线性 DP 与背包

冉雨杭

2023 年 8 月 6 日

线性 DP

定义

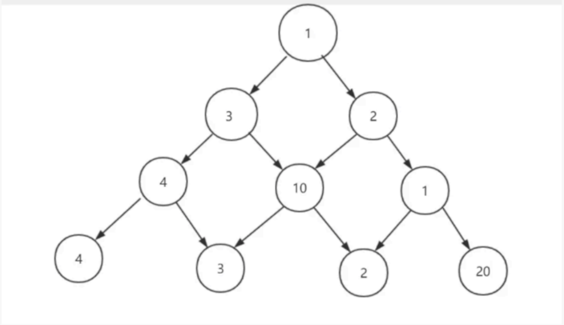
线性 DP 是动态规划问题中的一类问题，指状态之间有线性关系的动态规划问题

斐波那契数列

$f_i = f_{i-1} + f_{i-2}$

数字三角形

- 给定一个 n 行由整数组成的数字三角形，从顶部出发，每次可以移动到该点下面的两个结点之一，走到最后一层终止，价值是所有走过点权值之和，求最大路径价值
- $n \leq 500, -10000 \leq a_{i,j} \leq 10000$



数字三角形

状态

- 令 $dp_{i,j}$ 表示从起点走到第 i 行第 j 列时，最大的路径价值和是多少

转移

- $dp_{i,j} = \max(dp_{i-1,j-1}, dp_{i-1,j}) + a_{i,j}$

最长上升子序列 (简单版)

- 给定一个长度为 n 的序列，求严格单调递增的最长子序列的长度
- $n \leq 10^3, -10^9 \leq a_i \leq 10^9$

最长上升子序列 (简单版)

状态

- dp_i 表示以 i 结尾的最长上升子序列的长度

转移

- $dp_i = \max\limits_{j<i \;\&\; a_j<a_i} dp_j + 1$

复杂度

- 时间复杂度：$O(n^2)$

最长公共子序列

- 给定两个长度分别为 n 和 m 的字符串 A 和 B，求既是 A 又是 B 的子序列最长是多少
- $n, m \leq 3000$

最长公共子序列

状态

- $dp_{i,j}$ 表示第一个串前 i 个位置，第二个串前 j 个位置的最长子序列长度

转移

- $dp_{i,j} \leftarrow dp_{i-1,j}$
- $dp_{i,j} \leftarrow dp_{i,j-1}$
- $A_i == B_j : dp_{i,j} \leftarrow dp_{i-1,j-1} + 1$

拓展

- 计算不同子序列个数
- 计算本质不同子序列个数

背包问题

定义

- 背包问题：有 n 件物品，每件物品有一定的价值，获取每件物品都需要一定的代价，背包问题就是在遵守一定的规则的情况下，获取最高的价值

分类

- 01 背包
- 完全背包
- 多重背包
- 分组背包

题意

- 有 n 件物品，背包容量为 V
- 第 i 个物品体积是 v_i，价值是 w_i
- 每个物品只能使用一次
- 体积不超过 V，最多能拿多大价值的物品
- $n, V \leq 1000, v_i \leq 1000, w_i \leq 10^9$

01 背包

状态

- 令 $dp_{i,j}$ 表示在前 i 个物品中选，总重量不超过 j 时最大能选到的价值是多少

转移

- $dp_{i,j} \leftarrow \max(dp_{i-1,j}, dp_{i-1,j-v_i} + w_i)$

复杂度

- 时间复杂度：$O(nV)$
- 空间复杂度：$O(nV)$

01 背包

- 时间复杂度似乎不能再优化了，空间复杂度呢？
- 发现只需要前一次的数组，可以只用开两个大小为 $V+1$ 的数组

```
for (int i = 1; i <= n; i++) {
    int o = i & 1;
    for (int j = 0; j <= V; j++) {
        if(j + v[i] <= V) dp[o][j + v[i]] = max(dp[o][j + v[i]], dp[o ^ 1][j] + w[i]);
        dp[o][j] = max(dp[o][j], dp[o ^ 1][j]);
    }
}
```

- 能否只开一个长度为 $V+1$ 的数组呢？

01 背包

- 只用一个长度为 $V+1$ 需要非常小心枚举的顺序
- 倒着枚举即可

```
for (int i = 1; i <= n; i++) {
    for (int j = V; j >= v[i]; j--) {
        dp[j] = max(dp[j], dp[j - v[i]] + w[i]);
    }
}
```

完全背包

题意

- 有 n 件物品，背包容量为 V
- 第 i 个物品体积是 v_i，价值是 w_i
- 每个物品可以使用无限次
- 体积不超过 V，最多能拿多大价值的物品
- $n, V \leq 1000, v_i \leq 1000, w_i \leq 10^9$

