

动态规划科普

Claris

中国膜 Q 学会

2019 年 1 月 22 日

写在前面

- 科普向。

写在前面

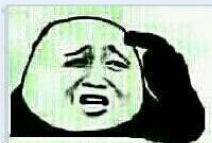
- 科普向。
- 题目都比较简单，欢迎现场秒题。

写在前面

- 科普向。
- 题目都比较简单，欢迎现场秒题。

【绿名】[quailty\(864852302\)](#) 18:46:10

都简单啊



01 背包

n 个物品，每个物品可选可不选，体积为 v_i ，价值为 w_i 。

01 背包

n 个物品，每个物品可选可不选，体积为 v_i ，价值为 w_i 。
求总体积不超过 m 的情况下能拿走物品总价值的最大值。

01 背包

n 个物品，每个物品可选可不选，体积为 v_i ，价值为 w_i 。
求总体积不超过 m 的情况下能拿走物品总价值的最大值。

- $n, m \leq 1000$ 。

Solution

- 状态：设 f_i 表示总体积为 i 的最大总价值。

Solution

- 状态：设 f_i 表示总体积为 i 的最大总价值。
- 依次考虑每个物品，要么选要么不选。

Solution

- 状态：设 f_i 表示总体积为 i 的最大总价值。
- 依次考虑每个物品，要么选要么不选。
- $f_i = \max(f_i, f_{i-v} + w)$ 。

Solution

- 状态：设 f_i 表示总体积为 i 的最大总价值。
- 依次考虑每个物品，要么选要么不选。
- $f_i = \max(f_i, f_{i-v} + w)$ 。
- 需要按照 i 从大到小的顺序更新，确保每个物品只会选一次。

Solution

- 状态：设 f_i 表示总体积为 i 的最大总价值。
- 依次考虑每个物品，要么选要么不选。
- $f_i = \max(f_i, f_{i-v} + w)$ 。
- 需要按照 i 从大到小的顺序更新，确保每个物品只会选一次。
- 时间复杂度 $O(nm)$ 。

01 背包计数

n 个物品，每个物品可选可不选，体积为 v_i 。

01 背包计数

n 个物品，每个物品可选可不选，体积为 v_i 。

求总体积不超过 m 的情况下拿走物品的方案数。

01 背包计数

n 个物品，每个物品可选可不选，体积为 v_i 。

求总体积不超过 m 的情况下拿走物品的方案数。

- $n, m \leq 1000$ 。

Solution

- 状态：设 f_i 表示总体积为 i 的方案数。

Solution

- 状态：设 f_i 表示总体积为 i 的方案数。
- 依次考虑每个物品，要么选要么不选。

Solution

- 状态：设 f_i 表示总体积为 i 的方案数。
- 依次考虑每个物品，要么选要么不选。
- $f_{i+} = f_{i-v_0}$

Solution

- 状态：设 f_i 表示总体积为 i 的方案数。
- 依次考虑每个物品，要么选要么不选。
- $f_{i+} = f_{i-v_0}$ 。
- 需要按照 i 从大到小的顺序更新，确保每个物品只会选一次。

Solution

- 状态：设 f_i 表示总体积为 i 的方案数。
- 依次考虑每个物品，要么选要么不选。
- $f_{i+} = f_{i-v}$ 。
- 需要按照 i 从大到小的顺序更新，确保每个物品只会选一次。
- 时间复杂度 $O(nm)$ 。

EXT

- 如何支持删除物品？

EXT

- 如何支持删除物品？
- 注意到加入物品的顺序不影响结果。

EXT

- 如何支持删除物品？
- 注意到加入物品的顺序不影响结果。
- 假设被删除的物品是最后一次加入的，那么倒过来还原即可。

EXT

- 如何支持删除物品？
- 注意到加入物品的顺序不影响结果。
- 假设被删除的物品是最后一次加入的，那么倒过来还原即可。
- $f_{i-} = f_{i-v_0}$

EXT

- 如何支持删除物品？
- 注意到加入物品的顺序不影响结果。
- 假设被删除的物品是最后一次加入的，那么倒过来还原即可。
- $f_{i-} = f_{i-v_0}$
- 需要按照 i 从小到大的顺序更新。

完全背包

n 个物品，每个物品可以无限选，体积为 v_i ，价值为 w_i 。

完全背包

n 个物品，每个物品可以无限选，体积为 v_i ，价值为 w_i 。
求总体积不超过 m 的情况下能拿走物品总价值的最大值。

完全背包

n 个物品，每个物品可以无限选，体积为 v_i ，价值为 w_i 。
求总体积不超过 m 的情况下能拿走物品总价值的最大值。

- $n, m \leq 1000$ 。

Solution

- 状态：设 f_i 表示总体积为 i 的最大总价值。

Solution

- 状态：设 f_i 表示总体积为 i 的最大总价值。
- 依次考虑每个物品，要么选要么不选。

Solution

- 状态：设 f_i 表示总体积为 i 的最大总价值。
- 依次考虑每个物品，要么选要么不选。
- $f_i = \max(f_i, f_{i-v} + w)$ 。

Solution

- 状态：设 f_i 表示总体积为 i 的最大总价值。
- 依次考虑每个物品，要么选要么不选。
- $f_i = \max(f_i, f_{i-v} + w)$ 。
- 需要按照 i 从小到大的顺序更新，意为要么停止选，要么接着多选一个。

Solution

- 状态：设 f_i 表示总体积为 i 的最大总价值。
- 依次考虑每个物品，要么选要么不选。
- $f_i = \max(f_i, f_{i-v} + w)$ 。
- 需要按照 i 从小到大的顺序更新，意为要么停止选，要么接着多选一个。
- 时间复杂度 $O(nm)$ 。

多重背包

n 个物品，每个物品可以选不超过 k_i 个，体积为 v_i ，价值为 w_i 。

多重背包

n 个物品，每个物品可以选不超过 k_i 个，体积为 v_i ，价值为 w_i 。

求总体积不超过 m 的情况下能拿走物品总价值的最大值。

多重背包

n 个物品，每个物品可以选不超过 k_i 个，体积为 v_i ，价值为 w_i 。

求总体积不超过 m 的情况下能拿走物品总价值的最大值。

- $n, m \leq 1000$ 。

Solution

- 二进制拆分，将一个物品拆成 $O(\log k)$ 个 01 背包的物品。

Solution

- 二进制拆分，将一个物品拆成 $O(\log k)$ 个 01 背包的物品。
- 时间复杂度 $O(nm \log k)$ 。

Solution

- 二进制拆分，将一个物品拆成 $O(\log k)$ 个 01 背包的物品。
- 时间复杂度 $O(nm \log k)$ 。
- 当然也可以按余数分组分别单调队列， $O(nm)$ 。

分组背包

n 个物品，每个物品只能选一个，体积为 v_i ，种类为 k_i 。

分组背包

n 个物品，每个物品只能选一个，体积为 v_i ，种类为 k_i 。
求总体积恰好 m 的情况下能拿走物品种类数的最大值。

分组背包

n 个物品，每个物品只能选一个，体积为 v_i ，种类为 k_i 。
求总体积恰好 m 的情况下能拿走物品种类数的最大值。

- $n, m \leq 1000$ 。

Solution

- 将所有物品按 k 分组。

Solution

- 将所有物品按 k 分组。
- 状态： $f_{i,j,k,s}$ 表示考虑前 i 组，这一组内考虑了前 j 个物品，总体积为 k ，第 i 组物品是否被选择的情况为 s 时，种类数的最大值。

