区间 dp 和树型 dp

冉雨杭

2023 年 8 月 7 日

区间 DP

定义

- 区间 dp 是线性 dp 的扩展。
- 区间 dp 问题往往可以用状态 $f(l, r)$ 表示区间 $[l, r]$ 内的" 最优答案"。
- 这类 dp 转移通常有合并性，即计算 $f(l, r)$ 时，可以通过枚举中间点 mid，$f(l, r)$ 的答案可以由小区间合并得到。
- 计算区间 dp 时，通常按区间长度从小到大的顺序计算/记忆化搜索

最长回文子序列

- 给定一个长度为 n 的字符串
- 求它最长的回文子序列是多少？
- $n \leq 3000$

最长回文子序列

- 令 $f(l, r)$ 表示区间 $[l, r]$ 内最长回问子序列的长度
- $l = r$ 时 $f(l, r) = 1$，$l = r + 1$ 时 $f(l, r) = 0$
- 如果 $s[l] == s[r]$，则 $f(l, r) = f(l + 1, r - 1) + 2$
- 否则 $f(l, r) = max(f(l + 1, r), f(l, r - 1))$
- 复杂度 $O(n^2)$

区间 DP

```
for (int len = 1; len <= n; len++) {
    for (int l = 1; l + len - 1 <= r; l++) {
        int r = l + len - 1;
        for (int mid = l; mid < r; mid++) {
            f[l][r] = merge(f[l][mid], f[mid + 1][r]);
        }
    }
}
```

矩阵乘法

- 给定 n 个矩阵 $A_1, A_2, \cdots, A_n$
- 每个矩阵是 h_i 行 c_i 列，保证对于 $i > 1$ 有 $c_i = h_{i+1}$，
- 现在你可以决定乘法的运算顺序，一个 a 行 b 列的矩阵和一个 b 行 c 列的矩阵相乘需要 abc 次运算
- 求最少运算次数
- $n \leq 500$

矩阵乘法

- $f(l, r)$ 表示区间 $[l, r]$ 都完成乘法后所需的最小次数是多少
- 枚举最后一次合并的矩阵是哪两个子区间合并来的
- $f(l, r) = \sum_{mid} f(l, mid) + f(mid + 1, r) + h_l * c_{mid} * c_r$
- 时间复杂度：$O(n^3)$

石子合并

- 在一个 N 堆石子排成一排，现要将石子有次序地合并成一堆，规定每次只能选相邻的 2 堆合并成新的一堆，并将新的一堆的石子数，记为该次合并的得分。
- 求合并成一堆所需要的最小得分
- $N \leq 500$

石子合并

- $f(l, r)$ 表示区间 $[l, r]$ 都合并完成的最小代价是多少
- 枚举最后一次合并是哪两个子区间合并来的
- $f(l, r) = \sum_{mid} f(l, mid) + f(mid + 1, r) + sum[r] - sum[l - 1]$
- 时间复杂度：$O(n^3)$

括号匹配

- 有 n 种括号
- 定义一个字符串是" 合法" 的如下:
- 空串是合法的。
- 如果 s 是合法的，(s) 是合法的，其中 $()$ 必须是同种类型的左右括号
- 如果 s, t 是合法的，st 是合法的。
- 给一个字符串 S，求它最长的合法子序列。
- $|S| \leq 500$。

括号匹配

- 用 $f(l, r)$ 表示 $s[l, r]$ 的最长合法子序列长度。
- 如果 $l = r$，则 $f(l, r) = 0$。
- 否则 $s[l, r]$ 的最长合法子序列有两种情况：
- 如果 l 和 r 是同类的左右括号，且这两个字符在合法子序列中匹配，$f(l, r) = f(l + 1, r - 1) + 2$。
- 由两段合法子序列拼接而成，

$$f(l, r) = \max_{l \le mid < r} f(l, mid) + f(mid + 1, r).$$

- 复杂度：$O(|S|^3)$。

三角划分

- 给定一个 n 个点的凸多边形
- 每个顶点有一个权值
- 划分成 $n-2$ 个三角形，每个三角形的权值是顶点权值的乘积，划分代价是所有三角形的权值和
- 求最小权值划分
- $n \leq 500$

三角划分

- 用 $f(l, r)$ 表示顶点 $i, i+1, \cdots, j$ 三角划分的最小代价
- 如果 $l+1=r$，则 $f(l, r)=0$
- 否则枚举划分点：

$$f(l, r)=\min_{k} f(l, k)+f(k, r)+a_l * a_k * a_r$$

- 复杂度 $O(n^3)$

CF 1132F

- 给你一个串 s，每次可以花费 1 的代价删去一个子串，要求子串的每一位为同一个字符。
- 例如 abcddcba 中的 dd
- 求删去整个串的最小代价
- $1 \leq |s| \leq 500$

