动态规划选讲

Claris

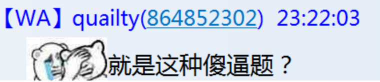
中国膜 Q 学会

2019 年 1 月 23 日

写在前面

- 题目都比较简单，欢迎现场秒题。

写在前面

- 题目都比较简单，欢迎现场秒题。
- 

Knapsack

n 个物品，m 容量的背包。每个物品最多选一个，求能背走的最大价值。

Knapsack

n 个物品，m 容量的背包。每个物品最多选一个，求能背走的最大价值。

Nephren【雀魂配信专用】(864852302) 13:58:11

- 这题挺简单的，趁早 A 了吧
- 

Knapsack

n 个物品，m 容量的背包。每个物品最多选一个，求能背走的最大价值。

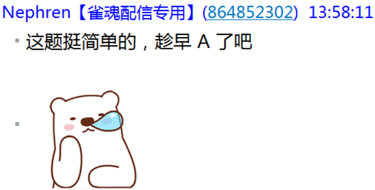
Nephren【雀魂配信专用】(864852302) 13:58:11

- 这题挺简单的，趁早 A 了吧
- 

- $n \leq 500, m \leq 10^{17}$。保证数据随机生成。

Knapsack

n 个物品，m 容量的背包。每个物品最多选一个，求能背走的最大价值。



- $n \le 500, m \le 10^{17}$。保证数据随机生成。
- Source：Petrozavodsk Winter-2018. Carnegie Mellon U Contest

Solution

- 01 背包裸题。

Solution

- 01 背包裸题。
- 设 $f_{i,j}$ 表示考虑了前 i 个物品，容量为 j 的最大体积。

Solution

- 01 背包裸题。
- 设 $f_{i,j}$ 表示考虑了前 i 个物品，容量为 j 的最大体积。
- 优化：对于两个状态 A, B，如果 A 比 B 容量小且价值大，那么 B 状态不优。

Solution

- 01 背包裸题。
- 设 $f_{i,j}$ 表示考虑了前 i 个物品，容量为 j 的最大体积。
- 优化：对于两个状态 A, B，如果 A 比 B 容量小且价值大，那么 B 状态不优。
- 每次转移后去掉所有这样的不优状态即可。

Solution

- 01 背包裸题。
- 设 $f_{i,j}$ 表示考虑了前 i 个物品，容量为 j 的最大体积。
- 优化：对于两个状态 A, B，如果 A 比 B 容量小且价值大，那么 B 状态不优。
- 每次转移后去掉所有这样的不优状态即可。
- 一共 $O(2^n)$ 个状态，正交凸包上期望 $O(n^2)$ 个状态。

Solution

- 01 背包裸题。
- 设 $f_{i,j}$ 表示考虑了前 i 个物品，容量为 j 的最大体积。
- 优化：对于两个状态 A, B，如果 A 比 B 容量小且价值大，那么 B 状态不优。
- 每次转移后去掉所有这样的不优状态即可。
- 一共 $O(2^n)$ 个状态，正交凸包上期望 $O(n^2)$ 个状态。
- 时间复杂度 $O(n^3)$。

烧桥计划

给定一个长度为 n 的序列 $a_1, a_2, \ldots, a_n$ 和一个参数 m。

烧桥计划

给定一个长度为 n 的序列 $a_1, a_2, \ldots, a_n$ 和一个参数 m。

你要从中删掉若干个位置 $p_1, p_2, \ldots, p_k$ $(1 \le p_1 < p_2 < \ldots < p_k \le n)$，耗费 $\sum_{i=1}^{k} i \times a_{p_i}$ 的代价。

烧桥计划

给定一个长度为 n 的序列 $a_1, a_2, \ldots, a_n$ 和一个参数 m。

你要从中删掉若干个位置 $p_1, p_2, \ldots, p_k$ $(1 \le p_1 < p_2 < \ldots < p_k \le n)$，耗费 $\sum_{i=1}^{k} i \times a_{p_i}$ 的代价。

上一步会把序列分割成 $k+1$ 段，对于剩下的每段求和，如果某一段的和 $sum > m$，则要额外支付 sum 的代价。

烧桥计划

给定一个长度为 n 的序列 $a_1, a_2, \ldots, a_n$ 和一个参数 m。

你要从中删掉若干个位置 $p_1, p_2, \ldots, p_k$ $(1 \le p_1 < p_2 < \ldots < p_k \le n)$，耗费 $\sum_{i=1}^{k} i \times a_{p_i}$ 的代价。

上一步会把序列分割成 $k+1$ 段，对于剩下的每段求和，如果某一段的和 $sum > m$，则要额外支付 sum 的代价。

k 是你任选的，求最小总代价。

烧桥计划

给定一个长度为 n 的序列 $a_1, a_2, ..., a_n$ 和一个参数 m。

你要从中删掉若干个位置 $p_1, p_2, ..., p_k$ $(1 \le p_1 < p_2 < ... < p_k \le n)$，耗费 $\sum_{i=1}^{k} i \times a_{p_i}$ 的代价。

上一步会把序列分割成 $k+1$ 段，对于剩下的每段求和，如果某一段的和 $sum > m$，则要额外支付 sum 的代价。

k 是你任选的，求最小总代价。

- $n \le 100000, 1000 \le a_i \le 2000$。

烧桥计划

给定一个长度为 n 的序列 $a_1, a_2, \ldots, a_n$ 和一个参数 m。

你要从中删掉若干个位置 $p_1, p_2, \ldots, p_k$ $(1 \le p_1 < p_2 < \ldots < p_k \le n)$，耗费 $\sum_{i=1}^{k} i \times a_{p_i}$ 的代价。

