# 动态规划算法作业

**姓名：** 叶子宁 **学号：** 1120231313

## 整数线性规划问题

### 题意

考虑下面的整数线性规划问题. 即给定序列 $a$，$b$ 和 $c$，其中 $a_i$ 和 $b$ 是整数，$n$ 是正整数.
求:

$$\max\left\{\sum_{i=1}^{n} c_i x_i \mid \sum_{i=1}^{n} a_i x_i \leq b, x_i \geq 0, x_i \in Z\right\}$$

### 解法

其实就是一个背包问题，$b$ 为背包容量，$a_i$ 为物品体积，$c_i$ 为物品价值，$x_i$ 为物品数量. 我们只需要对每一个物体做一次完全背包即可.

#### 时间复杂度

$dp$ 状态数为 $O(b)$
对于每一个物体，我们需要 $O(b)$ 的时间复杂度，故总时间复杂度为 $O(nb)$

### 伪代码

```
int integer_linear_programming(int n, int a[], int b[], int c[])
{
    int *dp = (int *)malloc(sizeof(int) * (b + 1));
    for (int i = 1; i ⩽ n; ++i)
        for (int j = a[i]; j ⩽ b; ++j)
            dp[j] = max(dp[j], dp[j - a[i]] + c[i]);
    return dp[b];
}
```

## 数字三角形问题

### 题意

#### 问题描述

给定一个有 $n$ 行数字组成的数字三角形，如下图所示。试设计一个算法，计算出从三角形的顶至底的一条路径，使该路径经过的数字和最大。

#### 算法设计

对于给定的 $n$ 行数字组成的三角形，计算从三角形顶至底的路径经过的数字和的最大值。

#### 数据输入

由文件 `input.txt` 提供输入数据。文件的第 1 行是数字三角形的行数 $n$，$1 \leq n \leq 100$。接下来 $n$ 行是数字三角形各行中的数字。所有数字在 $0 \sim 99$ 之间。

### 结果输出

将计算结果输出到文件 **output.txt**，文件第 1 行中的数是计算出的最大值。

## 解法

利用动态规划解决该问题，每一个点只可能从上一行的两个点转移过来，故有：

$$dp[i][j] = \max\{dp[i-1][j], dp[i-1][j-1]\} + a[i][j]$$

### 时间复杂度

涉及到二维状态 $O(n^2)$，每个状态转移的时间复杂度为 $O(1)$，故总时间复杂度为 $O(n^2)$

## 伪代码

```
#define N 105
int n, a[N][N], dp[N][N];
void read_data()
{
    file *fp = fopen("input.txt", "r");
    for (int i = 1; i ⩽ n; ++i)
        for (int j = 1; j ⩽ i; ++j)
            fscanf(fp, "%d", &a[i][j]);
    fclose(fp);
}
void write_data(int ans)
{
    file *fp = fopen("output.txt", "w");
    fprintf(fp, "%d\n", ans);
    fclose(fp);
}
int numberTriangle()
{
    for (int i = 1; i ⩽ n; ++i)
    {
        dp[i][1] = dp[i - 1][1] + a[i][1];
        for (int j = 2; j < i; ++j)
            dp[i][j] = max(dp[i - 1][j], dp[i - 1][j - 1]) + a[i][j];
        dp[i][i] = dp[i - 1][i - 1] + a[i][i];
    }
    int ans = 0;
    for (int i = 1; i ⩽ n; ++i)
        ans = max(ans, dp[n][i]);
    return ans;
}
int main()
{
    read_data();
    int ans = numberTriangle();
    write_data(ans);
    return 0;
}
```

# 游艇出租问题

## 题意

### 问题描述

长江游艇俱乐部在长江上设置了 $n$ 个游艇出租站 $1, 2, ..., n$。
游客可在这些游艇出租站租用游艇，并在下游的任何一个游艇出租站归还游艇。
游艇出租站 $i$ 到出租站 $j$ 之间的租金为 $r(i, j)$，$1 < i < j < n$。
试设计一个算法，计算出从游艇出租站 1 到游艇出租站 $n$ 所需的最少租金，并分析算法的计算复杂性。

### 算法设计

对于给定的游艇出租站 $i$ 到游艇出租站 $j$ 的租金 $r(i, j)$，$1 = i < j = n$，计算出从游艇出租站 1 到 $n$ 所需的最少租金。

### 数据输入

由文件 **input.txt** 提供输入数据。文件的第 1 行有一个正整数 $n$，$n < 200$，表示有 $n$ 个游艇出租站。接下来 $n - 1$ 行是 $r(i, j)$，$1 < i < j < n$。

### 结果输出

将计算出的游艇出租站 1 到 $n$ 最少租金输出到文件 **output.txt**。

## 解法

对于每个站可能从在它之前的任意一个站到达
顺序遍历，有无后效性
设$dp[i]$表示从 1 到 i 的最小租金
故有:

$$dp[1] = 0$$

$$\forall i > 1 \forall j < i, dp[i] = \min\{dp[j] + r(j, i)\}$$

### 时间复杂度

状态数为 $O(n)$，每个状态转移的时间复杂度为 $O(n)$，故总时间复杂度为 $O(n^2)$

## 伪代码

```
#define N 205
int n, r[N][N], dp[N];
void read_data()
{
    file *fp = fopen("input.txt", "r");
    for (int i = 2; i ⩽ n; ++i)
        for (int j = 1; j < i; ++j)
            fscanf(fp, "%d", &r[j][i]);
    fclose(fp);
}
void write_data(int ans)
{
    file *fp = fopen("output.txt", "w");
    fprintf(fp, "%d\n", ans);
    fclose(fp);
}
```

```
int minCost()
{
    memset(dp, 0x3f, sizeof(dp));
    dp[1] = 0;
    for (int i = 2; i ≤ n; ++i)
        for (int j = 1; j < i; ++j)
            dp[i] = min(dp[i], dp[j] + r[j][i]);
    return dp[n];
}
int main()
{
    read_data();
    int ans = minCost();
    write_data(ans);
    return 0;
}
```