动态规划

树形 DP、状压 DP 及动规的常用优化

郑欣

2024 年 7 月 28 日

1. 树形 DP
   - 树形背包
   - 基环树 DP
   - 换根 DP
2. 状压 DP
3. DP 优化

例题 1

Luogu P1352　没有上司的舞会（树上最大权点独立集）

给一棵树，i 的点权为 a_i，求带权最大点独立集。即选出权值之和尽可能大的点，使得没有一条边的两个顶点同时选中。

范围：$n \le 10^6$。

约定 1 为根节点。用 $f_{u,0/1}$ 表示 T_u（u 为根的子树）的最大点独立集大小，其中 1 表示选 u，0 表示不选 u。

分两种情况讨论：

- 如果不选 u，那么 u 的儿子选不选都可以。即 $f_{u,0} = \sum_{v\in \mathrm{son}_u} \max\{f_{v_0}, f_{v_1}\}$。
- 如果选了 u，那么 u 能贡献 a_u，但 u 的儿子都不能选。即 $f_{u,1} = a_u + \sum_{v\in \mathrm{son}_u} f_{v_0}$。

最后结果为 $\max\{f_{1,0}, f_{1,1}\}$。在 DFS 的过程中转移，复杂度 $O(n)$。

例题 1

Code

```
vector<array<int, 2>> f(n + 1);

void dfs(int u, int p) {
    f[u][1] = a[u];
    for (auto v: G[u]) {
        if (v == p) continue;
        dfs(v, u);
        f[u][0] += max(f[v][0], f[v][1]);  // 转移 1: 选 u
        f[u][1] += f[v][0];                // 转移 2: 不选 u
    }
}

int main() {
    dfs(1, 1);
    cout << max(f[1][0], f[1][1]) << endl;
}
```

树形 DP 的一般过程：

- 如果没根就随便选个根出来。
- 状态：$f_{u,s}$，表示只考虑 u 为根节点的子树的最优解，状态 s 一般与子树外面无关。

  $f_{u,0/1}$：是否选择点 u 时，T_u（u 为根的子树）的最大点独立集大小。
- 转移：将所有 $v \in \mathrm{son}_u$ 为根的子树的结果合并到 u 上。基于树的递归结构转移。

  $f_{u,0} = \sum_{v \in \mathrm{son}_u} \max\{f_{v_0}, f_{v_1}\}$：如果不选 u，那么儿子选不选都可以。

  $f_{u,1} = a_u + \sum_{v \in \mathrm{son}_u} f_{v_0}$：如果选了 u，那么 u 的儿子都不能选。
- 最后结果就是根节点的 DP 值。

例题 2

SDOI2006　保安站岗（树上最小权支配集）

给一棵树，每个点有点权 a_i。要求选出权值和尽可能小的关键点，使得所有点都被覆盖。称一个点被覆盖，若存在一个关键点距离不超过 1。

范围：$n \le 10^6$。

$f_{u,i}$ 表示 T_u 内关键点的最小权值。其中 $i \in \{-1,0,1\}$，$i=-1$ 表示 u 没被覆盖，$i=0$ 表示 u 被覆盖但不是关键点，$i=1$ 表示 u 是关键点。

- 如果选了 u，则 $f_{u,1} = a_u + \sum_{v\in \mathrm{son}_u} \min_{-1\le i\le 1} f_{v,i}$。
- 如果不选 u，分下面两种情况：
  - u 未被覆盖：$f_{u,-1} = \sum_{v\in \mathrm{son}_u} \min_{-1\le i\le 1} f_{v,i}$。
  - u 被覆盖，则 u 至少有一个儿子 w 被选中：$f_{u,0} = \min_{w\in \mathrm{son}_u} \left(f_{w,1} + \sum_{w\ne v\in \mathrm{son}_u} \min\{f_{v,0}, f_{v,1}\}\right)$。

最后答案为 $\min\{f_{1,0}, f_{1,1}\}$。复杂度 $O(n)$。

Luogu P2279 消防局的设立（加强）

给一棵树，每个点有点权 a_i。要求选出权值和尽可能小的关键点，使得所有点都被覆盖。称一个点被覆盖，若存在一个关键点距离不超过 k。

范围：$n,k \le 10^3$。

$f_{u,i}$ 表示 u 为根节点、T_u 内的点能覆盖到 u 上方第 i 层的条件下，T_u 内关键点的最小权值。其中 $-k \le i \le k$，且 $i < 0$ 时表示 u 没被覆盖。