上一步会把序列分割成 $k+1$ 段，对于剩下的每段求和，如果某一段的和 $sum > m$，则要额外支付 sum 的代价。

k 是你任选的，求最小总代价。

- $n \le 100000, 1000 \le a_i \le 2000$。
- Source： BZOJ 5424

Solution

- 我会 $O(n^3)$！

Solution

- 我会 $O(n^3)$！
- 设 $f_{i,j}$ 表示考虑了前 i 个位置，第 i 个位置被删除，一共删除了 j 个位置的最小代价。

Solution

- 我会 $O(n^3)$！
- 设 $f_{i,j}$ 表示考虑了前 i 个位置，第 i 个位置被删除，一共删除了 j 个位置的最小代价。
- 枚举上一个删除的位置进行转移。

Solution

- 我会 $O(n^2)$！

Solution

- 我会 $O(n^2)$!
- 设 s_i 为 $a_1 + a_2 + \ldots + a_i$。

Solution

- 我会 $O(n^2)$！
- 设 s_i 为 $a_1 + a_2 + \ldots + a_i$。
- $f_{i,j} = \min\{f_{k,j-1} + cost(s_{i-1} - s_k)\} + j \times a_i$。

Solution

- 我会 $O(n^2)$！
- 设 s_i 为 $a_1 + a_2 + \ldots + a_i$。
- $f_{i,j} = \min\{f_{k,j-1} + cost(s_{i-1} - s_k)\} + j \times a_i$。
- 按照 $s_{i-1} - s_k$ 和 m 的大小关系可以把 k 分成两类。

Solution

- 我会 $O(n^2)$！
- 设 s_i 为 $a_1 + a_2 + \ldots + a_i$。
- $f_{i,j} = \min\{f_{k,j-1} + cost(s_{i-1} - s_k)\} + j \times a_i$。
- 按照 $s_{i-1} - s_k$ 和 m 的大小关系可以把 k 分成两类。
- 第一类是个前缀，可以维护前缀 min。

Solution

- 我会 $O(n^2)$！
- 设 s_i 为 $a_1+a_2+\ldots+a_i$。
- $f_{i,j}=\min\{f_{k,j-1}+cost(s_{i-1}-s_k)\}+j\times a_i$。
- 按照 $s_{i-1}-s_k$ 和 m 的大小关系可以把 k 分成两类。
- 第一类是个前缀，可以维护前缀 min。
- 第二类是个后缀，可以单调队列。

Solution

- 我发现数据范围很诡异！$1000 \le a_i \le 2000$。

Solution

- 我发现数据范围很诡异！$1000 \le a_i \le 2000$。
- 假设第一步什么也没做，那么我们最多需要 $2000n$ 的代价。

Solution

- 我发现数据范围很诡异！$1000 \le a_i \le 2000$。
- 假设第一步什么也没做，那么我们最多需要 $2000n$ 的代价。
- 如果第一步选择了 k 个位置，那么至少需要 $\frac{k(k+1)}{2} \times 1000$ 的代价。

Solution

- 我发现数据范围很诡异！$1000 \le a_i \le 2000$。
- 假设第一步什么也没做，那么我们最多需要 $2000n$ 的代价。
- 如果第一步选择了 k 个位置，那么至少需要 $\frac{k(k+1)}{2} \times 1000$ 的代价。
- 可以发现能更新答案的 k 是 $O(\sqrt{n})$ 级别。

Solution

- 我发现数据范围很诡异！$1000 \le a_i \le 2000$。
- 假设第一步什么也没做，那么我们最多需要 $2000n$ 的代价。
- 如果第一步选择了 k 个位置，那么至少需要 $\frac{k(k+1)}{2} \times 1000$ 的代价。
- 可以发现能更新答案的 k 是 $O(\sqrt{n})$ 级别。
- 时间复杂度 $O(n\sqrt{n})$。

Road Connectivity

n 个点的无向图，一开始是空的。每次操作会随机选择一条边 $(u, v)(1 \le u < v \le n)$，翻转它的存在状态，即存在 $\to$ 不存在，不存在 $\to$ 存在。

Road Connectivity

n 个点的无向图，一开始是空的。每次操作会随机选择一条边 $(u, v)(1 \leq u < v \leq n)$，翻转它的存在状态，即存在 $\rightarrow$ 不存在，不存在 $\rightarrow$ 存在。

q 次询问，每次给定 l, r，求操作次数在 $[l, r]$ 之间将这个图连通的概率。

Road Connectivity

n 个点的无向图，一开始是空的。每次操作会随机选择一条边 $(u,v)(1 \le u < v \le n)$，翻转它的存在状态，即存在 $\to$ 不存在，不存在 $\to$ 存在。

q 次询问，每次给定 l, r，求操作次数在 $[l, r]$ 之间将这个图连通的概率。

- $n \le 5$。

Road Connectivity

n 个点的无向图，一开始是空的。每次操作会随机选择一条边 $(u, v)(1 \leq u < v \leq n)$，翻转它的存在状态，即存在 $\rightarrow$ 不存在，不存在 $\rightarrow$ 存在。

q 次询问，每次给定 l, r，求操作次数在 $[l, r]$ 之间将这个图连通的概率。

- $n \leq 5$。
- $q \leq 1000$。

Road Connectivity

n 个点的无向图，一开始是空的。每次操作会随机选择一条边 $(u, v)(1 \le u < v \le n)$，翻转它的存在状态，即存在 $\rightarrow$ 不存在，不存在 $\rightarrow$ 存在。

q 次询问，每次给定 l, r，求操作次数在 $[l, r]$ 之间将这个图连通的概率。

- $n \le 5$。
- $q \le 1000$。
- $0 \le l \le r \le 10^{15}$。

Road Connectivity

n 个点的无向图，一开始是空的。每次操作会随机选择一条边 $(u, v)(1 \le u < v \le n)$，翻转它的存在状态，即存在 $\to$ 不存在，不存在 $\to$ 存在。

q 次询问，每次给定 l, r，求操作次数在 $[l, r]$ 之间将这个图连通的概率。

- $n \le 5$。
- $q \le 1000$。
- $0 \le l \le r \le 10^{15}$。
- Source：XIX Open Cup GP of Udmurtia

