的点，使得路径为蓝色路径从开头在这个点的部分拼接上这个点后面的第一个红色点开始到结尾的部分。

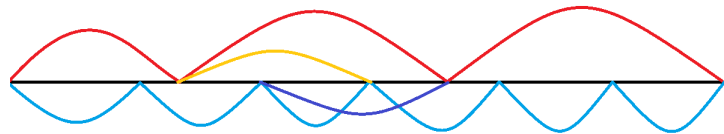
考虑这个点需要满足的条件:



取蓝-红的这一条路径。那么一定存在一个蓝色路径上的点，使得路径为蓝色路径从开头在这个点的部分拼接上这个点后面的第一个红色点开始到结尾的部分。

考虑这个点需要满足的条件:

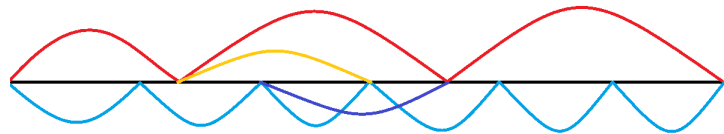
1. 这个点之后的下一个蓝色点比红色点更近。



取蓝-红的这一条路径。那么一定存在一个蓝色路径上的点，使得路径为蓝色路径从开头在这个点的部分拼接上这个点后面的第一个红色点开始到结尾的部分。

考虑这个点需要满足的条件:

1. 这个点之后的下一个蓝色点比红色点更近。
2. 这样拼接出来的路径长度为需要的长度。

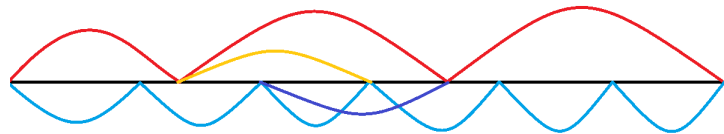


取蓝-红的这一条路径。那么一定存在一个蓝色路径上的点，使得路径为蓝色路径从开头在这个点的部分拼接上这个点后面的第一个红色点开始到结尾的部分。

考虑这个点需要满足的条件：

1. 这个点之后的下一个蓝色点比红色点更近。
2. 这样拼接出来的路径长度为需要的长度。

因为一定有解，所以找一个这样的点即可。



取蓝-红的这一条路径。那么一定存在一个蓝色路径上的点，使得路径为蓝色路径从开头在这个点的部分拼接上这个点后面的第一个红色点开始到结尾的部分。

考虑这个点需要满足的条件：

1. 这个点之后的下一个蓝色点比红色点更近。
2. 这样拼接出来的路径长度为需要的长度。

因为一定有解，所以找一个这样的点即可。

这个调整的复杂度为 $O(n)$ 。

如何证明一个序列是凸序列/满足四边形不等式:

如何证明一个序列是凸序列/满足四边形不等式：
可以直接证明要求的式子或者使用上面的结论。

如何证明一个序列是凸序列/满足四边形不等式：
可以直接证明要求的式子或者使用上面的结论。
最简单有效的方式：猜一个，然后随机数据暴力验证。

如何证明一个序列是凸序列/满足四边形不等式：
可以直接证明要求的式子或者使用上面的结论。
最简单有效的方式：猜一个，然后随机数据暴力验证。
在绝大多数情况下有效。

如何证明一个序列是凸序列/满足四边形不等式：

可以直接证明要求的式子或者使用上面的结论。

最简单有效的方式：猜一个，然后随机数据暴力验证。

在绝大多数情况下有效。

反例：某道题随机数据拍了一小时找出了反例，最后那个做法 60。

最后一个例题

给一个长度为 m 的字符串 T , 对于每个 i , 求出有多少个长度为 n 的字符串 S , 满足 $LCS(S, T) = i$, 模 $10^9 + 7$ 。 LCS 指最长公共子序列。
 $n \leq 1000, m \leq 15$, 字符集大小为 4。