CF 1132F

- 用 $f(l, r)$ 表示删除 $s[l, \cdots, r]$ 子串所需要花费的最小代价
- 只删旁边一个字符：$f(l, r) \leftarrow min(f(l+1, r)+1, f(l, r-1))+1$
- 枚举合并点：$f(l, r) \leftarrow min_{s[k]==s[l]} f(l+1, k-1)+f(k, r)$
- 复杂度 $O(|S|^3)$

P1063

- 给定 n 个矩阵 $A_1, A_2, \cdots, A_n$
- 每个矩阵是 h_i 行 c_i 列，保证对于 $i > 1$ 有 $c_i = h_{i+1}$，
- 现在你可以决定乘法的运算顺序，一个 a 行 b 列的矩阵和一个 b 行 c 列的矩阵相乘需要 abc 次运算
- 矩阵是环形放置的，保证最后一个矩阵可以和第一个矩阵进行乘法运算
- 求最少运算次数
- $n \leq 100$

P1063

- 我会枚举断点，然后再进行区间 dp!
- 复杂度 $O(n^4)$
- 还能再进行优化吗

P1063

- 破环为链，环形问题最常用的技巧！
- 将原来的内容复制一份放在后面，相当于在这个环上走了两圈
- 以任何一个点为起点的走法都被这个是这个长度为 $2n$ 的序列的一个连续子序列
- 最后枚举所有 $f_{i,i+n-1}$ 更新答案即可
- 复杂度 $O(n^3)$

树形 dp

- 树形 dp，即在树上进行的 dp。由于树固有的递归性质，树形 dp 一般都是递归进行的。
- 在树上的 dp，一般是先解决子树的答案，然后求出整棵树的答案。

树上最大独立集

- 独立集：在图中两两互不相邻的顶点构成的点的集合。
- 最大独立集：所有独立集中点个数最多的集合。
- 树上最大独立集问题：给定一棵大小为 n 的树，求它的最大独立集的大小。
- $n \leq 10^6$

树上最大独立集

- 好像可以直接贪心，能选就尽量选！
- 是正确的吗？

树上最大独立集

- $dp_{u,0/1}$ 表示考虑 u 及其子树内，u 这个点没选/选了的时候最大独立集是多少
- $dp_{u,1} = (\sum_{v \in son_u} dp_{v,0}) + 1$
- $dp_{u,0} = \sum_{v \in son_u} max(dp_{v,1}, dp_{v,0})$
- 复杂度：$O(n)$

树上最小点覆盖

- 点覆盖：在图中的点的集合，保证图中任意一条边的两个端点都至少有一个点属于该集合。
- 最小点覆盖：所有点覆盖中点个数最少的集合。
- 树上最小点覆盖问题：给定一棵大小为 n 的树，求它的最小点覆盖的大小。
- $n \leq 10^6$。

树上最小点覆盖

- $dp_{u,0/1}$ 表示考虑 u 及其子树内，u 这个点没选/选了的时候最小点覆盖是多少
- $dp_{u,1} = (\sum_{v \in son_u} min(dp_{v,0}, dp_{v,1}) + 1$
- $dp_{u,0} = \sum_{v \in son_u} dp_{v,1}$
- 复杂度：$O(n)$

树的直径

- 给定一个 n 个点的树，求它的直径是多少
- 直径长度等于最长的简单路径长度
- $n \leq 10^6$

树的直径

- 树的直径有两种做法：两次 bfs，树 dp
- 先任意找一个点作为起点进行 bfs，然后找到它能走到的最远的点
- 再用那个最远的点作为起点 bfs，能走到的最远距离即为直径

树的直径

```
auto bfs = [&](int s) {
    std::vector<int> dp(n, -1);
    dp[s] = 0;
    std::queue<int> que;
    que.emplace(s);
    while(!que.empty()) {
        int u = que.front();
        que.pop();
        for (auto v : adj[u]) {
            if (dp[v] == -1) {
                dp[v] = dp[u] + 1;
                que.emplace(v);
            }
        }
    }
    return std::max_element(dp.begin(), dp.end()) - dp.begin();
};
int s = bfs(0);
int e = bfs(s);
```

树的直径

- dp_u 表示从 u 开始往下走最深能走到哪里
- 可以在求 dp 的过程中顺便求出直径

```
void dfs(int u, int fa) {
    for (auto v : adj[u]) {
        if(v == fa) continue;
        dfs(v, u);
        res = max(res, dp[u] + dp[v] + 1);
        dp[u] = max(dp[u], dp[v] + 1);
    }
}
```

P3304

- 求树的直径以及有多少条边是所有直径的必须边
- $n \leq 2 \times 10^5$

P3304

引理 1

- 所有直径都会经过同一个中点

引理 2

- 答案一定是某条直径上的一个区间

P3304

- 先 dfs 求出直径及其端点
- 找到中点开始 dfs，并且找到每个子树内的最长路
- 如果一个点往其它子树分叉也能有相同的长度，那么对应的这条边就不是必经边
- 复杂度 $O(n)$

