基础动态规划模型

Robert_JYH
洛谷网校

洛谷
www.luogu.com.cn

动态规划(DP)

- 对问题各状态维度进行分阶段、有顺序、无重复、决策性的遍历求解
- 三要素：状态、阶段、决策
- 三个基本条件：子问题重叠性、无后效性、最优子结构性质

动态规划(DP)

1.状态类型/结构:

序列、背包、区间、坐标、环形……

2.转移方式: 递推、记忆化搜索

3.优化:

(1)预处理、前缀和

(2)状态量: 跳过无用状态、改进状态表示、利用约束关系

(3)空间: 滚动数组

P6702 [COCI2010-2011#7] ŠEĆER

Mirko 需要向某地的一家糖果店运送 n 颗糖。

Mirko 可以使用两种类型的包装盒子：

- 一种是可以装 3 颗糖的 1 号包装；
- 另一种是可以装 5 颗糖的 2 号包装。

Mirko 想让包装盒尽可能少。例如，他要运送 18 颗糖，则可以使用 6 个 1 号包装。但是，最优策略是 3 个 2 号包装和 1 个 1 号包装。这样总共有 4 个包装。

请你帮助 Mirko 找到需要包装盒最少的方案。

- $3 \leq n \leq 5000$

P6702 [COCI2010-2011#7] ŠEĆER

- 状态：设$f[i]$表示装i颗糖所需最少包装数
- 决策：考虑本次用的是1号包装，还是2号包装。
  $$f[i] = \min(f[i], f[i-3]+1, f[i-5]+1)$$
- 答案：$f[n]$

P6702 [COCI2010-2011#7] ŠEĆER

```
memset(f,0x3f3f,sizeof f);
f[0] = 0;
for (int i = 1; i <= n; i++)
{
    if (i >= 3)
        f[i] = min(f[i], f[i - 3] + 1);
    if (i >= 5)
        f[i] = min(f[i], f[i - 5] + 1);
}
```

一、线性DP

回顾：经典模型

- 简单递推
  - Fibonacci数列
- 序列类问题
  - 单个序列：最长上升子序列
  - 多个序列：最长公共子序列
- 坐标类/多维问题
  - 数字金字塔
- 环形问题的处理

P10376 [GESP202403 六级] 游戏

- 你有四个正整数 n, a, b, c，并准备用它们玩一个简单的小游戏。
- 在一轮游戏操作中，你可以选择将 n 减去 a，或是将 n 减去 b。游戏将会进行多轮操作，直到当 $n \leq c$ 时游戏结束。
- 你想知道游戏结束时有多少种不同的游戏操作序列。两种游戏操作序列不同，当且仅当游戏操作轮数不同，或是某一轮游戏操作中，一种操作序列选择将 n 减去 a，而另一种操作序列选择将 n 减去 b。如果 $a = b$，也认为将 n 减去 a 与将 n 减去 b 是不同的操作。
- 由于答案可能很大，你只需要求出答案对 $10^9 + 7$ 取模的结果。
- $1 \leq a, b, c \leq 2 \times 10^5$

P10376 [GESP202403 六级] 游戏

- 设$dp[i]$表示从数值i出发，经过若干次减a或减b操作，使得最终数值不超过c的不同操作序列的数量。
- 如果当前的数值$i \leq c$，则不需要进行任何操作，只有一种操作序列（即不操作）。
- 即有：$dp[i] = 1(i \leq c)$
- 如果当前的数值$i > c$，则可以选择减去a或减去b。
- 即有：$dp[i] = dp[i-a] + dp[i-b](i > c)$

P10376 [GESP202403 六级] 游戏

```
cin >> n >> a >> b >> c;
for (int i = 0; i <= n; i++)
{
    if (i <= c)
        f[i] = 1;
    else
        f[i] = (f[max(i - a, 0)] + f[max(i - b, 0)]) % mod;
}
cout << f[n] << endl;
```

[ARC174A] A Multiply

- 给出 N, C。你可以选择一个区间 $[l, r]$，让 $A_l, \ldots, A_r$ 全部乘上 C。至多一次操作后，求A数组总和的最大值。
- $1 \leq N \leq 3 \times 10^5$, $|C|, |A_i| \leq 10^6$

