

01 背包问题 ,完全背包问题：运行代码输入数据（绿色）的形式请如下所示（回车换行）。

输入物品数量4

输入背包最大承重8

输入一组物品的重量和价值：2 3

输入一组物品的重量和价值：3 4

输入一组物品的重量和价值：4 5

输入一组物品的重量和价值：5 6

最大价值： 10

背包中物体：

2

4

The screenshot displays two windows. The left window is a web browser showing a document titled '01背包问题' (01 Knapsack Problem). It contains a table for the '标记函数取值' (Marking function value) and a section for '最优解的追踪——标记函数法' (Optimal solution tracking - Marking function method). The right window is a Python IDE showing the implementation of the 01 Knapsack problem. The code defines a 'package' function and a 'show' function. The input data is entered in the console, and the output shows the maximum value and the items in the backpack.

01背包问题

最优解的追踪——标记函数法

标记函数取值：

rec[i,j]	C=0	1	2	3	4	5	6	7	8
1	0	0	1	1	1	1	1	1	1
2	0	0	0	0	1	1	1	1	1
3	0	0	0	0	0	1	0	1	1
4	0	0	0	0	0	0	0	0	1

从末尾开始分析：

$rec[4,8]=1 \Rightarrow$ 物品4在背包中

$rec[4-1, 8-w_4]=rec[3, 3]=0 \Rightarrow$ 物品3不在背包中

$rec[3-1, 3]=rec[2, 3]=1 \Rightarrow$ 物品2在背包中

$rec[2-1, 3-w_2]=rec[1, 3-3]=0 \Rightarrow$ 物品1不在背包中

故，背包中含有物品2和4。

01背包问题

最优解的追踪——直接由备忘录推导

备忘录：

DP[i][j]	C=0	1	2	3	4	5	6	7	8
1	0	0	1	1	1	1	1	1	1
2	0	0	0	0	1	1	1	1	1
3	0	0	0	0	0	1	0	1	1
4	0	0	0	0	0	0	0	0	1

```
def package(n, c, w, v):  
    # n = 物品数 c = 背包最大承重 w = 物品重量 v = 物品价值  
    value = [[0 for j in range(c+1)] for i in range(n+1)]  
    for i in range(1, n+1):  
        for j in range(1, c+1):  
            value[i][j] = value[i-1][j]  
            if j >= w[i-1] and value[i][j] < value[i-1][j-w[i-1]] + v[i-1]:  
                value[i][j] = value[i-1][j-w[i-1]] + v[i-1]  
    return value  
  
def show(n, c, w, value):  
    print("最大值:", value[n][c])  
    x = [False for i in range(n)]  
    j = c  
    for i in range(n, 0, -1):  
        if value[i][j] > value[i-1][j]:  
            x[i-1] = True  
            j -= w[i-1]  
    print('背包中物体: ')  
    for i in range(n):  
        if x[i]:  
            print(i+1, end=' ')  
            package() = for i in range(1, n+1) : for j in range(1, c+1) : if j >= w[i-1] and value[i][j] < value[i-1][j-w[i-1]] + v[i-1]: value[i][j] = value[i-1][j-w[i-1]] + v[i-1]
```

Run: 01package.py

D:\WVPY\algo\venv\Scripts\python.exe D:\WVPY\algo\01package.py

输入物品数量4

输入背包最大承重8

输入一组物品的重量和价值: 2 3

输入一组物品的重量和价值: 3 4

输入一组物品的重量和价值: 4 5

输入一组物品的重量和价值: 5 6

最大价值: 10

背包中物体:

2 4

Process finished with `exit` code 0

The image shows a screenshot of an IDE (likely PyCharm) with a Python script named `complete_knapsack.py` open. The script implements a dynamic programming solution for a knapsack problem. The code is as follows:

```
31
32 if __name__ == '__main__':
33     n = input('输入物品数量')
34     c = input('输入背包最大承重')
35     w = []
36     v = []
37     for i in range(0, int(n)):
38         a, b = input("输入一组物品的重量和价值: ").split()
39         w.append(int(a))
40         v.append(int(b))
41     value = package(int(n), int(c), w, v)
42     print("最大价值: ", value[int(n)][int(c)])
43     #show(int(n), int(c), w, value)
44
45
```

