动态规划初步

Claris

Hangzhou Dianzi University

2018 年 2 月 2 日

Overview

- 动态规划是一种思想。

Overview

- 动态规划是一种思想。
- 最优子结构 (状态的确立)。

Overview

- 动态规划是一种思想。
- 最优子结构 (状态的确立)。
- 无后效性。

Overview

- 动态规划是一种思想。
- 最优子结构 (状态的确立)。
- 无后效性。
- 状态转移方程。

Overview

- 动态规划是一种思想。
- 最优子结构 (状态的确立)。
- 无后效性。
- 状态转移方程。
- 递归法与递推法。

Maximum sum with swaps

给定一个长度为 n 的序列 $a_1, a_2, ..., a_n$ ，你有 k 次机会选择一个 x 和 $y(1 \leq x < y \leq n)$ ，交换 a_x 和 a_y。

Maximum sum with swaps

给定一个长度为 n 的序列 $a_1, a_2, \ldots, a_n$ ，你有 k 次机会选择一个 x 和 $y(1 \leq x < y \leq n)$ ，交换 a_x 和 a_y。

k 次机会不一定要全部用完，你需要操作完之后选择一个区间 $[l, r]$ 使得 $a_l + a_{l+1} + \ldots + a_r$ 最大。

Maximum sum with swaps

给定一个长度为 n 的序列 $a_1, a_2, ..., a_n$，你有 k 次机会选择一个 x 和 $y(1 \leq x < y \leq n)$，交换 a_x 和 a_y。

k 次机会不一定要全部用完，你需要操作完之后选择一个区间 $[l, r]$ 使得 $a_l + a_{l+1} + ... + a_r$ 最大。

- $n \leq 10000$。

Maximum sum with swaps

给定一个长度为 n 的序列 $a_1, a_2, \ldots, a_n$，你有 k 次机会选择一个 x 和 $y(1 \le x < y \le n)$，交换 a_x 和 a_y。

k 次机会不一定要全部用完，你需要操作完之后选择一个区间 $[l, r]$ 使得 $a_l + a_{l+1} + \ldots + a_r$ 最大。

- $n \le 10000$。
- $k \le 10$。

Maximum sum with swaps

给定一个长度为 n 的序列 $a_1, a_2, ..., a_n$，你有 k 次机会选择一个 x 和 $y(1 \le x < y \le n)$，交换 a_x 和 a_y。

k 次机会不一定要全部用完，你需要操作完之后选择一个区间 $[l, r]$ 使得 $a_l + a_{l+1} + ... + a_r$ 最大。

- $n \le 10000$。
- $k \le 10$。
- Source：XVII Open Cup named after E.V. Pankratiev. Grand Prix of Eurasia

k=0 Solution

- $k = 0$ 时，只需要求最大子段和。

k=0 Solution

- $k = 0$ 时，只需要求最大子段和。
- 状态：f_i 表示考虑 $[1, i]$，且选中的区间包括 i 的最大子段和。

k=0 Solution

- $k = 0$ 时，只需要求最大子段和。
- 状态：f_i 表示考虑 $[1, i]$，且选中的区间包括 i 的最大子段和。
- 无后效性：与之前从何处开始无关，只关心最大和。

k=0 Solution

- $k = 0$ 时，只需要求最大子段和。
- 状态：f_i 表示考虑 $[1, i]$，且选中的区间包括 i 的最大子段和。
- 无后效性：与之前从何处开始无关，只关心最大和。
- 转移方程：$f_i = \max(f_{i-1}, 0) + a_i$，意为要么从 i 处开始，要么从 $i - 1$ 延伸过来。

k=0 Solution

- $k = 0$ 时，只需要求最大子段和。
- 状态：f_i 表示考虑 $[1, i]$，且选中的区间包括 i 的最大子段和。
- 无后效性：与之前从何处开始无关，只关心最大和。
- 转移方程：$f_i = \max(f_{i-1}, 0) + a_i$，意为要么从 i 处开始，要么从 $i - 1$ 延伸过来。
- $ans = \max_{i=1}^{n} f_i$。

k=0 Solution

- $k = 0$ 时，只需要求最大子段和。
- 状态：f_i 表示考虑 $[1, i]$，且选中的区间包括 i 的最大子段和。
- 无后效性：与之前从何处开始无关，只关心最大和。
- 转移方程：$f_i = \max(f_{i-1}, 0) + a_i$，意为要么从 i 处开始，要么从 $i - 1$ 延伸过来。
- $ans = \max_{i=1}^{n} f_i$。
- 时间复杂度 $O(n)$。