完全背包

- 发现按照前面那样，转移的时候还需要枚举当前这个物品选几个，复杂度就变高了
- 想想能否通过巧妙设计枚举顺序来解决这个问题

完全背包

状态

令 dp_j 表示此时已经装了重量为 j 的物品，能获得最大的价值和是多少

```
for (int i = 1; i <= n; i++) {
    for (int j = 0; j + v[i] <= V; j++) {
        dp[j + v[i]] = max(dp[j + v[i]], dp[j] + w[i]);
    }
}
```

复杂度

- 时间复杂度 $O(nV)$
- 空间复杂度 $O(V)$

- 正着枚举为什么是正确的
- 01 背包倒着枚举，完全背包正着枚举

多重背包

题意

- 有 n 件物品，背包容量为 V
- 第 i 个物品体积是 v_i，价值是 w_i
- 每个物品有 s_i 个
- 体积不超过 V，最多能拿多大价值的物品
- $n, V \leq 1000, v_i \leq 1000, w_i \leq 10^9$

多重背包

- 再枚举选多少个物品，复杂度 $O(nV^2)$
- 复杂度太高，想办法优化

多重背包

- 再枚举选多少个物品，复杂度 $O(nV^2)$
- 复杂度太高，想办法优化

优化

- 二进制优化
- 单调队列优化

二进制优化

- 想办法把 s_i 分成 log(新物品)，满足总和等于 s_i，且对于任意 $x \leq s_i$ 都能用这 log 个物品组合出来

```
cin >> v >> w >> c;

int k = 1;
while(k <= c) {
    // 加入一个大小为k * v, 价值为k * w的物品
    c -= k;
    k *= 2;
}
if (c > 0) {
    // 加入一个大小为c * v, 价值为c * w的物品
}
```

- 时间复杂度：$O(nVlogV)$
- 如何证明所有 $x \leq s_i$ 个物品都能被这 log 个新物品组合出来

单调队列优化

- 我们先假设物品的大小都为 1
- 令 dp_j 表示放了重量 j 的物品，最大价值和是多少
- $dp_j = \max\limits_{k \le s_i} dp_{j-k} + k * w[i]$

```
for (int i = 1; i <= n; i++){
    for (int j = V; j >= 0; j--) {
        for(int k = 1; k <= min(j, s[i]); k++){
            dp[j] = max(dp[j], dp[j - k] + k * w[i]);
        }
    }
}
```

单调队列优化

- 可以发现对于某个固定的 s_i，枚举 j 后，令 $t = j - k$, 可行的 t 是一个长度为 k 的滑窗
- $k = j - t$，$dp_j \leftarrow dp_t + (j - t) * w_i$
- 转移就是找窗口中最大的 $dp_t - t * w_i$

单调队列优化

- 对于两个位置 t_1, t_2，如果有 $t_1 < t_2$ 且 $dp_{t_1} - t_1 * w_i < dp_{t_2} - t_2 * w_i$，当 t_2 在转移范围内时，t_1 永远不可能再被作为最优的 t
- 此时可以把 t_1 pop 出去，队首永远是可行的最优解
- 所有元素只会被加入队列一次，也只会被删除一次
- 复杂度是线性的

单调队列优化

- 前面假设物品的大小都是 1，如果现在是 v_i 呢
- 按照余数分类，相同余数的一起处理

```
for (int i = 1; i <= n; i++){
    for (int rem = 0; rem < v[i]; rem++) {
        head = 1;
        tail = 0;
        for (int k = 0; k <= (V - rem) / v[i]; k++) {
            int tmp = dp[k * v[i] + rem] - k * w[i];
            while(head <= tail && Q[tail] <= tmp) tail--;
            Q[++tail] = tmp;
            I[tail] = k;
            while(head <= tail && k - I[head] > s[i]) {
                head++;
            }
            dp[k * v[i] + rem] = max(dp[k * v[i] + rem], dp[I[head] * v[i] + rem] + (k
                 - I[head]) * w[i]);

        }
    }
}
```

CF 95E

- 有一个 n 个点 m 条边的图
- 幸运数是指的只由 4 和 7 构成的数字
- 问至少连多少条边能够成一个大小为幸运数的连通块
- $n, m \leq 10^5$

CF 95E

- 先把初始的连通块求出来
- 问题转换成了有一些物品，每个物品有一个体积，问至少选多少个物品它们的体积和是幸运数

CF 95E

- 看起来是一个背包问题，我们来 dp!
- 复杂度 $O(n^2)$
- 超时了，怎么优化？

CF 95E

- 发现一个性质，所有的物品体积加起来是等于 n 的
- 我们把物品分成两类，体积小于 $\sqrt{n}$ 和体积大于等于 $\sqrt{n}$
- 第一部分最多有根号个数字，第二部分最多有根号个物品
- 想想两部分分别是什么问题