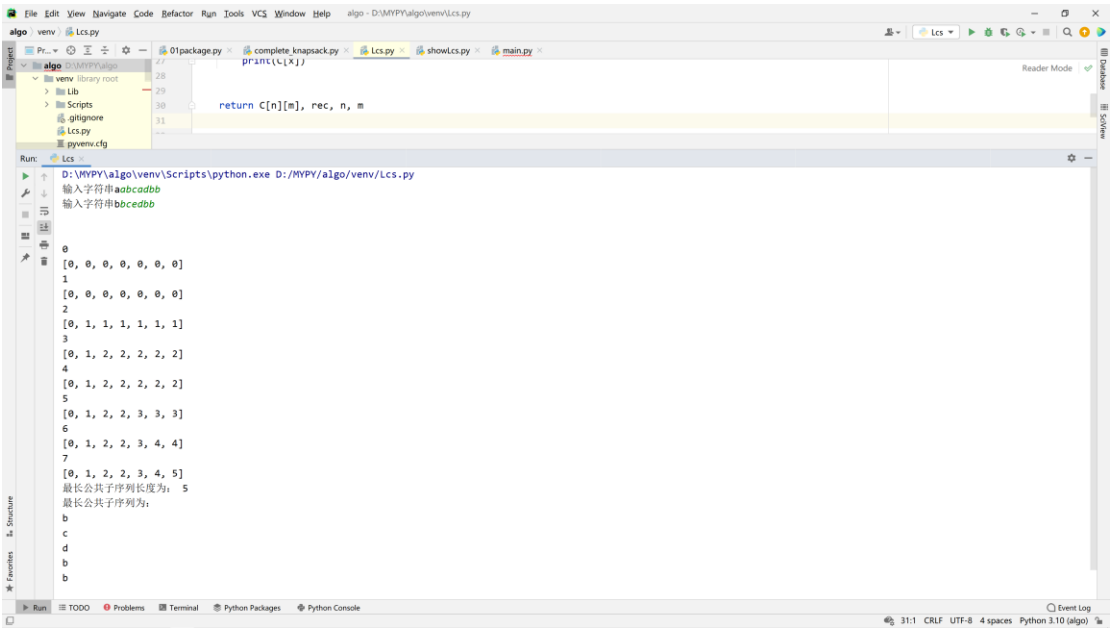
The script is executed, and the output is shown in the Run window:

```
Run: complete_knapsack x
D:\MYPY\algo\venv\Scripts\python.exe D:/MYPY/algo/complete_knapsack.py
输入物品数量5
输入背包最大承重16
输入一组物品的重量和价值: 5 12
输入一组物品的重量和价值: 4 3
输入一组物品的重量和价值: 7 10
输入一组物品的重量和价值: 2 3
输入一组物品的重量和价值: 6 6
最大价值: 36

Process finished with exit code 0
```

The IDE interface includes a Project view on the left showing the file structure, a Run window at the bottom, and a status bar at the very bottom indicating the current file encoding (UTF-8) and line endings (CRLF).