Naive Solution

- $k \leq 10$ 时，考虑暴力做法。

Naive Solution

- $k \le 10$ 时，考虑暴力做法。
- 枚举最终的区间 $[l, r]$。

Naive Solution

- $k \leq 10$ 时，考虑暴力做法。
- 枚举最终的区间 $[l, r]$。
- 不断将 $[l, r]$ 内最小的数换成外面最大的数。

Naive Solution

- $k \leq 10$ 时，考虑暴力做法。
- 枚举最终的区间 $[l, r]$。
- 不断将 $[l, r]$ 内最小的数换成外面最大的数。
- 即有 $t \leq k$ 个 $[l, r]$ 内的数不算入，而有另外 t 个 $[l, r]$ 外的数算入。

Solution

- 从左往右依次考虑每个 a_i。

Solution

- 从左往右依次考虑每个 a_i。
- 当前的 i 与 $[l, r]$ 的关系只有 3 种：$i < l, l \leq i \leq r, r < i$。

Solution

- 从左往右依次考虑每个 a_i。
- 当前的 i 与 $[l, r]$ 的关系只有 3 种：$i < l, l \leq i \leq r, r < i$。
- 状态：$f_{i,j,k,S}$ 表示考虑了 $[1, i]$，有 j 个不在 $[l, r]$ 内的数算入，k 个在 $[l, r]$ 的数不算入，i 与 $[l, r]$ 的关系为 S 的最大和。

Solution

- 从左往右依次考虑每个 a_i。
- 当前的 i 与 $[l,r]$ 的关系只有 3 种：$i<l$, $l\le i\le r$, $r<i$。
- 状态：$f_{i,j,k,S}$ 表示考虑了 $[1,i]$，有 j 个不在 $[l,r]$ 内的数算入，k 个在 $[l,r]$ 的数不算入，i 与 $[l,r]$ 的关系为 S 的最大和。
- 转移：从 i 递推到 $i+1$，对于每个状态考虑 j,k,S 的变化。

Solution

- 从左往右依次考虑每个 a_i。
- 当前的 i 与 $[l,r]$ 的关系只有 3 种：$i<l, l\le i\le r, r<i$。
- 状态：$f_{i,j,k,S}$ 表示考虑了 $[1,i]$，有 j 个不在 $[l,r]$ 内的数算入，k 个在 $[l,r]$ 的数不算入，i 与 $[l,r]$ 的关系为 S 的最大和。
- 转移：从 i 递推到 $i+1$，对于每个状态考虑 j,k,S 的变化。
- 时间复杂度 $O(nk^2)$。

货币系统

一个货币系统要求一共有 m 种货币，并且将它们按照币值从小到大排好序以后，前一个货币币值乘上 x 等于后一个货币币值，$x \in \{2, 3, 4, 5\}$，且最小的币值一定为 1。

货币系统

一个货币系统要求一共有 m 种货币，并且将它们按照币值从小到大排好序以后，前一个货币币值乘上 x 等于后一个货币币值，$x \in \{2, 3, 4, 5\}$，且最小的币值一定为 1。

请设计一个货币系统，使得它表示总币值为 n 的钱所需的货币总张数最少。

货币系统

一个货币系统要求一共有 m 种货币，并且将它们按照币值从小到大排好序以后，前一个货币币值乘上 x 等于后一个货币币值，$x \in \{2, 3, 4, 5\}$，且最小的币值一定为 1。