- 如果选了 u，则 u 能覆盖到上方 k 层：$f_{u,k} = a_u + \sum_{v\in \mathrm{son}_u} \min_{-k\le i\le k} f_{v,i}$。
- 如果不选 u，分下面两种情况：
  - u 未被覆盖。假设 u 被覆盖到第 $i<0$ 层，则 u 的所有儿子都至少覆盖第 $i+1$ 层：$f_{u,i} = \sum_{v\in \mathrm{son}_u} \min_{i<j\le k} f_{v,j} \quad (i<0)$。
  - u 被覆盖。假设 u 能覆盖第 $i \ge 0$ 层，则 u 至少有一个儿子 w 能覆盖 $i+1$ 层，且其余儿子能覆盖 $-i$ 层：$f_{u,i} = \min_{w\in \mathrm{son}_u} \left(\min_{i<j\le k} f_{w,j} + \sum_{w\ne v\in \mathrm{son}_u} \min_{-i\le j\le k} f_{v,j} \right) \quad (i \ge 0)$。

最后答案为 $\min_{0\le i\le k} f_{1,i}$。

例题 3

$g_{u,i}$ 表示 u 为根节点、T_u 内的点能覆盖到 u 上方至少第 i 层的条件下，T_u 内关键点的最小权值。则 g_u 是 f_u 的后缀 min：$g_{u,i} = \min_{i \le j \le k} f_{u,j}$。

- 如果选了 u，则 u 能覆盖到上方 k 层：$f_{u,k} = a_u + \sum_{v \in \mathrm{son}_u} g_{v,-k}$。
- 如果不选 u，分下面两种情况：
  - u 未被覆盖。假设 u 被覆盖到第 $i < 0$ 层，则 u 的所有儿子都至少覆盖第 $i+1$ 层。即 $f_{u,i} = \sum_{v \in \mathrm{son}_u} g_{v,i+1} \quad (i < 0)$。
  - u 被覆盖。假设 u 能覆盖第 $i \ge 0$ 层，则 u 至少有一个儿子 w 能覆盖 $i+1$ 层，且其余儿子能覆盖 $-i$ 层。即 $f_{u,i} = \min_{w \in \mathrm{son}_u} \left(g_{w,i+1} + \sum_{w \ne v \in \mathrm{son}_u} g_{v,-i} \right) \quad (i \ge 0)$。

最后答案为 $g_{1,0}$。复杂度 $O(nk)$。

例题 4

CF 461B　Appleman and Tree

给一棵树，每个顶点被黑白染色。要求将树划分成若干连通块，使得每个连通块有且仅有一个黑点。求方案数模 10^9+7。

范围：$n \le 10^5$。

用 $f_{u,k,i}$ 表示 u 为根、只考虑前 k 个儿子的子树的划分的方案数，且 u 所在的连通块恰好有 $i \in \{0,1\}$ 个黑点（f 的第二维实际可以省略）。

- 初始状态：若 u 是黑点，则 $f_{u,0,1}=1$，否则 $f_{u,0,0}=1$。
- 状态转移：考虑第 $k+1$ 个儿子 v。

  $f_{u,k+1,1}=f_{u,k,0}f_{v,1}+f_{u,k,1}(f_{v,0}+f_{v,1})$：可以从下面 3 种情况转移过来：
  - u 内还没有黑点，但 v 内有黑点，不删 (u,v)。
  - u 内已经有黑点，且 v 内也有黑点，必须删 (u,v)。
  - u 内已经有黑点，但 v 还没黑点。为了让 v 也有黑点，不删 (u,v)。

  $f_{u,k+1,0}=f_{u,k,0}(f_{v,0}+f_{v,1})$：$u$ 内还没有黑点，若 v 内有黑点则删 (u,v)，否则不删。

最后答案为 $f_{1,1}$。

例题 5

Luogu P2015　二叉苹果树

给一棵树，每条边有边权。要求保留恰好 q 条边，使得根节点所在连通块边权和最大。

范围：$n, q \le 10^3$

令每个点的点权 w_i 为连接父亲的那条边的边权（根节点点权为 0），则题目可以转化为保留 $q+1$ 个点，使得点权和最大。

用 $f_{u,k,i}$ 表示 u 只考虑前 k 个儿子的子树在保留 i 个点时的点权之和最大值。

- 初始状态：
  - 选 u：$f_{u,0,1} = w_u$。
  - 不选 u：$f_{u,0,0} = 0$。
- 状态转移：考虑第 $k+1$ 个儿子 v。假设 u 和前 $k+1$ 个儿子一共选了 i 个点，枚举 v 选的点数 j，则有 $f_{u,k+1,i} = \max_{1 \le j \le \min\{\mathrm{siz}_v, i-1\}} f_{u,k,i-j} + f_{v,j}$。

