

实验四实验报告-回溯与分支限界法

计算机84 席卓然 2186113559

一、问题描述

给定1个1000行×20列的0-1矩阵，对于该矩阵的任意1列，其中值为1的元素的数量不超过10%。设有两个非空集合A和B，每个集合由矩阵的若干列组成。集合A和B互斥是指对于矩阵的任意一行，同时满足下列2个条件：

- 1) 若A中有一个或多个元素在这一行上的值是1，则B中的元素在这一行全部是0；
- 2) 若B中有一个或多个元素在这一行上的值是1，则A中的元素在这一行全部是0。

请你设计一个算法，找出一对互斥集合A和B，使得A和B包含的列的总数最大。

二、问题分析

本问题要求将20列的矩阵按列加入到A或B集合中(或都不加入)，并且要求加入后使得两集合中的列满足集合互斥的要求。容易想到使用回溯法，对于每一步的选择有：加入A，加入B，不加入这3种选择，按照这种选择构造解空间树，同时按照递归回溯方法对最优解进行求解。

三、算法设计

在主方法中定义需要用到的数据：`self.A/self.B`记录当前放入A/B中的列的序号；`self.res_A/self.res_B`记录最优解的A/B的列序号；`set_A/set_B`保存当前A/B中已经有1的行的位置；由于题目要求对于最优解的策略要求，需要A/B的长度，A B长度的差的绝对值，以及一个记录最优的A,B长度和的`bestlen`。因为需要使用递归回溯的方法进行求解，因此使用一个递归地`backtrack`函数进行主方法的求解。如果当前最优值小于`sum(A_len+B_len)`，进行最优值得修改，相等时则通过给定的3种择优策略进行判断是否需要进行修改。回溯后得到结果。

四、算法实现

```

import sys
class Solution():
    def main(self,List):
        self.A = [] #A保存当前A中放入的列数
        self.B = []
        self.res_A = [] #res_A保存最优解的A
        self.res_B = []
        self.set_A = set() #set_A保存当前A中已经有1的行
        self.set_B = set()
        self.A_len = 0
        self.B_len = 0
        self.absAB = 20
        self.sum_A = 1000 #解A的个数
        self.bestlen = 0 #最优解对应的长度
        self.dic = {}
        self.row = len(List)
        self.column = len(List[0])
        for i in range(self.column):
            self.dic[i] = set()
            for j in range(self.row):
                if List[j][i] == 1:
                    self.dic[i].add(j)
                    # dic[i]保存了第i列有1的所有位置
        if self.judge():
            self.backtrack(0)
        return [self.res_A,self.res_B]

    def copy(self,set1): #集合的复制
        set2 = set()
        for i in set1:
            set2.add(i)
        return set2

    def intersection(self,set1,set2): #求集合的交集
        set3 = set()
        for i in set1:
            set3.add(i)
        for j in set2:
            set3.add(j)
        return set3

    def compare(self,set1,set2): # set1与set2没有重复元素 true;反之false;
        for i in set1:
            if i in set2:
                return False
        return True

    def sub(self,set1,set2): # 集合减法
        set3 = set()
        for i in set1:
            if i not in set2:

```

```

        set3.add(i)
    return set3

def judge(self):
    for i in range(self.column):
        for j in range(i+1,self.column):
            if self.compare(self.dic[i],self.dic[j]):
                return True
    return False

def backtrack(self,i):
    if i>=20:
        if len(self.A)==0 or len(self.B)==0:
            return
        if(len(self.A)+len(self.B)>self.bestlen):
            self.res_A = self.A.copy()
            self.res_B = self.B.copy()
            self.A_len = len(self.res_A)
            self.B_len = len(self.res_B)
            self.absAB = abs(self.A_len-self.B_len)
            self.bestlen = self.A_len+self.B_len
            sum = 0
            for every in self.res_A:
                sum += every
            self.sum_A = sum

            # 3个取最优的策略
        elif(len(self.A)+len(self.B)==self.bestlen):
            if(abs(len(self.A)-len(self.B))<self.absAB):
                self.res_A = self.A.copy()
                self.res_B = self.B.copy()
                self.A_len = len(self.res_A)
                self.B_len = len(self.res_B)
                self.absAB = abs(self.A_len - self.B_len)
                sum = 0
                for every in self.res_A:
                    sum += every
                self.sum_A = sum
            elif(abs(len(self.A)-len(self.B))==self.absAB):
                if(len(self.A)>self.A_len):
                    self.res_A = self.A.copy()
                    self.res_B = self.B.copy()
                    self.A_len = len(self.res_A)
                    self.B_len = len(self.res_B)
                    sum = 0
                    for every in self.res_A:
                        sum += every
                    self.sum_A = sum
                elif(len(self.A)==self.A_len):
                    sum = 0
                    for every in self.A:

