DP 入门

Contents

Contents

初始动态规划

想要回答”什么是动态规划”这种问题从来是很困难的，动态规划有着广阔的内涵，我们或许可以这样概括:
求解一类最优化问题，将原问题划分成子问题，子问题有非常工整的结构.

初始动态规划

想要回答”什么是动态规划”这种问题从来是很困难的，动态规划有着广阔的内涵，我们或许可以这样概括：
求解一类最优化问题，将原问题划分成子问题，子问题有非常工整的结构.

DP 术语：状态，状态转移方程.
子问题图：以状态为点，以状态之间的转移为边.
子问题图是一个 DAG.

初始动态规划

想要回答"什么是动态规划"这种问题从来是很困难的, 动态规划有着广阔的内涵, 我们或许可以这样概括:
求解一类最优化问题, 将原问题划分成子问题, 子问题有非常工整的结构.

DP 术语: 状态, 状态转移方程.
子问题图: 以状态为点, 以状态之间的转移为边.
子问题图是一个 DAG.

DP 的顺序:

- 递推 (bottom-up method)
- 记忆化搜索 (top-down with memoziation)

初始动态规划

想要回答" 什么是动态规划" 这种问题从来是很困难的, 动态规划有着广阔的内涵, 我们或许可以这样概括:
求解一类最优化问题, 将原问题划分成子问题, 子问题有非常工整的结构.

DP 术语: 状态, 状态转移方程.
子问题图: 以状态为点, 以状态之间的转移为边.
子问题图是一个 DAG.

DP 的顺序:

- 递推 (bottom-up method)
- 记忆化搜索 (top-down with memoziation)

问题的内涵: 最值, 方案, 方案数.

Contents

一般思路

一般 dp 形式为前 i 个元素，通过考虑对于最后一个元素的选择来转移.

最大子段和

已知 $\{a_n\}$, 求

$$\max_{1\le l\le r\le n}\{\sum_{i=l}^{r} a_i\}$$

$n \le 10^7$

最大子段和

解法一：考虑最短负前缀，要么是最大子段和是其前缀，要么与最大子段和无交.

最大子段和

解法一：考虑最短负前缀，要么是最大子段和是其前缀，要么与最大子段和无交.

解法二：DP，f_i 表示以 i 为结尾的最大子段和 (可能为空)，$f_i = \max(f_{i-1}, 0) + a_i$.

最大子段和

解法一：考虑最短负前缀，要么是最大子段和是其前缀，要么与最大子段和无交.

解法二：DP，f_i 表示以 i 为结尾的最大子段和 (可能为空)，$f_i = \max(f_{i-1}, 0) + a_i$.

解法三：考虑前缀和，问题变成求 $\max_{0\le i<j\le n}\{s_j - s_i\} = \max_{1\le j\le n}\{s_j - \min_{0\le i<j}\{s_i\}\}$

动态最大子段和

维护一个序列，支持

- 单点修改
- 查询区间最大子段和

$n, m \leq 3 * 10^5$

动态最大子段和

维护一个序列，支持

- 单点修改
- 查询区间最大子段和

$n, m \leq 3 * 10^5$

使用线段树，每个区间维护:

- 包含左端点的最大子段和
- 包含右端点的最大子段和
- 最大子段和
- 区间和

最长上升子序列

一个长度为 s 的上升子序列 P 被定义为
$P = (i_1, \cdots, i_s), 1 < i_1 < \cdots < i_s, a_{i_1} < a_{i_2} < \cdots < a_{i_s}$, 求最长的上升子序列的长度是多少?
$n \leq 10^6$

最长上升子序列

解法一: $f(i)$ 表示以 i 结尾的最长上升子序列.

$$f(i) = \max_{1 \le j < i}\{f(j)\}$$

使用树状数组求前缀 max.
Q: 为什么在这个题中树状数组可以求前缀 max?