[ARC174A] A Multiply

乘C操作对答案的影响为$(C-1)\times sum[l,r]$

考虑到 C 的取值范围可能为正或负，我们需要分情况讨论：

当 $C>0$ 时：应选择一段和最大的且大于零的区间。

当 $C\le 0$ 时：应选择一段和最小的且小于零的区间。

分类讨论后，我们只需要求一个最大（最小）子段和即可。

[ARC174A] A Multiply

复习：如何求最大子段和。

- 设$f[i]$表示到第i个数为止的最大子段和。
- 当$f[i-1] \leq 0$时，延续之前的子段不优，$f[i] = num[i]$
- 当$f[i-1] > 0$时，延续之前的子段更优，$f[i] = num[i] + f[i-1]$
- 综合有，$f[i] = num[i] + \max(f[i-1], 0)$

最小子段和同理，设$g[i]$表示到第i个数为止的最小子段和,有$g[i] = num[i] + \min(g[i-1], 0)$

分类讨论后DP即可，时间复杂度$O(n)$。

[ARC174A] A Multiply

```
    if (c > 0)
    {
        // 最大子段和
        for (int i = 1; i <= n; i++)
        {
            f[i] = num[i] + max(f[i - 1], 0LL);
            ans = max(ans, f[i]);
        }
    }
```

[ARC174A] A Multiply

```
else
{
    // 最小子段和
    for (int i = 1; i <= n; i++)
    {
        f[i] = num[i] + min(f[i - 1], 0LL);
        ans = min(ans, f[i]);
    }
}
cout << sum + (c - 1) * ans;
```

P9173 [COCI2022-2023#4] Zrinka

- 给你两个长度分别为 n 和 m 的数组，它们只由 0 和 1 组成。
- 你的任务是用偶数替换每个 0，用奇数替换每个 1。
- 替换之后，两个数组都应该是单调递增的且所有元素均大于 0，并且你最多可以使用每个正整数一次，使用的最大数字要尽可能的小。
- $n, m \leq 5000$

P9173 [COCI2022-2023#4] Zrinka

- 本题是一个双序列DP的问题。
- 设$f[i][j][0/1]$表示a数组替换前i个，b数组替换前j个，最大数字在$0/1$数组时，最小的最大数字。
- 需要考虑从$f[i-1][j][0/1], f[i][j-1][0/1]$四个状态分别如何转移过来。
- 如果从$f[i-1][j][0]$转移过来，若$a[i-1]$与$a[i]$相同，则需要使用奇偶性相同的数字，$f[i][j][0] = f[i-1][j][0] + 2$；若不同，则使用奇偶性不同的数字，$f[i][j][0] = f[i-1][j][0] + 1$。
- 如果从$f[i-1][j][1]$转移过来，则考虑$a[i]$与$b[j]$是否相同，相同则转移时需要额外+1。
- 从$f[i][j-1][0/1]$时转移时同理。

P9173 [COCI2022-2023#4] Zrinka

```
if (i > 0)
{
    f[i][j][0] = min(f[i][j][0],
        f[i - 1][j][0] + 1 + (bool)(a[i] == a[i - 1]));
    f[i][j][0] = min(f[i][j][0],
        f[i - 1][j][1] + 1 + (bool)(a[i] == b[j]));
}
if (j > 0)
{
    f[i][j][1] = min(f[i][j][1],
        f[i][j - 1][1] + 1 + (bool)(b[j] == b[j - 1]));
    f[i][j][1] = min(f[i][j][1],
        f[i][j - 1][0] + 1 + (bool)(b[j] == a[i]));
}
```

