插头DP

建模专题

回顾：轮廓线DP

S记录轮廓线上**格点**的状态
在一个格点决策
计算下一个轮廓线的状态S1
转移状态（滚动数组）

1567

铺地砖加强版

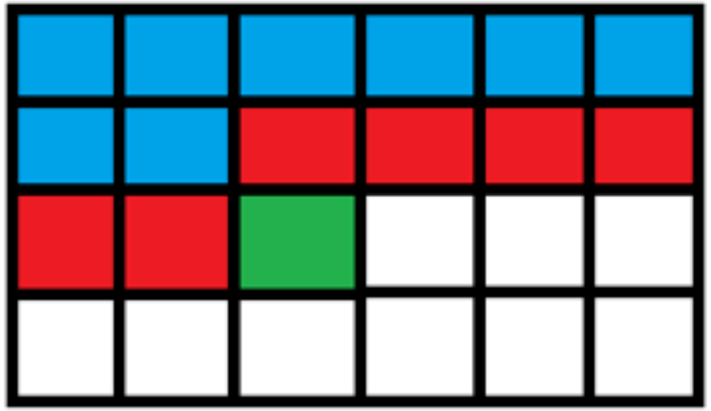
用1*1和1*2的地砖盖满M*N的区域有多少种方法?
（N<=100,M<=10）
地图用01表示 1表示可以放置，0表示不可以
要求1*1的地砖个数在[c,d]之间 (c,d<=20)
答案对1e9+7取模

输入样例:
2 2 0 0
11
11

输出样例:
2

输入样例:
4 5 3 5
11111
11011
10101
11111

输出样例:
954



1.状态表示

F[i][j][S][t]:
涂到Aij,
轮廓线状态为S,
用了t个1*1
滚动→f[p][S][t]

2.决策→轮廓线转移

决策1:1*1
转移到F[p^1][S1][t+1]

决策2:1*2横放
转移到F[p^1][S1][t]

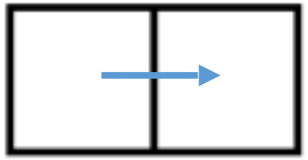
决策3:1*2竖放
转移到F[p^1][S1][t]

决策4: 不放
转移到F[p^1][S1][t]

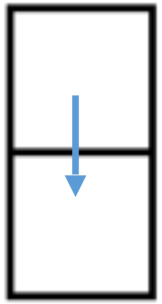
效率:O(N*M*2^M*D)