最长上升子序列

解法一：$f(i)$ 表示以 i 结尾的最长上升子序列.

$$f(i) = \max_{1 \le j < i} \{f(j)\}$$

使用树状数组求前缀 max.
Q: 为什么在这个题中树状数组可以求前缀 max?

解法二：按 j 从小到大的顺序求 $f(j)$, 对于每一个 k, 使用一个数组 g 维护 $\min\{a_i\}, f(i) = k$.

最长上升子序列

解法一：$f(i)$ 表示以 i 结尾的最长上升子序列.

$$f(i) = \max_{1 \le j < i}\{f(j)\}$$

使用树状数组求前缀 max.
Q: 为什么在这个题中树状数组可以求前缀 max?

解法二：按 j 从小到大的顺序求 $f(j)$, 对于每一个 k, 使用一个数组 g 维护 $\min\{a_i\}, f(i) = k$.

Q: 如何统计最长子序列的方案数?

最长上升子序列

解法一: $f(i)$ 表示以 i 结尾的最长上升子序列.

$$f(i) = \max_{1 \le j < i}\{f(j)\}$$

使用树状数组求前缀 max.
Q: 为什么在这个题中树状数组可以求前缀 max?

解法二: 按 j 从小到大的顺序求 $f(j)$, 对于每一个 k, 使用一个数组 g 维护 $\min\{a_i\}, f(i) = k$.

Q: 如何统计最长子序列的方案数?

在 g 中套一个 vector, vector 每个元素记录历史权值大小和方案数的前缀和.

最长公共子序列

已知两个整数序列 $\{a_n\}$, $\{b_n\}$, 求它们的最长公共子序列.
长度为 s 的公共子序列定义为
$(i_1, i_2, \cdots, i_s)$, $i_1 < i_2 < \cdots < i_s$, $a_{i_1} = b_{i_1}$,
$a_{i_2} = b_{i_2}, \cdots, a_{i_n} = b_{i_n}$.
$n \le 5000$

最长公共子序列

已知两个整数序列 $\{a_n\}$, $\{b_n\}$, 求它们的最长公共子序列.
长度为 s 的公共子序列定义为
$(i_1, i_2, \cdots, i_s)$, $i_1 < i_2 < \cdots < i_s$, $a_{i_1} = b_{i_1}$, $a_{i_2} = b_{i_2}, \cdots, a_{i_n} = b_{i_n}$.
$n \leq 5000$

$f(i, j)$ 表示 a_{1-i}, b_{1-j} 的最长公共子序列.

$$f(i,j) = \min \begin{cases} f(i-1, j-1) & , a_i = b_j \\ f(i, j-1) \\ f(i-1, j) \end{cases}$$

最长公共上升子序列

最优矩阵链乘

有 n 个矩阵 $A_1, A_2, \cdots, A_n$，每个矩阵的大小是 $p_i \times q_i$，求计算 $\prod_{i=1}^{n} A_i$ 最少需要多少次整数乘法？有多少种方案来计算这个乘积？

$n \leq 5000$

最优矩阵链乘

有 n 个矩阵 $A_1, A_2, \cdots, A_n$，每个矩阵的大小是 $p_i \times q_i$，求计算 $\prod_{i=1}^{n} A_i$ 最少需要多少次整数乘法？有多少种方案来计算这个乘积？
$n \le 5000$

$f(l,r)$ 表示把 $[l,r]$ 的矩阵乘起来的最少需要的整数乘法次数.

$$f(l,r) = \min_{l \le i < r}\{f(l,i) + f(i+1,r) + p_l q_i q_r\}$$

方案数：Catlan 数

Contents

一般思路

状态为区间.
常见的有两种转移方法:

- 考虑区间端点处的选择.
- 考虑区间的划分.

后者复杂度可能较高, 可以通过四边形不等式优化.