CF 95E

- 第一部分是一个多重背包
- 第二部分是一个 01 背包
- 每部分的复杂度都是 $O(n\sqrt{n})$ 的
- 复杂度 $O(n\sqrt{n})$

CF 95E

- 有一个 n 个点 m 条边的图
- 幸运数是指的只由 4 和 7 构成的数字
- 问至少连多少条边能够成一个大小为幸运数的连通块
- $n, m \leq 10^5$

分组背包

题意

- 有 n 组物品，背包容量为 V
- 第 i 组物品有 s_i 个，每个物品体积是 v_{ij}，价值是 w_{ij}
- 每组只能使用一个物品
- 体积不超过 V，最多能拿多大价值的物品
- $n, V \leq 1000, v_{ij} \leq 1000, w_{ij} \leq 10^9, \sum_n \leq 1000$

分组背包

- 与 01 背包的做法非常类似
- 倒着枚举

```
for (int i = 1; i <= n; i++) {
    for (int j = V; j >= 0; j--) {
        for (int k = 0; k < s[i]; k++) {
            if(j >= v[i][k]) {
                dp[j] = max(dp[j], dp[j - v[i][k]] + w[i][k]);
            }
        }
    }
}
```

存在性 01 背包即其优化

题意

- 有 n 件物品，背包容量为 V
- 第 i 个物品体积是 v_i
- 每个物品可以使用一次
- 问对于 $0 \le x \le V$，是否存在一种方法使得恰好体积是 x
- $n \le 1000, V \le 10^6, v_i \le 10^6$

存在性 01 背包即其优化

- 将存在设置为 1，不存在设置为 0，现在有一个长度为 $V + 1$ 的 01 向量
- 每次选一个物品就是将这个向量左移了 v_i 位，最后 v_i 位都设置成 0，超出部分截断
- 不选就是这个向量不变

bitset

- 是 c++ 中一个快速做 01 位运算的类型
- 操作复杂度是：$O(len/w)$，w 一般为 32 或者 64
- 最后总复杂度为：$O(nV/w)$
- 代码非常简单

```
std::bitset<1000005> dp;
dp[0] = 1;
for (int i = 1; i <= n; i += 1) {
    dp = dp | (dp << v[i]);
}
```

计数类 01 背包

题意

- 有 n 件物品，背包容量为 V
- 第 i 个物品体积是 v_i
- 每个物品可以使用一次
- 问对于 $0 \leq x \leq V$，求有多少种方案使得最后的体积恰好为 x
- $n \leq 1000, V \leq 1000, v_i \leq 1000$

计数类 01 背包

- dp_i 表示体积为 i 的方案数是多少
- 把前面取 max 改成加法取模即可

- 如果现在询问对于每一个物品 i，排除这个物品后剩下 $n-1$ 个物品内，体积为 $j(0 \leq j \leq V)$ 的方案数有多少呢

计数类 01 背包

- 如果现在询问对于每一个物品 i，排除这个物品后剩下 $n-1$ 个物品内，体积为 $j(0 \le j \le V)$ 的方案数有多少呢
- 我会！每次重新算一次背包
- 复杂度 $O(n^2V)$
- 还有办法优化吗

计数类 01 背包

- 最后方案数和物品的顺序是无关的
- 删除谁就假设谁再最后一个
- 考虑怎么加进去的，怎么反过来删掉即可

```
void add(int v) {
    for (int i = V; i >= v; i--) {
        dp[i] = (dp[i] + dp[i - v]) % mod;
    }
}

void del(int v) {
    for (int i = v; i <= V; i--) {
        dp[i] = (dp[i] + mod - dp[i - v]) % mod
    }
}
```

P1507 NASA 的食物计划

- 给定 n 种食物，每种体积为 h_i，质量 t_i，含卡路里 k_i, 现在最多装 h 体积和 t 重量的食物，问最多能含多少卡路里
- $n \leq 50, h, t, h_i, t_i \leq 400, k_i \leq 500$

P1734 最大约数和

- 选取和不超过 S 的若干个不同的正整数，使得所有数的约数（不含它本身）之和最大
- $S \leq 1000$

P1757 通天之分组背包

- 自 01 背包问世之后，小 A 对此深感兴趣。一天，小 A 去远游，却发现他的背包不同于 01 背包，他的物品大致可分为 k 组，每个物品重量 a_i, 价值 b_i，属于第 c_i 组。每组中的物品相互冲突，现在，他想知道最大的利用价值是多少。
- $n, m \le 1000, k \le 100, a_i, b_i, c_i$ 在 int 范围内

P1586 四方定理

- 任意一个正整数 n，求有多少种方案可以将其分解为不超过四个整数的平方和
- $n \leq 32768, t \leq 100$

P1776 宝物筛选

- 多重背包模版题

习题

- P1048
- U280382
- P2979
- P1020
- P2285
- P1725
- P4933
- P2758

谢谢!