dp 套 dp

这种问题可以看成，给出一个较为简单的 dp，你需要求出有多少种输入使得这个 dp 给出某一个结果。

dp 套 dp

这种问题可以看成，给出一个较为简单的 dp，你需要求出有多少种输入使得这个 dp 给出某一个结果。

考虑求 LCS 的方式，设 $dp_{i,j}$ 表示 S 串的前 i 位和 T 串的前 j 位的 LCS。

dp 套 dp

这种问题可以看成，给出一个较为简单的 dp，你需要求出有多少种输入使得这个 dp 给出某一个结果。

考虑求 LCS 的方式，设 $dp_{i,j}$ 表示 S 串的前 i 位和 T 串的前 j 位的 LCS。显然有 $dp_{i,j} \leq dp_{i,j+1} \leq dp_{i,j} + 1$ ，则 dp_i 的差分是一个 01 序列。又因为 $|T|$ 很小，因此 dp_i 的差分只有 $2^{|T|}$ 种，可以直接用二进制数存。

dp 套 dp

这种问题可以看成，给出一个较为简单的 dp，你要求出有多少种输入使得这个 dp 给出某一个结果。

考虑求 LCS 的方式，设 $dp_{i,j}$ 表示 S 串的前 i 位和 T 串的前 j 位的 LCS。显然有 $dp_{i,j} \leq dp_{i,j+1} \leq dp_{i,j} + 1$ ，则 dp_i 的差分是一个 01 序列。又因为 $|T|$ 很小，因此 dp_i 的差分只有 $2^{|T|}$ 种，可以直接用二进制数存。

因此设 $f_{i,state}$ 表示填了 S 串的前 i 位，当前 LCS 中 dp_i 的状态为 $state$ 的方案数，转移可以预处理。

dp 套 dp

这种问题可以看成，给出一个较为简单的 dp，你要求出有多少种输入使得这个 dp 给出某一个结果。

考虑求 LCS 的方式，设 $dp_{i,j}$ 表示 S 串的前 i 位和 T 串的前 j 位的 LCS。显然有 $dp_{i,j} \leq dp_{i,j+1} \leq dp_{i,j} + 1$ ，则 dp_i 的差分是一个 01 序列。又因为 $|T|$ 很小，因此 dp_i 的差分只有 $2^{|T|}$ 种，可以直接用二进制数存。

因此设 $f_{i,state}$ 表示填了 S 串的前 i 位，当前 LCS 中 dp_i 的状态为 $state$ 的方案数，转移可以预处理。

复杂度 $O((n+m)2^m|\Sigma|)$

dp 套 dp

这种问题可以看成，给出一个较为简单的 dp，你要求出有多少种输入使得这个 dp 给出某一个结果。

考虑求 LCS 的方式，设 $dp_{i,j}$ 表示 S 串的前 i 位和 T 串的前 j 位的 LCS。显然有 $dp_{i,j} \leq dp_{i,j+1} \leq dp_{i,j} + 1$ ，则 dp_i 的差分是一个 01 序列。又因为 $|T|$ 很小，因此 dp_i 的差分只有 $2^{|T|}$ 种，可以直接用二进制数存。

因此设 $f_{i,state}$ 表示填了 S 串的前 i 位，当前 LCS 中 dp_i 的状态为 $state$ 的方案数，转移可以预处理。

复杂度 $O((n+m)2^m|\sum|)$

另外一个非常经典的题目是：求出 $[l, r]$ 中的所有数中，将一个数的每一位分开得到一个序列，这个序列的 LIS 等于某个数的数数量。

dp 套 dp

这种问题可以看成，给出一个较为简单的 dp，你要求出有多少种输入使得这个 dp 给出某一个结果。

考虑求 LCS 的方式，设 $dp_{i,j}$ 表示 S 串的前 i 位和 T 串的前 j 位的 LCS。显然有 $dp_{i,j} \leq dp_{i,j+1} \leq dp_{i,j} + 1$ ，则 dp_i 的差分是一个 01 序列。又因为 $|T|$ 很小，因此 dp_i 的差分只有 $2^{|T|}$ 种，可以直接用二进制数存。

