**暨南大学本科实验报告专用纸**

课程名称 算法分析与设计实验 成绩评定

实验项目名称 数字三角形问题 指导教师 李展

实验项目编号 实验· 实验项目类型 综合性 实验地点

学生姓名 张印祺 学号 2018051948

学院 信息科学技术 系 计算机科学 专业 网络工程

实验时间 2020 年 4月 15日 下午～ 4 月 15 日下温度 ℃湿度

1. 问题描述

给定由n个数字组成的数字三角形，试设计一种算法计算从三角形的顶至底的一条路径，使得该算法经过的数字总和最大。数字三角形形状如图所示：

3

2 14

10 10 5

16 6 8 14

1. 算法思路

使用动态规划，用一个列表记录每一个点的路径最大和,具体思路如下：

1.如果这个结点在三角形顶部，则直接记录。

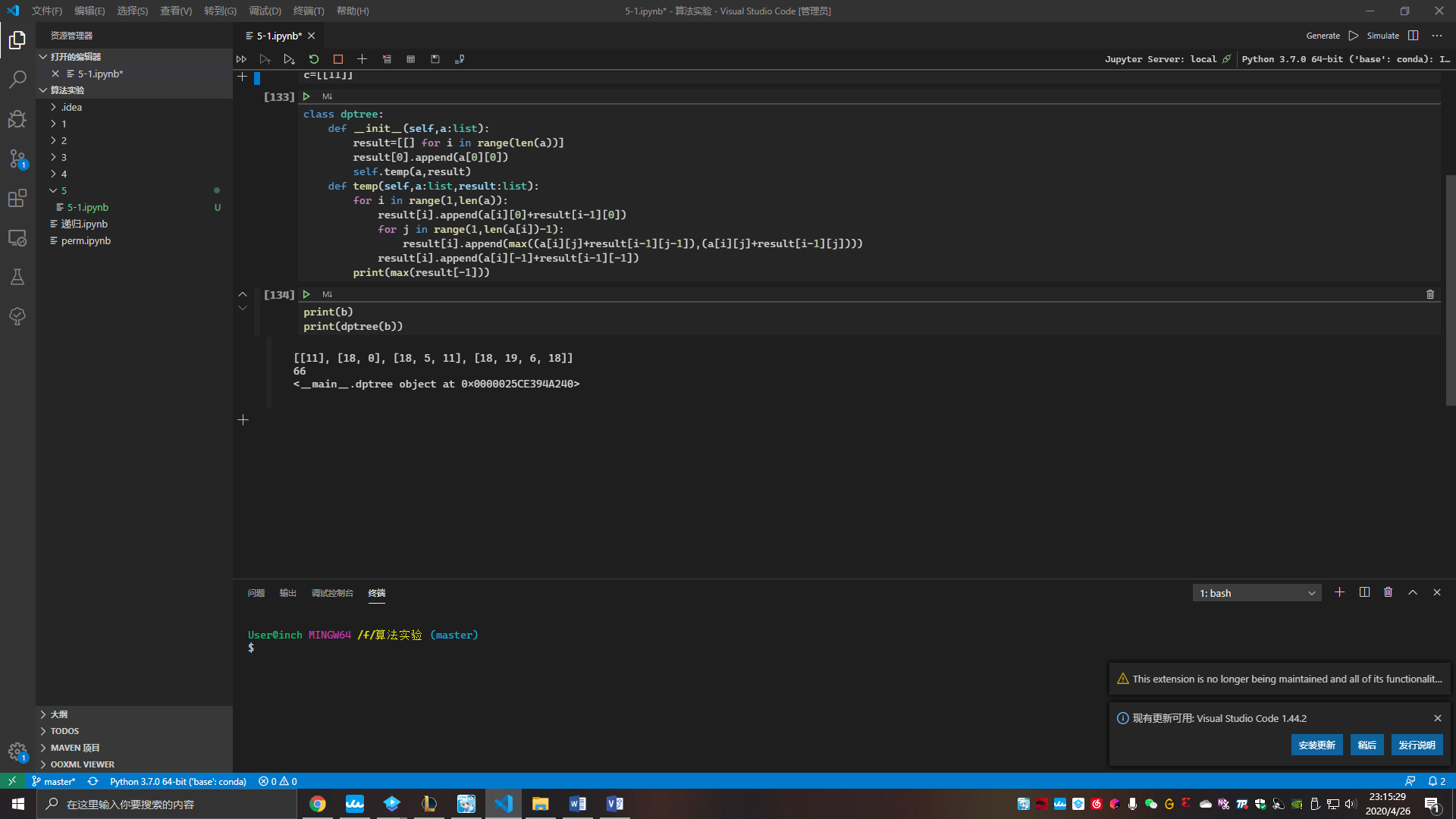
2.如果结点不在顶部，在temp列表对应位置记录其与上方相邻两数之和的较大值。

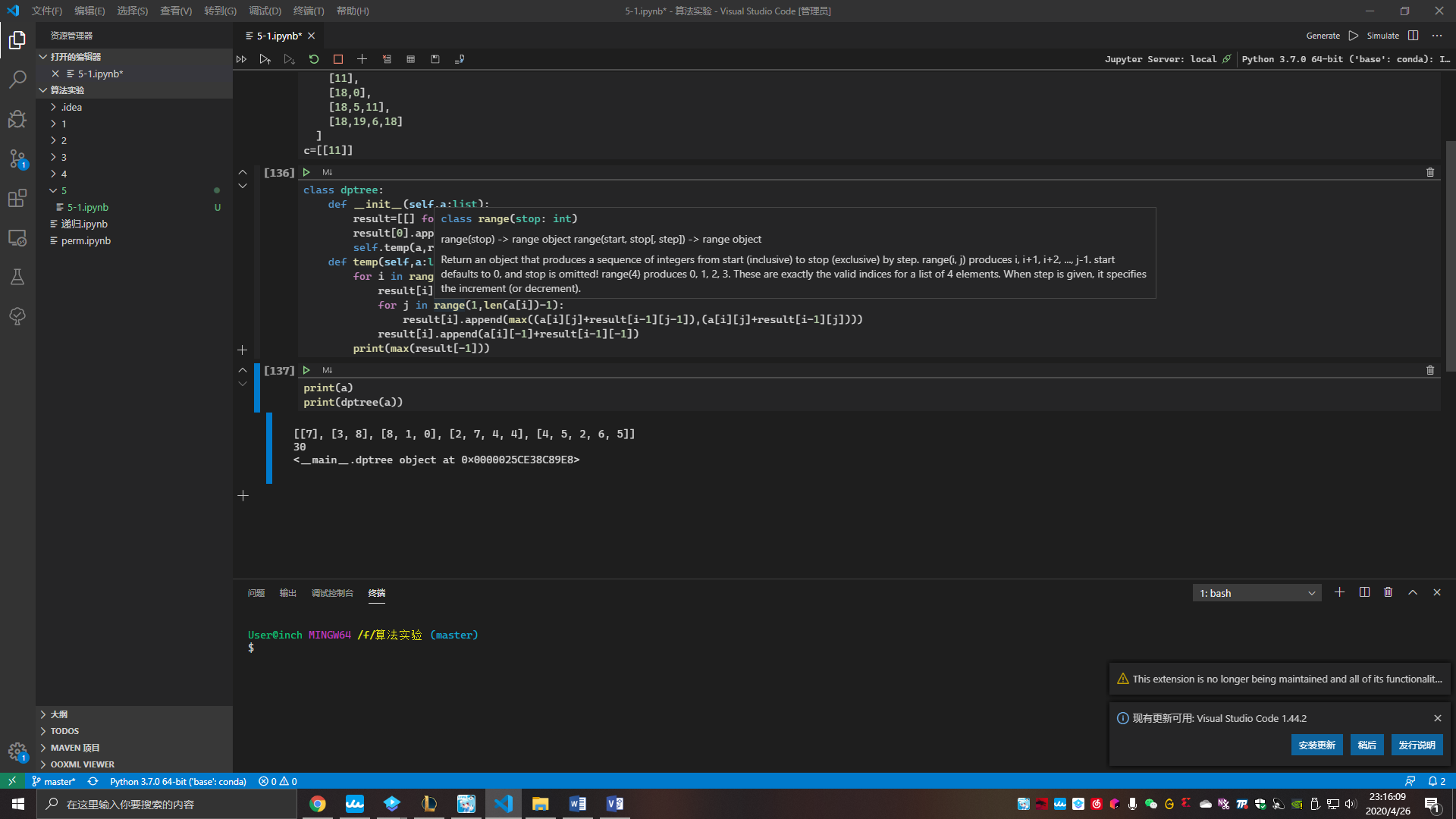
3.遍历列表底部，选取最大值。

1. 流程图



1. 测试结果





1. 实验总结

本算法如果运用了动态规划的递归算法，对每一个结点，返回其数值和最大的底层结点数值的和，这样算法的时间复杂度为，因为会有相同的子结点被重复计算。

然而本算法运用了记录列表，这样就能记录被计算过的结点值，从而时间复杂度降低到。

这里给出另一种思路：由底至上。

从三角形底部开始计算，对每两个结点，选取数值最大的结点，和这两个结点对应的上层结点相加，数值记录在原来的列表上。每次扫描一层，最后顶点的值就是最大路径和。这种算法的时间复杂度也是，相比于之前的算法，空间复杂度少了一个记录列表，并且不需要递归算法。

**暨南大学本科实验报告专用纸(附页)**

1. 附录：源代码

class dptree:

    def \_\_init\_\_(self,a:list):

        result=[[] for i in range(len(a))]

        result[0].append(a[0][0])

        self.temp(a,result)

    def temp(self,a:list,result:list):

        for i in range(1,len(a)):

            result[i].append(a[i][0]+result[i-1][0])

            for j in range(1,len(a[i])-1):

                result[i].append(max((a[i][j]+result[i-1][j-1]),(a[i][j]+result[i-1][j])))

            result[i].append(a[i][-1]+result[i-1][-1])

        print(max(result[-1]))

print(a)

print(dptree(a))

测试用例：

a=[

    [7],

    [3,8],

    [8,1,0],

    [2,7,4,4],

    [4,5,2,6,5]

  ]

b=[

    [11],

    [18,0],

    [18,5,11],

    [18,19,6,18]

  ]

c=[[11]]