P16229 [蓝桥杯 2026 省 A] 外卖配送
题解

1 题意概述

将 N 个订单按固定顺序分成若干批次依次配送。站点有 M 种交通工具，第 i 种的路途耗时为 A_i，箱体拥挤系数为 B_i。若某批次包含 L 个订单且选用第 i 种交通工具，该批次耗时为：

$$L \cdot A_i + \frac{L(L-1)}{2} \cdot B_i$$

此外，从第二个批次起，每开启一个新批次需额外花费固定的 X 秒折返。求送完所有订单的最小总耗时。

2 第一层思考：预处理单批次最优耗时

先不考虑批次之间的分配问题，聚焦一个子问题：如果某批次恰好要送 L 个订单，最少要花多少时间？

交通工具的选择仅取决于 L，与其他批次无关。因此，对于每个 L，我们遍历所有 M 种交通工具，取最小耗时即可。定义：

$$C_L = \min_{1 \le i \le M} \left(L \cdot A_i + \frac{L(L-1)}{2} \cdot B_i \right)$$

C_L 的含义是：单独配送一个包含 L 个订单的批次所需的最小耗时（不含折返）。这一步的时间复杂度为 $O(NM)$。

3 第二层思考：发现完全背包结构

现在的问题变成了：将 N 个订单分成若干批，第一批不需要折返，后续每批额外花费 X 秒。怎样分批使得总耗时最小？

3.1 统一折返代价的技巧

为了让每一批的代价形式一致，我们使用一个常见技巧：**假设每一批都需要折返**（每批的代价统一为 $C_L + X$），最终答案再减去多算的第一批的那一次 X。

这样，每一批的代价就只取决于该批的订单数 L，是一个” 物品” 的概念了。

3.2 转化为完全背包

经过上述统一，问题可以精确地描述为：

> 有 N 种” 物品”，第 L 种物品的” 重量” 为 L，” 代价” 为 $C_L + X$。每种物品可以使用任意多次。需要恰好装满容量为 N 的背包，求最小总代价。

这正是一个**完全背包问题**。定义状态 f_w 为恰好配送前 w 个订单的最小总耗时（含统一折返），转移方程为：

$$f_w = \min_{1 \le L \le w} (f_{w-L} + C_L + X)$$

初始条件 $f_0 = 0$，最终答案为 $f_N - X$。

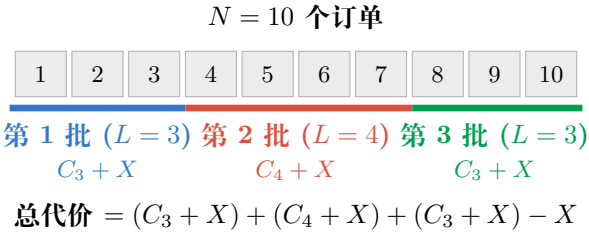

Figure 1: 将 10 个订单分成 3 批的示意图。每批统一计入折返代价 X，最后减去首批多算的一次 X。批次大小 L 对应完全背包中的” 物品重量”。

3.3 完全背包的代码范式

标准的完全背包写法是：外层枚举物品种类 L，内层从 L 到 N 正序枚举容量 w。正序枚举保证了同一种物品可以被重复选取：

```
for (int L = 1; L <= N; ++L)
  for (int w = L; w <= N; ++w)
    dpN[w] = min(dpN[w], dp[L] + dpN[w - L] + X);
```

这和 0-1 背包的唯一区别就在于内层循环是**正序**而非逆序。

4 关于完全背包时间复杂度的讨论

读者可能会注意到：这道题的 DP 部分是一个双重循环，时间复杂度为 $O(N^2)$。这和” 完全背包的时间复杂度是 $O(nW)$”（n 为物品种类数，W 为背包容量）是一致的吗?