关路灯

一条马路上有 n 盏路灯，第 i 盏路灯在位置 a_i，每秒钟会消耗 w_i 的能源．现在你在位置 x 处，每秒钟可以移动一个单位距离．当你走到路灯所在的位置后你可以把路灯关掉，现在你需要把所有的路灯都关掉，此时消耗的最少能源是多少？
$n \leq 5000$

关路灯

一条马路上有 n 盏路灯，第 i 盏路灯在位置 a_i，每秒钟会消耗 w_i 的能源．现在你在位置 x 处，每秒钟可以移动一个单位距离．当你走到路灯所在的位置后你可以把路灯关掉，现在你需要把所有的路灯都关掉，此时消耗的最少能源是多少？
$n \leq 5000$

关掉的路灯必然是一个区间．
$f(l, r, 0/1)$ 表示关掉第 l 到第 r 个路灯，都关上之后在左/右边的方案数．

$$f(l, r, 0) = \min \begin{cases} f(l+1, r, 0) + a_{l+1} - a_l \\ f(l+1, r, 1) + a_r - a_l \end{cases}$$

$$f(l, r, 1) = \min \begin{cases} f(l, r-1, 0) + a_r - a_l \\ f(l, r-1, 1) + a_r - a_{r-1} \end{cases}$$

石子合并

有 n 堆石子排成一排，第 i 堆石子的重量为 a_i，你可以把相邻的两堆石子合并成一堆，合并的代价是石子重量和．最小化把
这 n 堆石子合并成一堆的方案数．
$n \le 400$

石子合并: 区间 DP

令 $f(l, r)$ 表示将第 l 堆到第 r 堆石子合并的最小代价.

$$f(l, r) = \min_{l \le i < r} \{ f(l, i) + f(i + 1, r) \} + \sum_{i=l}^{r} a_i$$

最优二叉搜索树

有 1-n 这 n 个数，对于 i 而言有 a_i 的概率会询问 i，对其建立一个最优二叉搜索树使得查询的期望深度最小.

$n \leq 500$

Contents

一般形式

一般处理环有两种方法:

- 将环倍长，将环的条件改对区间长度的限制条件
- 枚举环中的一点，破环为链

环形最大子段和

在大小为 n 的环 (环上每个元素是一个正数) 上选一个子段 (不能有重复元素), 使得子段和最大.

$n \leq 10^7$

环形最大子段和

在大小为 n 的环 (环上每个元素是一个正数) 上选一个子段 (不能有重复元素), 使得子段和最大.
$n \le 10^7$

将环倍长, DP 不再适用, 但前缀和的方法依然可行.

$$\max_{0 \le l < r \le 2n \wedge r - l \le n} \{s_r - s_l\}$$

需要使用单调队列优化.

环上最大带权独立集

在大小为 n 的环上选择若干个不相邻的数，使得其和最大.
$n \le 10^7$

环上最大带权独立集

在大小为 n 的环上选择若干个不相邻的数，使得其和最大.
$n \leq 10^7$

枚举第一个数选或者不选，即可破环为链.

Contents

一般形式

往往在 LCA 处统计信息：链，连通点集，子树...

一般形式

往往在 LCA 处统计信息: 链, 连通点集, 子树...

LCA 的性质:

$$LCA(a, b) = LCA(b, a)$$

$$LCA(LCA(a, b), c) = LCA(a, LCA(b, c))$$

所以我们可以定义 $LCA(A), A \subset V$.

一般形式

往往在 LCA 处统计信息: 链, 连通点集, 子树...

LCA 的性质:

$$LCA(a, b) = LCA(b, a)$$

$$LCA(LCA(a, b), c) = LCA(a, LCA(b, c))$$

所以我们可以定义 $LCA(A), A \subset V$.
Q: 树上连通点集的 LCA 是否一定属于这个连通点集?