Solution

- 考虑反面：求 $[l, r]$ 不连通的概率。

Solution

- 考虑反面：求 $[l, r]$ 不连通的概率。
- 也就是从空图开始自由生长到 $l-1$，然后生长到 r，满足中间任何时刻都不连通。

Solution

- 考虑反面：求 $[l, r]$ 不连通的概率。
- 也就是从空图开始自由生长到 $l-1$，然后生长到 r，满足中间任何时刻都不连通。
- $f_{i,S}$ 表示 i 次操作后邻接矩阵为 S 的概率。

Solution

- 考虑反面：求 $[l, r]$ 不连通的概率。
- 也就是从空图开始自由生长到 $l-1$，然后生长到 r，满足中间任何时刻都不连通。
- $f_{i,S}$ 表示 i 次操作后邻接矩阵为 S 的概率。
- 两个阶段的转移可以分别用矩阵表示。

Solution

- 对于多个询问的情况，预处理出转移矩阵的 2 的幂次方。

Solution

- 对于多个询问的情况，预处理出转移矩阵的 2 的幂次方。
- 每次询问只需要用 $O(\log r)$ 个矩阵乘以向量。

Solution

- 对于多个询问的情况，预处理出转移矩阵的 2 的幂次方。
- 每次询问只需要用 $O(\log r)$ 个矩阵乘以向量。
- 状态数 $m = 2^{\frac{n(n-1)}{2}}$，时间复杂度 $O(m^3 \log r + qm^2 \log r)$。

Solution

- 对于多个询问的情况，预处理出转移矩阵的 2 的幂次方。
- 每次询问只需要用 $O(\log r)$ 个矩阵乘以向量。
- 状态数 $m = 2^{\frac{n(n-1)}{2}}$，时间复杂度 $O(m^3 \log r + qm^2 \log r)$。
- 好像不太能过。

Solution

- 把图的标号去掉，对于两个状态 S, T，如果把 S 的点重标号后能得到 T，就把这两个状态合并。

Solution

- 把图的标号去掉，对于两个状态 S, T，如果把 S 的点重标号后能得到 T，就把这两个状态合并。
- $n = 5$ 时最多只有 34 个本质不同的图。

Solution

- 把图的标号去掉，对于两个状态 S, T，如果把 S 的点重标号后能得到 T，就把这两个状态合并。
- $n = 5$ 时最多只有 34 个本质不同的图。
- 

Rikka with Subsequences

给定一个长度为 n 的序列 q ，每个位置是 $[1, n]$ 之间的整数。

Rikka with Subsequences

给定一个长度为 n 的序列 q，每个位置是 $[1, n]$ 之间的整数。

给定 $n \times n$ 的 01 矩阵 g，定义一个序列 $a_1, a_2, \ldots, a_m$ 是好的，当且仅当且对于任意的 $1 \leq i < m$，$g_{a_i, a_{i+1}} = 1$ 恒成立。

Rikka with Subsequences

给定一个长度为 n 的序列 q，每个位置是 $[1, n]$ 之间的整数。

给定 $n \times n$ 的 01 矩阵 g，定义一个序列 $a_1, a_2, \ldots, a_m$ 是好的，当且仅当且对于任意的 $1 \le i < m$，$g_{a_i, a_{i+1}} = 1$ 恒成立。

假设有一个 `std::map` 保存了 q 的每个好的子序列的出现次数，你需要统计它们的出现次数的立方和。

Rikka with Subsequences

给定一个长度为 n 的序列 q，每个位置是 $[1, n]$ 之间的整数。

给定 $n \times n$ 的 01 矩阵 g，定义一个序列 $a_1, a_2, \ldots, a_m$ 是好的，当且仅当且对于任意的 $1 \le i < m$，$g_{a_i, a_{i+1}} = 1$ 恒成立。

假设有一个 `std::map` 保存了 q 的每个好的子序列的出现次数，你需要统计它们的出现次数的立方和。

- $n \le 200$。

Rikka with Subsequences

给定一个长度为 n 的序列 q，每个位置是 $[1, n]$ 之间的整数。

给定 $n \times n$ 的 01 矩阵 g，定义一个序列 $a_1, a_2, \ldots, a_m$ 是好的，当且仅当且对于任意的 $1 \le i < m$，$g_{a_i, a_{i+1}} = 1$ 恒成立。

假设有一个 `std::map` 保存了 q 的每个好的子序列的出现次数，你需要统计它们的出现次数的立方和。

- $n \le 200$。
- Source：2018-2019 ICPC, Asia Xuzhou Regional Contest

Solution

- $x^3 = \sum_{i=1}^{x} \sum_{j=1}^{x} \sum_{k=1}^{x} 1$。

Solution

- $x^3 = \sum_{i=1}^{x} \sum_{j=1}^{x} \sum_{k=1}^{x} 1$。
- 题目等价于把 q 复制成三份 a, b, c，求 a, b, c 的公共好子序列的方案数。

Solution

- 我会 $O(n^6)$！

Solution

- 我会 $O(n^6)$！
- 设 $f_{i,j,k}$ 表示考虑了 $a[1..i], b[1..j], c[1..k]$，公共好子序列的最后一项分别为 a_i, b_j, c_k 的方案数。

Solution

- 我会 $O(n^6)$！
- 设 $f_{i,j,k}$ 表示考虑了 $a[1..i], b[1..j], c[1..k]$，公共好子序列的最后一项分别为 a_i, b_j, c_k 的方案数。
- 枚举下一个位置 x, y, z 转移。

Solution

- 细化状态转移：同时枚举 x, y, z 可以看作三个阶段：

  (1) 将 i 移动到 x。

  (2) 将 j 移动到 y。

  (3) 将 k 移动到 z。

Solution

- 细化状态转移：同时枚举 x, y, z 可以看作三个阶段：
  (1) 将 i 移动到 x。
  (2) 将 j 移动到 y。
  (3) 将 k 移动到 z。
- 在第一步完成后，可以根据 g_{a_i,a_x} 是否是 1 来判断这个序列是不是好的。