P10262 [GESP样题 六级] 亲朋数

- 给定一串长度为 L、由数字 0~9组成的数字串 S。容易知道，它的连续子串儿共有 $L(L+1)/2$个。如果某个子串对应的数（允许有前导零）是 p 的倍数，则称该子串为数字串 S 对于 p 的亲朋数。
- 例如，数字串 S 为“ 12342 ”、p 为 2，则在 15 个连续子串中，亲朋数有“ 12 ”、“ 1234 ”、“ 12342 ”、“ 2 ”、“ 234 ”、“ 2342 ”、“ 34 ”、“ 342 ”、“ 4 ”、“ 42 ”、“ 2 ”等共 11个。注意其中“ 2 ”出现了 2 次，但由于其在 S 中的位置不同，记为不同的亲朋数。
- 现在，告诉你数字串 S 和正整数 p，你能计算出有多少个亲朋数吗？
- $1 \leq p \leq 128, 1 \leq L \leq 10^6$

P10262 [GESP样题 六级] 亲朋数

- 设$f[i][j]$表示所有以位置i结尾的子串模p的余数为j的方案数。
- 答案即为所有的$f[i][0]$相加。

- 考虑每个$f[i-1][j]$可以转移到哪里，设当前位置数字为$a[i]$，则$f[i][(j \times 10 + a[i]) \bmod p] \mathrel{+}= f[i-1][j]$。

P10262 [GESP样题 六级] 亲朋数

```
for (int i = 0; i < n; i++)
{
    num = s[i] - '0';
    for (int j = 0; j < p; j++)
        pre[j] = f[j], f[j] = 0;
    for (int j = 0; j < p; j++)
        f[(j * 10 + num) % p] += pre[j];
    f[num % p]++;
    ans += f[0];
}
```

P8816 [CSP-J 2022] 上升点列

- 在一个二维平面内，给定 n 个整数点，此外你还可以自由添加 k个整数点。
- 你在自由添加 k 个点后，还需要从 $n+k$个点中选出若干个整数点并组成一个序列，使得序列中任意相邻两点间的欧几里得距离恰好为 1 而且横坐标、纵坐标值均单调不减，即 $x_{i+1}-x_i=1, y_{i+1}=y_i$或 $y_{i+1}-y_i=1, x_{i+1}=x_i$。
- 请给出满足条件的序列的最大长度。

- $1 \le n \le 500$，$0 \le k \le 100$。对于所有给定的整点，其横纵坐标 $1 \le x_i, y_i \le 10^9$，且保证所有给定的点互不重合。对于自由添加的整点，其横纵坐标不受限制。

P8816 [CSP-J 2022] 上升点列

- 我们先把点以x为第一关键字，y为第二关键字排序。
- 设$f[i][u]$表示考虑到点i，添加了u个点时的最大长度。
- 考虑上一次从j点转移过来，他们之间需要添加$dist = a[i].x - a[j].x + a[i].y - a[j].y - 1$个点才能满足转移条件，则有：
$$f[i][u] = \max(f[i][u], f[j][u - dist] + dist + 1)$$
- 时间复杂度$O(n^2k)$

P8816 [CSP-J 2022] 上升点列

```
for (int i = 2; i <= n; i++)
    {
        for (int j = i - 1; j >= 1; j--)
        {
            if (a[j].y > a[i].y)
                continue;
            int dist = a[i].x - a[j].x + a[i].y - a[j].y - 1;
            for (int u = dist; u <= k; u++)
                f[i][u] = max(f[i][u], f[j][u - dist] + dist + 1);
        }
    }
```

二、坐标类/多维DP

P8356 「WHOI-1」数列计数

这种数列满足下面这一条神奇的性质:

- $a_0 = 0$。
- $\forall i \in [1, n]$ 均有 $a_i = a_{i-1} + x$ 或者 $a_i = a_{i-1} + y$。
- $\forall i \in [1, n], p \nmid a_i$（$a_i$ 不是p的倍数）。

求这样的数列的数量。

两个数列不同，当且仅当他们有一个下标存储的元素不同。

$n \leq 10^4, x, y, p \leq 10^9$

P8356 「WHOI-1」数列计数

- 当x与y相等时，可以直接生成数列判断是否合法。

```
ans = 1;
for (int i = 1; i <= n; i++)
{
    if (i * x % p == 0)
    {
        ans = 0;
        break;
    }
}
```