Solution

- 将所有物品按 k 分组。
- 状态： $f_{i,j,k,s}$ 表示考虑前 i 组，这一组内考虑了前 j 个物品，总体积为 k ，第 i 组物品是否被选择的情况为 s 时，种类数的最大值。
- 按照状态定义转移。

Solution

- 将所有物品按 k 分组。
- 状态： $f_{i,j,k,s}$ 表示考虑前 i 组，这一组内考虑了前 j 个物品，总体积为 k ，第 i 组物品是否被选择的情况为 s 时，种类数的最大值。
- 按照状态定义转移。
- 依次考虑每个物品，要么选要么不选。

Solution

- 将所有物品按 k 分组。
- 状态： $f_{i,j,k,s}$ 表示考虑前 i 组，这一组内考虑了前 j 个物品，总体积为 k ，第 i 组物品是否被选择的情况为 s 时，种类数的最大值。
- 按照状态定义转移。
- 依次考虑每个物品，要么选要么不选。
- 时间复杂度 $O(nm)$ 。

树形依赖背包

以 1 为根的树上有 n 个节点，每个节点有一个物品，体积 v_i ，价值 w_i 。

树形依赖背包

以 1 为根的树上有 n 个节点，每个节点有一个物品，体积 v_i ，价值 w_i 。

选了一个点就必须选它的父亲。

树形依赖背包

以 1 为根的树上有 n 个节点，每个节点有一个物品，体积 v_i ，价值 w_i 。

选了一个点就必须选它的父亲。

求总体积不超过 m 的情况下能拿走物品总价值的最大值。

树形依赖背包

以 1 为根的树上有 n 个节点，每个节点有一个物品，体积 v_i ，价值 w_i 。

选了一个点就必须选它的父亲。

求总体积不超过 m 的情况下能拿走物品总价值的最大值。

- $n, m \leq 1000$ 。

Solution

- 按照 DFS 的顺序进行 DP。

Solution

- 按照 DFS 的顺序进行 DP。
- 往下搜的时候，强行将儿子选入背包中。

Solution

- 按照 DFS 的顺序进行 DP。
- 往下搜的时候，强行将儿子选入背包中。
- 往上回溯的时候，可以选择要这棵子树的 DP 值，或者不要。

Solution

- 按照 DFS 的顺序进行 DP。
- 往下搜的时候，强行将儿子选入背包中。
- 往上回溯的时候，可以选择要这棵子树的 DP 值，或者不要。
- 时间复杂度 $O(nm)$ 。

LIS

给定一个长度为 n 的序列 a_1, a_2, \dots, a_n , 从中选出尽量多的位置 , 使得它们的值严格递增。

LIS

给定一个长度为 n 的序列 a_1, a_2, \dots, a_n , 从中选出尽量多的位置 , 使得它们的值严格递增。

- $n \leq 1000$ 。

Solution

- 状态：设 f_i 表示考虑前 i 个位置，且选择第 i 个位置时，数量的最大值。

Solution

- 状态：设 f_i 表示考虑前 i 个位置，且选择第 i 个位置时，数量的最大值。
- 决策：枚举上一个位置接上去。

Solution

- 状态：设 f_i 表示考虑前 i 个位置，且选择第 i 个位置时，数量的最大值。
- 决策：枚举上一个位置接上去。
- 无后效性：与之前选了什么数无关，只考虑最后一个数。

Solution

- 状态：设 f_i 表示考虑前 i 个位置，且选择第 i 个位置时，数量的最大值。
- 决策：枚举上一个位置接上去。
- 无后效性：与之前选了什么数无关，只考虑最后一个数。
- 转移方程： $f_i = \max(f_j) + 1$ ，其中 $1 \leq j < i$ 且 $a_j < a_i$ 。

Solution

- 状态：设 f_i 表示考虑前 i 个位置，且选择第 i 个位置时，数量的最大值。
- 决策：枚举上一个位置接上去。
- 无后效性：与之前选了什么数无关，只考虑最后一个数。
- 转移方程： $f_i = \max(f_j) + 1$ ，其中 $1 \leq j < i$ 且 $a_j < a_i$ 。
- 时间复杂度 $O(n^2)$ 。可以优化到 $O(n \log n)$ 。

Solution

- 维护一个递增的数组 q_1, q_2, \dots, q_m , 一开始为空 , 即 $m = 0$ 。

Solution

- 维护一个递增的数组 q_1, q_2, \dots, q_m , 一开始为空 , 即 $m = 0$ 。
- 从左往右依次考虑每个数 a_i 。

Solution

- 维护一个递增的数组 q_1, q_2, \dots, q_m , 一开始为空 , 即 $m = 0$ 。
- 从左往右依次考虑每个数 a_i 。
- 若 $a_i > q_m$, 直接将其作为 q_{m+1} 。

Solution

- 维护一个递增的数组 q_1, q_2, \dots, q_m , 一开始为空 , 即 $m = 0$ 。
- 从左往右依次考虑每个数 a_i 。
- 若 $a_i > q_m$, 直接将其作为 q_{m+1} 。
- 否则在 q 中二分查找出大于等于 a_i 的最小的数 , 将其替换为 a_i 。

Solution

- 维护一个递增的数组 q_1, q_2, \dots, q_m , 一开始为空 , 即 $m = 0$ 。
- 从左往右依次考虑每个数 a_i 。
- 若 $a_i > q_m$, 直接将其作为 q_{m+1} 。
- 否则在 q 中二分查找出大于等于 a_i 的最小的数 , 将其替换为 a_i 。
- 最终答案为 m , 但是方案不是 q_1, q_2, \dots, q_m 。

Solution

- 维护一个递增的数组 q_1, q_2, \dots, q_m , 一开始为空 , 即 $m = 0$ 。
- 从左往右依次考虑每个数 a_i 。
- 若 $a_i > q_m$, 直接将其作为 q_{m+1} 。
- 否则在 q 中二分查找出大于等于 a_i 的最小的数 , 将其替换为 a_i 。
- 最终答案为 m , 但是方案不是 q_1, q_2, \dots, q_m 。
- 时间复杂度 $O(n \log n)$ 。

k-LIS

给定一个长度为 n 的序列 a_1, a_2, \dots, a_n , 从中选出 k 个递增子序列 , 使得总长度最大。

k-LIS

给定一个长度为 n 的序列 a_1, a_2, \dots, a_n , 从中选出 k 个递增子序列 , 使得总长度最大。

- $n \leq 100000$ 。

k-LIS

给定一个长度为 n 的序列 a_1, a_2, \dots, a_n , 从中选出 k 个递增子序列 , 使得总长度最大。

- $n \leq 100000$ 。
- $k \leq 50$ 。

Solution

- 维护 k 个上述的数组 $q1[], q2[], q3[], \dots, qk[]$ 。

Solution

- 维护 k 个上述的数组 $q1[], q2[], q3[], \dots, qk[]$ 。
- 从左往右依次考虑每个数 a_i 。

Solution

- 维护 k 个上述的数组 $q1[], q2[], q3[], \dots, qk[]$ 。
- 从左往右依次考虑每个数 a_i 。
- 若 $a_i > q1_m$, 直接将其作为 $q1_{m+1}$ 。