请设计一个货币系统，使得它表示总币值为 n 的钱所需的货币总张数最少。

- $n \leq 10^{18}$。

货币系统

一个货币系统要求一共有 m 种货币，并且将它们按照币值从小到大排好序以后，前一个货币币值乘上 x 等于后一个货币币值，$x \in \{2, 3, 4, 5\}$，且最小的币值一定为 1。

请设计一个货币系统，使得它表示总币值为 n 的钱所需的货币总张数最少。

- $n \le 10^{18}$。
- $m \le 100$。

货币系统

一个货币系统要求一共有 m 种货币，并且将它们按照币值从小到大排好序以后，前一个货币币值乘上 x 等于后一个货币币值，$x \in \{2, 3, 4, 5\}$，且最小的币值一定为 1。

请设计一个货币系统，使得它表示总币值为 n 的钱所需的货币总张数最少。

- $n \leq 10^{18}$。
- $m \leq 100$。
- Source：BZOJ 4265

Solution

- 如果知道了相邻两个币值的比值 x，那么因为都是倍数关系，可以将 n 写成特殊的 m 进制。

Solution

- 如果知道了相邻两个币值的比值 x，那么因为都是倍数关系，可以将 n 写成特殊的 m 进制。
- 即：将 n 进制分解，使得每一位的和最小。

Solution

- 如果知道了相邻两个币值的比值 x，那么因为都是倍数关系，可以将 n 写成特殊的 m 进制。
- 即：将 n 进制分解，使得每一位的和最小。
- 进制分解的方法：从最低位开始，不断将 n 对 x 的余数取出，然后将 n 除以 x。

Solution

- 如果知道了相邻两个币值的比值 x，那么因为都是倍数关系，可以将 n 写成特殊的 m 进制。
- 即：将 n 进制分解，使得每一位的和最小。
- 进制分解的方法：从最低位开始，不断将 n 对 x 的余数取出，然后将 n 除以 x。
- 注意到只跟当前的 n 以及已经确定的位数有关。

Solution

- 状态：$f_{i,j}$ 表示从低到高确定了 i 个 x，且 n 被除成了 j 时，已经分解出来的和的最小值。

Solution

- 状态：$f_{i,j}$ 表示从低到高确定了 i 个 x，且 n 被除成了 j 时，已经分解出来的和的最小值。
- 转移：枚举 $x \in \{2,3,4,5\}$，$f_{i,j} + j \bmod x \to f_{i+1,\lfloor \frac{j}{x} \rfloor}$。

Solution

- 状态：$f_{i,j}$ 表示从低到高确定了 i 个 x，且 n 被除成了 j 时，已经分解出来的和的最小值。
- 转移：枚举 $x \in \{2,3,4,5\}$，$f_{i,j} + j \bmod x \to f_{i+1,\lfloor \frac{j}{x} \rfloor}$。
- 初始状态：$f_{1,n} = 0$。

Solution

- 状态：$f_{i,j}$ 表示从低到高确定了 i 个 x，且 n 被除成了 j 时，已经分解出来的和的最小值。
- 转移：枚举 $x \in \{2, 3, 4, 5\}$，$f_{i,j} + j \bmod x \to f_{i+1,\lfloor \frac{j}{x} \rfloor}$。
- 初始状态：$f_{1,n} = 0$。
- $ans = \min(f_{m,j} + j)$。

Solution

- 状态：$f_{i,j}$ 表示从低到高确定了 i 个 x，且 n 被除成了 j 时，已经分解出来的和的最小值。
- 转移：枚举 $x \in \{2,3,4,5\}$，$f_{i,j} + j \bmod x \to f_{i+1,\lfloor \frac{j}{x} \rfloor}$。
- 初始状态：$f_{1,n} = 0$。
- $ans = \min(f_{m,j} + j)$。
- 状态数：$j = \lfloor \frac{n}{p} \rfloor$，其中 p 只由 2,3,5 构成，共 $O(\log^3 n)$ 种。