最后结果为 $f_{1,q+1}$。

树形背包

基本形式

给定 n 个物品，每个物品有体积 v_i 和价值 w_i。物品之间构成一棵树，如果要选某个物品必须选它的父亲。有一容积为 m 的背包，要求选择一些物品放入背包，使得物品总体积不超过 m 的前提下，价值总和最大。

- 初始状态：$f_{u,0} = 0, f_{u,v_u} = w_u$，其它 $f_{u,i} = -\infty$。
- 状态转移：枚举 $v \in \mathrm{son}_u$，对于所有 $1 \le i \le \min\{\mathrm{siz}_u, m\}$ 和 $1 \le j \le \min\{\mathrm{siz}_v, i - v_u\}$，令 $f_{u,i} \leftarrow \max\{f_{u,i}, f_{u,i-j} + f_{v,j}\}$。

复杂度 $O(nm)$，不是 $O(nm^2)$。

Code

```
vector<int> siz(n + 1);  // 子树物品的价值之和
vector<vector<ll>> f(n + 1, vector<ll>(m + 1));

void dfs(int u, int p) {
    siz[u] = v[u];
    f[u][v[u]] = a[u];  // 初始状态，只选物品 u
    for (auto v: G[u]) {
        if (v == p) continue;
        dfs(v, u);
        siz[u] += siz[v];  // 此时 siz_u 为当前背包大小上界
        for (int i = min(siz[u], m); i >= 1; --i)
            for (int j = 1; j <= siz[v] && i - j >= v[u]; ++j)
                // 选出体积 j 价值 f_{v,j} 的物品，i − j ≥ v_u 是因为 u 必须选
                f[u][i] = max(f[u][i], f[u][i - j] + f[v][j]);
    }
}
```

例题

Luogu P2015　二叉苹果树

给一棵树，每条边有边权（物品的价值）。要求保留恰好 q 条边（物品体积为 1，背包容量为 q），使得根节点所在连通块（如果某个点在连通块中，它的父亲一定也在）边权和（价值总和）最大。

CTSC1997　选课

有 n 门课程，第 i 门课程的学分为 a_i（物品的价值），每门课程有零门或一门先修课，有先修课的课程需要先学完其先修课，才能学习该课程（物品的依赖关系）。一位学生要学习 m 门课程（物品体积为 1，背包容量为 m），求其能获得的最多学分（价值总和）。

CF 461B　Appleman and Tree（加强）

给一棵树，每个顶点被黑白染色。要求将树划分成若干连通块，使得每个连通块恰好有 k 个黑点。求方案数模 10^9+7。

范围：$n, k \le 10^3$。

基环树

基环树：

- n 个点 n 条边的连通图。
- 图里有且仅有一个环，环的每个点上挂一棵树。
- 每个点有且仅有一个出度（内向基环树森林）。

转化成树处理（例如删一条环边、对树和环分开处理）。

例题 6

ZJOI2008　骑士

给一个基环树森林，每个点有点权 a_i，求最大权点独立集。

范围：$n \le 10^6$

分别考虑每个连通块，最后把答案加起来。

找到一个环边 (r,s)，把 (r,s) 删除（注意可能有重边）得到一棵 r 为根的树，令 $f_{u,i,j}$ 表示 u 为根的子树的答案，其中 i 表示是否选 u，j 表示是否选 s。

- 当 $u \neq s$ 时：
  - 选 u：$f_{u,1,j} = a_u + \sum_{v \in \text{son}_u} f_{v,0,j}$。
  - 不选 u：$f_{u,0,j} = \sum_{v \in \text{son}_u} \max\{f_{v,0,j}, f_{v,1,j}\}$。
- 当 $u = s$ 时，$f_{u,0,1} = f_{u,1,0} = -\infty$，其它转移与 $u \neq s$ 相同。

最后答案为 $\max\{f_{r,0,0}, f_{r,0,1}, f_{r,1,0}\}$，即不能同时选 r 和 s。

换根 DP

一般的树形 DP 是确定一个根，然后进行 DP。如果要将每个点都作为根 DP 一遍，复杂度会变成 $O(n^2)$。

先任选一个点 u 作为根 DP。考虑将根从 u 转移到 $v \in \mathrm{son}_u$，我们发现只有 f_u 和 f_v 的值会改变：

- f_u 少了 T_v（即 v 为根的子树）的贡献；
- f_v 多了 $T_u \setminus T_v$ 的贡献。

因此可以用 DFS 通过较低的复杂度更新根节点的 DP 值。

例题 7

POI2008　STA-Station

给一棵树，求出一个根节点，使得所有点到根的距离之和最大。

范围：$n \leq 10^6$

首先考虑 1 为根的 DP。用 f_u 表示 u 为根的子树中，所有点到 u 的距离之和。

转移：$f_u = \sum_{v \in \mathrm{son}_u} (f_v + \mathrm{siz}_v)$，其中 siz_v 表示 v 为根的子树大小，有 $\mathrm{siz}_u = 1 + \sum_{v \in \mathrm{son}_u} \mathrm{siz}_v$。

