GESP C++ 六级认证（二）

搜索算法专题 - 代码实战

深度优先搜索 · 广度优先搜索 · 二叉树搜索

相关题目

- P10376 [GESP202403 六级] 游戏
- P10377 [GESP202403 六级] 好斗的牛

P10376 [GESP202403 六级] 游戏

题意

游戏从正整数 n 开始，每轮可以选择将 n 减去 a 或减去 b，游戏在 $n \leq c$ 时结束。求不同的游戏操作序列数量，即使 $a = b$，减去 a 和减去 b 也被视为不同操作。答案对 $10^9 + 7$ 取模。

数据范围：$1 \leq a, b, c \leq n \leq 2 \times 10^5$。

分析

这是一个典型的动态规划问题。我们需要计算从 n 出发，通过不断减去 a 或 b 到达 $\leq c$ 状态的所有不同路径数量。

状态定义：
设 $dp[i]$ 表示从 n 开始到达数字 i 的不同操作序列数量。

状态转移：
对于当前数字 i，我们可以选择减去 a 或减去 b：

- 如果 $i - a \leq c$，说明这一步会直接结束游戏，所有路径都汇聚到状态 c
- 如果 $i - a > c$，则转移到状态 $i - a$
- 同理处理减去 b 的情况

状态转移方程为：

$$dp[i-a] = dp[i-a] + dp[i] \quad (\text{当 } i-a > c)$$

$$dp[c] = dp[c] + dp[i] \quad (\text{当 } i-a \le c)$$

流程：

1. 初始化 $dp[n] = 1$（起点）
2. 从 n 到 $c+1$ 倒序遍历
3. 对于每个 i，分别处理减去 a 和减去 b 的情况
4. 最终 $dp[c]$ 即为所有结束状态的方案数之和

时间复杂度：$O(n)$，每个状态最多被访问一次。

参考代码

```cpp
const int N = 2e5 + 10;
const int mod = 1e9 + 7;
int dp[N], n, a, b, c;

int main() {
    cin >> n >> a >> b >> c;
    // 初始化：从n开始的方案数为1
    dp[n] = 1;
    // 从n到c+1倒序遍历
    for (int i = n; i > c; i--) {
        // 处理减去a的情况
        if (i - a <= c)
            dp[c] = (dp[c] + dp[i]) % mod;  // 直接结束游戏
        else
            dp[i - a] = (dp[i - a] + dp[i]) % mod;  // 转移到新状态
        // 处理减去b的情况
        if (i - b <= c)
            dp[c] = (dp[c] + dp[i]) % mod;  // 直接结束游戏
        else
            dp[i - b] = (dp[i - b] + dp[i]) % mod;  // 转移到新状态
    }
    cout << dp[c] << endl;
    return 0;
}
```

记忆化搜索版本：

```
const int N = 2e5 + 10, mod = 1e9 + 7;
int mem[N], n, a, b, c;

// 记忆化搜索：计算从x开始的方案数
int dfs(int x) {
    if (x <= c)  // 终止条件
        return 1;
    if (mem[x] != -1)  // 记忆化
        return mem[x];
    // 递归计算两种选择的方案数之和
    return mem[x] = (dfs(x - a) + dfs(x - b)) % mod;
}

int main() {
    memset(mem, -1, sizeof mem);  // 初始化记忆化数组
    cin >> n >> a >> b >> c;
    cout << dfs(n) << endl;
    return 0;
}
```

P10377 [GESP202403 六级] 好斗的牛

题意

有 10^9 个牛棚排成一排，需要安置 n 头牛。第 i 头牛的攻击范围是 (a_i, b_i)，意味着如果它左边 a_i 个牛棚或右边 b_i 个牛棚内有其他牛，它就会挑事。要求选择一段连续的牛棚，使得能够安置所有牛且没有牛会挑事，求这段连续牛棚的最短长度。

数据范围：$1 \le n \le 9$，$1 \le a_i, b_i \le 10^3$。

分析

1. **问题转化**：由于 n 很小，可以考虑枚举所有牛的排列顺序。对于每种排列，计算安置这些牛所需的最小连续牛棚长度。关键在于确定牛之间的最小间隔。

2. **关键观察**：当按顺序放置牛时，相邻两头牛之间需要满足攻击范围的要求。具体地，如果前一头牛（编号为 $last$）放在位置 x，当前牛（编号为 i）放在位置 y（$y > x$），则为了确保两头牛都不挑事，间隔必须至少为 $\max(b_{last}, a_i)$。这是因为：

   - 前一头牛 $last$ 的右攻击范围为 b_{last}，所以 $y - x - 1 \geq b_{last}$（即中间空牛棚数至少为 b_{last}）。
   - 当前牛 i 的左攻击范围为 a_i，所以 $y - x - 1 \geq a_i$。
     因此，最小间隔为 $\max(b_{last}, a_i)$，从而 $y = x + \max(b_{last}, a_i) + 1$。

3. **算法步骤**：
   - 使用 DFS 枚举所有 $n!$ 种排列顺序。
   - 对于每种排列，从第一头牛开始放置（位置为 1），然后依次计算后续牛的位置：如果前一头牛在位置 x，则下一头牛在位置 $x + \max(b_{last}, a_i) + 1$。
   - 最后一头牛的位置即为所需连续牛棚的长度（因为牛棚从位置 1 开始连续）。
   - 在所有排列中取最小长度作为答案。
4. **正确性证明**：通过枚举所有排列，确保考虑到所有可能的安置顺序。递推公式保证了相邻牛之间的间隔满足攻击范围要求，且由于攻击范围的对称性，非相邻牛之间也会自动满足条件（因为间隔更大）。

时间复杂度：$O(n! \cdot n)$，在 $n \leq 9$ 时可行。

参考代码

```
const int N = 15;
const long long INF = 0x3f3f3f3f3f3f3f3f;
long long ans;
int a[N], b[N], n;
bool vis[N];

// DFS枚举所有排列：p表示当前已放置牛的数量，last表示上一头牛的编号，cost表示当前最后一头牛的位置
void dfs(int p, int last, long long cost) {
    if (p > n) { // 所有牛都已放置
        ans = min(ans, cost); // 更新最小长度
        return;
    }
    for (int i = 1; i <= n; i++)
        if (!vis[i]) { // 如果牛i未被放置
            vis[i] = true;
            if (p == 1)
                // 放置第一头牛，位置为1
                dfs(p + 1, i, 1);
            else
                // 放置后续牛：新位置 = 原位置 + 最大攻击范围 + 1
                dfs(p + 1, i, cost + max(b[last], a[i]) + 1);
            vis[i] = false; // 回溯
        }
}
```

```cpp
int main() {
    cin >> n;
    for (int i = 1; i <= n; i++) cin >> a[i];
    for (int i = 1; i <= n; i++) cin >> b[i];
    ans = INF;
    dfs(1, 0, 1);
    // 开始DFS，初始状态：放置第1头牛，上一头牛编号为0（无效），当前位置为1
    cout << ans << endl;
    return 0;
}
```

GESP C++ 六级 - 搜索算法

代码实战