Solution

- 细化状态转移：同时枚举 x, y, z 可以看作三个阶段：
  (1) 将 i 移动到 x。
  (2) 将 j 移动到 y。
  (3) 将 k 移动到 z。
- 在第一步完成后，可以根据 g_{a_i,a_x} 是否是 1 来判断这个序列是不是好的。
- 所以第一步完成后 i 没有必要记录，j, k 同理。

Solution

- 细化状态转移：同时枚举 x, y, z 可以看作三个阶段：
  (1) 将 i 移动到 x。
  (2) 将 j 移动到 y。
  (3) 将 k 移动到 z。
- 在第一步完成后，可以根据 g_{a_i,a_x} 是否是 1 来判断这个序列是不是好的。
- 所以第一步完成后 i 没有必要记录，j, k 同理。
- 新状态：$f_{i,j,k,t}$ 表示目前正在移动第 $t(0 \leq t \leq 2)$ 个变量的方案数。

Solution

- 细化状态转移：同时枚举 x, y, z 可以看作三个阶段：
  (1) 将 i 移动到 x。
  (2) 将 j 移动到 y。
  (3) 将 k 移动到 z。
- 在第一步完成后，可以根据 g_{a_i,a_x} 是否是 1 来判断这个序列是不是好的。
- 所以第一步完成后 i 没有必要记录，j, k 同理。
- 新状态：$f_{i,j,k,t}$ 表示目前正在移动第 $t(0 \leq t \leq 2)$ 个变量的方案数。
- 时间复杂度 $O(n^4)$。

Solution

- 其实从枚举 x 也是没有必要的。

Solution

- 其实从枚举 x 也是没有必要的。
- 简化转移：要么将 x 增加 1，要么钦定这个 x 就是我们想要的 x，开始 y 的转移。

Solution

- 其实从枚举 x 也是没有必要的。
- 简化转移：要么将 x 增加 1，要么钦定这个 x 就是我们想要的 x，开始 y 的转移。
- 时间复杂度 $O(n^3)$。

Non-palindromic cutting

给定一个长度为 n 的字符串 S。

Non-palindromic cutting

给定一个长度为 n 的字符串 S。

将 S 划分为若干段非空连续子串，使得每段都不是回文串。

Non-palindromic cutting

给定一个长度为 n 的字符串 S。
将 S 划分为若干段非空连续子串，使得每段都不是回文串。
求最多能划分成多少段。

Non-palindromic cutting

给定一个长度为 n 的字符串 S。
将 S 划分为若干段非空连续子串，使得每段都不是回文串。
求最多能划分成多少段。

- $n \leq 200000$。

Non-palindromic cutting

给定一个长度为 n 的字符串 S。
将 S 划分为若干段非空连续子串，使得每段都不是回文串。
求最多能划分成多少段。

- $n \leq 200000$。
- Source：Ural 2057

Solution

- 设 f_i 表示前 i 个最多划分成多少段，$f_i = \max(f_j + 1)$，其中 $j < i$ 且 $[j+1, i]$ 不是回文串。

Solution

- 设 f_i 表示前 i 个最多划分成多少段，$f_i = \max(f_j + 1)$，其中 $j < i$ 且 $[j+1, i]$ 不是回文串。
- 发现非常难转移。

Solution

- 设 f_i 表示前 i 个最多划分成多少段，$f_i = \max(f_j + 1)$，其中 $j < i$ 且 $[j+1, i]$ 不是回文串。
- 发现非常难转移。
- 

Solution

- 考虑无解的情形，只能是 aaaaa,aaabaaa,abababa 这三种情况。

Solution

- 考虑无解的情形，只能是 aaaaa,aaabaaa,abababa 这三种情况。
- 重新定义转移：$f_i = \max(f_j + 1)$，其中 $j < i$ 且 $[j + 1, i]$ 有解，那么最优解不变。

Solution

- 考虑无解的情形，只能是 aaaaa,aaabaaa,abababa 这三种情况。
- 重新定义转移：$f_i = \max(f_j + 1)$，其中 $j < i$ 且 $[j+1, i]$ 有解，那么最优解不变。
- 可行的 j 可以根据 aaaaa、abababa 分奇偶得出两个上界。

Solution

- 考虑无解的情形，只能是 aaaaa,aaabaaa,abababa 这三种情况。
- 重新定义转移：$f_i = \max(f_j + 1)$，其中 $j < i$ 且 $[j+1, i]$ 有解，那么最优解不变。
- 可行的 j 可以根据 aaaaa、abababa 分奇偶得出两个上界。
- 对于 aaabaaa，最多只有一个 j 不可行，因此 DP 的同时分奇偶维护出前缀最大值和次大值即可 $O(1)$ 转移。

Solution

- 考虑无解的情形，只能是 aaaaa,aaabaaa,abababa 这三种情况。
- 重新定义转移：$f_i = \max(f_j + 1)$，其中 $j < i$ 且 $[j+1, i]$ 有解，那么最优解不变。
- 可行的 j 可以根据 aaaaa、abababa 分奇偶得出两个上界。
- 对于 aaabaaa，最多只有一个 j 不可行，因此 DP 的同时分奇偶维护出前缀最大值和次大值即可 $O(1)$ 转移。
- 时间复杂度 $O(n)$。

Modern Art Plagiarism

给定两棵无根树 A 和 B，判断是否存在 A 的一个子连通块和 B 同构。

Modern Art Plagiarism

给定两棵无根树 A 和 B，判断是否存在 A 的一个子连通块和 B 同构。

- $n \leq 100$。

Modern Art Plagiarism

给定两棵无根树 A 和 B，判断是否存在 A 的一个子连通块和 B 同构。

- $n \leq 100$。
- Source： Google Code Jam 2008 APAC Onsites

Solution

- 枚举 A 的树根转化为有根树，并钦定 B 的根为 1，变成有根树同构。

Solution

- 枚举 A 的树根转化为有根树，并钦定 B 的根为 1，变成有根树同构。
- 设 $f_{i,j}$ 表示 A 中是否有个以 i 为根的子连通块和 B 以 j 为根的子树同构。