换根：考虑根节点从 u 变为 $v \in \mathrm{son}_v$，则

- f_u 变为 $f'_u = f_u - (f_v + \mathrm{siz}_v)$，$\mathrm{siz}_u$ 变为 $\mathrm{siz}'_u = \mathrm{siz}_u - \mathrm{siz}_v$。
- f_v 变为 $f'_v = f_v + (f'_u + \mathrm{siz}'_u)$，$\mathrm{siz}_v$ 变为 $\mathrm{siz}'_v = \mathrm{siz}_v + \mathrm{siz}'_u$。

因此将 f_v 更新为 $f_u + \mathrm{siz}_u - 2\mathrm{siz}_v$，$\mathrm{siz}_v$ 更新为 siz_u 即可。

状压 DP

DP 的状态不止可以是数，也可以是一个集合。

状态压缩：把集合、函数、排列等状态压缩成一个整数，以实现用数组存储。

比如可以用 $[0, 2^n)$ 存一个全集为 $\{0, \ldots, n-1\}$ 的集合 S，第 i 个二进制为 1 则表示 $i \in S$。

题目特征：有一维特别小。

NOIP2016　愤怒的小鸟

平面上 n 个点，求最少几条过原点向下的抛物线，即 $y = ax^2 + bx\ (a < 0)$，可以经过所有点。

范围：$n \le 18$

用 f_S 表示覆盖 S 内所有点所需的最少抛物线个数。

- 初始状态：$f_\varnothing = 0$。
- 状态转移：
  - 抛物线至少经过两个点：$f_S \leftarrow \min\{f_S, f_{S \setminus P_{i,j}} + 1\}\ (1 \le i < j \le n)$，其中 $P_{i,j}$ 表示过 i, j 的抛物线能够经过的点集（如果无法同时经过 i, j 则 $P_{i,j} = \varnothing$）。可以 $O(n^3)$ 预处理。
  - 抛物线只经过一个点：$f_S \leftarrow \min\{f_S, f_{S \setminus \{i\}} + 1\}\ (1 \le i \le n)$。

最后答案为 $f_{\{0,\ldots,n-1\}}$。

复杂度 $O(n^2 2^n)$，约定转移顺序可以做到 $O(n2^n)$。

例题 1

Code

```
// 预处理过 i,j 的抛物线能覆盖的点集 P_{i,j}
for (int i = 0; i < n; i++)
    for (int j = i + 1; j < n; j++)
        for (int k = 0; k < n; k++)
            if (hit(i, j, k))         // 表示过 i,j 的抛物线经过 k
                p[i][j] |= 1 << k;    // P_{i,j} ← P_{i,j} ∪ {k}

for (int s = 1; s < (1 << n); s++) {
    for (int i = 0; i < n; ++i)
        for (int j = i + 1; j < n; ++j)
            f[s] = min(f[s], f[s & ~p[i][j]] + 1);  // 转移 1: 至少经过 i,j
    for (int i = 0; i < n; ++i)
        f[s] = min(f[s], f[s & ~(1 << i)] + 1);     // 转移 2: 只经过 i
}

// 最后结果为全集 f_{{0,1,...,n-1}}
cout << f[(1 << n) - 1] << endl;
```

集合运算

用一个 int 变量 s 描述集合 S，s 的第 i 个二进制为 1 表示 $i \in S$。

假设 $S \subseteq U = \{0, \ldots, n-1\}$。

- $\varnothing$： `0`
- $\{0, \ldots, n-1\}$： `(1 << n) - 1` （注意优先级！）
- $\{i\}$： `1 << i`
- $S \cap T$： `s & t`
- $S \cup T$： `s | t`
- $S \oplus T$： `s ^ t`
- $\{i+k : i \in S\}\ (k \geq 0)$： `s << k`
- $\{i-k : i \in S\}\ (k \geq 0)$： `s >> k`

集合运算

- $U \setminus S$： `((1 << n) - 1) ^ s`
- $S \setminus T$： `s & ~t`
- $i \in S$： `s >> i & 1`
- $S \subseteq T$： `(s & ~t) == 0` 或 `(s & t) == s`
- $\min S$： `s & -s` （lowbit 操作）
- $|S|$： `__builtin_popcountll(s)`