因此设 $f_{i,state}$ 表示填了 S 串的前 i 位，当前 LCS 中 dp_i 的状态为 $state$ 的方案数，转移可以预处理。

复杂度 $O((n+m)2^m|\sum|)$

另外一个非常经典的题目是：求出 $[l, r]$ 中的所有数中，将一个数的每一位分开得到一个序列，这个序列的 LIS 等于某个数的数数量。

相信大家能自己推出这题的做法。

dp 套 dp

这种问题下，可以将内层的 dp 看成一个有限状态确定型自动机 (DFA)。

dp 套 dp

这种问题下，可以将内层的 dp 看成一个有限状态确定型自动机 (DFA)。DFA 包含若干个状态，每个状态对于接下来不同的输入有不同的转移边。某一个状态为起始状态，还有若干接受状态，即停止在这些状态的输入是满足要求的。

dp 套 dp

这种问题下，可以将内层的 dp 看成一个有限状态确定型自动机 (DFA)。DFA 包含若干个状态，每个状态对于接下来不同的输入有不同的转移边。某一个状态为起始状态，还有若干接受状态，即停止在这些状态的输入是满足要求的。

非常多的 dp 问题都可以看成 DFA 的形式，在 dp 套 dp 的问题中，内层 dp 得到的 DFA 状态数一般较小，或者通过合并等价状态得到的状态数很少。(这部分由于篇幅原因咕咕咕了，可以看今年集训队论文或营员交流)

dp 套 dp

这种问题下，可以将内层的 dp 看成一个有限状态确定型自动机 (DFA)。DFA 包含若干个状态，每个状态对于接下来不同的输入有不同的转移边。某一个状态为起始状态，还有若干接受状态，即停止在这些状态的输入是满足要求的。

非常多的 dp 问题都可以看成 DFA 的形式，在 dp 套 dp 的问题中，内层 dp 得到的 DFA 状态数一般较小，或者通过合并等价状态得到的状态数很少。（这部分由于篇幅原因咕咕咕了，可以看今年集训队论文或营员交流）

这类问题中一个更加经典的问题：[ZJOI2019] 麻将。

喜闻乐见.jpg

杂题

喜闻乐见.jpg

因为上面那些东西大部分题目都是板子，所以下面几乎不会用到上面的东西。

杂题

喜闻乐见.jpg

因为上面那些东西大部分题目都是板子，所以下面几乎不会用到上面的东西。

因为我很菜，大部分题都很良心，但不保证难度按顺序排列。

杂题

喜闻乐见.jpg

因为上面那些东西大部分题目都是板子，所以下面几乎不会用到上面的东西。

因为我很菜，大部分题都很良心，但不保证难度按顺序排列。

本来这里会有一点集训队作业的 AGC 题，但因为某些原因就换掉了：

喜闻乐见.jpg

因为上面那些东西大部分题目都是板子，所以下面几乎不会用到上面的东西。

因为我很菜，大部分题都很良心，但不保证难度按顺序排列。

本来这里会有一点集训队作业的 AGC 题，但因为某些原因就换掉了：



感觉可能在场的人都做过集训队作业(?)

一道题

给一个长度为 n 的非负整数序列 a ，你需要将其划分成 k 段。定义第 i 段的和为 su_i ，最大值为 mx_i ，你的划分方案需要满足

$\forall i \in \{1, 2, \dots, n-1\}, |su_i - su_{i+1}| \leq \max(mx_i, mx_{i+1})$ 。

构造一个方案或输出无解。

$3 \leq k \leq n \leq 10^5, 0 \leq a_i \leq 5 \times 10^4$

一道题

考虑一种划分方案，如果存在相邻两段使得
 $|su_i - su_{i+1}| > \max(mx_i, mx_{i+1})$ ，不妨设 $su_i > su_{i+1}$ 。

一道题

考虑一种划分方案，如果存在相邻两段使得

$|su_i - su_{i+1}| > \max(mx_i, mx_{i+1})$ ，不妨设 $su_i > su_{i+1}$ 。

将第 i 段的最后一个非零元素以及之后的部分划分给第 $i+1$ 段，因为这个元素的大小一定小于 $su_i - su_{i+1}$ ，设这样调整后两段的和为 su'_i, su'_{i+1} ，则有 $0 \leq su_{i+1} < su'_i, su'_{i+1} < su_i$ ，且 $su_i + su_{i+1} = su'_i + su'_{i+1}$ 。