公共子序列 (Lcs.py)， 形式如下：



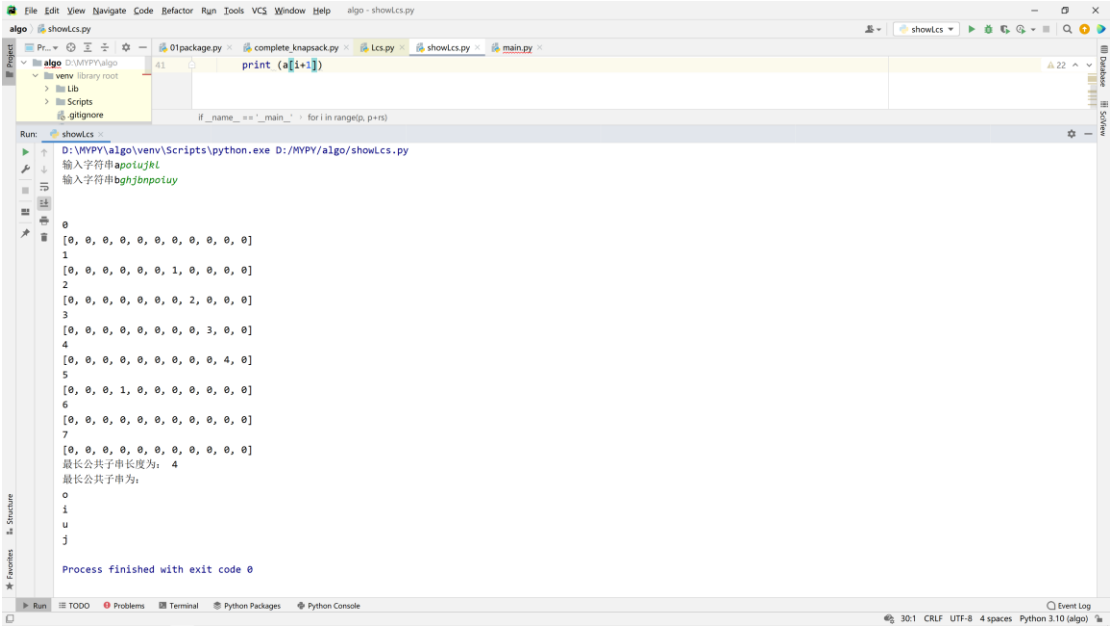
```
File Edit View Navigate Code Refactor Run Tools VCS Window Help algo - D:\MYPY\algo\venv\Lcs.py
Project Pr... 01package.py complete_knapsack.py Lcs.py showLcs.py main.py
D:\MYPY\algo\venv library root
Lib
Scripts
gitignore
Lcs.py
pyvenv.cfg

Run: Lcs
D:\MYPY\algo\venv\Scripts\python.exe D:/MYPY/algo/venv/Lcs.py
输入字符串aabcadb
输入字符串bbcedbb

0
[0, 0, 0, 0, 0, 0, 0]
1
[0, 0, 0, 0, 0, 0, 0]
2
[0, 1, 1, 1, 1, 1, 1]
3
[0, 1, 2, 2, 2, 2, 2]
4
[0, 1, 2, 2, 2, 2, 2]
5
[0, 1, 2, 3, 3, 3, 3]
6
[0, 1, 2, 3, 4, 4, 4]
7
[0, 1, 2, 3, 4, 5]
最长公共子序列长度为: 5
最长公共子序列为:
b
c
b

Run | Run | Run | Problems | Terminal | Python Packages | Python Console | Event Log
30:1 CRLF UTF-8 4 spaces Python 3.10 (algo)
```

公共子串 (showLcs)， 形式如下：



```
File Edit View Navigate Code Refactor Run Tools VCS Window Help algo - showLcs.py
Project Pr... 01package.py complete_knapsack.py Lcs.py showLcs.py main.py
D:\MYPY\algo\venv library root
Lib
Scripts
gitignore

Run: showLcs
D:\MYPY\algo\venv\Scripts\python.exe D:/MYPY/algo/showLcs.py
输入字符串apoiu
输入字符串bghjbnpoiuy

0
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
1
[0, 0, 0, 0, 0, 0, 1, 0, 0, 0]
2
[0, 0, 0, 0, 0, 0, 2, 0, 0, 0]
3
[0, 0, 0, 0, 0, 0, 3, 0, 0, 0]
4
[0, 0, 0, 0, 0, 0, 4, 0, 0, 0]
5
[0, 0, 0, 1, 0, 0, 0, 0, 0, 0]
6
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
7
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
最长公共子串长度为: 4
最长公共子串为:
o
i

Process finished with exit code 0

Run | Run | Run | Problems | Terminal | Python Packages | Python Console | Event Log
30:1 CRLF UTF-8 4 spaces Python 3.10 (algo)
```

综上，请按形式输入，第一次使用 python 写算法，有很多语法也是刚在网上学习的。笨拙之处见谅。