Solution

- 维护 k 个上述的数组 $q1[], q2[], q3[], \dots, qk[]$ 。
- 从左往右依次考虑每个数 a_i 。
- 若 $a_i > q1_m$, 直接将其作为 $q1_{m+1}$ 。
- 否则在 $q1$ 中二分查找出大于等于 a_i 的最小的数 x , 将其替换为 a_i 。

Solution

- 维护 k 个上述的数组 $q1[], q2[], q3[], \dots, qk[]$ 。
- 从左往右依次考虑每个数 a_i 。
- 若 $a_i > q1_m$, 直接将其作为 $q1_{m+1}$ 。
- 否则在 $q1$ 中二分查找出大于等于 a_i 的最小的数 x , 将其替换为 a_i 。
- 对于被替换的数 x , 将其放入下一层数组 $q2$ 继续做这个过程。

Solution

- 维护 k 个上述的数组 $q1[], q2[], q3[], \dots, qk[]$ 。
- 从左往右依次考虑每个数 a_i 。
- 若 $a_i > q1_m$, 直接将其作为 $q1_{m+1}$ 。
- 否则在 $q1$ 中二分查找出大于等于 a_i 的最小的数 x , 将其替换为 a_i 。
- 对于被替换的数 x , 将其放入下一层数组 $q2$ 继续做这个过程。
- 最终答案为 k 个数组的元素个数之和。

Solution

- 维护 k 个上述的数组 $q1[], q2[], q3[], \dots, qk[]$ 。
- 从左往右依次考虑每个数 a_i 。
- 若 $a_i > q1_m$, 直接将其作为 $q1_{m+1}$ 。
- 否则在 $q1$ 中二分查找出大于等于 a_i 的最小的数 x , 将其替换为 a_i 。
- 对于被替换的数 x , 将其放入下一层数组 $q2$ 继续做这个过程。
- 最终答案为 k 个数组的元素个数之和。
- 时间复杂度 $O(kn \log n)$ 。

Arcade

给定平面上 n 个点，你要控制遥控车从 $(0,0)$ 出发，以 V_x 的水平速度，速率不高于 V_y 的垂直速度进行移动。

Arcade

给定平面上 n 个点，你要控制遥控车从 $(0,0)$ 出发，以 V_x 的水平速度，速率不高于 V_y 的垂直速度进行移动。

求最多能经过多少个点。

Arcade

给定平面上 n 个点，你要控制遥控车从 $(0,0)$ 出发，以 V_x 的水平速度，速率不高于 V_y 的垂直速度进行移动。

求最多能经过多少个点。

- $n \leq 100000$ 。

Arcade

给定平面上 n 个点，你要控制遥控车从 $(0,0)$ 出发，以 V_x 的水平速度，速率不高于 V_y 的垂直速度进行移动。

求最多能经过多少个点。

- $n \leq 100000$ 。
- $1 \leq V_x, V_y \leq 2^{30}$

Arcade

给定平面上 n 个点，你要控制遥控车从 $(0, 0)$ 出发，以 V_x 的水平速度，速率不高于 V_y 的垂直速度进行移动。

求最多能经过多少个点。

- $n \leq 100000$ 。
- $1 \leq V_x, V_y \leq 2^{30}$
- Source : Petrozavodsk Winter Series 2010–2011 Perm SU Contest

Solution

- 每个点下一步能到达的点在平面上是一个范围。

Solution

- 每个点下一步能到达的点在平面上是一个范围。
- 旋转坐标系变成右上角。

Solution

- 每个点下一步能到达的点在平面上是一个范围。
- 旋转坐标系变成右上角。
- 转化为 LIS 问题。

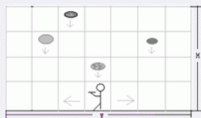
免费的馅饼

SERKOI 最新推出了一种叫做“免费馅饼”的游戏:游戏在一个舞台上进行。舞台的宽度为 W 格 (从左到右依次用 1 到 w 编号), 游戏者占一格。开始时游戏者可以站在舞台的任意位置, 手里拿着一个托盘。下图为天幕的高度为 4 格时某一个时刻游戏者接馅饼的情景。

游戏开始后, 从舞台天幕顶端的格子中不断出现馅饼并垂直下落。游戏者左右移动去接馅饼。游戏者每秒可以向左或向右移动一格或两格, 也可以站在原地不动。

当馅饼在某一时刻恰好到达游戏者所在的格子中, 游戏者就收集到了这块馅饼。当馅饼落在一个游戏者不在的格子里时该馅饼就消失。

写一个程序, 帮助我们的游戏者收集馅饼, 使得所收集馅饼的分数之和最大。



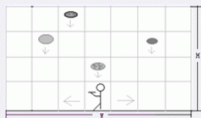
免费的馅饼

SERKOI 最新推出了一种叫做“免费馅饼”的游戏:游戏在一个舞台上进行。舞台的宽度为 W 格 (从左到右依次用 1 到 w 编号), 游戏者占一格。开始时游戏者可以站在舞台的任意位置, 手里拿着一个托盘。下图为天幕的高度为 4 格时某一个时刻游戏者接馅饼的情景。

游戏开始后, 从舞台天幕顶端的格子中不断出现馅饼并垂直下落。游戏者左右移动去接馅饼。游戏者每秒可以向左或向右移动一格或两格, 也可以站在原地不动。

当馅饼在某一时刻恰好到达游戏者所在的格子中, 游戏者就收集到了这块馅饼。当馅饼落在一个游戏者不在的格子里时该馅饼就消失。

写一个程序, 帮助我们的游戏者收集馅饼, 使得所收集馅饼的分数之和最大。



■ $n \leq 100000$.

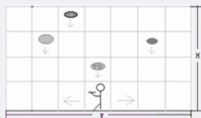
免费的馅饼

SERKOI 最新推出了一种叫做“免费馅饼”的游戏:游戏在一个舞台上进行。舞台的宽度为 W 格 (从左到右依次用 1 到 w 编号), 游戏者占一格。开始时游戏者可以站在舞台的任意位置, 手里拿着一个托盘。下图为天幕的高度为 4 格时某一个时刻游戏者接馅饼的情景。

游戏开始后, 从舞台天幕顶端的格子中不断出现馅饼并垂直下落。游戏者左右移动去接馅饼。游戏者每秒可以向左或向右移动一格或两格, 也可以站在原地不动。

当馅饼在某一时刻恰好到达游戏者所在的格子中, 游戏者就收集到了这块馅饼。当馅饼落在一个游戏者不在的格子里时该馅饼就消失。

写一个程序, 帮助我们的游戏者收集馅饼, 使得所收集馅饼的分数之和最大。



■ $n \leq 100000$ 。

■ Source : BZOJ 2131

Solution

- 将每个馅饼看成 (t_i, p_i) , 即在 t_i 时刻落到舞台上第 p_i 个位置。

Solution

- 将每个馅饼看成 (t_i, p_i) , 即在 t_i 时刻落到舞台上第 p_i 个位置。
- 那么一个点 i 能到达点 $j(t_i < t_j)$, 当且仅当
$$2t_j - 2t_i \geq |p_i - p_j|。$$