一道题

考虑一种划分方案，如果存在相邻两段使得

$|su_i - su_{i+1}| > \max(mx_i, mx_{i+1})$ ，不妨设 $su_i > su_{i+1}$ 。

将第 i 段的最后一个非零元素以及之后的部分划分给第 $i+1$ 段，因为这个元素的大小一定小于 $su_i - su_{i+1}$ ，设这样调整后两段的和为

su'_i, su'_{i+1} ，则有 $0 \leq su_{i+1} < su'_i, su'_{i+1} < su_i$ ，且 $su_i + su_{i+1} = su'_i + su'_{i+1}$ 。

容易发现，此时对于任意大于 1 的实数 k ，都有

$su_i^k + su_{i+1}^k > (su'_i)^k + (su'_{i+1})^k$ 。因此取一个 $\sum su_i^k$ 最小的划分方案，这个方案一定不能再被调整，因此它一定满足条件。

一道题

考虑一种划分方案，如果存在相邻两段使得

$|su_i - su_{i+1}| > \max(mx_i, mx_{i+1})$ ，不妨设 $su_i > su_{i+1}$ 。

将第 i 段的最后一个非零元素以及之后的部分划分给第 $i+1$ 段，因为这个元素的大小一定小于 $su_i - su_{i+1}$ ，设这样调整后两段的和为

su'_i, su'_{i+1} ，则有 $0 \leq su_{i+1} < su'_i, su'_{i+1} < su_i$ ，且 $su_i + su_{i+1} = su'_i + su'_{i+1}$ 。

容易发现，此时对于任意大于 1 的实数 k ，都有

$su_i^k + su_{i+1}^k > (su'_i)^k + (su'_{i+1})^k$ 。因此取一个 $\sum su_i^k$ 最小的划分方案，这个方案一定不能再被调整，因此它一定满足条件。

为了方便，接下来取 $k = 2$ 。

一道题

设 $f_{l,r}$ 表示这一段内元素的和的平方，显然它满足四边形不等式。

一道题

设 $f_{l,r}$ 表示这一段内元素的和的平方, 显然它满足四边形不等式。
根据上面的结论, 设 $dp_{i,j}$ 表示将前 i 个元素分成 j 段, 每一段元素和的平方的最小值。则 dp_n 是一个下凸函数。可以使用 wqs 二分。

一道题

设 $f_{l,r}$ 表示这一段内元素的和的平方，显然它满足四边形不等式。
根据上面的结论，设 $dp_{i,j}$ 表示将前 i 个元素分成 j 段，每一段元素和的平方的最小值。则 dp_n 是一个下凸函数。可以使用 wqs 二分。
和的平方拆开后的形式可以直接斜率优化，因此没有必要再做决策单调性的 $O(n \log n)$ 做法。

一道题

设 $f_{l,r}$ 表示这一段内元素的和的平方，显然它满足四边形不等式。
根据上面的结论，设 $dp_{i,j}$ 表示将前 i 个元素分成 j 段，每一段元素和的平方的最小值。则 dp_n 是一个下凸函数。可以使用 wqs 二分。
和的平方拆开后的形式可以直接斜率优化，因此没有必要再做决策单调性的 $O(n \log n)$ 做法。
最后输出方案的部分可以参考之前讲的。

一道题

设 $f_{l,r}$ 表示这一段内元素的和的平方，显然它满足四边形不等式。
根据上面的结论，设 $dp_{i,j}$ 表示将前 i 个元素分成 j 段，每一段元素和的平方的最小值。则 dp_n 是一个下凸函数。可以使用 wqs 二分。
和的平方拆开后的形式可以直接斜率优化，因此没有必要再做决策单调性的 $O(n \log n)$ 做法。
最后输出方案的部分可以参考之前讲的。
复杂度 $O(n \log v)$ 。