Solution

- 枚举 A 的树根转化为有根树，并钦定 B 的根为 1，变成有根树同构。
- 设 $f_{i,j}$ 表示 A 中是否有个以 i 为根的子连通块和 B 以 j 为根的子树同构。
- 转移？

Solution

- 枚举 A 的树根转化为有根树，并钦定 B 的根为 1，变成有根树同构。
- 设 $f_{i,j}$ 表示 A 中是否有个以 i 为根的子连通块和 B 以 j 为根的子树同构。
- 转移？
- 将 i 和 j 的儿子之间 DP 值为 1 的连边，匈牙利算法判断是否存在二分图完美匹配。

波浪

考虑 1 到 n 的一个排列 $a_1, a_2, \ldots, a_n$ ，定义它的波浪值为 $\sum_{i=1}^{n-1} |a_i - a_{i+1}|$。

波浪

考虑 1 到 n 的一个排列 $a_1, a_2, \ldots, a_n$，定义它的波浪值为 $\sum_{i=1}^{n-1} |a_i - a_{i+1}|$。

给定 m，求有多少排列的波浪值不小于 m。

波浪

考虑 1 到 n 的一个排列 $a_1, a_2, \dots, a_n$，定义它的波浪值为 $\sum_{i=1}^{n-1} |a_i - a_{i+1}|$。

给定 m，求有多少排列的波浪值不小于 m。

- $n \leq 100$。

波浪

考虑 1 到 n 的一个排列 $a_1, a_2, \dots, a_n$，定义它的波浪值为 $\sum_{i=1}^{n-1} |a_i - a_{i+1}|$。

给定 m，求有多少排列的波浪值不小于 m。

- $n \le 100$。
- Source：ZJOI 2012

Solution

- 如何 DP 一个排列？

Solution

- 如何 DP 一个排列？
- 我会 $O(2^n)$ 状压 DP！

Solution

- 如何 DP 一个排列？
- 我会 $O(2^n)$ 状压 DP！
- $n \leq 100$。

Solution

- 如何 DP 一个排列？
- 我会 $O(2^n)$ 状压 DP！
- $n \leq 100$。
- 

Solution

- 假如已经知道了最后的排列，考虑从小到大依次将 1 到 n 这些数填到对应的位置上。

Solution

- 假如已经知道了最后的排列，考虑从小到大依次将 1 到 n 这些数填到对应的位置上。
- 这个序列在这个过程中是一段段区间，且最多有一个区间占了最左端，最多有一个区间占了最右端。

Solution

- 假如已经知道了最后的排列，考虑从小到大依次将 1 到 n 这些数填到对应的位置上。
- 这个序列在这个过程中是一段段区间，且最多有一个区间占了最左端，最多有一个区间占了最右端。
- 状态：$f[i][j][k][t]$ 表示插入了 1 到 i，有 j 个区间，这 j 个区间中占了 $k=0/1/2$ 个最左/最右端，波浪值为 t 的方案数。

Solution

- 假如已经知道了最后的排列，考虑从小到大依次将 1 到 n 这些数填到对应的位置上。
- 这个序列在这个过程中是一段段区间，且最多有一个区间占了最左端，最多有一个区间占了最右端。
- 状态：$f[i][j][k][t]$ 表示插入了 1 到 i，有 j 个区间，这 j 个区间中占了 $k=0/1/2$ 个最左/最右端，波浪值为 t 的方案数。
- 状态数 $O(n^4)$。

Solution

- 转移？

Solution

- 转移？
- (1) 新开一个区间。
  (2) 扩展之前某个区间，即放到它的左边/右边。
  (3) (1) 或 (2) 的基础上占据最左/最右的格子。
  (4) 合并两个区间。

Solution

- 转移？
- (1) 新开一个区间。
  (2) 扩展之前某个区间，即放到它的左边/右边。
  (3) (1) 或 (2) 的基础上占据最左/最右的格子。
  (4) 合并两个区间。
- 转移 $O(1)$。

Solution

- 转移？
- (1) 新开一个区间。
  (2) 扩展之前某个区间，即放到它的左边/右边。
  (3) (1) 或 (2) 的基础上占据最左/最右的格子。
  (4) 合并两个区间。
- 转移 $O(1)$。
- 时间复杂度 $O(n^4)$。

Solution

- 等等！转移的时候怎么算波浪值？

Solution

- 等等！转移的时候怎么算波浪值？
- 因为不知道相邻的数值具体是多少，考虑每次将 i 增加 1 时，都将未知的相邻差值增加 1，可以很方便地由区间个数得到。

Solution

- 等等！转移的时候怎么算波浪值？
- 因为不知道相邻的数值具体是多少，考虑每次将 i 增加 1 时，都将未知的相邻差值增加 1，可以很方便地由区间个数得到。
- 每个区间有左右两个相邻差值，除了最左/最右两个格子，所以波浪值增加量为 $2j-k$。

小 C 的独立集

给定一棵仙人掌，请选择最多的点，使得任意两点不相邻。

小 C 的独立集

给定一棵仙人掌，请选择最多的点，使得任意两点不相邻。

- $n \leq 50000$。

小 C 的独立集

给定一棵仙人掌，请选择最多的点，使得任意两点不相邻。

- $n \leq 50000$。
- Source： BZOJ 4316

Solution

- 按照 DFS 生成树的方式遍历这棵仙人掌，那么每条非树边对应一个环，且是祖孙关系。

Solution

- 按照 DFS 生成树的方式遍历这棵仙人掌，那么每条非树边对应一个环，且是祖孙关系。
- 那么每个点到父亲的边最多属于一个简单环，只需要记录这个环底部的点是否选择即可。

Solution

- 按照 DFS 生成树的方式遍历这棵仙人掌，那么每条非树边对应一个环，且是祖孙关系。
- 那么每个点到父亲的边最多属于一个简单环，只需要记录这个环底部的点是否选择即可。
- 设 $f[i][j][k]$ 表示考虑 DFS 生成树中 i 的子树，i 点选择情况为 j，i 到 i 父亲这条边所在环底部的点选择情况为 k 时，最多能选几个点。

