

2016-2017 第一学期

《算法设计与分析》 期末考核

专业 计算机科学与技术 班级 计科 141 软件(对)

学号 20141515404 姓名 刘博

一、项目描述

最小长度电路板排列问题

问题描述:最小长度电路板排列问题是大规模电子系统设计中提出的实际问题。该问题的提法是,讲 n 块电路板以最佳的排列方案插入带有 n 个插槽的机箱中。 n 块电路板的不同的排列方式对应于不同的电路板插入方案。

设 $B=\{1,2,\dots,n\}$ 是 n 块电路板的集合。集合 $L=\{N_1,N_2,\dots,N_m\}$ 是 n 块电路板的 m 个连接块。其中每个连接块 N_i 是 B 的一个子集,且 N_i 中的电路板用一根导线连接在一起。在最小长度电路板排列问题中,连接块的长度是指该连接块中第 1 块电路板到最后 1 块电路板之间的距离。

设计要求:对于给定的电路板连接块,设计一个分支限界算法,找出所给 n 个电路板的最佳排列,使得 m 个连接块中最大长度达到最小。

二、算法设计

算法 `bbBoards` 是解电路板排列问题的优先队列式分支限界法的主体,首先考虑 $s=n-1$ 的情形,此时已排定 $n-1$ 块电路板,故当前扩展结点是一个排列树中的叶结点的父结点, x 表示相应于该叶结点的电路板排列,当 $s<n-1$ 时,算法依次产生当前扩展结点的所以儿子结点,对于当前扩展结点的每一个儿子结点 `node`,计算出其相应的密度 `node.cd`,当 `node.cd<bestd` 时,将该儿子结点 `node` 插入到活结点优先队列中,而当 `node.cd>=bestd` 时,以 `node` 为根的子树中不可能比当前最优解 `bestx` 更好的解,故可将结点 `node` 舍去。

三、程序

```
import java.util.Collections;
import java.util.LinkedList;
import java.util.Scanner;
public class Bboard {
    public static int bbBoards(int[][] board,int m,int[] bestx){
        int n=board.length-1;
        LinkedList<Heapno> heap =new LinkedList<Heapno>();
        Heapno enode=new Heapno(0,new int[m+1],0,new int[n+1]);
        int[] total = new int[m+1];
        for(int i=1;i<=n;i++){
            enode.x[i]=i;
            for(int j=1;j<=m;j++)
```

```

        total[j]+=board[i][j];
    }
    int bestd=m+1;
    int[] x=null;
    do{
        if(enode.s==n-1){
            int ld=0;
            for(int j=1;j<=m;j++)
                ld+= board[enode.x[n]][j];
            if(ld<bestd){
                x=enode.x;
                bestd=Math.max(ld, enode.cd);
            }else{
                for(int i=enode.s+1;i<=n;i++){
                    Heapno node=new Heapno(0,new int[m+1],0,new int[n+1]);
                    for(int j=0;j<=m;j++)
                        node.now[j]=enode.now[j]+board[enode.x[i]][j];
                    int ld=0;
                    for(int j=1;j<=m;j++){
                        if(node.now[j]>0 && total[j]!=node.now[j])
                            ld++;
                    }
                    node.cd=Math.max(ld,enode.cd);
                    if(node.cd<bestd){
                        node.s=enode.s+1;
                        for(int j=1;j<=n;j++)
                            node.x[j]=enode.x[j];
                        node.x[node.s]=enode.x[i];
                        node.x[i]=enode.x[node.s];
                        heap.add(node);
                        Collections.sort(heap);
                    }
                }
            }
        }
    }

    enode=(Heapno)heap.poll();
    }while(enode!=null&&enode.cd<bestd);
    for(int i=1;i<=n;i++)
        bestx[i]=x[i];
    return bestd;
}

public static void main(String[] args){
    int m=5;    //链接
    int n=8;    //电路板
    int[][] b=new int[n+1][m+1];