Solution

- 将每个馅饼看成 (t_i, p_i) , 即在 t_i 时刻落到舞台上第 p_i 个位置。
- 那么一个点 i 能到达点 $j(t_i < t_j)$, 当且仅当
$$2t_j - 2t_i \geq |p_i - p_j|。$$
- 每个点下一步能到达的点在平面上是一个范围。

Solution

- 将每个馅饼看成 (t_i, p_i) , 即在 t_i 时刻落到舞台上第 p_i 个位置。
- 那么一个点 i 能到达点 $j(t_i < t_j)$, 当且仅当
$$2t_j - 2t_i \geq |p_i - p_j|。$$
- 每个点下一步能到达的点在平面上是一个范围。
- 旋转坐标系变成右上角。

Solution

- 将每个馅饼看成 (t_i, p_i) , 即在 t_i 时刻落到舞台上第 p_i 个位置。
- 那么一个点 i 能到达点 $j(t_i < t_j)$, 当且仅当
$$2t_j - 2t_i \geq |p_i - p_j|。$$
- 每个点下一步能到达的点在平面上是一个范围。
- 旋转坐标系变成右上角。
- 转化为 LIS 问题。

Team Building

两个人 A 和 B 分别有 n 和 m 张卡牌，每张牌上有一个数字。

Team Building

两个人 A 和 B 分别有 n 和 m 张卡牌，每张牌上有一个数字。

两人需要分别选出 k 张卡牌，然后将它们按数字排序后一一配对比较大小。

Team Building

两个人 A 和 B 分别有 n 和 m 张卡牌，每张牌上有一个数字。

两人需要分别选出 k 张卡牌，然后将它们按数字排序后一一配对比较大小。

问有多少种方案，使得 A 的 k 张卡牌的数字每一张都比 B 对应的牌要大。

Team Building

两个人 A 和 B 分别有 n 和 m 张卡牌，每张牌上有一个数字。

两人需要分别选出 k 张卡牌，然后将它们按数字排序后一一配对比较大小。

问有多少种方案，使得 A 的 k 张卡牌的数字每一张都比 B 对应的牌要大。

- $n, m \leq 30000$ 。

Team Building

两个人 A 和 B 分别有 n 和 m 张卡牌，每张牌上有一个数字。

两人需要分别选出 k 张卡牌，然后将它们按数字排序后一一配对比较大小。

问有多少种方案，使得 A 的 k 张卡牌的数字每一张都比 B 对应的牌要大。

- $n, m \leq 30000$ 。
- $k \leq 10$ 。

Team Building

两个人 A 和 B 分别有 n 和 m 张卡牌，每张牌上有一个数字。

两人需要分别选出 k 张卡牌，然后将它们按数字排序后一一配对比较大小。

问有多少种方案，使得 A 的 k 张卡牌的数字每一张都比 B 对应的牌要大。

- $n, m \leq 30000$ 。
- $k \leq 10$ 。
- Source : BZOJ 4742

Solution

- 将两个人的卡牌混在一起从大到小排序，对于数字相同的情况，把 B 的牌放在前面。

Solution

- 将两个人的卡牌混在一起从大到小排序，对于数字相同的情况，把 B 的牌放在前面。
- 那么问题转化为，从排序后的序列中选择 k 个 A 和 B ，使得任意一个前缀中 A 的个数不比 B 的个数少。

Solution

- 将两个人的卡牌混在一起从大到小排序，对于数字相同的情况，把 B 的牌放在前面。
- 那么问题转化为，从排序后的序列中选择 k 个 A 和 B ，使得任意一个前缀中 A 的个数不比 B 的个数少。
- 设 $f_{i,j,x}$ 表示考虑前 i 张牌，选了 j 个 A ， A 的个数减去 B 的个数为 x 的情况。

Solution

- 将两个人的卡牌混在一起从大到小排序，对于数字相同的情况，把 B 的牌放在前面。
- 那么问题转化为，从排序后的序列中选择 k 个 A 和 B ，使得任意一个前缀中 A 的个数不比 B 的个数少。
- 设 $f_{i,j,x}$ 表示考虑前 i 张牌，选了 j 个 A ， A 的个数减去 B 的个数为 x 的情况。
- 枚举第 $i+1$ 张牌是否选择进行转移，状态有效当且仅当 $0 \leq j \leq k$ 且 $x \geq 0$ 。

Solution

- 将两个人的卡牌混在一起从大到小排序，对于数字相同的情况，把 B 的牌放在前面。
- 那么问题转化为，从排序后的序列中选择 k 个 A 和 B ，使得任意一个前缀中 A 的个数不比 B 的个数少。
- 设 $f_{i,j,x}$ 表示考虑前 i 张牌，选了 j 个 A ， A 的个数减去 B 的个数为 x 的情况。
- 枚举第 $i+1$ 张牌是否选择进行转移，状态有效当且仅当 $0 \leq j \leq k$ 且 $x \geq 0$ 。
- $ans = f_{n+m,k,0}$ 。

Solution

- 将两个人的卡牌混在一起从大到小排序，对于数字相同的情况，把 B 的牌放在前面。
- 那么问题转化为，从排序后的序列中选择 k 个 A 和 B ，使得任意一个前缀中 A 的个数不比 B 的个数少。
- 设 $f_{i,j,x}$ 表示考虑前 i 张牌，选了 j 个 A ， A 的个数减去 B 的个数为 x 的情况。
- 枚举第 $i+1$ 张牌是否选择进行转移，状态有效当且仅当 $0 \leq j \leq k$ 且 $x \geq 0$ 。
- $ans = f_{n+m,k,0}$ 。
- 时间复杂度 $O((n+m)(k^2 + \log(n+m)))$ 。

石子合并

n 堆石子从左往右排成一行，第 i 堆有 a_i 个石子。

石子合并

n 堆石子从左往右排成一排，第 i 堆有 a_i 个石子。

每次只能将相邻两堆石子合并，这一步的操作代价为两堆石子数量之和。

石子合并

n 堆石子从左往右排成一排，第 i 堆有 a_i 个石子。

每次只能将相邻两堆石子合并，这一步的操作代价为两堆石子数量之和。

求合并所有石子成一堆的最小总代价。

石子合并

n 堆石子从左往右排成一排，第 i 堆有 a_i 个石子。

每次只能将相邻两堆石子合并，这一步的操作代价为两堆石子数量之和。

求合并所有石子成一堆的最小总代价。

- $n \leq 300$ 。

Solution

- 因为只能合并相邻的石子，因此任意局面下一堆石子一定来自于原来的一个区间。

Solution

- 因为只能合并相邻的石子，因此任意局面下一堆石子一定来自于原来的一个区间。
- 设 $f[l][r]$ 表示将 a_l, a_{l+1}, \dots, a_r 合并成一堆石子的最小代价，则 $ans = f[1][n]$ 。