Solution

- 按照 DFS 生成树的方式遍历这棵仙人掌，那么每条非树边对应一个环，且是祖孙关系。
- 那么每个点到父亲的边最多属于一个简单环，只需要记录这个环底部的点是否选择即可。
- 设 $f[i][j][k]$ 表示考虑 DFS 生成树中 i 的子树，i 点选择情况为 j，i 到 i 父亲这条边所在环底部的点选择情况为 k 时，最多能选几个点。
- 在发现每个环时，更新 k 的状态；在每个环的最顶端那条边处判断是否可以转移。

Solution

- 按照 DFS 生成树的方式遍历这棵仙人掌，那么每条非树边对应一个环，且是祖孙关系。
- 那么每个点到父亲的边最多属于一个简单环，只需要记录这个环底部的点是否选择即可。
- 设 $f[i][j][k]$ 表示考虑 DFS 生成树中 i 的子树，i 点选择情况为 j，i 到 i 父亲这条边所在环底部的点选择情况为 k 时，最多能选几个点。
- 在发现每个环时，更新 k 的状态；在每个环的最顶端那条边处判断是否可以转移。
- 时间复杂度 $O(n)$，代码很短。

共鸣

给定平面上 n 个白点和 m 个黑点，请挑选其中若干个白点，使得它们的凸包内部或边上都没有黑点，且面积最大。

共鸣

给定平面上 n 个白点和 m 个黑点，请挑选其中若干个白点，使得它们的凸包内部或边上都没有黑点，且面积最大。

- $n \leq 100$。

共鸣

给定平面上 n 个白点和 m 个黑点，请挑选其中若干个白点，使得它们的凸包内部或边上都没有黑点，且面积最大。

- $n \leq 100$。
- $m \leq 100$。

共鸣

给定平面上 n 个白点和 m 个黑点，请挑选其中若干个白点，使得它们的凸包内部或边上都没有黑点，且面积最大。

- $n \leq 100$。
- $m \leq 100$。
- Source：BZOJ 3778

Solution

- 如何 DP 一个凸包？

Solution

- 如何 DP 一个凸包？
- 凸包上相邻的有向边的极角一定是递增的！

Solution

- 如何 DP 一个凸包？
- 凸包上相邻的有向边的极角一定是递增的！
- 枚举 $O(n^2)$ 对有向边，将它们按照极角从小到大排序。

Solution

- 如何 DP 一个凸包？
- 凸包上相邻的有向边的极角一定是递增的！
- 枚举 $O(n^2)$ 对有向边，将它们按照极角从小到大排序。
- 枚举一个点 S 作为起点，设 $f[x]$ 表示终点为 x 的最大面积。

Solution

- 如何 DP 一个凸包？
- 凸包上相邻的有向边的极角一定是递增的！
- 枚举 $O(n^2)$ 对有向边，将它们按照极角从小到大排序。
- 枚举一个点 S 作为起点，设 $f[x]$ 表示终点为 x 的最大面积。
- 按照极角从小到大依次松弛 $O(n^2)$ 条边，每次加入一个三角形，用 $f[S]$ 更新答案。

Solution

- 如何 DP 一个凸包？
- 凸包上相邻的有向边的极角一定是递增的！
- 枚举 $O(n^2)$ 对有向边，将它们按照极角从小到大排序。
- 枚举一个点 S 作为起点，设 $f[x]$ 表示终点为 x 的最大面积。
- 按照极角从小到大依次松弛 $O(n^2)$ 条边，每次加入一个三角形，用 $f[S]$ 更新答案。
- 时间复杂度 $O(n^3)$。

DP 套 DP

- DP 套 DP 是给定一个 DP 问题 A，用另一个 DP B 去计算一种可能的 A 的输入，使得 A 的 DP 结果为 x。

DP 套 DP

- DP 套 DP 是给定一个 DP 问题 A，用另一个 DP B 去计算一种可能的 A 的输入，使得 A 的 DP 结果为 x。
- 一般的做法是直接将 A 这个 DP 每个状态的 DP 值作为状态进行 DP。

Subsequences

给定 n 个小写字符串，考虑 $n!$ 种连接它们的顺序，问有多少种连接顺序最后得到的字符串有偶数个本质不同的子序列。

Subsequences

给定 n 个小写字符串，考虑 $n!$ 种连接它们的顺序，问有多少种连接顺序最后得到的字符串有偶数个本质不同的子序列。

- $n \leq 20$。

Subsequences

给定 n 个小写字符串，考虑 $n!$ 种连接它们的顺序，问有多少种连接顺序最后得到的字符串有偶数个本质不同的子序列。

- $n \leq 20$。
- $\sum len \leq 100000$。

Subsequences

给定 n 个小写字符串，考虑 $n!$ 种连接它们的顺序，问有多少种连接顺序最后得到的字符串有偶数个本质不同的子序列。

- $n \leq 20$。
- $\sum len \leq 100000$。
- 时限 5 秒。

Subsequences

给定 n 个小写字符串，考虑 $n!$ 种连接它们的顺序，问有多少种连接顺序最后得到的字符串有偶数个本质不同的子序列。

- $n \leq 20$。
- $\sum len \leq 100000$。
- 时限 5 秒。
- Source：XIX Open Cup GP of Siberia

Solution

- 考虑如何计算本质不同的子序列数量。

Solution

- 考虑如何计算本质不同的子序列数量。
- 设 $f[i]$ 表示以字符 i 为结尾的本质不同子序列数量，sum 表示本质不同子序列数量 (包括空串)。

Solution

- 考虑如何计算本质不同的子序列数量。
- 设 $f[i]$ 表示以字符 i 为结尾的本质不同子序列数量，sum 表示本质不同子序列数量 (包括空串)。
- 初始值：$f[i] = 0, sum = 1$。