树上背包

- 给定一棵有 n 个节点的点权树
- 要求你从中选出 m 个节点，使得这些选出的节点的点权和最大
- 一个节点能被选当且仅当其父亲节点被选中，根节点可以直接选
- $n, m \leq 3000$

树上背包

- $dp_{u,i}$ 表示 u 及其子树内选了 i 个点的最大权值和是多少
- 每次枚举一个子树 v，然后相当于做了一次背包合并
- 写出如下代码：
- 复杂度 $O(n^3)$

```
void dfs(int u, int fa) {
    dp[u][1] = a[u];
    for (auto v : adj[u]) {
        if (v == fa) continue;
        dfs(v, u);
        for (int i = n; i >= 1; i--) {
            for (int j = i; j >= 0; j--) {
                dp[u][i + j] = min(dp[u][i + j], dp[u][i] + dp[v][j]);
            }
        }
    }
}
```

树上背包

- 改两个小地方，可以直接优化复杂度到 $O(n^2)$

```
void dfs(int u, int fa) {
    dp[u][1] = a[u];
    sz[u] = 1;
    for (auto v : adj[u]) {
        if (v == fa) continue;
        dfs(v, u);
        for (int i = sz[u]; i >= 1; i--) {
            for (int j = sz[v]; j >= 0; j--) {
                dp[u][i + j] = min(dp[u][i + j], dp[u][i] + dp[v][j]);
            }
        }
        sz[u] += sz[v];
    }
}
```

- 怎么分析这个复杂度呢？

树上背包

- 任意两个点只会被枚举到一次
- 会在什么地方被枚举到？

树上背包

- 任意两个点只会被枚举到一次
- 会在什么地方被枚举到？
- 任意两个点只会在它们的 LCA 处被枚举到一次

CF 23E

- 给一棵 n 个点的树，现在你需要删除一些边，一种删除方案的价值是所有连通块大小的乘积，问最大价值是多少？
- $n \leq 700$

CF 23E

- $dp_{u,i}$ 表示 u 及其子树内，与 u 连着的连通块大小是 i 时，不包含 u 的连通块的乘积最大是多少
- 转移枚举子树及大小，树背包合并即可
- 复杂度：$O(n^2)$

CF GYM 102992 M

- 给定一个 n 个点的树，1 为根
- 每个点有一个怪兽
- 你可以用超能力免费消除 k 个怪兽
- 剩下每个点 u 存活的怪兽你需要花费 $a_u + \sum_{v存活 \& v \in son_u} a_v$
- 求消灭所有怪兽的最少花费，输出 $0 \le k \le n$ 的所有答案
- $n \le 2000$

CF GYM 102992 M

- $dp_{u,i,0/1}$ 表示 u 及其子树内删除了 i 个点，u 这个点没删/删了的最小花费是多少
- 初始化：$dp_{u,0,0} = 0, dp_{u,1,1} = a_u$
- $dp_{u,i+j,0} \leftarrow dp_{u,i,0} + dp_{v,j,0}$
- $dp_{u,i+j,0} \leftarrow dp_{u,i,0} + dp_{v,j,1}$
- $dp_{u,i+j,1} \leftarrow dp_{u,i,1} + dp_{v,j,0}$
- $dp_{u,i+j,1} \leftarrow dp_{u,i,1} + dp_{v,j,1} + a_v$

P4170 涂色

- 假设你有一条长度为 5 的木板，初始时没有涂过任何颜色。你希望把它的 5 个单位长度分别涂上红、绿、蓝、绿、红色，用一个长度为 5 的字符串表示这个目标：RGBGR。
- 每次你可以把一段连续的木板涂成一个给定的颜色，后涂的颜色覆盖先涂的颜色。例如第一次把木板涂成 RRRRR，第二次涂成 RGGGR，第三次涂成 RGBGR 达到目标。
- 用尽量少的涂色次数达到目标。
- $n \leq 50$

P3146

- 给定一个 $1 \times n$ 的地图，在里面玩 2048，每次可以合并相邻两个（数值范围 $[1, 40]$）。问序列中出现的最大数字的值最大是多少
- 注意合并后的数值并非加倍而是 $+1$，2 和 2 合成的是 3
- $n \leq 248$

Gym 101505F

- 给定 n 个节点的树。你要选择 k 个节点，使得每个被选择的节点周围至少有一个被选择的节点。求方案数
- $k, n \leq 200$

P2014 选课

- 在大学里每个学生，为了达到一定的学分，必须从很多课程里选择一些课程来学习，在课程里有些课程必须在某些课程之前学习，如高等数学总是在其它课程之前学习。现在有 n 门功课，每门课有个学分，每门课有一门或没有直接先修课（若课程 a 是课程 b 的先修课即只有学完了课程 a，才能学习课程 b)。一个学生要从这些课程里选择 m 门课程学习，问他能获得的最大学分是多少？
- $n, m \leq 300, s_i \leq 20, k_i \leq n$

习题

- CF607B
- P3205
- P1063
- P1070
- P2015
- P2014
- P1352

谢谢!