Solution

- 状态：$f_{i,j}$ 表示从低到高确定了 i 个 x，且 n 被除成了 j 时，已经分解出来的和的最小值。
- 转移：枚举 $x \in \{2, 3, 4, 5\}$，$f_{i,j} + j \bmod x \to f_{i+1,\lfloor \frac{j}{x} \rfloor}$。
- 初始状态：$f_{1,n} = 0$。
- $ans = \min(f_{m,j} + j)$。
- 状态数：$j = \lfloor \frac{n}{p} \rfloor$，其中 p 只由 2,3,5 构成，共 $O(\log^3 n)$ 种。
- 用 std::map 存储状态即可，时间复杂度 $O(m \log^3 n)$。

储能表

给定 n, m, k，求：

$$\sum_{i=0}^{n-1} \sum_{j=0}^{m-1} \max((i \ xor \ j) - k, 0)$$

储能表

给定 n, m, k，求：

$$\sum_{i=0}^{n-1} \sum_{j=0}^{m-1} \max((i \ xor \ j) - k, 0)$$

- $n, m, k \leq 10^{18}$。

储能表

给定 n, m, k，求：

$$\sum_{i=0}^{n-1}\sum_{j=0}^{m-1}\mathrm{max}((i\ xor\ j)-k, 0)$$

- $n, m, k \leq 10^{18}$。
- Source：SDOI 2016

Solution

- 既然 *xor* 是二进制的运算，那么不妨从二进制的角度来考虑这个问题。

Solution

- 既然 *xor* 是二进制的运算，那么不妨从二进制的角度来考虑这个问题。
- 简化问题：转化为统计所有 $i \; xor \; j > k$ 的方案数以及异或和的和。

Solution

- 既然 *xor* 是二进制的运算，那么不妨从二进制的角度来考虑这个问题。
- 简化问题：转化为统计所有 $i \text{ xor } j > k$ 的方案数以及异或和的和。
- 首先将 i, j, k, n, m 高位补 0 对齐，共 $O(\log n)$ 位。

Solution

- 既然 *xor* 是二进制的运算，那么不妨从二进制的角度来考虑这个问题。
- 简化问题：转化为统计所有 $i \text{ xor } j > k$ 的方案数以及异或和的和。
- 首先将 i, j, k, n, m 高位补 0 对齐，共 $O(\log n)$ 位。
- 假如知道了 i，如何比较和 n 的大小？

Solution

- 既然 xor 是二进制的运算，那么不妨从二进制的角度来考虑这个问题。
- 简化问题：转化为统计所有 $i \ xor \ j > k$ 的方案数以及异或和的和。
- 首先将 i, j, k, n, m 高位补 0 对齐，共 $O(\log n)$ 位。
- 假如知道了 i，如何比较和 n 的大小？
- 从最高位开始逐一比对，直到不同，这个过程中始终有 $i \leq n$。

Solution

- 既然 xor 是二进制的运算，那么不妨从二进制的角度来考虑这个问题。
- 简化问题：转化为统计所有 $i\ xor\ j > k$ 的方案数以及异或和的和。
- 首先将 i, j, k, n, m 高位补 0 对齐，共 $O(\log n)$ 位。
- 假如知道了 i，如何比较和 n 的大小？
- 从最高位开始逐一比对，直到不同，这个过程中始终有 $i \leq n$。
- 假如知道了 i 和 j，如何比较 $i\ xor\ j$ 和 k 的大小？同理。

Solution

- 既然 xor 是二进制的运算，那么不妨从二进制的角度来考虑这个问题。
- 简化问题：转化为统计所有 $i\ xor\ j > k$ 的方案数以及异或和的和。
- 首先将 i, j, k, n, m 高位补 0 对齐，共 $O(\log n)$ 位。
- 假如知道了 i，如何比较和 n 的大小？
- 从最高位开始逐一比对，直到不同，这个过程中始终有 $i \le n$。
- 假如知道了 i 和 j，如何比较 $i\ xor\ j$ 和 k 的大小？同理。
- DP 从高到低枚举数位的过程。

