动态规划科普 (Preview)

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。
- 

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。
- $f_i += f_{i-v}$。

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。

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。
- $f_i - = f_{i-v}$。

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。
- $f_i - = f_{i-v}$。
- 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。

- $n, m \leq 1000$。

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 $a_1, a_2, \ldots, a_n$，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 $a_1, a_2, ..., a_n$ ，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 $1 \le j < i$ 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 $q_1, q_2, ..., q_m$ ，一开始为空，即 $m = 0$。

Solution

- 维护一个递增的数组 $q_1, q_2, \ldots, q_m$ ，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 $q_1, q_2, ..., q_m$ ，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 $q_1, q_2, \ldots, q_m$ ，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 $q_1, q_2, ..., q_m$ ，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 $q_1, q_2, ..., q_m$。
- 时间复杂度 $O(n \log n)$。

k-LIS

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

- $n \leq 100000$。
- $k \leq 50$。

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

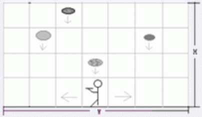
免费的馅饼

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。



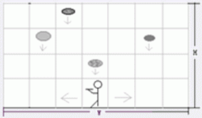
免费的馅饼

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。



- $n \leq 100000$。

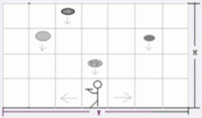
免费的馅饼

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

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

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。

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 $a_l, a_{l+1}, \ldots, a_r$ 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 $a_l, a_{l+1}, \ldots, a_r$ 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 $f[l][r] = \min(f[l][i] + f[i+1][r] + sum(l, i) + sum(i+1, r)) = \min(f[l][i] + f[i+1][r]) + sum(l, r)$，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 $a_l, a_{l+1}, \ldots, a_r$ 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 $f[l][r] = \min(f[l][i] + f[i+1][r] + sum(l, i) + sum(i+1, r)) = \min(f[l][i] + f[i+1][r]) + sum(l, r)$，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。

- $n \leq 100$。

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。

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

Solution

- 相比上一个问题多了数量的限制。

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

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

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。

- $2 \le n \le 16$。

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。

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

Solution

- 从左往右依次填人。

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。

- $n, m \leq 100000$。
- $k \leq 16$。

数位和

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

数位和

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

- $n \leq 10^{100}$。

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

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。

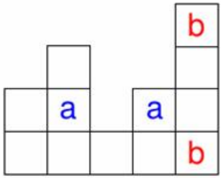
- $1 \leq A \leq B \leq 10^{18}$。
- Source：SPOJ 10606

Periodni

n 个底边长度为 1，高度为 h_i 的长条拼成一个棋盘。

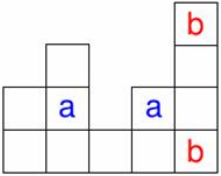
Periodni

n 个底边长度为 1，高度为 h_i 的长条拼成一个棋盘。



Periodni

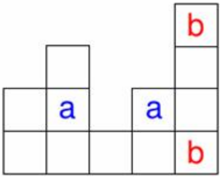
n 个底边长度为 1，高度为 h_i 的长条拼成一个棋盘。



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。

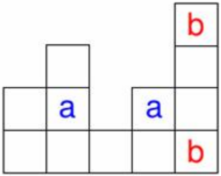


两个车能相互攻击当且仅当它们在同一行或者同一列，且它们之间所有格子均存在。

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。



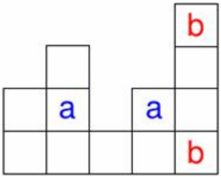
两个车能相互攻击当且仅当它们在同一行或者同一列，且它们之间所有格子均存在。

现在要在棋盘上放置恰好 k 个相互不攻击的车，求方案数。

- $n, k \le 500, h_i \le 10^6$。

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。

- $n, k \le 500, h_i \le 10^6$。
- Source： COCI 2008

Thank you!