**中山大学计算机学院**

**人工智能**

**本科生实验报告**

课程名称：Artificial Intelligence

|  |  |  |  |
| --- | --- | --- | --- |
| 学号 | **22336327** | 姓名 | **庄云皓** |

# 实验题目





# 实验内容

**二分查找**

**算法原理：**对于一个有序递增序列，选序列中间的数字与目标值比较，如果相等则返回答案，否则如果中间值小于目标值，则排除中间数字及其左的数字；如果中间值大于目标值，则排除中间数字及其左的数字。

注意while中的条件是low<=high而非low<high。我们使用low < high时，如果找到目标元素，循环可能会在执行完low++操作后立即结束，这样下一次循环的low值就会比high小，导致跳过了一次数组元素的访问，从而可能漏掉了一些情况（尤其是当目标元素是数组的最后一个元素时）

因为

**关键代码展示：**

while(low <= high):

        mid = int(low +(high - low)>>1)#注意索引要是整数

        if(target < nums[mid]):

            high = mid - 1

        elif(target > nums[mid]):

            low = mid + 1

        else:

            return mid

return -1

**创新点&优化**：使用mid = int(low +(high - low)//2)防止溢出

若其中有重复元素：返回最左边的查找到的元素的下标

    while(low <= high):

        mid = int(low +(high - low)//2)#注意整除

        if(target < nums[mid]):

            high = mid - 1

        elif(target > nums[mid]):

            low = mid + 1

        else:

            while( mid > low and nums[mid-1]==nums[mid]):

                mid = mid - 1

            return mid

    return -1

对开始时就数组为空或查找元素不存在进行判断：

if nums == None or nums[0]>target or nums[high]<target:

        return -1

**矩阵加法**

**算法原理：**

矩阵加法是矩阵中每个元素相加，矩阵乘法是

先初始化C矩阵为全0的矩阵

通过遍历矩阵中每个元素按照对应的规则实现。

关键代码展示：

def MatrixAdd(*A*, *B*):

    C = [ [0 for \_ in range(len(A[0]))] for \_ in range(len(A))]

    for i in range(len(A)):

        for j in range(len(A[0])):

            C[i][j] = A[i][j] + B[i][j]

    return C

def MatrixMul(*A*, *B*):

n = len(A)

    C = [ [0 for \_ in range(n)] for \_ in range(n)]

    for i in range(n):

        for j in range(n):

            for k in range (n):

                C[i][j] += A[i][k] \* B[k][j]#k

    return C

**创新点&优化：**在矩阵加法中可采用列表推导式改进，使代码看上去更加简洁。

C=[[(A[i][j]+B[i][j]) for j in range(len(A[0])] for i in range(A)]

当不是n\*n矩阵时可以加上判断条件看看是否能相乘

if(len(A[0])!=len(B)):

        print("不能相乘")

        return None

    C = [ [0 for \_ in range(len(B[0]))] for \_ in range(len(A))]

    for i in range(len(A)):

        for j in range(len(B[0])):

            for k in range (len(B)):

                C[i][j] += A[i][k] \* B[k][j]#k

**字典遍历**

**算法原理：**如下面的代码所示，将列表从dict1通过dict1.items()函数中取出key,value并倒过来形成一个generator object，再用将其转化为dict

关键代码展示：

inv = dict([val,key] for key,val in dict1.items())#括号的位置

参考链接[5]还有另外实现的方式：

inv = dict(zip(dict1.values(),dict2.keys()))

