G Ugly Windows——by Arthur

Description

1)Sheryl works for a software company in the country of Brada. Her job is to develop a Windows operating system. People in Brada are incredibly conservative. They even never use graphical monitors! So Sheryl's operating system has to run in text mode and windows in that system are formed by characters. Sheryl decides that every window has an ID which is a capital English letter(**'A' to 'Z'**). Because every window had a unique ID, there can't be more than 26 windows at the same time.And as you know, all windows are rectangular.

2)If a window has no parts covered by other windows, we call it a "top window"(the frame is also considered as a part of a windows). Usually, the top windows are the windows that interact with user most frequently. Assigning top windows more CPU time and higher priority will result in better user exeriences. Given the screen presented as Figs followed, can you tell Sheryl which windows are top window?

```
.........................
....AAAAAAAAAAAAA.........
....A..........A.........
....A.......BBBBBBBBBB....
....A.......B.........BCCC.
....AAAAAAAAB.........B..C.
........C...BBBBBBBBBB..C.
........CCCCCCCCCCCCCCCCC.
.........................
          Fig.1


.........................
....DDDDDDDDDDDDD.........
....D..........D.........
....D..........D.........
....D..........D..AAA...
....DDDDDDDDDDDDD..A.A...
...................AAA...
          Fig.2
```

Input

1)The input contains serveral test cases.
2)Each test case begins with two integers, **n** and **m**(1<=n,m<=100), indicating that the screen has **n** lines, and each lines consists of **m** characters.
3)The following **n** lines describe the whole screen you see. Each line contains m characters.For characters which are not any window frame, we just replace them with **'.'**.
4)It is guaranteed that:

```
__1)There is at least one window on the screen.
__2)Any window's frame is at least 3x3.
__3)No part of any window is outside the screen.
```

Output

1)For each test case, output the IDs of all top windows in a lines without blanks and in alphabet order.

Sample Input

```
9 26
```

```
.........................
....AAAAAAAAAAAAA.........
....A...........A.........
....A.......BBBBBBBBBB....
....A.......B.........BCCC.
....AAAAAAAAB.........B..C.
........C...BBBBBBBBBB..C.
........CCCCCCCCCCCCCCCCC.
.........................
7 25
.........................
....DDDDDDDDDDDDD.........
....D...........D.........
....D...........D.........
....D...........D..AAA...
....DDDDDDDDDDDDD..A.A...
...................AAA...
```

Sample Output

```
B
AD
```

解题分析

神一样的题目，矩形相交的终极版本——
你的目的就是判断这一大堆矩形，是否互相有所重叠

首先由于题目中所述，矩形不会超过 26 个，且他们只会由不重复的 26 个字母中的某几个构成。这个条件让题目的难度大大降低。
这里给出一种方案：

① 将图表中的矩形转换出来成为正统的矩形，找到顶角并且转换为点坐标。在这过程中进行第一层判断，判断这个矩形的完整性（这个用来解决角覆盖问题）

② 根据这些矩形的坐标在图表之中遍历对应的边，一旦某条边上出现不属于该矩形的字符，那么说明这个矩形被覆盖了。（这个和上一次 Rectangle 那道题的思路比较类似，这一步用来解决边覆盖问题）
③ 最后判断一次是否该矩形在某一个矩形的内部，参见代码中所述。
④ 上面三个判断中，只要有一个符合条件，那么该矩形就是被覆盖的，它对应的字母便计入考虑范围内。