答案是**完全一致**的。

在本题中，” 物品” 是批次大小 $L \in \{1, 2, \ldots, N\}$，共有 $n = N$ 种物品；” 背包容量”$W = N$（需要恰好配送 N 个订单）。所以完全背包的时间复杂度 $O(nW) = O(N \times N) = O(N^2)$，恰好与代码中双重循环的复杂度吻合。

更一般地，完全背包的时间复杂度**确实**是” 物品种类数 × 背包容量”。我们来直观理解为什么：

- 外层循环枚举的是” 用哪种物品来更新”，共 n 轮。
- 内层循环枚举的是” 更新哪个容量的状态”，共 W 步。
- 每一步的转移是 $O(1)$ 的（直接比较取 min）。

因此总时间为 $O(nW)$。这与 0-1 背包的复杂度形式相同——区别仅在于内层循环的枚举方向（正序 vs 逆序），不影响循环次数。

如图 2 所示，完全背包的 DP 表可以看作一个 $n \times W$ 的网格，每个格子恰好被访问一次。

5 完整算法流程

1. **预处理**：对每个批次大小 $L \in [1, N]$，遍历 M 种交通工具，计算 C_L。时间 $O(NM)$。
2. **完全背包 DP**：以批次大小为物品、订单总数为容量，求最小代价。时间 $O(N^2)$。
3. **输出**：$f_N - X$。

6 代码实现

```
#include <bits/stdc++.h>
using namespace std;
using ll = long long;

static constexpr ll INF = (1ll << 61) - 1;
```

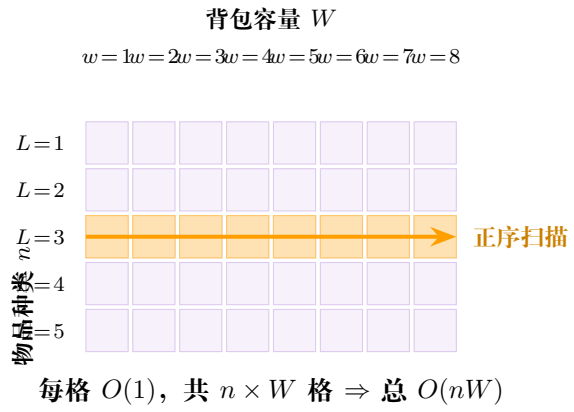

Figure 2: 完全背包的 DP 表示意图。外层遍历 n 种物品（行），内层正序遍历容量 W（列），每格恰好访问一次，总复杂度为 $O(nW)$。

```cpp
struct Solve {
  ll N, M, X;
  vector<array<ll, 2> > abls;
  vector<ll> dp;   // dp[L] = 单批送 L 个订单的最小耗时
  vector<ll> dpN;  // dpN[w] = 送前 w 个订单的最小总耗时（含统一折返）

  Solve() {
    cin >> N >> M >> X;
    abls.resize(M);
    for (int i = 0; i < M; ++i) {
      cin >> abls[i][0] >> abls[i][1];
    }
    // 预处理单批次最优耗时
    dp.resize(N + 2, INF);
    for (int L = 1; L <= N; ++L) {
      for (int i = 0; i < M; ++i) {
        auto [a, b] = abls[i];
        dp[L] = min(dp[L], a * L + ((L * (L - 1)) / 2) * b);
      }
    }
    // 完全背包：外层枚举物品（批次大小），内层正序枚举容量（订单数）
    dpN.resize(N + 2, INF);
    dpN[0] = 0;
    for (int L = 1; L <= N; ++L) {
      for (int w = L; w <= N; ++w) {
        dpN[w] = min(dpN[w], dp[L] + dpN[w - L] + X);
      }
    }
    cout << dpN[N] - X << "\n";
  }
};

signed main() {
  ios::sync_with_stdio(false);
  cin.tie(nullptr);
  Solve solve;
  return 0;
}
```

7 复杂度分析

- **时间复杂度**：预处理 C_L 需 $O(NM)$，完全背包 DP 需 $O(N^2)$。总计 $O(NM + N^2)$。在 $N, M \le 5000$ 的范围下，运算量约 5×10^7，C++ 下可在百毫秒内完成。

- **空间复杂度**：$O(N+M)$，用于存储 C_L 数组、DP 数组和交通工具属性。