例题 2

Luogu P1171　售货员的难题（旅行商问题，TSP）

给一个有向带权图，求一个哈密顿回路，使得边权之和最小。

范围：$n \le 20$

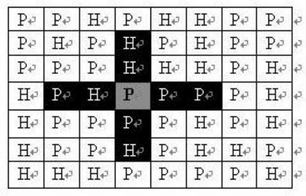
不妨设从 0 号点出发最后回到 0。用 $f_{S,i}$ 表示当前在点 i，且已经访问过的点集为 S 的最短路。

- 初始状态：$f_{\{0\},0} = 0$。
- 状态转移：枚举边 (i,j)。考虑从 i 走到 j，则 $f_{S,j} = \min_i(f_{S\setminus\{j\},i} + w(i,j))$。

最后答案为 $\min_i(f_{\{0,\ldots,n-1\},i} + w(i,0))$，因为从 i 回到 0。复杂度 $O(n^2 2^n)$。

NOI2001　炮兵阵地

给一个 $n \times m$ 的网格，有些格点上有障碍物（H）。你需要在非障碍物的格点（P）上放尽可能多的炮兵，使得炮兵之间无法相互攻击。每个炮兵能攻击到左右两格和上下两格。



P	P	H	P	H	H	P	P
P	H	P	H	P	H	P	P
P	P	P	H	H	H	P	H
H	P	H	P	P	P	P	H
H	P	P	P	P	H	P	H
H	P	P	H	P	H	H	P
H	H	H	P	P	P	P	H

范围：$n \leq 100, m \leq 10$

例题 3

用 $f_{i,S,T}$ 表示第 $1 \sim i$ 行一共放了多少炮兵。其中 S, T 分别表示第 i 行和 $i-1$ 行的炮兵位置集合，且必须满足：

- 同一行的炮兵不能相互攻击：$((S+1) \cup (S+2)) \cap S = \varnothing$（可以预处理所有合法的 S），T 同理。
- 两行间的炮兵不能相互攻击：$S \cap T = \varnothing$。
- S, T 所在的位置不能有障碍物。

初始状态：对于所有合法的 (S, T)，$f_{2,S,T} = |S| + |T|$（也可以令 $f_{0,\varnothing,\varnothing} = 0$）。

状态转移：对于所有合法的 R（第 $i+1$ 行的位置集合）和 S，枚举合法的 T。若 $T \cap (R \cup S) = \varnothing$，则 $f_{i,S,T}$ 可以转移到 $f_{i+1,R,S}$。即 $f_{i+1,R,S} = \max_T (f_{i,S,T} + |R|)$。

答案为 $\max_{S,T} f_{n,S,T}$。或者多转移两行，答案为 $f_{n+2,\varnothing,\varnothing}$。

时间复杂度（远小于）$O(n4^m)$。通过滚动数组优化，空间 $O(4^m)$。

例题 4

AtCoder DP U　Grouping

将 n 个物品分成若干组，若 i 和 j 被分到一组则获得 $a_{i,j}$ 分数（可能 < 0）。求分数总和最大值。

范围：$n \le 16$

用 f_S 表示集合 S 内的物品分组得到的最大分数总和。初始状态 $f_\varnothing = 0$。

对于 S 的非空子集 $T \subseteq S$，S 可以由 $S \setminus T$ 的最优分组加上 T 单独作为一组转移。于是可以枚举 $T \subseteq S$，则 $f_S = \max_{\varnothing \ne T \subseteq S}(f_{S \setminus T} + w_T)$，其中 w_T 表示 T 分为一组的分数，可以 $O(n^2 2^n)$ 预处理。

时间复杂度 $O(3^n)$：

$$\sum_{S \subseteq [n]} 2^{|S|} = \sum_{i=0}^{n} \binom{n}{i} 2^i = 3^n$$

Code (枚举子集)

```
for (int s = 0; s < (1 << n); ++s)
    for (int t = s; t; t = s & (t - 1))   // 枚举 S 的所有子集 T (T ≠ ∅)
        f[s] = max(f[s], f[s ^ t] + w[t]);
```

例题 5

NOIP2017　宝藏

给定一个有向带权连通图，求一棵 r 为根的生成树，最小化每条边的边权和根节点到这条边经过的顶点数（包括根节点）的乘积之和。

范围：$n \le 12, m \le 1000$

考虑一个个点加入生成树，用 $f_{S,d}$ 记录生成树的总代价，其中 S 表示树上有哪些点，d_S 表示每个点在树上的深度。考虑将点 i 连接 $j \in S$，则 S 转移到 $S \cup \{i\}$，i 的深度为 $d_j + 1$，边 (i,j) 贡献 $d_i \cdot w(i,j)$ 的代价。