Solution

- 因为只能合并相邻的石子，因此任意局面下一堆石子一定来自于原来的一个区间。
- 设 $f[l][r]$ 表示将 a_l, a_{l+1}, \dots, a_r 合并成一堆石子的最小代价，则 $ans = f[1][n]$ 。
- 一堆石子最后一步一定是由两堆石子合并而成，所以转移为
$$f[l][r] = \min(f[l][i] + f[i+1][r] + sum(l, i) + sum(i+1, r)) = \min(f[l][i] + f[i+1][r]) + sum(l, r), \text{ 其中 } l \leq i < r。$$

Solution

- 因为只能合并相邻的石子，因此任意局面下一堆石子一定来自于原来的一个区间。
- 设 $f[l][r]$ 表示将 a_l, a_{l+1}, \dots, a_r 合并成一堆石子的最小代价，则 $ans = f[1][n]$ 。
- 一堆石子最后一步一定是由两堆石子合并而成，所以转移为
$$f[l][r] = \min(f[l][i] + f[i+1][r] + sum(l, i) + sum(i+1, r)) = \min(f[l][i] + f[i+1][r]) + sum(l, r), \text{ 其中 } l \leq i < r。$$
- 利用前缀和 $O(1)$ 计算 sum ，总时间复杂度为 $O(n^3)$ 。

Pangu and Stones

n 堆石子从左往右排成一排，第 i 堆有 a_i 个石子。

Pangu and Stones

n 堆石子从左往右排成一排，第 i 堆有 a_i 个石子。

每次只能将连续若干堆石子合并成一堆，这一步的操作代价为这些堆石子数量之和，要求选择的堆数在 $[L, R]$ 之间。

Pangu and Stones

n 堆石子从左往右排成一排，第 i 堆有 a_i 个石子。

每次只能将连续若干堆石子合并成一堆，这一步的操作代价为这些堆石子数量之和，要求选择的堆数在 $[L, R]$ 之间。

求合并所有石子成一堆的最小总代价。

Pangu and Stones

n 堆石子从左往右排成一行，第 i 堆有 a_i 个石子。

每次只能将连续若干堆石子合并成一堆，这一步的操作代价为这些堆石子数量之和，要求选择的堆数在 $[L, R]$ 之间。

求合并所有石子成一堆的最小总代价。

- $n \leq 100$ 。

Pangu and Stones

n 堆石子从左往右排成一行，第 i 堆有 a_i 个石子。

每次只能将连续若干堆石子合并成一堆，这一步的操作代价为这些堆石子数量之和，要求选择的堆数在 $[L, R]$ 之间。

求合并所有石子成一堆的最小总代价。

- $n \leq 100$ 。
- Source : 2017 ACM/ICPC 北京站

Solution

- 相比上一个问题多了数量的限制。

Solution

- 相比上一个问题多了数量的限制。
- 设 $f[l][r][k]$ 表示准备将 a_l, a_{l+1}, \dots, a_r 合并成一堆石子的最小代价，最后一步合并操作由 k 堆石子组成。
设 $g[l][r]$ 表示将 a_l, a_{l+1}, \dots, a_r 合并成一堆石子的最小代价。

Solution

- 相比上一个问题多了数量的限制。
- 设 $f[l][r][k]$ 表示准备将 a_l, a_{l+1}, \dots, a_r 合并成一堆石子的最小代价，最后一步合并操作由 k 堆石子组成。
设 $g[l][r]$ 表示将 a_l, a_{l+1}, \dots, a_r 合并成一堆石子的最小代价。
- 对于 f ，枚举最后一堆所在位置进行转移：
$$f[l][r][k] = \min(f[l][x][k-1] + g[x+1][r]), l \leq x < r.$$

Solution

- 相比上一个问题多了数量的限制。
- 设 $f[l][r][k]$ 表示准备将 a_l, a_{l+1}, \dots, a_r 合并成一堆石子的最小代价，最后一步合并操作由 k 堆石子组成。

设 $g[l][r]$ 表示将 a_l, a_{l+1}, \dots, a_r 合并成一堆石子的最小代价。

- 对于 f ，枚举最后一堆所在位置进行转移：

$$f[l][r][k] = \min(f[l][x][k-1] + g[x+1][r]), l \leq x < r.$$

- 对于 g ，枚举堆数进行转移：

$$g[l][r] = \min(f[l][r][x]) + \text{sum}(l, r), L \leq x \leq R.$$

Solution

- 相比上一个问题多了数量的限制。
- 设 $f[l][r][k]$ 表示准备将 a_l, a_{l+1}, \dots, a_r 合并成一堆石子的最小代价，最后一步合并操作由 k 堆石子组成。
设 $g[l][r]$ 表示将 a_l, a_{l+1}, \dots, a_r 合并成一堆石子的最小代价。
- 对于 f ，枚举最后一堆所在位置进行转移：
$$f[l][r][k] = \min(f[l][x][k-1] + g[x+1][r]), l \leq x < r。$$
- 对于 g ，枚举堆数进行转移：
$$g[l][r] = \min(f[l][r][x]) + \text{sum}(l, r), L \leq x \leq R。$$
- 时间复杂度 $O(n^4)$ 。

Zuma2

n 个珠子从左往右排成一排，第 i 个颜色为 a_i 。

Zuma2

n 个珠子从左往右排成一排，第 i 个颜色为 a_i 。

每次你可以选择连续的一段珠子，要求必须同色，且长度不少于 k ，将它们消去，左右的会合并。

Zuma2

n 个珠子从左往右排成一排，第 i 个颜色为 a_i 。

每次你可以选择连续的一段珠子，要求必须同色，且长度不少于 k ，将它们消去，左右的会合并。

不会发生链式反应，左右合并后即使会消也需要你指定这次操作。

Zuma2

n 个珠子从左往右排成一排，第 i 个颜色为 a_i 。

每次你可以选择连续的一段珠子，要求必须同色，且长度不少于 k ，将它们消去，左右的会合并。

不会发生链式反应，左右合并后即使会消也需要你指定这次操作。

求消掉所有珠子的最少操作次数。

Zuma2

n 个珠子从左往右排成一排，第 i 个颜色为 a_i 。

每次你可以选择连续的一段珠子，要求必须同色，且长度不少于 k ，将它们消去，左右的会合并。

不会发生链式反应，左右合并后即使会消也需要你指定这次操作。

求消掉所有珠子的最少操作次数。

- $n \leq 100, k \leq 6$ 。

Zuma2

n 个珠子从左往右排成一排，第 i 个颜色为 a_i 。

每次你可以选择连续的一段珠子，要求必须同色，且长度不少于 k ，将它们消去，左右的会合并。

不会发生链式反应，左右合并后即使会消也需要你指定这次操作。

求消掉所有珠子的最少操作次数。

- $n \leq 100, k \leq 6$ 。
- Source : BZOJ 2220

Solution

- 考虑一个区间 $[l, r]$, l 一定是最后一步被消去的。

Solution

- 考虑一个区间 $[l, r]$, l 一定是最后一步被消去的。
- 设 $f[l][r][k]$ 表示 $(l, r]$ 消完 , 除了和 l 同色的剩 k 个的最小代价。