参考代码

```
#include<iostream>
#include<cstdio>
#include<string>
#include<cstring>
```

```
#define MaxSize 101
using namespace std;
typedef struct
{
    int x;
    int y;
}Point;    //定义点
typedef struct
{
    Point p[4];
}Rect;    //定义矩形

bool checker[26];  //26 个矩形的判断
Rect r[26];        //最多 26 个矩形
int m,n;          //点阵的长宽
char rec[MaxSize][MaxSize];   //点阵
void Init()    //初始化部分
{
   for(int i=0;i<26;i++)
   {
      for(int j=0;j<4;j++)
         r[i].p[j].x=r[i].p[j].y=-1;//四个点坐标都置为-1，为不合法坐标
      checker[i]=false;   //初始时认为没有任何一个矩形被覆盖
   }
}


bool checkPoint(int i)   //判断矩形合法
{
    for(int s=0;s<4;s++)  //对四个点进行判断
      if(r[i].p[s].x==-1||r[i].p[s].y==-1)  //出现不合法坐标
        return false;   //说明这个矩形不完整，被角覆盖，返回 false
    return true;      //否则返回 true，表示未被覆盖
}



bool checkIntact(int i)   //判断是否有边覆盖
{//四条边依次扫描，看边是否被截断
   int temp=r[i].p[0].y;
   for(int s=r[i].p[0].x;s<r[i].p[1].x;s++)
     if(rec[temp][s]!=rec[temp][s+1])  return false;
   temp=r[i].p[1].x;
   for(int s=r[i].p[1].y;s<r[i].p[2].y;s++)
```

```
    if(rec[s][temp]!=rec[s+1][temp])  return false;
  temp=r[i].p[2].y;
  for(int s=r[i].p[2].x;s>r[i].p[3].x;s--)
    if(rec[temp][s]!=rec[temp][s-1])  return false;
  temp=r[i].p[3].x;
  for(int s=r[i].p[3].y;s>r[i].p[0].y;s--)
    if(rec[s][temp]!=rec[s-1][temp])  return false;
  return true; //未被覆盖，返回 true
}

bool checkInside(int i)//判断是否有在内部的情况
{
    for(int s=0;s<26;s++)
      if(s!=i)
          if(r[s].p[0].x>r[i].p[0].x&&r[s].p[0].y>r[i].p[0].y&&r[s].
          p[0].x<r[i].p[2].x&&r[s].p[0].y<r[i].p[2].y)
               return false;//上面这一大坨判断大家自己看一下吧 TT
    return true;  //如果未出现内含矩形，则未被覆盖
}

void checkTop()  //整体的判断是否有矩形被覆盖
{
    for(int i=0;i<26;i++)   //26 个全都要搜
      if(checkPoint(i))     //如果未出现角覆盖，判断边覆盖
         if(checkIntact(i))   //如果未出现边覆盖，判断内含情况
           if(checkInside(i))  //如果内含也没有
              checker[i]=true;   //这个矩形未被覆盖
}









void setPoint()   //将点阵转化为矩形
{
    for(int i=0;i<n;i++)
          for(int j=0;j<m;j++)
              if(rec[i][j]!='.')//出现不是’.’的
              {
```

```
                    if(j+1<m&&i+1<n&&rec[i][j+1]==rec[i][j]&&rec[i+1][j]
                    ==rec[i][j])
                     {//第一个顶点
                       r[rec[i][j]-'A'].p[0].x=i;
                       r[rec[i][j]-'A'].p[0].y=j;
                     }

                     if(i+1<n&&j-1>=0&&rec[i][j-1]==rec[i][j]&&rec[i+1][
                     j]==rec[i][j])
                     {//第二个顶点
                       r[rec[i][j]-'A'].p[1].x=i;
                       r[rec[i][j]-'A'].p[1].y=j;
                     }

                      if(i-1>=0&&j-1>=0&&rec[i][j-1]==rec[i][j]&&rec[i-1
                      ][j]==rec[i][j])
                     {//第三个顶点
                       r[rec[i][j]-'A'].p[2].x=i;
                       r[rec[i][j]-'A'].p[2].y=j;
                     }

                     if(i-1>=0&&j+1<m&&rec[i][j+1]==rec[i][j]&&rec[i-1][
                     j]==rec[i][j])
                     {//第四个顶点
                       r[rec[i][j]-'A'].p[3].x=i;
                       r[rec[i][j]-'A'].p[3].y=j;
                     }
                  }
}
```

```
int main()
{
    while(cin>>n>>m)
    {
        Init();
        for(int i=0;i<n;i++)
```

```
            for(int j=0;j<m;j++)
                cin>>rec[i][j];
        setPoint();//将点阵转换为矩形顶点坐标
        checkTop();   //判断是否相交
        for(int i=0;i<26;i++)//从 A 到 Z 扫描一遍
            if(checker[i])//如果未被覆盖，则输出对应字母
                cout<<static_cast<char>('A'+i);
        cout<<endl;
    }
}
```