P8356 「WHOI-1」数列计数

- 当x与y不相等时，设$f[i][j]$表示执行了i次$+x$，j次$+y$的方案数。
- 此时$a[i+j]=ix+jy$
- 若$a[i+j]$是p的倍数，则$f[i][j]=0$。

```
f[0][0] = 1;
for (int i = 0; i <= n; i++)
{
    for (int j = 0; i + j <= n; j++)
    {
        if (i == 0 && j == 0)
            continue;
        if ((i * x + j * y) % p == 0)
        {
            f[i & 1][j] = 0;
        }
```

P8356 「WHOI-1」数列计数

- 当x与y不相等时，设$f[i][j]$表示执行了i次$+x$，j次$+y$的方案数。
- 此时$a[i+j]=ix+jy$
- 若$a[i+j]$不是p的倍数，则类似数字三角形的转移，考虑本次$+x$还是$+y$即可。

```
else
{
    if (i == 0)
        f[i & 1][j] = f[i & 1][j - 1];
    if (j == 0)
        f[i & 1][j] = f[i & 1 ^ 1][j];
    if (i > 0 && j > 0)
        f[i & 1][j] = (f[i & 1 ^ 1][j] + f[i & 1][j - 1]) % mod;
}
if (i + j == n)
{
    ans = (ans + f[i & 1][j]) % mod;
}
```

P7160 「dWoi R1」Sixth Monokuma's Son

- 题目首先定义矩阵环为，给定一个矩阵 A，初始全为白色，在其中选定一个子矩阵 A1标黑，再在 A1内选定一个子矩阵 A2标白，就会形成一个矩阵环。注意，矩阵环至少上下左右都有被选定的部分，且整个矩阵环不是一个长方形的矩阵。
- 假设 + 为黑，- 为白，下面这个就是矩阵环：

```
---++++++--
---++--+--
---++++++--
---++++++--
----------
```

- 因此，矩阵环会出现上，下，左，右四条边，每个方向有多少个涂黑的部分，就是那个方向的厚度。比如对于第一张符合要求的图，上方，右方的厚度为 1，左方，下方的厚度为 2。
- **注意，一个完整的矩阵不是一个矩阵环。**

P7160 「dWoi R1」Sixth Monokuma's Son

- 机望的机械眼能扫到一片 $n \times m$ 的区域，第 i 行第 j 列发现了 $a_{i,j}$ 的不对劲值。
- 因为机望被外部力量折磨的厉害，所以机望只能锁定一个矩阵环进行查看。机望想求助于你，他想让你锁定一个矩阵环，使得这个矩阵环中的所有位置的不对劲值的和最大，**上方，下方的厚度为 1 且上方的那一行在整个区域的第一行，下方的那一行在整个区域的最后一行**。至于左右的厚度，机望不限制更多要求。
- $n \leq 10, 1 \leq m \leq 10^5, |a_{i,j}| \leq 100$

P7160 「dWoi R1」Sixth Monokuma's Son

- 我们可以把矩阵环分为左中右三部分：

```
++++++
++ --+
++ --+
++++++
```

- 左右两部分是完全由+组成的矩形，中间部分为首尾行是+，中间是-的矩形。
- 设$f[i][0/1/2]$表示考虑到第i列，矩阵前$1/2/3$个部分时的最大值。
- 预处理 $sum[i]$ 表示当前列之和， $cur[i]$ 表示当前列首尾元素之和。
- 则有：

P7160 「dWoi R1」Sixth Monokuma's Son

```
for (int i = 1; i <= m; i++)
{
    f[i][0] = max(f[i - 1][0], 0) + sum[i]; // 左
    if (i != 1)
    {
        if (i != m)
            f[i][1] = max(f[i - 1][1], f[i - 1][0]) + cur[i]; // 中
        f[i][2] = max(f[i - 1][2], f[i - 1][1]) + sum[i];     // 右
        ans = max(ans, f[i][2]);
    }
}
```

三、区间DP

所求答案与一段区间有关，常用$f[l,r]$表示状态，长度/左端点/决策点3层循环

区间DP

- 区间动态规划问题的基本思想是，对于每个区间，他们的最优值都是由几段更小区间的最优值得到，于是可以分而治之。
- 状态：一般设定$f[i][j]$表示$[i,j]$这个区间的最优值。
- 决策与转移：大区间是由小区间推导出来的，于是需要枚举分界点转移，并附加代价。