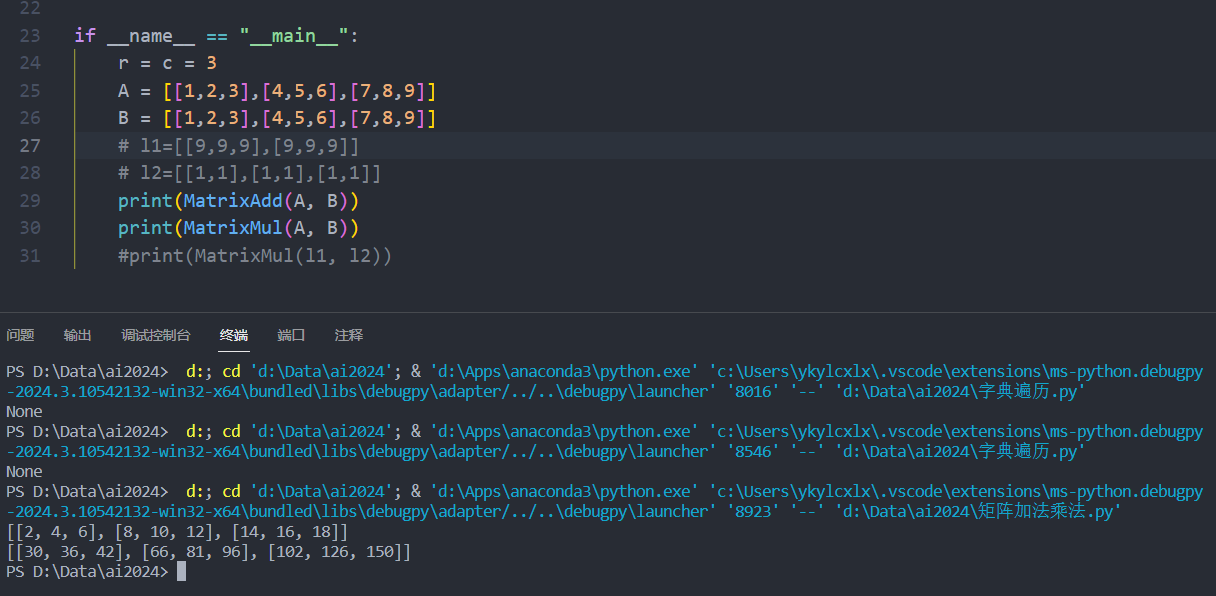
Zip函数对象中对应的元素打包成一个个元组，然后返回由这些元组组成的列表，最后再用dict函数将其转化为字典。

# 实验结果及分析

1.1二分查找



1.2 矩阵相乘相加



1.3 字典遍历



# 参考资料

[1][【二分查找】详细图解\_二分查找法流程图-CSDN博客](https://blog.csdn.net/qq_45978890/article/details/116094046)

[2][二分法查找及有重复值的二分法\_重复有序数组二分查找-CSDN博客](https://blog.csdn.net/qq_24629159/article/details/88630655)

[3][算法 | 【二分查询】进阶与优化 ——区间查询、递归查询、0.618优化... - 我叫RT - 博客园 (cnblogs.com)](https://www.cnblogs.com/TaoR320/p/12680094.html)

[4][Python列表实现矩阵的创建、输入输出、转化转置、加减乘运算并设计一个矩阵计算器GUI界面\_python的列表可以进行矩阵运算吗-CSDN博客](https://blog.csdn.net/weixin_56115549/article/details/124428729)

[5][Python将字典中的键值对反转方法\_py 字典颠倒键值-CSDN博客](https://blog.csdn.net/ITYTI/article/details/85098699" \l ":~:text=Python%E5%B0%86%E5%AD%97%E5%85%B8%E4%B8%AD%E7%9A%84%E9%94%AE%E5%80%BC%E5%AF%B9%E5%8F%8D%E8%BD%AC%E6%96%B9%E6%B3%95%E6%9C%89%E4%B8%89%E7%A7%8D%EF%BC%9A%E7%AC%AC%E4%B8%80%E7%A7%8D%EF%BC%9Adict= {"a":1,"b":2,"c":3}inverse_dic= {}for key,val in dict.items ():,inverse_dic [val]=key%E7%AC%AC%E4%BA%8C%E7%A7%8D%EF%BC%9Adict_list= {"a":1,"b":2,"c":3}inverse_dict=dict ([val,key] for k_py %E5%AD%97%E5%85%B8%E9%A2%A0%E5%80%92%E9%94%AE%E5%80%BC)