【TTR-201】「TT OI」 Round I

注：为了防止抄袭题解的事件发生，本场比赛的题解仅提供思路&核心代码，不提供完整源代码！！！

T4 小吃街

其实吧……这题我也不太会……

所以以下是 ~~AI生成~~ 的题解：

问题分析

我们需要在总花费不超过 q 元的前提下，尽可能多地购买小吃街中的菜。每道菜有各自的价格，但必须在购买菜之前支付所在店的茶位费 s_i（题目描述中明确指出：在店里买小吃需要先付茶位费）。

关键点：

- 如果要在某家店点菜，必须先支付该店的茶位费。
- 每家店有若干道菜，每道菜有独立价格。
- 总花费 = 所选店铺的茶位费之和 + 所选菜的价格之和。
- 目标是最大化购买的菜的数量，而不是最小化花费。

思路框架

由于 $n \leq 30$，每家店的菜数 $m_i \leq 30$，且每道菜价格 $t_{i,j} \leq 50$，总预算 $q \leq 30$，可以考虑用动态规划（背包问题的变形）求解。

核心难点

如果简单地按照“花费-最大菜数”去背包，状态定义会比较复杂，因为茶位费只有在选了该店的至少一道菜时才需要支付。

更清晰的思路是：

按店铺划分阶段，在每个阶段决定是否在这家店点菜，如果点菜，则必须先支付茶位费，然后在该店的菜中选择若干道（可以不选，但不选的话就不需要付茶位费）。

状态设计

定义 $dp[k][c]$ 表示考虑前 k 家店，花费恰好 c 元时，能买到的最大菜数。

对于第 i 家店（有 s_i 茶位费，以及 m_i 道菜，每道菜价格已知）：

1. **不在第 i 家店点任何菜**：直接继承上一阶段的状态。
2. **在第 i 家店点若干道菜**：先支付茶位费 s_i，然后在该店的菜品集合中做一次“01背包”来选择菜（目标：在花费一定的情况下，最大化菜的数量）。

转移方式

- 对于每家店，先考虑不点菜的情况，即 $dp[i][c] = dp[i-1][c]$。
- 再考虑点菜的情况：在支付茶位费后，用该店的菜品进行内部选择，更新花费与菜数的对应关系，合并到总状态中。

实际操作时，可以用“分组背包”的思维，将每家店看作一个组，组内物品包括：

- 不点菜（花费0，菜数0）
- 点若干道菜的各个方案（花费 = 茶位费 + 所选菜总价，菜数 = 所选菜的数量）

具体步骤

1. 初始化 $dp[0][0] = 0$，其余为负无穷（表示不可达）。
2. 遍历每家店 $i = 1 \to n$：
   - 先复制上一阶段的状态。
   - 计算在该店点菜的情况：
     - 对该店的菜品进行 01 背包，得到 $f[x]$ 表示在该店点菜总价为 x 时能买到的最大菜数（注意这里的 x 只是菜的价格，不含茶位费）。
     - 对于每个可能的 x（从 0 到该店菜品总价上限），如果 $f[x]$ 有意义（即能选出若干道菜总价为 x），则对于总花费 c（来自上一阶段），若上一阶段花费 c' 可达，则新的花费 $c' + s_i + x$ 可达，菜数增加 $f[x]$。
3. 最终在所有 $dp[n][c]$（$c \le q$）中找出最大的菜数即为答案。

边界情况

- 当 $n = 0$ 时，没有店铺，输出 0。
- 某家店茶位费过高或所有菜价格过高，导致无法点任何菜，则跳过该店点菜的情况。
- 注意“点0道菜”在内部01背包中对应花费0、菜数0，这是合理的（这样就不需要付茶位费，相当于不在该店消费）。

复杂度分析

- 店铺数 $n \le 30$。
- 总预算 $q \le 30$。
- 每家店菜品数 $m_i \le 30$，菜价 ≤ 50，因此在该店内部01背包的花费上限不超过 $30 \times 50 = 1500$（但实际由于 q 很小，超过 q 的部分可以剪枝）。
- 总体复杂度大致为 $O(n \cdot q \cdot \sum m_i)$，在题目限制下完全可行。

这样，我们就可以在给定预算下，通过动态规划求出最多能买到的菜的数量。