3.确定初始结束状态

插头：相邻的格点叫做一个插头



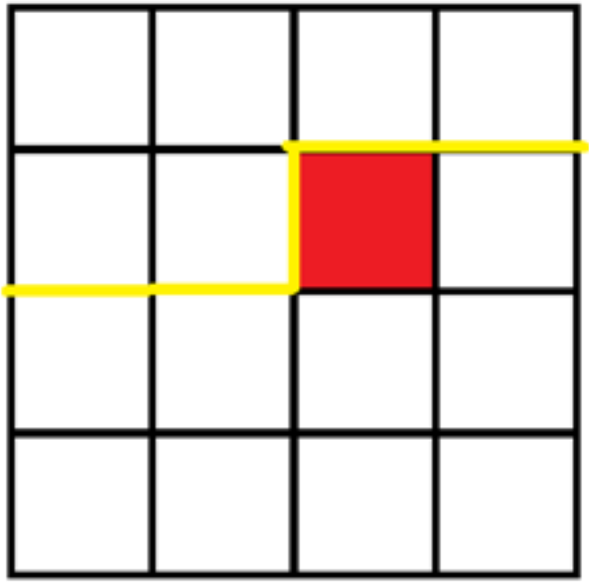
右插头



下插头

插头DP：连通性状态压缩

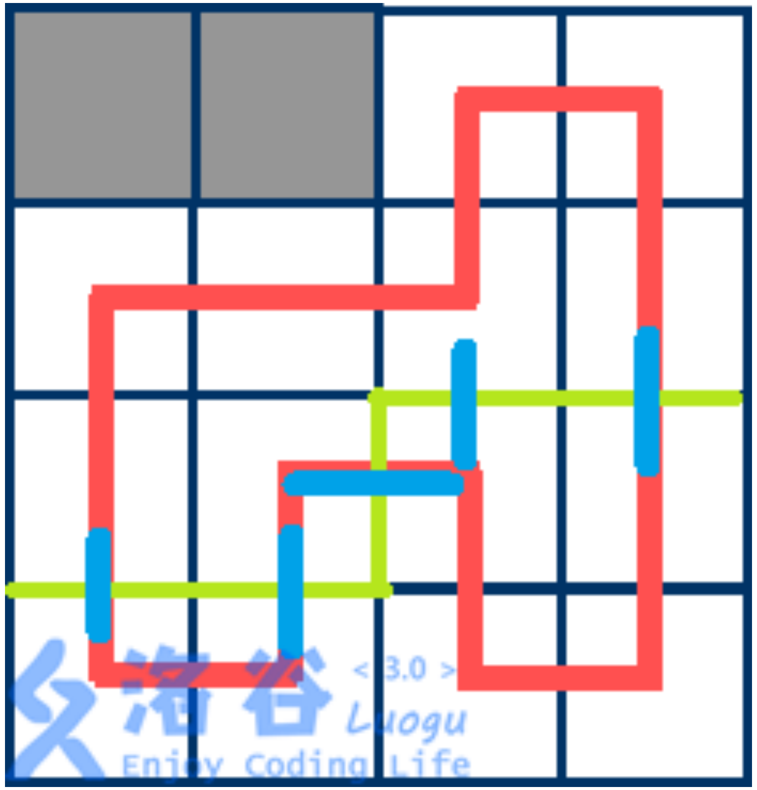
插头DP的轮廓线：不是格点而是线段

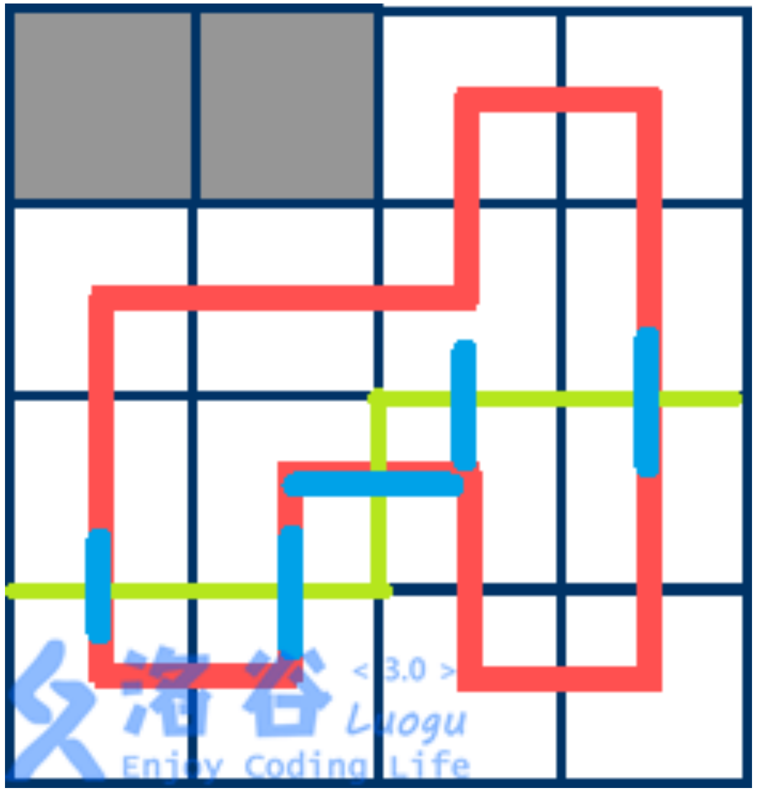


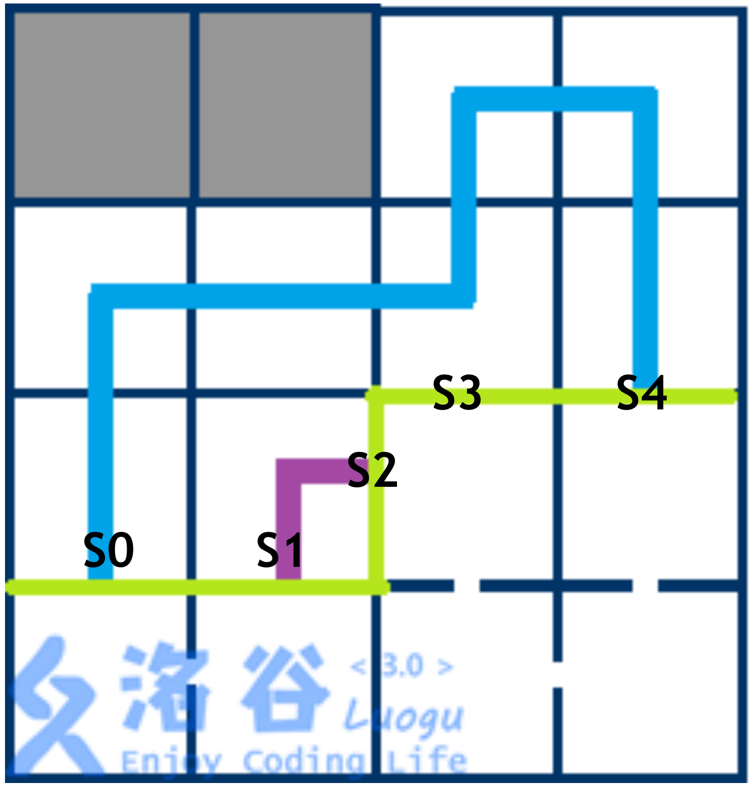