最优连通子集

给定一颗树，每个点都有权值，在上面选择一些相互之间均能够连通的点，使得这些点的权值和最大。

树的直径

求一棵 n 个节点有边权的树的直径 (距离最远的点对的距离),
(不) 保证树的边权非负?
$n \leq 4 * 10^6$

树的直径

求一棵 n 个节点有边权的树的直径 (距离最远的点对的距离),
(不) 保证树的边权非负?
$n \leq 4 * 10^6$

权值非负: 贪心.
从任意一个点 x 出发, 找到距离 x 最远的点 u, 再找到距离 u 最远的点 v, u 和 v 之间的距离便是直径.

树的直径

求一棵 n 个节点有边权的树的直径 (距离最远的点对的距离), (不) 保证树的边权非负?
$n \le 4 * 10^6$

权值非负: 贪心.
从任意一个点 x 出发, 找到距离 x 最远的点 u, 再找到距离 u 最远的点 v, u 和 v 之间的距离便是直径.

权值可以为负: DP. 令 f_i 表示从 i 往下走的最远距离.

$$f_i = \max_{j \in child_i} \{f_j + w(i,j)\}$$

$$ans = \max_i \max_{j \neq k \in child_i} \{f_j + w(i,j) + f_k + w(i,k)\}$$

在每个节点求一个最大值和次大值.

树上最远距离

求树中距离每个点最远的点的距离.

$n \leq 4 * 10^6$

树上最远距离

求树中距离每个点最远的点的距离.
$n \leq 4 * 10^6$

先 dp 出上一道题的 f, 再以 g_i 表示 i 往上走的最远距离.

$$g_i = \max \begin{cases} g_{parent_i} \\ f_j + w_{j,parent_i} & j \in sibling_i \end{cases} + w_{i,parent_i}$$

树上最大独立集

在树上选出最多的点使得它们两两不相邻.
有/无点权.
$n \leq 10^7$

树上最大独立集

在树上选出最多的点使得它们两两不相邻.
有/无点权.
$n \leq 10^7$

无点权: 贪心.
不断地选叶节点, 删掉与之相邻的节点.

树上最大独立集

在树上选出最多的点使得它们两两不相邻.
有/无点权.
$n \leq 10^7$

无点权: 贪心.
不断地选叶节点, 删掉与之相邻的节点.

带点权: DP. $f_{i,0/1}$ 表示以 i 为根的子树, i 不选/选的答案.

$$f_{i,1} = \sum_{j \in child_i} f_{j,0}$$

$$f_{i,0} = \sum_{j \in child_i} \max\{f_{j,0}, f_{j,1}\}$$

$$ans = \max\{f_{root,0}, f_{root,1}\}$$

树上背包

例题：[JSOI2016] 最佳团体

JSOI 信息学代表队一共有 N 名候选人，这些候选人从 1 到 N 编号。方便起见，JYY 的编号是 0 号。每个候选人都由一位编号比他小的候选人 R_i 推荐。如果 $R_i = 0$，则说明这个候选人是 JYY 自己看上的。

为了保证团队的和谐，JYY 需要保证，如果招募了候选人 i，那么候选人 R_i 也一定需要在团队中。当然了，JYY 自己总是在团队里的。每一个候选人都有一个战斗值 P_i，也有一个招募费用 S_i。JYY 希望招募 K 个候选人（JYY 自己不算），组成一个性价比最高的团队。也就是，这 K 个被 JYY 选择的候选人的总战斗值与总招募费用的比值最大。

$1 \le K \le N \le 2500, 0 < S_i, P_i \le 10^4, 0 \le R_i < i$

树上背包

例题： Rikka with Game

很久很久以前，有一块圣地亚哥大陆，大陆上有 n 个生产金坷垃的城市和 $n-1$ 条道路。每一对城市都可以通过这些道路相互到达。

现在你想要建立你自己国家并让你的故乡——1 号城市成为首都。接着你会切断所有连接着在你国家中的城市和不在你国家中的城市的道路。为了社会的稳定，你的国家必须满足：

1.1 号城市必须在你的国家中。

2. 你的国家中的每一对城市都可以通过还没有被切断的道路相互到达。

实际上，在切断了道路之后，圣地亚哥大陆被分成了很多个联通块。每一个联通块都发展成了一个国家。为了世界的和平，你打算选择至多 k 个其他的国家（当然不可能是自己的国家了）建立外交关系。