上面算法的瓶颈是要存所有点的深度，但如果约定点必须按深度升序加入到树上，那么只需要记录最深层的点集。用 $f_{S,R,d}$ 记录树上最深的一层点 R 及其深度 d。转移需要枚举深度为 $d+1$ 的点集 T，将 T 中的每个点与 R 中距离最近的连接。于是有 $f_{S,R,d} + d \cdot w(T,R) \to f_{S\cup T,T,d+1}$。

事实上我们无需记录 DP 的第二维，因为 T 一定会连接到 S 中最深的点，否则不会使答案更优。因此可以 $f_{S,d} + d \cdot w(T,S) \to f_{S\cup T,d+1}$。其中 $w(T,S)$ 表示 T 的每个点连接到 S 中最近的点的代价之和，可以 $O(3^n)$ 预处理。最后答案为 $\min_{1\le d\le n} f_{V,d}$。

通过枚举 $S \cup T$ 的子集 T 转移，总时间复杂度 $O(n3^n)$。

例题 6

HDU 4336　Card Collector

有 n 张卡，每次买到第 i 张的期望为 p_i。求买到所有卡的期望次数。

范围：$n \le 20, \sum_i p_i \le 1$

期望 DP，用 f_S 表示当前拥有的卡为 S 时，期望还需要多少次能买到所有卡。

- 初始状态：如果所有卡都有了，那么不需要继续买卡，即 $f_{\{0,\dots,n-1\}} = 0$。
- 状态转移：假设买到第 i 张卡。如果 $i \in S$，那么期望还需要买 f_S 次；如果 $i \notin S$，那么期望需要买 $f_{S\cup\{i\}}$ 次。因此有 $f_S = 1 + \sum_{i\notin S} p_i \cdot f_{S\cup\{i\}} + \left(1 - \sum_{i\notin S} p_i\right) \cdot f_S$，化简得

$$f_S = \frac{1 + \sum_{i\notin S} p_i \cdot f_{S\cup\{i\}}}{\sum_{i\notin S} p_i}$$

答案为 $f_\varnothing$，时间复杂度 $O(n2^n)$。

例题 7

UVA 10032　Tug of War

n 个人拔河，每个人有体重 w_i。要求分成两组，使得两组人数差不超过 1，要求最小化体重差。

范围：$n \le 100, w_i \le 450$

令 $W = \sum_i w_i$，最小化体重差选出一组人，使得体重和尽可能接近 $W/2$。

假设没有人数差的限制，那么问题就变为 01 背包：$f_{i,s}$ 表示前 i 个人的体重之和是否可以为 s，则 $f_{i,s} = f_{i-1,s} \vee f_{i-1,s-w_i}$。

现在有人数限制，因此可以用 $f_{i,s}$ 记录使得体重之和为 s 的所有可能的人数数量（$f_{i,s}$ 是一个集合），则 $f_{i,s} = f_{i-1,s} \cup (f_{i-1,s-w_i} + 1)$。

最后找到满足 $n/2 \in f_{n,s}$，且最接近 $W/2$ 的 s 即可。

例题 8

HDU 3001 Travelling

给一个无向带权图，找一条边权总和最小的路径，使得这条路径经过每个点至少一次、且最多两次。

范围：$n \le 10$

如果要求路径经过每个点最多一次，那么问题就变为 TSP 问题：用 $f_{S,i}$ 表示当前在点 i，且已经访问过的点集为 S 的最短路。则 $f_{S,j} = \min_i(f_{S\setminus\{j\},i} + w(i,j))$。

考虑原问题，类似地可以记录每个点访问过的次数 $S : \{0,\dots,n-1\} \to \{0,1,2\}$，转移式变为 $f_{S,j} = \min_i(f_{S',i} + w(i,j))$，其中 $S'(j) = S(j) - 1$。

答案需要统计所有对任意 v 满足 $S(v) \ge 1$ 的 S。

轮廓线 DP：逐格状态转移，记录子问题的格子的边界。（太难了，只介绍一下这种思想）

HDU 1400　Mondriaan's Dream

有一个 $n \times m$ 的棋盘，要求在棋盘中放满 1×2 的骨牌，求方案数。

九省联考 2018　一双木棋

A 和 B 在一个 $n \times m$ 上的棋盘上轮流下棋，A 下黑棋，B 下白棋，一个棋子可以落下当且仅当这个棋子的左侧及上方的所有格子内都有棋子，棋盘填满时游戏结束。每个格子有 $a_{i,j}$ 和 $b_{i,j}$ 两个权值，A 的得分为黑子所在位置的 $a_{i,j}$ 之和，B 为白子所在位置的 $b_{i,j}$ 之和。双方都希望最大化自己的分数 − 对手的分数，求最优策略下 A, B 得分差。