Solution

- 状态：$f_{x,a,b,c}$ 表示从高到低确定了 i,j 的前 x 位，i 和 n 的大小关系为 a，j 和 m 的大小关系为 b，$i\ xor\ j$ 和 k 的大小关系为 c 的方案数。$g_{x,a,b,c}$ 表示对应的异或和的和。

Solution

- 状态：$f_{x,a,b,c}$ 表示从高到低确定了 i, j 的前 x 位，i 和 n 的大小关系为 a，j 和 m 的大小关系为 b，$i \ xor \ j$ 和 k 的大小关系为 c 的方案数。$g_{x,a,b,c}$ 表示对应的异或和的和。
- 转移：枚举 i 和 j 的下一位，计算出新的 a, b, c。

Solution

- 状态：$f_{x,a,b,c}$ 表示从高到低确定了 i,j 的前 x 位，i 和 n 的大小关系为 a，j 和 m 的大小关系为 b，i xor j 和 k 的大小关系为 c 的方案数。$g_{x,a,b,c}$ 表示对应的异或和的和。
- 转移：枚举 i 和 j 的下一位，计算出新的 a, b, c。
- $f_x \to f_{x+1}$，　$f_x, g_x \to g_{x+1}$。

Solution

- 状态：$f_{x,a,b,c}$ 表示从高到低确定了 i,j 的前 x 位，i 和 n 的大小关系为 a，j 和 m 的大小关系为 b，i xor j 和 k 的大小关系为 c 的方案数。$g_{x,a,b,c}$ 表示对应的异或和的和。
- 转移：枚举 i 和 j 的下一位，计算出新的 a, b, c。
- $f_x \to f_{x+1}$, $\quad f_x, g_x \to g_{x+1}$。
- 初始状态：

  $f_{0,i==n,j==m,i\ xor\ j==k} = 1, g_{0,i==n,j==m,i\ xor\ j==k} = 0$。

Solution

- 状态：$f_{x,a,b,c}$ 表示从高到低确定了 i,j 的前 x 位，i 和 n 的大小关系为 a，j 和 m 的大小关系为 b，$i\ xor\ j$ 和 k 的大小关系为 c 的方案数。$g_{x,a,b,c}$ 表示对应的异或和的和。
- 转移：枚举 i 和 j 的下一位，计算出新的 a,b,c。
- $f_x \to f_{x+1}, \quad f_x, g_x \to g_{x+1}$。
- 初始状态：

  $f_{0,i==n,j==m,i\ xor\ j==k} = 1, g_{0,i==n,j==m,i\ xor\ j==k} = 0$。
- $ans = g_{len,i<n,j<m,i\ xor\ j>k} - k \times f_{len,i<n,j<m,i\ xor\ j>k}$。

Solution

- 状态：$f_{x,a,b,c}$ 表示从高到低确定了 i,j 的前 x 位，i 和 n 的大小关系为 a，j 和 m 的大小关系为 b，i xor j 和 k 的大小关系为 c 的方案数。$g_{x,a,b,c}$ 表示对应的异或和的和。
- 转移：枚举 i 和 j 的下一位，计算出新的 a,b,c。
- $f_x \to f_{x+1}$, $f_x, g_x \to g_{x+1}$。
- 初始状态：

  $f_{0,i==n,j==m,i\ xor\ j==k} = 1, g_{0,i==n,j==m,i\ xor\ j==k} = 0$。
- $ans = g_{len,i<n,j<m,i\ xor\ j>k} - k \times f_{len,i<n,j<m,i\ xor\ j>k}$。
- 时间复杂度 $O(\log n)$。