每一个城市都有一个繁荣值 w_i，每一个国家的繁荣值等于在这个国家中的所有城市的繁荣值之和。你建立的国家稳定值等于你的国家的繁荣值加上 $a\times$ 所有和你建立外交关系的国家的繁荣值之和。

为了增加金坷垃的产量，你需要最大化你的国家的稳定值。

环套树上的 DP

环套树的定义: 有且仅有一个环的连通图 (若为有向图则是指其底环连通).
其实就是树形 DP+ 区间 DP.

环套树上最大独立集

在带点权的环套树上求最大独立集.
$n \leq 3 * 10^6$

环套树上最大独立集

在带点权的环套树上求最大独立集.
$n \le 3 * 10^6$

枚举环上一点选/不选, 将环套树破成树做即可.

环套树的直径

有边权的环套树上求两点间最小距离的最大值.

$n \leq 3 * 10^6$

环套树的直径

有边权的环套树上求两点间最小距离的最大值.
$n \leq 3 * 10^6$

直径分为两种情况: 经过环的和不经过环的.
处理出环上每一棵树中的直径 g_i 和最大带权深度 f_i.
将环倍长 (环长为 m, 设其节点编号为 1-m(倍长之后是 1-2m), a_i 为 i-1 号节点和 i 号节点之间的距离, $s_i = \sum_{j=2}^{i} a_j$).

$$ans = \max \begin{cases} s_r - s_l + f_l + f_r & r - l < m \land 0 < l < r \leq 2m \\ g_i \end{cases}$$

需要使用单调队列优化.

Contents

一般思路

DP 状态是一个集合 S.
常见转移:

- 枚举某一个元素选/不选.
- 枚举子集

Q: 如何枚举子集?

子集和

对于所有的$i, 0 \leq i \leq 2^n - 1$，求解$\sum_{j \subset i} a_j$。

$n \leq 2$^20

子集和

对于所有的i, $0 \leq i \leq 2^n - 1$，求解$\sum_{j \subset i} a_j$。

$n \leq 2$^20

$f(i, S)$ 表示在 S 中，前 i 个数是子集和，其余的数的选择确定的方案数.

$$f(i, S) = \begin{cases} f(i-1, S) & i \notin S \\ f(i-1, S) + f(i-1, S - \{i\}) & i \in S \end{cases}$$

可以使用滚动数组优化空间.

也可以理解为高维前缀和.

又见 LIS

1-n 的排列中有多少个满足 LIS 的长度为 k?

$n \leq 14$

$n \leq 16$

又见 LIS

1-n 的排列中有多少个满足 LIS 的长度为 k?
$n \leq 14$
$n \leq 16$

解法一：观察到输入的所有可能并不多，可以尝试搜索 + 打表．$O(n!)$

又见 LIS

1-n 的排列中有多少个满足 LIS 的长度为 k?
$n \leq 14$
$n \leq 16$

解法一: 观察到输入的所有可能并不多, 可以尝试搜索 + 打表. $O(n!)$
解法二: 我们从一个空排列开始考虑, 不断地把每一个数加入排列, 这时对于每一个数有三种情况:

- 已在排列中, 但不在g数组中.
- 已在排列中, 且在g数组中.
- 尚未在排列中.

我们需要使用三进制来表示状态.

又见 LIS

1-n 的排列中有多少个满足 LIS 的长度为 k?
$n \leq 14$
$n \leq 16$

解法一：观察到输入的所有可能并不多，可以尝试搜索 + 打表。$O(n!)$
解法二：我们从一个空排列开始考虑，不断地把每一个数加入排列，这时对于每一个数有三种情况：

- 已在排列中，但不在g数组中.
- 已在排列中，且在g数组中.
- 尚未在排列中.

