题目

D 钢管切割2

时间限制：1000ms 内存限制：65536kb

通过率：170/182 (93.41%) 正确率：170/285 (59.65%)

题目描述

给定一段长度为 n 英寸的钢管和一个价格表，该价格表表示长度为 i（$i = 1, 2, \ldots, n$）英寸的钢管的价格为 p_i。

你是一个慈善家，想要以最低的价格卖出钢管，你想知道相对于最大的总销售价格，卖出切割后的钢管最多能便宜多少价格。

输入

第一行一个正整数 n（$1 \leq n \leq 10^4$），表示钢管的总长度。

第二行 n 个正整数 $p_1, p_2, \ldots, p_n$（$1 \leq p_i \leq 10^7$），表示钢管的价格。

输出

一行一个正整数，表示最多便宜的价格。

思路

定义状态和子问题：

在这个问题中，每个状态可以表示为 (i, j)，其中 i 表示流水线编号（0、1、2 分别代表三条流水线），j 表示装配站编号（0 到 m-1）。

子问题是确定每个装配站 (i, j) 上所需的最小时间，而这个时间又依赖于前一个装配站的最小时间以及来自其他流水线的转移时间。

初始化：

首先，初始化第一个装配站 (0, 0)、(1, 0)、(2, 0) 的最小时间，分别为对应的装配站时间。

递归计算最小时间：

接下来，通过循环遍历装配站和流水线，逐步计算并填充动态规划表 dp。

对于每个 (i, j)，我们考虑两个方面：

更新当前流水线 (i, j) 上的最小时间：这是通过将前一个装配站 (i, j-1) 的最小时间与当前装配站 (i, j) 的装配时间相加得到的。dp[i][j] = dp[i][j-1] + p[i][j]。

更新来自其他流水线的最小时间：这是通过遍历其他流水线 (k)，其中 k != i，并考虑装配站 (i, j) 从流水线 (k) 移动过来的情况。这需要减去当前装配站 (i, j) 的装配时间，并加上流水线 (k) 到流水线 (i) 的移动时间，以及装配站 (k, j) 的最小时间。dp[i][j] = min(dp[i][j], dp[k][j] + t[k][i] - p[k][j] + p[i][j])。

这两个步骤合并到了一个嵌套循环中，遍历装配站 j 并在流水线 i 内进行更新。

找到最小时间：

在计算了每个装配站 (i, j) 的最小时间后，我们可以找到最小总时间。最小总时间可能出现在三条流水线中的任何一条。因此，我们遍历 i，找到 dp[i][m-1] 中的最小值，这表示在每条流水线的最后一个装配站上的最小总时间。

代码实现

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

const long long INF = 1e18;

int main() {
    long long T;
    cin >> T;
    while (T--) {
        long long m;
        cin >> m;

        // 三条流水线的装配站时间
        vector<vector<long long>> p(3, vector<long long>(m));
        for (long long i = 0; i < 3; ++i) {
            for (long long j = 0; j < m; ++j) {
                cin >> p[i][j];
            }
        }

        // 三条流水线的移动时间
        long long t[3][3];
        for (long long i = 0; i < 3; ++i) {
            for (long long j = 0; j < 3; ++j) {
                cin >> t[i][j];
            }
        }

        // dp[i][j] 表示在第 i 条流水线的第 j 个装配站时所需的最少时间
        vector<vector<long long>> dp(3, vector<long long>(m, INF));

        // 初始化第一个流水线的第一个装配站
        dp[0][0] = p[0][0];
        dp[1][0] = p[1][0];
        dp[2][0] = p[2][0];

        // 计算 dp 数组
        for (long long j = 0; j < m; ++j) {
            for (long long i = 0; i < 3; ++i) {
```

```
                if (j + 1 < m) {
                    dp[i][j + 1] = min(dp[i][j + 1], dp[i][j] + p[i][j + 1]);
                }
                if(j>0){
                for (long long k = 0; k < 3; ++k) {
                    if (k != i) {
                        dp[k][j] = min(dp[k][j], dp[i][j] + t[i][k] - p[i][j]
+p[k][j]);
                    }
                }
                }
            }
        }

        // 找到最小时间
        long long min_time = INF;
        for (long long i = 0; i < 3; ++i) {
            min_time = min(min_time, dp[i][m - 1]);
        }

        cout << min_time << endl;
    }

    return 0;
}
```

要注意的细节

设置最大值，要在合法范围外