一共有M+1个插头

回路计数

给你一个n×m的棋盘(有的格子存在障碍)，求经过所有非障碍格子的哈密顿回路个数。 (n,m<=12)$(n,m<=12)$







对应的S：

1	2	2	0	1

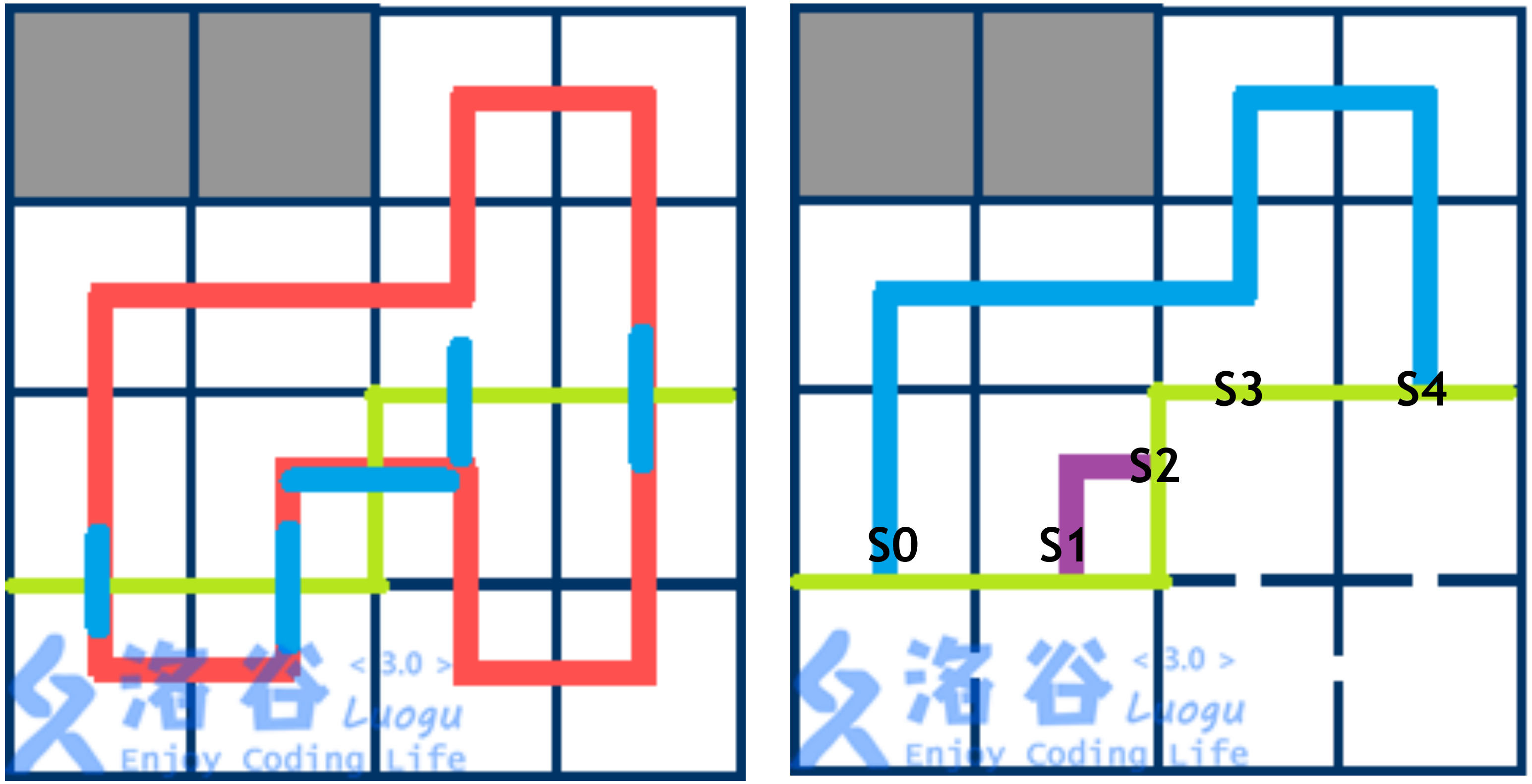
状态数太多

对应的S：

(	(	)	#	)

括号表示法
三进制表示
(四进制存储)

猜一猜：S是怎么
表示出来的?