座位安排

n 个人，圆桌上 m 个座位，每个座位只能坐一个人，已知每个人可以坐到哪些座位上。

座位安排

n 个人，圆桌上 m 个座位，每个座位只能坐一个人，已知每个人可以坐到哪些座位上。

两个人不能坐在相邻的座位上，也不会正对面坐下 (m 是奇数时不存在这种情况)。

座位安排

n 个人，圆桌上 m 个座位，每个座位只能坐一个人，已知每个人可以坐到哪些座位上。

两个人不能坐在相邻的座位上，也不会正对面坐下 (m 是奇数时不存在这种情况)。

求所有人都有座位坐的座位安排的方案数。

座位安排

n 个人，圆桌上 m 个座位，每个座位只能坐一个人，已知每个人可以坐到哪些座位上。

两个人不能坐在相邻的座位上，也不会正对面坐下 (m 是奇数时不存在这种情况)。

求所有人都有座位坐的座位安排的方案数。

- $n \leq 10$。

座位安排

n 个人，圆桌上 m 个座位，每个座位只能坐一个人，已知每个人可以坐到哪些座位上。

两个人不能坐在相邻的座位上，也不会正对面坐下 (m 是奇数时不存在这种情况)。

求所有人都有座位坐的座位安排的方案数。

- $n \leq 10$。
- $m \leq 1000$。

座位安排

n 个人，圆桌上 m 个座位，每个座位只能坐一个人，已知每个人可以坐到哪些座位上。

两个人不能坐在相邻的座位上，也不会正对面坐下 (m 是奇数时不存在这种情况)。

求所有人都有座位坐的座位安排的方案数。

- $n \leq 10$。
- $m \leq 1000$。
- Source： BZOJ 5095

Solution

- 当 m 是奇数时，只需考虑相邻两个位置不同时坐人。

Solution

- 当 m 是奇数时，只需考虑相邻两个位置不同时坐人。
- 状压 DP，设 $f_{i,j,k,l}$ 表示考虑前 i 个座位，还需要放下 j 集合的人，第 1 个座位是否坐人为 k，第 i 个座位是否坐人为 l 的方案数。

Solution

- 当 m 是奇数时，只需考虑相邻两个位置不同时坐人。
- 状压 DP，设 $f_{i,j,k,l}$ 表示考虑前 i 个座位，还需要放下 j 集合的人，第 1 个座位是否坐人为 k，第 i 个座位是否坐人为 l 的方案数。
- j 可以用一个 n 位二进制数表示。

Solution

- 当 m 是奇数时，只需考虑相邻两个位置不同时坐人。
- 状压 DP，设 $f_{i,j,k,l}$ 表示考虑前 i 个座位，还需要放下 j 集合的人，第 1 个座位是否坐人为 k，第 i 个座位是否坐人为 l 的方案数。
- j 可以用一个 n 位二进制数表示。
- 转移：枚举 j 中某个人，判断能否坐下。

Solution

- 当 m 是奇数时，只需考虑相邻两个位置不同时坐人。
- 状压 DP，设 $f_{i,j,k,l}$ 表示考虑前 i 个座位，还需要放下 j 集合的人，第 1 个座位是否坐人为 k，第 i 个座位是否坐人为 l 的方案数。
- j 可以用一个 n 位二进制数表示。
- 转移：枚举 j 中某个人，判断能否坐下。
- 时间复杂度 $O(nm2^n)$。

Solution

- 当 m 是偶数时，还要考虑不正对面。

Solution

- 当 m 是偶数时，还要考虑不正对面。
- 更换决策过程：令 $h = \frac{m}{2}$，同时考虑 i 和 $i + h$ 座位。

Solution

- 当 m 是偶数时，还要考虑不正对面。
- 更换决策过程：令 $h = \frac{m}{2}$，同时考虑 i 和 $i + h$ 座位。
- 转移：枚举 i 和 j 的下一位，计算出新的 a, b, c。

Solution

- 当 m 是偶数时，还要考虑不正对面。
- 更换决策过程：令 $h=\frac{m}{2}$，同时考虑 i 和 $i+h$ 座位。
- 转移：枚举 i 和 j 的下一位，计算出新的 a,b,c。
- 设 $f_{i,j,k,l}$ 表示考虑前 i 对座位，还需要放下 j 集合的人，1 和 $1+h$ 坐人情况为 k，i 和 $i+h$ 坐人情况为 l 的方案数。