P2858 [USACO06FEB] Treats for the Cows G/S

- 约翰购置了 N（$N \leq 2000$）份美味的零食来卖给奶牛们。每天约翰售出一份零食。
- 零食按照 $1, \ldots, N$ 编号，它们被排成一列放在一个很长的盒子里。盒子的两端都有开口，约翰每天可以从盒子的任一端取出最外面的一个。
- 第i份零食的初始价值为 V_i（$1 \leq V_i \leq 1000$）。
- 第 i 份零食如果在被买进后的第 a 天出售，则它的售价是 $V_i \times a$。
- 希望你能帮他计算一下，在这些零食全被卖出后，他最多能得到多少钱。

P2858 [USACO06FEB] Treats for the Cows G/S

- 设$f[l][r]$表示还剩$[l,r]$段没有卖出的最多钱数。
- 考虑枚举这次从左边还是从右边卖出，即有
- $f[i][j] = \max(f[i-1][j] + a[i-1] * k, f[i][j+1] + a[j+1] * k)$
- 时间复杂度$O(n^2)$。

P2858 [USACO06FEB] Treats for the Cows G/S

```
for (int i = 1; i <= n; i++)
{
    for (int j = n; j >= i; j--)
    {
        int k = n - (j - i + 1);
        f[i][j] = max(f[i - 1][j] + a[i - 1] * k,
                      f[i][j + 1] + a[j + 1] * k);
    }
}
//枚举最后卖哪个
for (int i = 1; i <= n; i++)
{
    ans = max(ans, f[i][i] + a[i] * n);
}
```

P9325 [CCC 2023 S2] Symmetric Mountains

Rebecca 有一张落基山脉的照片，上面排列着 $N(1 \le N \le 5000)$ 座山，从左向右的第 i 座山的高度是 $h_i(1 \le h_i \le 10^5)$ 。

Rebecca 截图保留照片的一个连续段，这张截图的不对称性定义为：处于截图上对称位置的山的高度差的绝对值之和（截图最左和最右的山的高度差，左数第二和右数第二的山的高度差，诸如此类的和）。

Rebecca 想要知道对于长度为 i 的截图，拥有的最小不对称性。请你对于 $1 \le i \le N$，输出这个值。

样例：

输入：1 3 5 6

输出：0 1 3 7

P9325 [CCC 2023 S2] Symmetric Mountains

设$f[l][r]$表示截图保留照片的$[l,r]$段时的不对称性。

这个值可以很方便地由$[l+1,r-1]$时的不对称性转移过来，因为此时头尾的l,r两座山恰好是计算不对称性时对应的两座山，只需由$[l+1,r-1]$段的答案加上l,r两座山高度差的绝对值即可。

即有：

$$f[l][r] = f[l+1][r-1] + abs(a[l] - a[r])$$

注意转移时需要先枚举长度，再枚举其中一个端点，以保证所有转移需要的值已被计算过。

时间复杂度$O(n^2)$。

P9325 [CCC 2023 S2] Symmetric Mountains

```
for (int len = 1; len <= n; len++) // 枚举长度
{
    int ans = inf;
    for (int l = 1; l + len - 1 <= n; l++) // 枚举左端点
    {
        int r = l + len - 1;
        f[l][r] = f[l + 1][r - 1] + abs(a[l] - a[r]);
        ans = min(ans, f[l][r]);
    }
    printf("%d ", ans);
}
```

四、背包问题

01背包

- 有 N 件物品和一个容量为 V 的背包。每件物品只有一件。放入第 i 件物品耗费的费用是 C_i，得到的价值是 W_i。求解将哪些物品装入背包可使价值总和最大。

- $F[i,v]=\max\{F[i-1,v],F[i-1,v-C_i]+W_i\}$
- 空间优化：滚动数组，倒序循环$F[v]=\max\{F[v],F[v-C_i]+W_i\}$
- 初始化：