以及各种哈密顿回路计数类问题：

- 网格图哈密顿回路 / 路径计数（Luogu P5056）；
- 网格图所有顶点被任意条 / 恰好 k 条回路覆盖的方案数；
- ……

数据结构优化 DP

利用数据结构实现快速转移：

- 前缀和；
- 树状数组；
- 线段树；
- 单调队列；
- ……

Luogu B3637　最长上升子序列

给一个长为 n 的序列 a，求最长上升子序列。

范围：$a \le 10^6$

用 f_i 表示 i 结尾的上升子序列的最长长度，则 $f_i = 1 + \max_{j<i,a_j<a_i} f_j$，复杂度 $O(n^2)$。

这里 $j < i, a_j < a_i$ 是经典的二维偏序结构，可以用树状数组维护最大值：

- 查询 f_j 最大值：在树状数组中查找 $x < a_i$ 中 T_x 的最大值。
- 更新树状数组：计算得到 f_i 后令 $T_{a_i} \leftarrow f_i$。

例题 2

NOIP2023　天天爱打卡

一共 n 天，每天可以选择是否跑步打卡，且不能连续打卡超过 k 天。初始时能量为 0，每打卡一次能量 $-d$。有 m 组目标 (x_i, y_i, v_i)，若 $[x_i - y_i + 1, x_i]$ 内均完成打卡，则能量 $+v_i$。求 n 天后的能量最大值。

范围：$n \le 10^9, m \le 10^5$

假设 $[x_i - y_i + 1, x_i]$ 的目标在第 x_i 天产生价值。

朴素 DP：用 f_i 表示第 i 天结束后的最大能量，则有两种转移：

- $f_i \leftarrow f_{i-1}$：第 i 天不打卡；
- $f_i \leftarrow f_{j-1} - (i-j)d + w(j+1, i)$ $(i-k \le j < i)$：表示第 j 天不打卡，且从 $j+1$ 打卡到 i 获得的最大能量。其中 $w(j, i)$ 表示 $[j, i]$ 打卡完成目标获得的能量.

我们只需考虑如何快速计算 $\max_{i-k \le j < i} f_{j-1} + jd + w(j+1, i)$。

例题 2

令

$$T_j = f_{j-1} + jd + w(j+1, i)$$

我们需要对每个 i 求出 T_j 在 $i-k \le j < i$ 上的最大值。

考虑如何从 $i-1$ 更新到 i。

右端点在 i，即 $x = i$ 的目标 (x, y, v)，当 $j+1 \le x-y+1$ 时会对 $w(j+1, i)$ 产生 v 的贡献。

因此需要遍历所有目标 (i, y, v)，对 $T_1, \ldots, T_{x-y}$ 区间 $+v$。区间加、区间求和操作可以通过线段树来完成。复杂度 $O(n \log n)$。

但是这题的 $n \le 10^9$。注意到我们打卡区间的两端一定都是关键点（即目标开始或结束的时间），因此可以对时间离散化，复杂度优化到 $O(m \log m)$。

例题 3

CF 713C　Sonya and Problem Wihtout a Legend

给一个长为 n 的序列 a，每次操作能选一个数，将它 $+1$ 或 -1。求最少的操作次数使得这个序列严格递增。

范围：$n \le 10^5, a_i \le 10^9$

我们考虑如何使这个序列单调不降，严格递增的情况类似。

朴素 DP：用 $f_i(x)$ 表示让前 i 位递增、且 $a_i \le x$ 所需的最少操作次数，则有

$$f_i(x) = \min_{y \le x}(|y - a_i| + f_{i-1}(y))$$

初始状态位 $f_0(x) = 0$。

注意到 $|x - a_i|$ 是下凸函数，且 $+, \min$ 运算都不会改变函数的凸性，因此 f_i, g_i 始终是凸的。

并且 f_i, g_i 只有 $O(n)$ 个拐点，因为每个 $|x - a_i|$ 只会给函数引入一个拐点，别的地方都是线性的。

Slope Trick：优先队列（或平衡树）维护拐点。

当拐点数量很少，且拐点前后斜率变化不大的时候，可以考虑通过维护极值点左侧拐点集合 L、右侧拐点集合 R、函数极值 y^* 代替维护整个函数。

这题中 $f_i(x)$ 递减，所以只需维护 L，即用 $L = \{x_0, x_1, x_2 \ldots\}$ $(x_0 \geq x_1 \geq x_2 \geq \ldots)$ 表示函数在 $(x_0, +\infty)$ 上斜率为 0，(x_1, x_0) 上斜率为 -1，(x_2, x_1) 上斜率为 -2，以此类推。