Solution

- 当 m 是偶数时，还要考虑不正对面。
- 更换决策过程：令 $h = \frac{m}{2}$，同时考虑 i 和 $i + h$ 座位。
- 转移：枚举 i 和 j 的下一位，计算出新的 a, b, c。
- 设 $f_{i,j,k,l}$ 表示考虑前 i 对座位，还需要放下 j 集合的人，1 和 $1 + h$ 坐人情况为 k，i 和 $i + h$ 坐人情况为 l 的方案数。
- 转移：枚举 j 中某个人，再枚举 i 和 $i + h$ 中的一个座位，判断能否坐下。

Solution

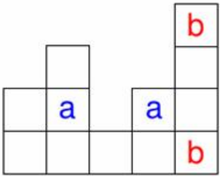
- 当 m 是偶数时，还要考虑不正对面。
- 更换决策过程：令 $h = \frac{m}{2}$，同时考虑 i 和 $i + h$ 座位。
- 转移：枚举 i 和 j 的下一位，计算出新的 a, b, c。
- 设 $f_{i,j,k,l}$ 表示考虑前 i 对座位，还需要放下 j 集合的人，1 和 $1 + h$ 坐人情况为 k，i 和 $i + h$ 坐人情况为 l 的方案数。
- 转移：枚举 j 中某个人，再枚举 i 和 $i + h$ 中的一个座位，判断能否坐下。
- 时间复杂度 $O(nm2^n)$。

Periodni

n 个底边长度为 1，高度为 h_i 的长条拼成一个棋盘。

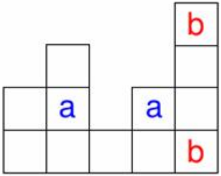
Periodni

n 个底边长度为 1 ，高度为 h_i 的长条拼成一个棋盘。



Periodni

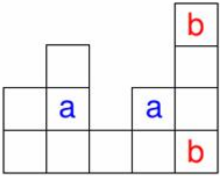
n 个底边长度为 1，高度为 h_i 的长条拼成一个棋盘。



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。

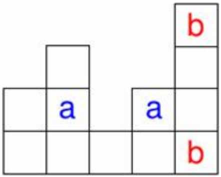


两个车能相互攻击当且仅当它们在同一行或者同一列，且它们之间所有格子均存在。

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。



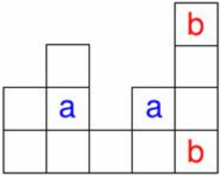
两个车能相互攻击当且仅当它们在同一行或者同一列，且它们之间所有格子均存在。

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，$f_{a,i} \times f_{b,j} \to f_{c,i+j}$。

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，$f_{a,i} \times f_{b,j} \to f_{c,i+j}$。
- 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，$f_{a,i} \times f_{b,j} \to f_{c,i+j}$。
- 然后考虑父亲矩形中放置多少个车。
- 时间复杂度 $O(n^3+h)$。

Solution

- 实现精良可以将复杂度优化到 $O(n^2 + h)$。

Solution

- 实现精良可以将复杂度优化到 $O(n^2+h)$。
- 设 $size_i$ 表示 i 子树中节点的个数，显然 $j \le size_i$。

Solution

- 实现精良可以将复杂度优化到 $O(n^2 + h)$。
- 设 $size_i$ 表示 i 子树中节点的个数，显然 $j \leq size_i$。
- 只枚举有意义的 j 即可将复杂度降低至平方。

Solution

- 实现精良可以将复杂度优化到 $O(n^2+h)$。
- 设 $size_i$ 表示 i 子树中节点的个数，显然 $j \le size_i$。
- 只枚举有意义的 j 即可将复杂度降低至平方。
- 证明：枚举量等同于两个点被合并的方案数 $=O(n^2)$。

题目提交

课上例题：

http://acm.hdu.edu.cn/diy/contest_show.php?cid=33137

课后习题：

http://acm.hdu.edu.cn/diy/contest_show.php?cid=33138

密码：

G*&GSF&*t387tr

Thank you!