设 $g[l][r]$ 表示 $[l, r]$ 消完的最小代价。

Solution

- 考虑一个区间 $[l, r]$, l 一定是最后一步被消去的。
- 设 $f[l][r][k]$ 表示 $(l, r]$ 消完 , 除了和 l 同色的剩 k 个的最小代价。
设 $g[l][r]$ 表示 $[l, r]$ 消完的最小代价。
- 对于 f , 有 $f[l][r][k] = \min(f[l][x][k] + g[x+1][r])$ 。

Solution

- 考虑一个区间 $[l, r]$, l 一定是最后一步被消去的。
- 设 $f[l][r][k]$ 表示 $(l, r]$ 消完 , 除了和 l 同色的剩 k 个的最小代价。

设 $g[l][r]$ 表示 $[l, r]$ 消完的最小代价。

- 对于 f , 有 $f[l][r][k] = \min(f[l][x][k] + g[x+1][r])$ 。
- 特别地 , 当 $a[l] = a[r]$ 时 , 有额外的转移 :
 $f[l][r][k] = \min(f[l][r][k], f[l][r-1][k-1])$ 。

Solution

- 考虑一个区间 $[l, r]$, l 一定是最后一步被消去的。
- 设 $f[l][r][k]$ 表示 $(l, r]$ 消完 , 除了和 l 同色的剩 k 个的最小代价。

设 $g[l][r]$ 表示 $[l, r]$ 消完的最小代价。

- 对于 f , 有 $f[l][r][k] = \min(f[l][x][k] + g[x+1][r])$ 。
- 特别地 , 当 $a[l] = a[r]$ 时 , 有额外的转移 :
$$f[l][r][k] = \min(f[l][r][k], f[l][r-1][k-1])$$
- 对于 g , 有 $g[l][r] = \min(f[l][r][k] + 1, g[l][x] + g[x+1][r])$ 。

Solution

- 考虑一个区间 $[l, r]$, l 一定是最后一步被消去的。
- 设 $f[l][r][k]$ 表示 $(l, r]$ 消完 , 除了和 l 同色的剩 k 个的最小代价。

设 $g[l][r]$ 表示 $[l, r]$ 消完的最小代价。

- 对于 f , 有 $f[l][r][k] = \min(f[l][x][k] + g[x+1][r])$ 。
- 特别地 , 当 $a[l] = a[r]$ 时 , 有额外的转移 :
$$f[l][r][k] = \min(f[l][r][k], f[l][r-1][k-1])$$
- 对于 g , 有 $g[l][r] = \min(f[l][r][k] + 1, g[l][x] + g[x+1][r])$ 。
- 时间复杂度 $O(n^3k)$ 。

Powodz

地面上有个水箱，俯视图是 $n \times m$ 的网格，相邻格子之间有墙，水箱外围墙的高度无限。

Powodz

地面上有个水箱，俯视图是 $n \times m$ 的网格，相邻格子之间有墙，水箱外围墙的高度无限。

已知水箱每个格子的水位是 $[0, H]$ 之间的整数，输入每堵墙的高度，求可能的水位情况总数。

Powodz

地面上有个水箱，俯视图是 $n \times m$ 的网格，相邻格子之间有墙，水箱外围墙的高度无限。

已知水箱每个格子的水位是 $[0, H]$ 之间的整数，输入每堵墙的高度，求可能的水位情况总数。

- $nm \leq 500000$ 。

Powodz

地面上有个水箱，俯视图是 $n \times m$ 的网格，相邻格子之间有墙，水箱外围墙的高度无限。

已知水箱每个格子的水位是 $[0, H]$ 之间的整数，输入每堵墙的高度，求可能的水位情况总数。

- $nm \leq 500000$ 。
- Source : POI 2018

Solution

- 考虑水位上涨的过程，只有最小生成树的树边有用。

Solution

- 考虑水位上涨的过程，只有最小生成树的树边有用。
- 求出 Kruskal 重构树，那么每个点的取值范围都是确定的。

Solution

- 考虑水位上涨的过程，只有最小生成树的树边有用。
- 求出 Kruskal 重构树，那么每个点的取值范围都是确定的。
- 考虑树形 DP，则对于一个点，要么所有点水位相同 (高于墙的高度)，要么还未发生合并 (低于墙的高度)。

Solution

- 考虑水位上涨的过程，只有最小生成树的树边有用。
- 求出 Kruskal 重构树，那么每个点的取值范围都是确定的。
- 考虑树形 DP，则对于一个点，要么所有点水位相同 (高于墙的高度)，要么还未发生合并 (低于墙的高度)。
- $dp[x] = up[x] - down[x] + 1 + dp[l[x]] \times dp[r[x]]$

Solution

- 考虑水位上涨的过程，只有最小生成树的树边有用。
- 求出 Kruskal 重构树，那么每个点的取值范围都是确定的。
- 考虑树形 DP，则对于一个点，要么所有点水位相同 (高于墙的高度)，要么还未发生合并 (低于墙的高度)。
- $dp[x] = up[x] - down[x] + 1 + dp[l[x]] \times dp[r[x]]$
- 时间复杂度 $O(nm \log(nm))$ 。

Sitting in Line

n 个人需要排成一排，第 i 个人手上的数为 a_i ，有些人位置已经确定，有些人位置不确定。

Sitting in Line

n 个人需要排成一排，第 i 个人手上的数为 a_i ，有些人位置已经确定，有些人位置不确定。

请安排不确定的人的位置，使得相邻两个人手上的数的乘积之和最大。

Sitting in Line

n 个人需要排成一排，第 i 个人手上的数为 a_i ，有些人位置已经确定，有些人位置不确定。

请安排不确定的人的位置，使得相邻两个人手上的数的乘积之和最大。

- $2 \leq n \leq 16$ 。

Sitting in Line

n 个人需要排成一排，第 i 个人手上的数为 a_i ，有些人位置已经确定，有些人位置不确定。

请安排不确定的人的位置，使得相邻两个人手上的数的乘积之和最大。

- $2 \leq n \leq 16$ 。
- Source : 2016“百度之星” - 初赛 (Astar Round2A)

Solution

- 从左往右依次填入。

Solution

- 从左往右依次填人。
- 不关心前面每个位置是谁，只关心哪些人还没有位置。

Solution

- 从左往右依次填人。
- 不关心前面每个位置是谁，只关心哪些人还没有位置。
- 设 $f[S][i]$ 表示 S 集合的人的位置已经确定，其中最靠右的是第 i 个人的最大乘积和。

Solution

- 从左往右依次填人。
- 不关心前面每个位置是谁，只关心哪些人还没有位置。
- 设 $f[S][i]$ 表示 S 集合的人的位置已经确定，其中最靠右的是第 i 个人的最大乘积和。
- 枚举下一个人转移。

Solution

- 从左往右依次填人。
- 不关心前面每个位置是谁，只关心哪些人还没有位置。
- 设 $f[S][i]$ 表示 S 集合的人的位置已经确定，其中最靠右的是第 i 个人的最大乘积和。
- 枚举下一个人转移。
- 时间复杂度 $O(2^n n^2)$ 。