  (1)恰好装满$F[]=\{0,-\infty,\dots\}$

  (2)价值最大$F[]=\{0\}$

P10987 [蓝桥杯 2023 国 Python A] 火车运输

- 钢厂有一辆用于运送废旧钢材的火车，它具有两节车厢，其中车厢 1 的最大载重量为 A，车厢 2 的最大载重量为 B。现在一共有 N件废旧钢材需要被运输，其中第 i 件钢材的重量为 w_i，为了最大化运输效率，车长想要一次性运输走重量尽可能多的钢材，请你帮助车长计算出一次运输最多可以带走多大重量的钢材。
- 每件钢材都是独立的不可分割的，只能被放置在某一节车厢中。在装载钢材时只需要考虑重量条件即可。
- $1 \leq N \leq 200, 1 \leq w_i, A, B \leq 1000$。

P10987 [蓝桥杯 2023 国 Python A] 火车运输

- 本题是01背包的直接应用，不过需要考虑两个背包。
- 设$f[j][k]$表示第一个背包容量为 j 并且第二个背包容量为 k 时，当前能装下的物品重量的最大值。费用和价值均为第i件物品的重量w_i。
- 分别考虑当前钢材分给哪个背包，即有：
- $f[j][k] = \max(f[j][k], f[j - w[i]][k] + w[i], f[j][k - w[i]] + w[i])$
- 时间复杂度$O(NAB)$。

P10987 [蓝桥杯 2023 国 Python A] 火车运输

```
for (int i = 1; i <= n; i++)
    for (int j = a; j >= 0; j--)
        for (int k = b; k >= 0; k--)
        {
            if (j >= w[i])
                f[j][k] = max(f[j][k], f[j - w[i]][k] + w[i]);
            if (k >= w[i])
                f[j][k] = max(f[j][k], f[j][k - w[i]] + w[i]);
            ans = max(ans, f[j][k]);
        }
```

完全背包与多重背包

- 完全背包：有 N 件物品和一个容量为 V 的背包。每件物品有无数件。放入第 i 件物品耗费的费用是 C_i，得到的价值是 W_i。求解将哪些物品装入背包可使价值总和最大。
- 正序循环

$$F[v] = \max\{F[v], F[v - C_i] + W_i\}$$

- 多重背包：有 N 件物品和一个容量为 V 的背包。每件物品有 M_i 件。放入第 i 件物品耗费的费用是 C_i，得到的价值是 W_i。求解将哪些物品装入背包可使价值总和最大。
- 当成01背包即可，可使用二进制拆分。单调队列优化将在提高组阶段学习。

- 混合背包：每件物品有一/多/无限件。为了时间效率，需要判断物品种类后将三种物品分开写（特别是完全+多重）。

分组背包

- 分组背包：物品被划分为 K 组，每组中的物品互相冲突，最多选一件。求解将哪些物品装入背包可使价值总和最大。
- 循环改为组号、倒序重量、组内物品（注意循环顺序）。

其它设问

- 求方案数：

1.求方案总数：初值$f[]=\{1,0,0,\ldots\}$，将max/min 改为求和

2.求最优方案数：两个数组，一个记录最大价值，一个记录最大价值的最优方案数

3.方案的存在性：$f[]$表示对应方案是否存在

- 输出方案：

1.最优方案：记录下每个状态的最优值是由状态转移方程的哪一项推出来的，换句话说，记录下它是由哪一个策略推出来的。便可根据这条策略找到上一个状态，从上一个状态接着向前推即可。

2.第k优方案：对于每个状态维护一个大小为k的解的队列，每次转移将两个队列合并后的前k优解记录。

[ABC286D] Money in Hand

- 有 n 种纸币，其中对于第 i 种纸币，它的面值是 a_i元，我们有 b_i张这种纸币。
- 请求出在不找零的情况下，用这些纸币能否正好付 x 元，如果能则输出 Yes，不能则输出 No。
- 保证 $1 \leq n \leq 50, 1 \leq x \leq 10^4, 1 \leq ai \leq 100, 1 \leq bi \leq 50$。