```

```

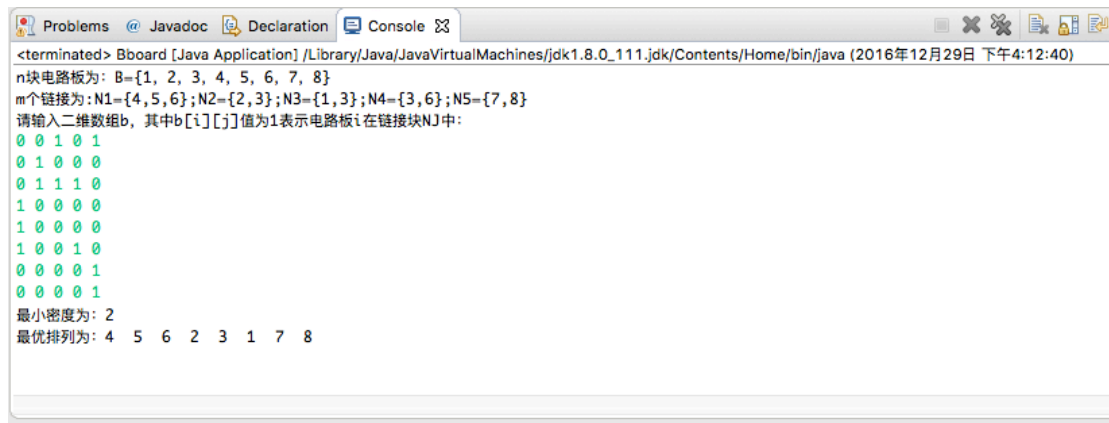
        System.out.println("n块电路板为: B={1, 2, 3, 4, 5, 6, 7, 8}");
        System.out.println("m个链接
为:N1={4,5,6};N2={2,3};N3={1,3};N4={3,6};N5={7,8}");
        System.out.println("请输入二维数组b, 其中b[i][j]值为1表示电路板i在
链接块NJ中: ");
        Scanner sc=new Scanner(System.in);
        for(int i=1;i<=n;i++){
            String str=sc.nextLine();
            String[] s=str.split(" ");
            for(int j=1;j<=m;j++){
                b[i][j]=Integer.parseInt(s[j-1]);
            }
        }
        int[] bestx=new int[n+1];
        int bestd=bbBoards(b,m,bestx);
        System.out.println("最小密度为: "+bestd);
        System.out.print("最优排列为: ");
        for(int j=1;j<bestx.length;j++){
            System.out.print(bestx[j]+" ");
        }
    }
}

class Heapno implements Comparable{
    int s;
    int cd;
    int[] now;
    int[] x;
    public Heapno(int cdd,int[] noww,int ss,int[] xx){
        cd=cdd;
        now=noww;
        s=ss;
        x=xx;
    }
    public int compareTo(Object x){
        int xcd=((Heapno)x).cd;
        if(cd<xcd) return -1;
        if(cd==xcd) return 0;
        return 1;
    }
}

```

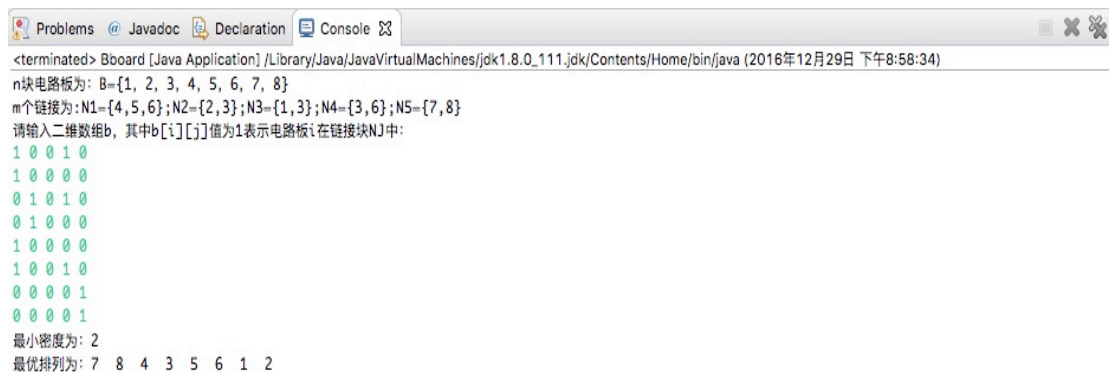
四、运行结果

结果 1



```
<terminated> Bboard [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_111.jdk/Contents/Home/bin/java (2016年12月29日 下午4:12:40)
n块电路板为: B={1, 2, 3, 4, 5, 6, 7, 8}
m个链接为: N1={4, 5, 6}; N2={2, 3}; N3={1, 3}; N4={3, 6}; N5={7, 8}
请输入二维数组b, 其中b[i][j]值为1表示电路板i在链接块NJ中:
0 0 1 0 1
0 1 0 0 0
0 1 1 1 0
1 0 0 0 0
1 0 0 0 0
1 0 0 1 0
0 0 0 0 1
0 0 0 0 1
最小密度为: 2
最优排列为: 4 5 6 2 3 1 7 8
```

结果 2



```
<terminated> Bboard [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_111.jdk/Contents/Home/bin/java (2016年12月29日 下午8:58:34)
n块电路板为: B={1, 2, 3, 4, 5, 6, 7, 8}
m个链接为: N1={4, 5, 6}; N2={2, 3}; N3={1, 3}; N4={3, 6}; N5={7, 8}
请输入二维数组b, 其中b[i][j]值为1表示电路板i在链接块NJ中:
1 0 0 1 0
1 0 0 0 0
0 1 0 1 0
0 1 0 0 0
1 0 0 0 0
1 0 0 1 0
0 0 0 0 1
0 0 0 0 1
最小密度为: 2
最优排列为: 7 8 4 3 5 6 1 2
```

五、复杂度分析

在解空间排列树的每个节点处, 算法花费 $O(m)$ 计算时间为每个儿子节点计算空间密度。因此, 计算密度所耗费的总计算时间为 $O(mn!)$ 。另外, 生成排列树需 $O(n!)$ 时间。每次更新当前最优解至少使 $bestd$ 减少 1, 而算法运行结束时 $bested \geq 0$ 。因此, 最优解被更新的次数为 $O(m)$, 更新当前最优解为 $O(mn)$ 时间。

综上可知, 解电路板排序问题的回溯算法所需要的计算时间为 $O(mn!)$ 。