旅行商问题

n 个点, m 条边的有向图, 其中有 k 个关键点。

旅行商问题

n 个点, m 条边的有向图, 其中有 k 个关键点。

求从 1 号点出发经过所有关键点至少一次后到达 n 号点的最短路。

旅行商问题

n 个点, m 条边的有向图, 其中有 k 个关键点。

求从 1 号点出发经过所有关键点至少一次后到达 n 号点的最短路。

■ $n, m \leq 100000$ 。

旅行商问题

n 个点, m 条边的有向图, 其中有 k 个关键点。

求从 1 号点出发经过所有关键点至少一次后到达 n 号点的最短路。

- $n, m \leq 100000$ 。
- $k \leq 16$ 。

Solution

- 通过 k 次 Dijkstra 在 $O(km \log m)$ 的时间里求出任意两个关键点之间的最短路。

Solution

- 通过 k 次 Dijkstra 在 $O(km \log m)$ 的时间里求出任意两个关键点之间的最短路。
- 是 $f[S][i]$ 表示经过了 S 集合的关键点，目前位于关键点 i 的最短路。

Solution

- 通过 k 次 Dijkstra 在 $O(km \log m)$ 的时间里求出任意两个关键点之间的最短路。
- 是 $f[S][i]$ 表示经过了 S 集合的关键点，目前位于关键点 i 的最短路。
- 枚举下一个经过的关键点转移。

Solution

- 通过 k 次 Dijkstra 在 $O(km \log m)$ 的时间里求出任意两个关键点之间的最短路。
- 是 $f[S][i]$ 表示经过了 S 集合的关键点，目前位于关键点 i 的最短路。
- 枚举下一个经过的关键点转移。
- 为了方便处理可以将起点和终点也作为关键点。

Solution

- 通过 k 次 Dijkstra 在 $O(km \log m)$ 的时间里求出任意两个关键点之间的最短路。
- 是 $f[S][i]$ 表示经过了 S 集合的关键点，目前位于关键点 i 的最短路。
- 枚举下一个经过的关键点转移。
- 为了方便处理可以将起点和终点也作为关键点。
- 时间复杂度 $O(km \log m + 2^k k^2)$ 。

数位和

设 $f(i)$ 为数字 i 十进制下每一位的和，给定 n ，求 $f(1) + f(2) + \dots + f(n)$ 。

数位和

设 $f(i)$ 为数字 i 十进制下每一位的和，给定 n ，求 $f(1) + f(2) + \dots + f(n)$ 。

- $n \leq 10^{100}$ 。

Solution

- 由于最多只有 100 位，假设 100 位都是 9，那么某个数的 f 值最大不会超过 900。

Solution

- 由于最多只有 100 位，假设 100 位都是 9，那么某个数的 f 值最大不会超过 900。
- 设 $f[i][j][0]$ 表示前 i 位（从高到低）的和为 j ，且目前已经小于 n 的数字的个数。
 $f[i][j][1]$ 表示前 i 位（从高到低）的和为 j ，且目前等于 n 的数字的个数。

Solution

- 由于最多只有 100 位，假设 100 位都是 9，那么某个数的 f 值最大不会超过 900。
- 设 $f[i][j][0]$ 表示前 i 位（从高到低）的和为 j ，且目前已经小于 n 的数字的个数。
 $f[i][j][1]$ 表示前 i 位（从高到低）的和为 j ，且目前等于 n 的数字的个数。
- 初始条件 $f[1][i][i == a[1]] = 1$

Solution

- 由于最多只有 100 位，假设 100 位都是 9，那么某个数的 f 值最大不会超过 900。
- 设 $f[i][j][0]$ 表示前 i 位（从高到低）的和为 j ，且目前已经小于 n 的数字的个数。
 $f[i][j][1]$ 表示前 i 位（从高到低）的和为 j ，且目前等于 n 的数字的个数。
- 初始条件 $f[1][i][i == a[1]] = 1$
- 时间复杂度 $O(\log^2 n)$ 。

Valley Number

当一个数字，从左到右依次看过去数字没有出现先递增接着递减的“山峰”现象，就被称作 Valley Number。它可以递增，也可以递减，还可以先递减再递增。在递增或递减的过程中可以出现相等的情况。

Valley Number

当一个数字，从左到右依次看过去数字没有出现先递增接着递减的“山峰”现象，就被称作 Valley Number。它可以递增，也可以递减，还可以先递减再递增。在递增或递减的过程中可以出现相等的情况。

比如，1, 10, 12, 212, 32122 都是 Valley Number。
121, 12331, 21212 则不是。

Valley Number

当一个数字，从左到右依次看过去数字没有出现先递增接着递减的“山峰”现象，就被称作 Valley Number。它可以递增，也可以递减，还可以先递减再递增。在递增或递减的过程中可以出现相等的情况。

比如，1, 10, 12, 212, 32122 都是 Valley Number。

121, 12331, 21212 则不是。

求 $[1, n]$ 内有多少整数是 Valley Number。

Valley Number

当一个数字，从左到右依次看过去数字没有出现先递增接着递减的“山峰”现象，就被称作 Valley Number。它可以递增，也可以递减，还可以先递减再递增。在递增或递减的过程中可以出现相等的情况。

比如，1, 10, 12, 212, 32122 都是 Valley Number。

121, 12331, 21212 则不是。

求 $[1, n]$ 内有多少整数是 Valley Number。

- $n \leq 10^{100}$ 。

Valley Number

当一个数字，从左到右依次看过去数字没有出现先递增接着递减的“山峰”现象，就被称作 Valley Number。它可以递增，也可以递减，还可以先递减再递增。在递增或递减的过程中可以出现相等的情况。

比如，1, 10, 12, 212, 32122 都是 Valley Number。

121, 12331, 21212 则不是。

求 $[1, n]$ 内有多少整数是 Valley Number。

- $n \leq 10^{100}$ 。
- Source : 2017 百度之星程序设计大赛 - 复赛

Solution

- 依旧从高位到低位考虑。

Solution

- 依旧从高位到低位考虑。
- 状态： $f[i][j][k][x][y]$ 表示考虑了最高的 i 位，第 i 位的数字是 j ，前面是否都是前导 0 的情况为 k ，上升/下降状态为 x ，是否已经小于 n 的情况为 y 的方案数。

Solution

- 依旧从高位到低位考虑。
- 状态： $f[i][j][k][x][y]$ 表示考虑了最高的 i 位，第 i 位的数字是 j ，前面是否都是前导 0 的情况为 k ，上升/下降状态为 x ，是否已经小于 n 的情况为 y 的方案数。
- 时间复杂度 $O(\log n)$ 。

Balanced Numbers

一个数被称为是平衡的数当且仅当对于所有出现过的数位，偶数出现奇数次，奇数出现偶数次。

Balanced Numbers

一个数被称为是平衡的数当且仅当对于所有出现过的数位，偶数出现奇数次，奇数出现偶数次。

给定 A, B ，请统计出 $[A, B]$ 内所有平衡的数的个数。

Balanced Numbers

一个数被称为是平衡的数当且仅当对于所有出现过的数位，偶数出现奇数次，奇数出现偶数次。

给定 A, B ，请统计出 $[A, B]$ 内所有平衡的数的个数。

- $1 \leq A \leq B \leq 10^{18}$ 。

Balanced Numbers

一个数被称为是平衡的数当且仅当对于所有出现过的数位，偶数出现奇数次，奇数出现偶数次。

给定 A, B ，请统计出 $[A, B]$ 内所有平衡的数的个数。

- $1 \leq A \leq B \leq 10^{18}$ 。
- Source : SPOJ 10606

Solution

- 设 $cal(n)$ 为 $[1, n]$ 中平衡的数的个数，那么
 $ans = cal(B) - cal(A - 1)$ 。

Solution

- 设 $cal(n)$ 为 $[1, n]$ 中平衡的数的个数，那么
 $ans = cal(B) - cal(A - 1)$ 。
- 考虑如何计算 $cal(n)$ ，对于每个数字，定义状态 0 为没出现过，1 为出现了奇数次，2 为出现了偶数次，那么可以用一个 10 位 3 进制数来表示所有数字的状态。