- 样例：
- 输入：

  2 19

  2 3

  5 6
- 输出： Yes

[ABC286D] Money in Hand

- 本题是多重背包的可行性问题。
- 设$f[i]$表示是否可以凑成金额i。
- 本题数据范围较小，因此可以把每张纸币都当成一个物品，对于第i种纸币，我们有b_i张这种纸币，那么执行b_i次01背包即可。
- 由于只需要知道凑成金额i是否可行，dp方程修改为：

$$f[i] = f[i] | f[i - a_i]$$

- 时间复杂度$O(n \ x \max b_i)$

[ABC286D] Money in Hand

```
while (n --)
{
    scanf ("%d%d", &a, &b);
    while (b --) // 拆成bi件物品
    {
        for (int i = m; i >= a; i --) // 倒序循环
        {
            f[i] |= f[i - a];
        }
    }
}
```

[ABC321F] #(subset sum = K) with Add and Erase

- 维护一个箱子，有 Q 个操作，分为如下类型：
- $+x$，表示向箱子中扔一个写有数字 x 的球。
- $-x$，表示从箱子中拿出一个写有数字 x 的球（保证存在这样的球）。
- 每一次操作后，计算：有多少种从箱子里拿球的方法，使得拿的球上写有的数字的总和为 K（并不需要真正拿出球）？
- $Q, x, K \leq 5000$

[ABC321F] #(subset sum = K) with Add and Erase

- 设$f[i]$表示使得拿的球上写有的数字的总和为 i的方案数。
- 如果只有+操作，是01背包统计方案数的模板题，边读边做01背包即可。逆序循环$f[i] = f[i] + f[i-x]$
- 考虑-操作，我们需要消去之前某次01背包的影响，我们需要执行+操作对于循环的逆操作，即正序循环$f[i] = f[i] - f[i-x]$，从而将影响消去。
- 时间复杂度$O(QK)$。

[ABC321F] #(subset sum = K) with Add and Erase

```
if (op == '+')
{
    // 逆序循环执行01背包
    for (int i = k; i >= x; i--)
        f[i] = (f[i] + f[i - x]) % mod;
}
else if (op == '-')
{
    // 正序循环消去影响
    for (int i = x; i <= k; i++)
        f[i] = (f[i] - f[i - x] + mod) % mod;
}
```

P8687 [蓝桥杯 2019 省 A] 糖果

- 糖果店的老板一共有 M 种口味的糖果出售。为了方便描述，我们将 M 种口味编号 $1 \sim M$。
- 小明希望能品尝到所有口味的糖果。遗憾的是老板并不单独出售糖果，而是 K 颗一包整包出售。
- 幸好糖果包装上注明了其中的 K 颗糖果的口味，所以小明可以在买之前就知道每包内的糖果口味。
- 给定 N 包糖果，请你计算小明最少买几包，就可以品尝到所有口味的糖果。
- $1 \leq N \leq 100$，$1 \leq M \leq 20$，$1 \leq K \leq 20$。

P8687 [蓝桥杯 2019 省 A] 糖果

- 由于 M 的最大值为 20，我们可以使用二进制数来表示某一包的糖果口味集合及小明希望品尝的糖果口味集合。具体地，每一种口味对应二进制位中的一位，集合中的某个口味存在则对应的位为 1，否则为 0。
- 设 $dp[S]$ 表示达到口味集合 S 所需的最少糖果包数。初始化时，$dp[0] = 0$，其他状态初始化为一个较大的值（如 INF）。
- 遍历每一包糖果，对每一个当前的状态 S，尝试购买这包糖果，得到新的状态 $S' = S \mid$ 包内糖果的二进制表示。更新 $dp[S'] = \min(dp[S'], dp[S] + 1)$。

- 本题用到了我们将在提高组阶段学习的状态压缩DP的基础思想。

P8687 [蓝桥杯 2019 省 A] 糖果

```
memset(dp, 0x3f3f, sizeof(dp));
dp[0] = 0;
for (int i = 0; i < N; i++) // 枚举每包糖果
    for (int S = 0; S < 1 << M; S++) { // 枚举所有状态
        if (dp[S] > MAXN)continue;
        dp[S | candy[i]] = min(dp[S | candy[i]], dp[S] + 1);
    }
```