Solution

- 考虑如何计算本质不同的子序列数量。
- 设 $f[i]$ 表示以字符 i 为结尾的本质不同子序列数量，sum 表示本质不同子序列数量 (包括空串)。
- 初始值：$f[i]=0, sum=1$。
- 考虑加入字符 x 后的变化：

  $f'[i]=f[i] (i \neq x)$

  $f'[x]=sum$，表示所有方案末尾都加上 x，而所有不加 x 的方案都可以通过去掉最后一个 x 然后加上一个 x 得到。

  $sum' = 2sum - f[x]$

Solution

- 考虑在模 2 意义下观察这个 DP：初始值：$f[i] = 0, sum = 1$。

Solution

- 考虑在模 2 意义下观察这个 DP：初始值：$f[i] = 0, sum = 1$。
- 考虑加入字符 x 后的变化：

  $f'[i] = f[i] (i \neq x)$

  $f'[x] = sum$

  $sum' = (2sum - f[x]) \bmod 2 = f[x]$

Solution

- 考虑在模 2 意义下观察这个 DP：初始值：$f[i]=0, sum=1$。
- 考虑加入字符 x 后的变化：

  $f'[i]=f[i](i\neq x)$

  $f'[x]=sum$

  $sum'=(2sum-f[x]) \bmod 2=f[x]$
- 可以发现任何时刻 f 与 sum 中有且仅有一个为 1。

Solution

- 考虑在模 2 意义下观察这个 DP：初始值：$f[i]=0, sum=1$。
- 考虑加入字符 x 后的变化：

  $f'[i]=f[i](i\neq x)$

  $f'[x]=sum$

  $sum'=(2sum-f[x]) \bmod 2=f[x]$
- 可以发现任何时刻 f 与 sum 中有且仅有一个为 1。
- 设 $dp[S][j]$ 表示考虑了 S 集合的字符串，f 与 sum 中 1 的位置在 j 的方案数，枚举下一个字符串转移。

Solution

- 考虑在模 2 意义下观察这个 DP：初始值：$f[i]=0, sum=1$。
- 考虑加入字符 x 后的变化：

  $f'[i]=f[i](i\neq x)$

  $f'[x]=sum$

  $sum'=(2sum-f[x]) \bmod 2=f[x]$
- 可以发现任何时刻 f 与 sum 中有且仅有一个为 1。
- 设 $dp[S][j]$ 表示考虑了 S 集合的字符串，f 与 sum 中 1 的位置在 j 的方案数，枚举下一个字符串转移。
- 时间复杂度 $O(27n2^n+27\sum len)$。

Independent Set

给定 m，请构造一棵点数不超过 15 的无根树，满足它的非空独立集个数恰好为 m。

Independent Set

给定 m，请构造一棵点数不超过 15 的无根树，满足它的非空独立集个数恰好为 m。

- $m \le 2000$。

Independent Set

给定 m，请构造一棵点数不超过 15 的无根树，满足它的非空独立集个数恰好为 m。

- $m \leq 2000$。
- Source： ZOJ 3951

Solution

- 假如知道了树的形态，那么可以树形 DP 求出独立集的个数。

Solution

- 假如知道了树的形态，那么可以树形 DP 求出独立集的个数。
- 设 $f[x][0/1]$ 表示考虑以 x 为根的子树，x 点选/不选时独立集的个数，转移显然。

Solution

- 假如知道了树的形态，那么可以树形 DP 求出独立集的个数。
- 设 $f[x][0/1]$ 表示考虑以 x 为根的子树，x 点选/不选时独立集的个数，转移显然。
- 为了让点数不超过 15，考虑 DP 出最少所需的点数。

Solution

- 假如知道了树的形态，那么可以树形 DP 求出独立集的个数。
- 设 $f[x][0/1]$ 表示考虑以 x 为根的子树，x 点选/不选时独立集的个数，转移显然。
- 为了让点数不超过 15，考虑 DP 出最少所需的点数。
- 设 $dp[i][j]$ 表示 $f[x][0]=i, f[x][1]=j$ 时，满足条件的树的点数的最小值，枚举儿子的情况转移。

Solution

- 假如知道了树的形态，那么可以树形 DP 求出独立集的个数。
- 设 $f[x][0/1]$ 表示考虑以 x 为根的子树，x 点选/不选时独立集的个数，转移显然。
- 为了让点数不超过 15，考虑 DP 出最少所需的点数。
- 设 $dp[i][j]$ 表示 $f[x][0]=i, f[x][1]=j$ 时，满足条件的树的点数的最小值，枚举儿子的情况转移。
- 状态数 $O(m^2)$。

动态线性 DP

- 序列上相邻几项之间进行转移的 DP。

动态线性 DP

- 序列上相邻几项之间进行转移的 DP。
- 一般方法是将 DP 改造成可以区间合并信息的方式。

动态线性 DP

- 序列上相邻几项之间进行转移的 DP。
- 一般方法是将 DP 改造成可以区间合并信息的方式。
- 比如知道 $[A, B]$ 和 $[B+1, C]$ 的 DP 值，可以很方便地算出 $[A, C]$ 的 DP 值，那么用线段树维护区间 DP 值就可以了。

Soldier Game

给定一个长度为 n 的序列 $a_1, a_2, \ldots, a_n$。

Soldier Game

给定一个长度为 n 的序列 $a_1, a_2, \ldots, a_n$。

你要把这个序列分成若干个连续区间，使得每个区间最多 2 个数。

Soldier Game

给定一个长度为 n 的序列 $a_1, a_2, \ldots, a_n$。

你要把这个序列分成若干个连续区间，使得每个区间最多 2 个数。

请最小化你的划分方案中，每个区间的和的极差。

Soldier Game

给定一个长度为 n 的序列 $a_1, a_2, \ldots, a_n$。

你要把这个序列分成若干个连续区间，使得每个区间最多 2 个数。

请最小化你的划分方案中，每个区间的和的极差。

- $n \leq 100000$。

Soldier Game

给定一个长度为 n 的序列 $a_1, a_2, \ldots, a_n$。

你要把这个序列分成若干个连续区间，使得每个区间最多 2 个数。

请最小化你的划分方案中，每个区间的和的极差。

- $n \le 100000$。
- $|a_i| \le 10^9$。

Soldier Game

给定一个长度为 n 的序列 $a_1, a_2, \dots, a_n$。

你要把这个序列分成若干个连续区间，使得每个区间最多 2 个数。

请最小化你的划分方案中，每个区间的和的极差。

- $n \le 100000$。
- $|a_i| \le 10^9$。
- Source： The 2018 ICPC Asia Qingdao Regional Contest