一道题

设 $f_{l,r}$ 表示这一段内元素的和的平方, 显然它满足四边形不等式。
根据上面的结论, 设 $dp_{i,j}$ 表示将前 i 个元素分成 j 段, 每一段元素和的平方的最小值。则 dp_n 是一个下凸函数。可以使用 wqs 二分。
和的平方拆开后的形式可以直接斜率优化, 因此没有必要再做决策单调性的 $O(n \log n)$ 做法。
最后输出方案的部分可以参考之前讲的。
复杂度 $O(n \log v)$ 。
这题的斜率二分边界需要到接近 10^{20}

一道现场过了 80% 的题

给一个长度为 $n - 1$ 的正整数数组 b ，称一个长度为 n 的正实数数组 a 为合法的，当且仅当它满足 $\forall i \in \{1, 2, \dots, n - 1\}, a_i * a_{i+1} \geq b_i$ 。
求出所有合法的 a 中 $\sum a_i$ 的最小值。
 $n \leq 3000, b_i \leq 40000$

一道题

给一棵 n 个点的树，随机一个 n 个点的排列 p ，然后进行如下操作：
维护一个集合 S ，初始为空集。顺序考虑排列中的每一个元素 p_i ，如果 $S \cup \{p_i\}$ 为树的一个独立集，则向 S 中加入 p_i ，否则不做操作。
求出操作结束后 $|S|$ 的期望乘上 $n!$ 的结果，模 $10^9 + 7$ 。
 $n \leq 200$

一道题

给定树 $T_1 = \{V, E_1\}$, $|V| = n$, 假设点集 V 能构成的生成树集合为 S , 求:

$$\sum_{T_2 \in S} |E_1 \cap E_2| * 2^{|E_1 \cap E_2|}$$

答案对 998244353 取模。

$$n \leq 2 \times 10^6$$

一道题

给定 c , 定义一个每个元素都为 $[1, c]$ 中整数的序列 a_1, \dots, a_k 的密度为最大的正整数 m , 使得每一个长度为 m 的每个元素为 $[1, c]$ 中整数的序列都是 a 的子序列。

给一个长度为 n 的序列 v , 考虑它的所有 $2^n - 1$ 个非空子序列, 求出这些序列中密度为 $0, 1, 2, \dots, n$ 的序列个数, 模 998244353。

$n, c \leq 3000$, 时间限制 6s。

提示: 先思考 $O(n^3)$ 的做法。

一道题

有一个 $2 \times n$ 的棋盘，有 c 种颜色。棋盘上有一些格子已经染了色。
你需要给剩下的格子染色，使得不存在两个相邻格子颜色相同，求染色
的方案数，模 $10^9 + 9$ 。
 $5 \leq n, c \leq 10^5$

一道题

给定 n, k 以及 n 对整数 l_i, r_i , k 个非负整数 c_i 。满足 $0 \leq l_i \leq r_i < 2^k$ 。
定义一个长度为 n 的非负整数序列 a 是合法的, 当且仅当 $\forall i, l_i \leq a_i \leq r_i$ 。
定义 $f(i) = \sum_{j=0}^{k-1} (\lfloor \frac{i}{2^j} \rfloor \bmod 2) * c_j$, 即 i 二进制表示上每一个为 1 的位对应的位权值和, 其中从低向高第 j 位的权值为 c_j 。
求出所有合法的序列中, $\sum_{i=1}^{n-1} f(a_i \oplus a_{i+1})$ 的最小值, 这里 \oplus 表示异或。
 $n, k \leq 50, c_j \leq 10^{12}$

一道题

有 n 个单词，第 i 个单词的使用次数为 v_i 。

有三种字符，这三种字符使用一次的代价分别为 2, 2, 3。

你需要给每个单词分配一个由这三种字符组成的字符串，满足任意一个分配的字符串不是另外一个分配的字符串的前缀。

定义一个字符串的代价为字符串每个字符代价的和，定义分配方案的代价为每个单词对应的字符串代价乘以这个字符串的使用次数的和。

求出所有分配方案中最小的代价，以及达到最小代价的分配方案数，方案数对 $10^9 + 9$ 取模。

$n \leq 30, v_i \leq 1000$

因为篇幅原因被放弃的题

给一棵 n 个点的树，每条边的边权在 $[0, 1]$ 间均匀随机。
求直径长度的期望，模 $10^9 + 7$ 。
 $n \leq 100$ ，时间限制 6s。