我们需要使用三进制来表示状态.
实际上我们不需要记哪些数字在排列中,转移时考虑下个数字和前面数字的相对关系就可以了.复杂度O(n2^n).

Contents

大容量的背包

n 个物品，每个物品有体积 v_i，价值 w_i，背包容量为 V，最大化价值总和.
$n \leq 40, V \leq 10^9$

大容量的背包

n 个物品, 每个物品有体积 v_i, 价值 w_i, 背包容量为 V, 最大化价值总和.
$n \leq 40, V \leq 10^9$

meet-in-middle: 把物品分成两半 A 和 B, 每一半枚举出其 2^n 种情况. 将 A 中的子集按体积排序. 枚举 B 中的所有情况, 在 A 中寻找加起来体积不超过 V, 且价值最大者.
可以在 A 中二分, 或者将 B 也按体积排序, 扫描 B 时最优的 A 即会是单调的.

大容量的背包

n 个物品, 每个物品有体积 v_i, 价值 w_i, 背包容量为 V, 最大化价值总和.
$n \leq 40, V \leq 10^9$

meet-in-middle: 把物品分成两半 A 和 B, 每一半枚举出其 2^n 种情况. 将 A 中的子集按体积排序. 枚举 B 中的所有情况, 在 A 中寻找加起来体积不超过 V, 且价值最大者.
可以在 A 中二分, 或者将 B 也按体积排序, 扫描 B 时最优的 A 即会是单调的.

$n \leq 500, V \leq 10^9, w_i \leq 500$

大容量的背包

n 个物品，每个物品有体积 v_i，价值 w_i，背包容量为 V，最大化价值总和.
$n \leq 40, V \leq 10^9$

meet-in-middle: 把物品分成两半 A 和 B，每一半枚举出其 2^n 种情况. 将 A 中的子集按体积排序. 枚举 B 中的所有情况，在 A 中寻找加起来体积不超过 V，且价值最大者.
可以在 A 中二分，或者将 B 也按体积排序，扫描 B 时最优的 A 即会是单调的.

$n \leq 500, V \leq 10^9, w_i \leq 500$

改为求 $f(i,j)$ 表示前 i 个物品，总价值为 j，体积最小是多少.

多重背包

n 个物品，每个物品有体积 v_i，价值 w_i，可以选至多 p_i 次，背包容量为 V，最大化价值总和.

$n, V \leq 1000$

$n, V \leq 5000$

多重背包

n 个物品，每个物品有体积 v_i，价值 w_i，可以选至多 p_i 次，背包容量为 V，最大化价值总和.

$n, V \leq 1000$

$n, V \leq 5000$

二进制拆分：把每个物品拆成其 $2^0, 2^1, \cdots, 2^{\lfloor log_2 p_i \rfloor - 1}, p_i - 2^{\lfloor log_2 p_i \rfloor} + 1$ 倍.

多重背包

n 个物品, 每个物品有体积 v_i, 价值 w_i, 可以选至多 p_i 次, 背包容量为 V, 最大化价值总和.
$n, V \leq 1000$
$n, V \leq 5000$

二进制拆分: 把每个物品拆成其 $2^0, 2^1, \cdots, 2^{\lfloor log_2 p_i \rfloor - 1}, p_i - 2^{\lfloor log_2 p_i \rfloor} + 1$ 倍.

单调队列优化

$$f(i,j) = \max_{0 < k \leq p_i \wedge kv_i \leq j} \{f(i-1, j-kv_i) + kw_i\}$$

Contents

参考资料

[1] 刘汝佳. 算法竞赛入门经典. 清华大学出版社.
[2] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein. Introduction to Algorithms(Third Edition). MIT Press.
[3] https://en.wikipedia.org/
[7] https://oi-wiki.org/