Solution

- 设 $cal(n)$ 为 $[1, n]$ 中平衡的数的个数，那么
 $ans = cal(B) - cal(A - 1)$ 。
- 考虑如何计算 $cal(n)$ ，对于每个数字，定义状态 0 为没出现过，1 为出现了奇数次，2 为出现了偶数次，那么可以用一个 10 位 3 进制数来表示所有数字的状态。
- 设 $f(i, S, 0)$ 表示从高到低填了前 i 位，所有数字的状态为 S ，且当前数字已经小于 n 的数字个数。 $f(i, S, 1)$ 表示从高到低填了前 i 位，所有数字的状态为 S ，且当前数字等于 n 的数字个数，然后 DP 即可。

Solution

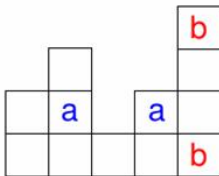
- 设 $cal(n)$ 为 $[1, n]$ 中平衡的数的个数，那么
 $ans = cal(B) - cal(A - 1)$ 。
- 考虑如何计算 $cal(n)$ ，对于每个数字，定义状态 0 为没出现过，1 为出现了奇数次，2 为出现了偶数次，那么可以用一个 10 位 3 进制数来表示所有数字的状态。
- 设 $f(i, S, 0)$ 表示从高到低填了前 i 位，所有数字的状态为 S ，且当前数字已经小于 n 的数字个数。 $f(i, S, 1)$ 表示从高到低填了前 i 位，所有数字的状态为 S ，且当前数字等于 n 的数字个数，然后 DP 即可。
- 时间复杂度 $O(3^{10} \log B)$ 。

Periodni

n 个底边长度为 1 , 高度为 h_i 的长条拼成一个棋盘。

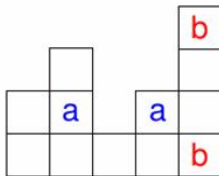
Periodni

n 个底边长度为 1 , 高度为 h_i 的长条拼成一个棋盘。



Periodni

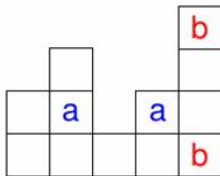
n 个底边长度为 1 , 高度为 h_i 的长条拼成一个棋盘。



两个车能相互攻击当且仅当它们在同一行或者同一列，且它们之间所有格子均存在。

Periodni

n 个底边长度为 1 , 高度为 h_i 的长条拼成一个棋盘。

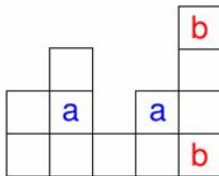


两个车能相互攻击当且仅当它们在同一行或者同一列，且它们之间所有格子均存在。

现在要在棋盘上放置恰好 k 个相互不攻击的车，求方案数。

Periodni

n 个底边长度为 1 , 高度为 h_i 的长条拼成一个棋盘。



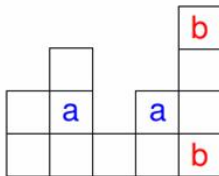
两个车能相互攻击当且仅当它们在同一行或者同一列，且它们之间所有格子均存在。

现在要在棋盘上放置恰好 k 个相互不攻击的车，求方案数。

- $n, k \leq 500, h_i \leq 10^6$ 。

Periodni

n 个底边长度为 1 , 高度为 h_i 的长条拼成一个棋盘。



两个车能相互攻击当且仅当它们在同一行或者同一列，且它们之间所有格子均存在。

现在要在棋盘上放置恰好 k 个相互不攻击的车，求方案数。

■ $n, k \leq 500, h_i \leq 10^6$ 。

■ Source : COCI 2008

Solution

- 不断将最矮的那一列以及下面的矩形取出，得到一棵树的结构。

Solution

- 不断将最矮的那一列以及下面的矩形取出，得到一棵树的结构。
- 假如已经知道了儿子中车的分布情况，那么父亲对应的矩形中还能放的车的列数是确定的。

Solution

- 不断将最矮的那一列以及下面的矩形取出，得到一棵树的结构。
- 假如已经知道了儿子中车的分布情况，那么父亲对应的矩形中还能放的车的列数是确定的。
- 一个长为 a ，宽为 b 的矩形中放置 k 个车的方案数为 $C(a, k)C(b, k)k!$ 。

Solution

- 不断将最矮的那一列以及下面的矩形取出，得到一棵树的结构。
- 假如已经知道了儿子中车的分布情况，那么父亲对应的矩形中还能放的车的列数是确定的。
- 一个长为 a ，宽为 b 的矩形中放置 k 个车的方案数为 $C(a, k)C(b, k)k!$ 。
- 树形 DP，设 $f_{i,j}$ 表示 i 子树中放了 j 个车的方案数。

Solution

- 不断将最矮的那一列以及下面的矩形取出，得到一棵树的结构。
- 假如已经知道了儿子中车的分布情况，那么父亲对应的矩形中还能放的车的列数是确定的。
- 一个长为 a ，宽为 b 的矩形中放置 k 个车的方案数为 $C(a, k)C(b, k)k!$ 。
- 树形 DP，设 $f_{i,j}$ 表示 i 子树中放了 j 个车的方案数。
- 首先将子树合并， $f_{a,i} \times f_{b,j} \rightarrow f_{c,i+j}$ 。

Solution

- 不断将最矮的那一列以及下面的矩形取出，得到一棵树的结构。
- 假如已经知道了儿子中车的分布情况，那么父亲对应的矩形中还能放的车的列数是确定的。
- 一个长为 a ，宽为 b 的矩形中放置 k 个车的方案数为 $C(a, k)C(b, k)k!$ 。
- 树形 DP，设 $f_{i,j}$ 表示 i 子树中放了 j 个车的方案数。
- 首先将子树合并， $f_{a,i} \times f_{b,j} \rightarrow f_{c,i+j}$ 。
- 然后考虑父亲矩形中放置多少个车。

Solution

- 不断将最矮的那一列以及下面的矩形取出，得到一棵树的结构。
- 假如已经知道了儿子中车的分布情况，那么父亲对应的矩形中还能放的车的列数是确定的。
- 一个长为 a ，宽为 b 的矩形中放置 k 个车的方案数为 $C(a, k)C(b, k)k!$ 。
- 树形 DP，设 $f_{i,j}$ 表示 i 子树中放了 j 个车的方案数。
- 首先将子树合并， $f_{a,i} \times f_{b,j} \rightarrow f_{c,i+j}$ 。
- 然后考虑父亲矩形中放置多少个车。
- 时间复杂度 $O(n^3 + h)$ 。

Thank you!