```

```

        sum += every
        if(sum<self.sum_A):
            self.res_A = self.A.copy()
            self.res_B = self.B.copy()
            self.sum_A = sum
        return
    return
return

# 剪枝条件
if len(self.A)+len(self.B)+self.column-i<self.bestlen:
    return
# 开始递归
if self.compare(self.sub(self.dic[i],self.set_A),self.set_B):
    temp1 = self.copy(self.set_A)
    self.A.append(i)
    self.set_A = self.copy(self.intersection(self.set_A,self.dic[i]))
    self.backtrack(i+1)
    self.set_A = self.copy(temp1)
    self.A = self.A[0:-1]
    # 放入A集合
    if self.compare(self.sub(self.dic[i],self.set_B),self.set_A):
        temp2 = self.copy(self.set_B)
        self.B.append(i)
        self.set_B = self.copy(self.intersection(self.set_B,self.dic[i]))
        self.backtrack(i+1)
        self.set_B = self.copy(temp2)
        self.B = self.B[0:-1]
        #放入B集合
        self.backtrack(i+1) #不放入该列
def read():
    infile = open("exp6_in.txt","r")
    outfile = open("out.txt", "w")
    s = infile.read()
    index = 0
    n = 0
    while True:
        if s[index]=='\n':
            index+=1
            break
        else:
            n = n*10+int(s[index])
            index+=1
    for _ in range(n):
        L = []
        for i in range(1000):
            List = []
            while True:
                if s[index] == '\n':
                    index += 1

```

```

        L.append(List)
        break
    if s[index] != ' ':
        List.append(int(s[index]))
    index += 1
S = Solution()
ans = S.main(L)
#print(ans)
outfile.write(' '.join(str(num) for num in ans[0])+'\n')
outfile.write(' '.join(str(num) for num in ans[1])+'\n')
infile.close()
outfile.close()
read()

```

五、运行结果

对于标准测试样例exp6_in.txt，通过运行程序得到的输出结果如下：

The screenshot shows the Visual Studio Code interface. The file explorer on the left lists various files, including exp1 through exp6. The main editor window displays the contents of exp6_out.txt, which contains 21 lines of output. The terminal at the bottom shows the PowerShell prompt, indicating the program was executed from the directory C:\Users\席卓然\Desktop\大三上课程资料\算法\上机.

```

exp6_out.txt
1 0 1 2 3 4 5 7 8 9 11
2 6 10 12 13 14 15 16 17 18 19
3 1 2 3 4 5 6 7 10 12 13 14 16 17 18 19
4 0 8 9 11 15
5 1 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
6 0 2
7 0 1 2 3 4 5 6 7 8 9 10 11 12 13 15 17 18 19
8 16
9 0 1 2 3 4 5 6 7 8 10 11 12 13 14 15 16 17 18 19
10 9
11 0 2 3 4 5 6 7 8 10 11 12 13 14 15 16 17 19
12 1 9 18
13 0 1 2 3 4 5 6 8 9 10 11 12 13 14 15 16 18 19
14 17
15 0 1 2 3 4 5 6 7 9 10 11 13 14 15 16 17 18 19
16 8 12
17 0 1 2 3 4 5 6 7 8 9 10 11 12 13 15 16 18 19
18 14
19
20
21

```

```

1: powershell
Windows PowerShell
版权所有 (C) Microsoft Corporation。保留所有权利。

尝试新的跨平台 PowerShell https://aka.ms/pscore6

PS C:\Users\席卓然\Desktop\大三上课程资料\算法\上机>

```

0 1 2 3 4 5 7 8 9 11
6 10 12 13 14 15 16 17 18 19
1 2 3 4 5 6 7 10 12 13 14 16 17 18 19
0 8 9 11 15
1 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
0 2
0 1 2 3 4 5 6 7 8 9 10 11 12 13 15 17 18 19
16
0 1 2 3 4 5 6 7 8 10 11 12 13 14 15 16 17 18 19
9
0 2 3 4 5 6 7 8 10 11 12 13 14 15 16 17 19
1 9 18
0 1 2 3 4 5 6 8 9 10 11 12 13 14 15 16 18 19
17
0 1 2 3 4 5 6 7 9 10 11 13 14 15 16 17 18 19
8 12
0 1 2 3 4 5 6 7 8 9 10 11 12 13 15 16 18 19
14

注意到最后一组输入是无解的。