Solution

- 显然最多 $O(n)$ 种可能的区间。

Solution

- 显然最多 $O(n)$ 种可能的区间。
- 将这些区间按照区间和从小到大排序，枚举区间和的下界 l。

Solution

- 显然最多 $O(n)$ 种可能的区间。
- 将这些区间按照区间和从小到大排序，枚举区间和的下界 l。
- 只需要找到在给定的下界下上界的最小可能值即可。

Solution

- 显然最多 $O(n)$ 种可能的区间。
- 将这些区间按照区间和从小到大排序，枚举区间和的下界 l。
- 只需要找到在给定的下界下上界的最小可能值即可。
- 设 $f[i][0/1]$ 表示考虑前 i 个数，第 i 个数是否已经配对的区间和的最大值的最小可能值，可以 $O(n)$ 算出。

Solution

- 显然最多 $O(n)$ 种可能的区间。
- 将这些区间按照区间和从小到大排序，枚举区间和的下界 l。
- 只需要找到在给定的下界下上界的最小可能值即可。
- 设 $f[i][0/1]$ 表示考虑前 i 个数，第 i 个数是否已经配对的区间和的最大值的最小可能值，可以 $O(n)$ 算出。
- 时间复杂度 $O(n^2)$，不能接受。

Solution

- 考虑将这个 DP 改写成区间合并的形式。

Solution

- 考虑将这个 DP 改写成区间合并的形式。
- 设 $f[0/1][0/1]$ 表示考虑某个区间，最左和最右两个数是否已经配对的区间和的最大值的最小可能值，则可以 $O(1)$ 合并两个区间。

Solution

- 考虑将这个 DP 改写成区间合并的形式。
- 设 $f[0/1][0/1]$ 表示考虑某个区间，最左和最右两个数是否已经配对的区间和的最大值的最小可能值，则可以 $O(1)$ 合并两个区间。
- 线段树维护，时间复杂度 $O(n \log n)$。

Solution

- 考虑将这个 DP 改写成区间合并的形式。
- 设 $f[0/1][0/1]$ 表示考虑某个区间，最左和最右两个数是否已经配对的区间和的最大值的最小可能值，则可以 $O(1)$ 合并两个区间。
- 线段树维护，时间复杂度 $O(n\log n)$。
- 

最大连通子块和

给出一棵 n 个点，以 1 为根的有根树，点有点权。m 次操作：

最大连通子块和

给出一棵 n 个点，以 1 为根的有根树，点有点权。m 次操作：

1. 修改某个点的点权。

最大连通子块和

给出一棵 n 个点，以 1 为根的有根树，点有点权。m 次操作：

1. 修改某个点的点权。
2. 求以 x 为根的子树的最大连通子块和。

最大连通子块和

给出一棵 n 个点，以 1 为根的有根树，点有点权。m 次操作：

1. 修改某个点的点权。
2. 求以 x 为根的子树的最大连通子块和。

- $n, m \leq 200000$。

最大连通子块和

给出一棵 n 个点，以 1 为根的有根树，点有点权。m 次操作：

1. 修改某个点的点权。
2. 求以 x 为根的子树的最大连通子块和。

- $n, m \leq 200000$。
- Source：BZOJ 5210

Solution

- 先不考虑动态维护，考虑如何计算答案。

Solution

- 先不考虑动态维护，考虑如何计算答案。
- 设 $dp[x]$ 表示 x 作为最高点的最大连通块和，则 $dp[x] = \sum(\max(dp[son], 0)) + a[x]$

Solution

- 先不考虑动态维护，考虑如何计算答案。
- 设 $dp[x]$ 表示 x 作为最高点的最大连通块和，则 $dp[x]=\sum(\max(dp[son],0))+a[x]$
- 设 $x+1$ 为 x 的重儿子，轻儿子们为 son，令 $h[x]=\sum(\max(dp[son],0))+a[x]$，则 $dp[x]=\max(h[x],h[x]+h[x+1],h[x]+h[x+1]+h[x+2],\ldots)$

Solution

- 先不考虑动态维护，考虑如何计算答案。
- 设 $dp[x]$ 表示 x 作为最高点的最大连通块和，则 $dp[x] = \sum(\max(dp[son], 0)) + a[x]$
- 设 $x+1$ 为 x 的重儿子，轻儿子们为 son，令 $h[x] = \sum(\max(dp[son], 0)) + a[x]$，则 $dp[x] = \max(h[x], h[x] + h[x+1], h[x] + h[x+1] + h[x+2], \dots)$
- 容易发现 $dp[x] = h$ 的最大前缀和，而 x 子树中的 dp 最大值则为 h 的最大子段和。

Solution

- 对于每次修改，假设修改的是点 x。

Solution

- 对于每次修改，假设修改的是点 x。
- 从 x 出发往上走，暴力枚举所有轻边，更新对应点的 h 以及轻儿子子树中最大 dp 值。

Solution

- 对于每次修改，假设修改的是点 x。
- 从 x 出发往上走，暴力枚举所有轻边，更新对应点的 h 以及轻儿子子树中最大 dp 值。
- 对于重链的维护，则是经典问题，只需要用线段树维护就可以了。

Solution

- 对于每次修改，假设修改的是点 x。
- 从 x 出发往上走，暴力枚举所有轻边，更新对应点的 h 以及轻儿子子树中最大 dp 值。
- 对于重链的维护，则是经典问题，只需要用线段树维护就可以了。
- 时间复杂度 $O(m\log^2 n)$。

Thank you!