- $+|x - a_i|$：在 L 中插入两个 a_i，表示斜率改变 2。此时 $(x_0, +\infty)$ 上斜率为 1，f_i 的最小值变为 $y^* + |x_0 - a_i|$。
- $\min_{y \leq x}$：即对 f_i 求前缀 min，从 L 中删除 x_0 即可，函数最小值不变。

APIO2014　Split the sequence

给定一个长为 n 的非负整数序列 a，你需要进行以下操作 m 次：选择一个至少两个元素的块（初始时只有一块，即整个序列），然后切成两段，获得两段元素和乘积的分数。求分数最大值。

范围：$n \le 10^5, m \le 200$

答案与操作顺序无关：a_i 和 a_j 只会在所在连通块被合并时贡献 a_ia_j 的分数。

假设每段元素和为 $b_1, \ldots, b_{m+1}$，则分数为 $\sum_{1\le i<j\le m+1} b_ib_j = \left(\left(\sum_i b_i\right)^2 - \sum_i b_i^2\right)/2$。我们只需最小化 $\sum_i b_i^2$ 即可。

考虑 DP。用 $f_{i,k}$ 表示 $[1,i]$ 分为 k 段答案的最小值，则有以下转移：

$$f_{i,k} = \min_{0\le j<i} f_{j,k-1} + (s_i - s_j)^2$$

其中 s 为 a 的前缀和，即 $s_i = \sum_{j\le i} a_j$。

答案为 $f_{n,m+1}$，时间复杂度 $O(n^2m)$，显然过不了。

$$f_{i,k} = \min_{0 \le j < i} f_{j,k-1} + (s_i - s_j)^2$$

假设从 j^* 转移是最优的，那么对于其它的任意 j 有：

$$\begin{aligned} & f_{j^*} + (s_i - s_{j^*})^2 \le f_j + (s_i - s_j)^2 \\ \Rightarrow \quad & f_{j^*} + s_{j^*}^2 - 2s_i s_{j^*} \le f_j + s_j^2 - 2s_i s_j \\ \Rightarrow \quad & (f_j + s_j^2) - (f_{j^*} + s_{j^*}^2) \ge 2s_i \cdot (s_j - s_{j^*}) \end{aligned}$$

把 j 处的状态看成一个点 (x_j, y_j)，其中 $x_j = s_j$，$y_j = f_j + s_j^2$。则

$$y_j - y_{j^*} \ge 2s_i \cdot (x_j - x_{j^*})$$

即其它所有点 (x, y) 都必须落在过 (x_{j^*}, y_{j^*}) 且斜率为 $2s_i$ 的直线 $y - y_{j^*} = 2s_i \cdot (x - x_{j^*})$ 的上方。

只需要维护 $\{(x_j, y_j) : j < i\}$ 的下凸壳和最优决策点 j^* 即可均摊 $O(1)$ 转移。复杂度 $O(nm)$。

怎么维护下凸壳？

用单调栈维护下凸壳上的所有点 $\{p_1, p_2, \dots, p_t\}$，保证直线 p_ip_{i+1} 的斜率递增。当插入一个新点 $p = (x_i, y_i)$ 时：

- 如果凸包内至少有两个点，比较 $p_{t-1}p_t$ 和 p_tp 的斜率。如果 p_tp 的斜率更小那么将 p_t 从栈内弹出。
- 重复这个操作直到 $p_{t-1}p_t$ 的斜率小于 p_tp，或栈内只有一个点。
- 将点 p 入栈。

在下凸壳上维护最优转移的位置 j^*。由于目标斜率 $2s_i$ 随 i 递增，j^* 单调右移。在凸包上找到第一个使得 p_jp_{j+1} 的斜率 $> 2s_i$ 的 j 即可。

如果有些题目斜率不单调，也可以通过二分在凸包上找到最优决策点。复杂度多一个 $\log n$。

如果再毒瘤一点 x_i 也不单调则需要离线或者动态凸包。

Bonus：$n \le 10^5, m \le n-1$

- 四边形不等式优化：满足 $w(a,c)+w(b,d)\le w(a,d)+w(b,c)$ 的转移有决策单调性。
- WQS 二分（凸优化）：对于 "恰好选 k 个" 的问题，如果答案关于 k 是凸的，可以给物品一个惩罚系数。二分这个系数直到恰好选了 k 个。
- 矩阵优化：放到最后一天讲。

Thanks