1.我们定义 个字符 的超集为: 若字符 A包含字符 B中的所有字符,且字符 A中某 字符 a的数 于a在字符 B中的数 ,那么A为B的“超集”。

如:若字符 A为 "abbccdd",字符 B为 "abcdd",那么A是B的“超集”。 若A为"abbccd",字符B为"abcdd",那么A 是B的“超集”。

现给定字符 A、B,判断A是否是B的“超集

答：

public boolean compare(String a, String b) {

if (TextUtils.isEmpty(b)) return true;

if (TextUtils.isEmpty(a)) return false;

char[] aChars = a.toCharArray();

char[] bChars = b.toCharArray();

List<Character> aList = new ArrayList<>();

for (char aChar : aChars) {

aList.add(aChar);

}

List<Character> bList = new ArrayList<>();

for (char bChar : bChars) {

bList.add(bChar);

}

int index;

for (int i = bList.size() - 1; i >= 0; i--) {

index = aList.indexOf(bList.get(i));

if (index != -1) {

bList.remove(i);

aList.remove(index);

} else {

return false;

}

}

return true;

}

2.　有一个 MxN 的棋盘,每个格子上有一个字母,现在按照下面的规则来构造单词:可以从任意一个位置开始,接着向该位置相邻的 8 个位置中选取下一个(不能选取已经选取过的位置),如此循环,构成一个字母序列,生成一个单词。

现在给你几个单词,尝试判断这几个单词是否能由此方法生成。

输入:

targets= ["ARE", "PENPIEAPPLE", "APPLEPEN", APPLE",

"LIPS", "RED", "AIR", "PLEASE"];

letters

= [['A','R','E'],

['I','P','D'],

['E','L','P']];

canGenerate(str): 如 果 可 以 用 以 上 规 则 生 成 该 词

(str),返回 true,否则返回 false。

输出:

以下词可以被生成:ARE, APPLE, RED, AIR。

A R E

I P D

E L P

答

for x in targets:

if canGenerate(x):

print(x)

3、给定⼀组数字，⼀组有9个数字，将这9个数字填写到3\*3的九宫格内；使得横， 竖，斜对⻆⼀条线上的三个数字之和相等；如果⽆解则打印⽆解；

解决思路：

已经有大量资料提供这种问题的解决方法。

矩阵中元素按照下面的方式排列：

横向为x(从0到n-1)，纵向为y方向（从0到n-1)

1. 第一个数放在X方向的中间位置

2. 其它数顺次放置各个位置，并依据如下原则：（假设第一个数是a,第二个数是b）

以a为中心的位置关系分别为：

左上|上|右上

左 |a | 右

左下 |下|右下

（1）b放在a的右上位置。a(x,y)–>b(x+1,y-1)

（2）如果仅有“右”位置超过边界，即x+1>n，则b(1,y-1)

（3）如果仅有“上”位置超过边界，即y-1<0，则b(x+1,n)

（4）如果“右”“上”位置都超过边界，即x+1>n,y-1

Python代码

def sudoku\_rule(n,sudoku):

tx = n/2

ty = 0

for i in range(n\*n):

sudoku[ty][tx] = i + 1

tx = tx+1

ty = ty-1

if ty<0 and tx>=n: #条件（4）

tx = tx-1

ty = ty+2

elif ty<0: #条件(3)

ty = n-1

elif tx>=n: #条件(2)

tx = 0

elif sudoku[ty][tx]!=0: #条件(5)

tx = tx-1

ty = ty+2

return sudoku

if \_\_name\_\_=="\_\_main\_\_":

n = 5

sudoku = [[0 for i in range(n)] for i in range(n)]

s = sudoku\_rule(n,sudoku)

for line in s:

print line

4、给定形如下⾯的矩阵，

1 1 1 1 1 1

1 1 0 0 0 1

1 0 0 0 1 0

1 1 0 1 1 1

0 1 0 1 0 0

1 1 1 1 1 1

上⾯矩阵的中的1代表海岸线，0代表⼩岛。求第⼆⼤⼩岛的⾯积（即被1中包围的0的 个数，如果只有⼀个⼩岛，输出最⼤⼩岛的⾯积）。

注意：

1. 仅求这样的0，该0所在⾏中被两个1包围，该0所在列中被两个1包围；

2. 输⼊矩阵中包含的⼩岛K >= 1;

样例输⼊：

1 1 1 1 1 1

1 1 0 0 0 1

1 0 0 0 1 0

1 1 0 1 1 1

0 1 0 1 0 0

1 1 1 1 1 1

#include<bits/stdc++.h>

using namespace std;

const int maxn = 2017;

int a[maxn][maxn];

int h[maxn],l[maxn],r[maxn];

int n,ans;

int main()

{

scanf("%d",&n);

for(int i=1;i<=n;i++)

for(int j=1;j<=n;j++)

scanf("%d",&a[i][j]);

for(int i=1;i<=n;i++)

{

for(int j=1;j<=n;j++)

if(a[i][j]) h[j]=0;

else h[j]++;

for(int j=1;j<=n;j++)

{

l[j]=j;

while(l[j]>1&&h[j]<=h[l[j]-1]) l[j]=l[l[j]-1];

}

for(int j=n;j>=1;j--)

{

r[j]=j;

while(r[j]<n&&h[j]<=h[r[j]+1]) r[j]=r[r[j]+1];

}

for(int j=1;j<=n;j++)

ans=max(h[j]\*(r[j]-l[j]+1),ans);

}

cout<<ans;

return 0;

}

题⽬3：

设计⼀个股票模拟交易系统。假设我们有⼀个很⽜叉的AI系统，已经预测到未来⼀段时间

内给定股票的价格，以数组来表示，它的第i个元素是⼀⽀给定的股票在第i天的价格。

假设：

1. 如果你最多只允许完成两次交易(⼀次交易是指：买⼊和卖出)；

2. 你有本⾦ K 单位（K >= 1），⼀单位本⾦可以购买 1 股票；这意味着你寻找的是 K 单位本⾦

条件下最⼤利润。

提示：K = 1 的时候最简单，可以先考虑。

设计⼀个算法来找出最⼤利润。

给定一个数组，它的第 i 个元素是一支给定股票第 i 天的价格。

如果你最多只允许完成一笔交易（即买入和卖出一支股票），设计一个算法来计算你所能获取的最大利润。

注意你不能在买入股票前卖出股票。

思路

引入两个变量，最小买入价格和最大利润，遍历数组，判断最大和最小值来得到最后的结果。

代码

class Solution:

def maxProfit(self, prices):

"""

:type prices: List[int]

:rtype: int

"""

if len(prices)<2:

return 0

min\_price = prices[0]

max\_profile = 0

for i in prices:

min\_price = min(min\_price, i)

max\_profile = max(max\_profile, i - min\_price)

return max\_profile