对应的S：

(	(	)	#	)

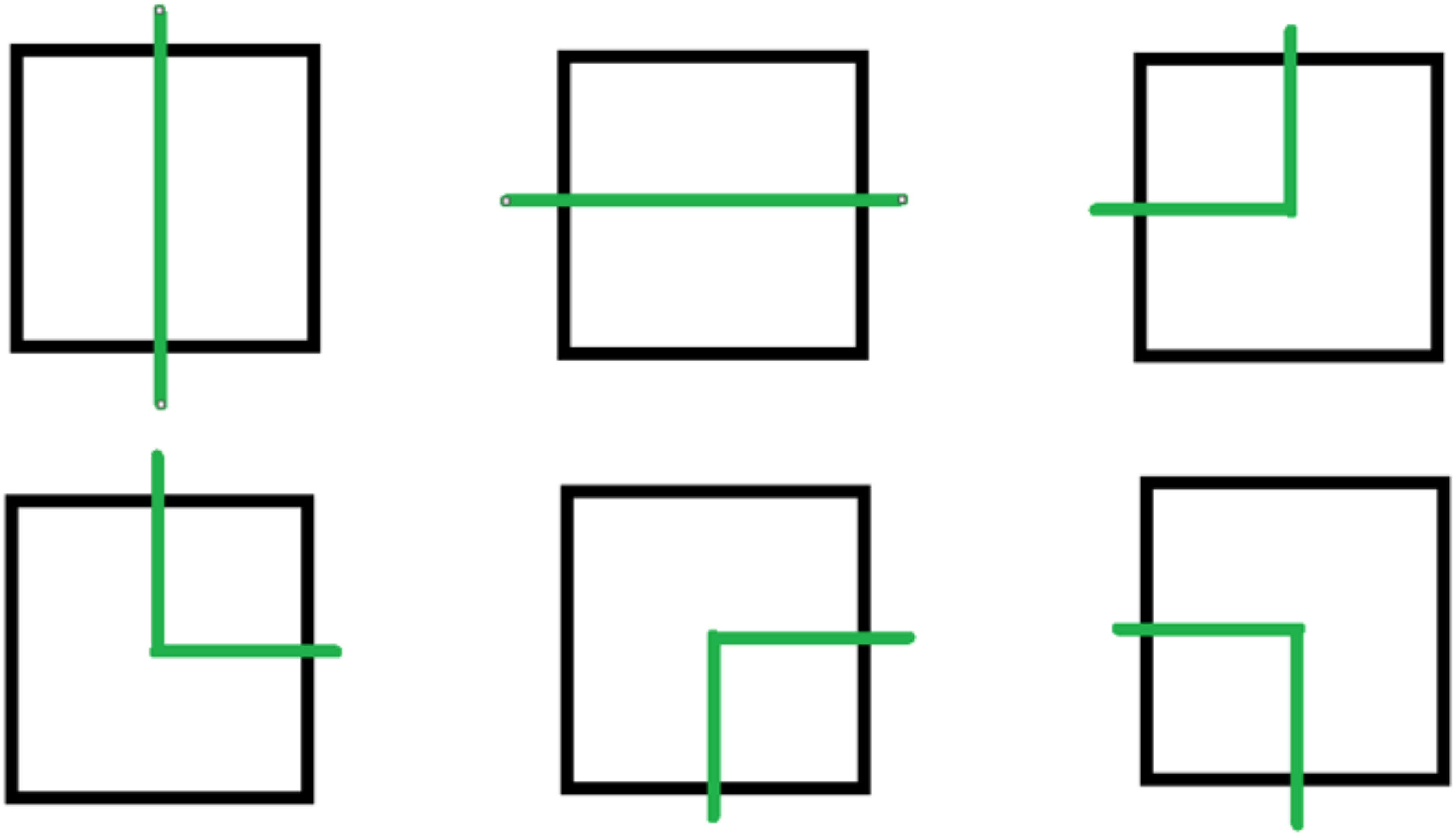
轮廓线的上方：所有线段都已经画完

决策：格点处的线段是什么样的?

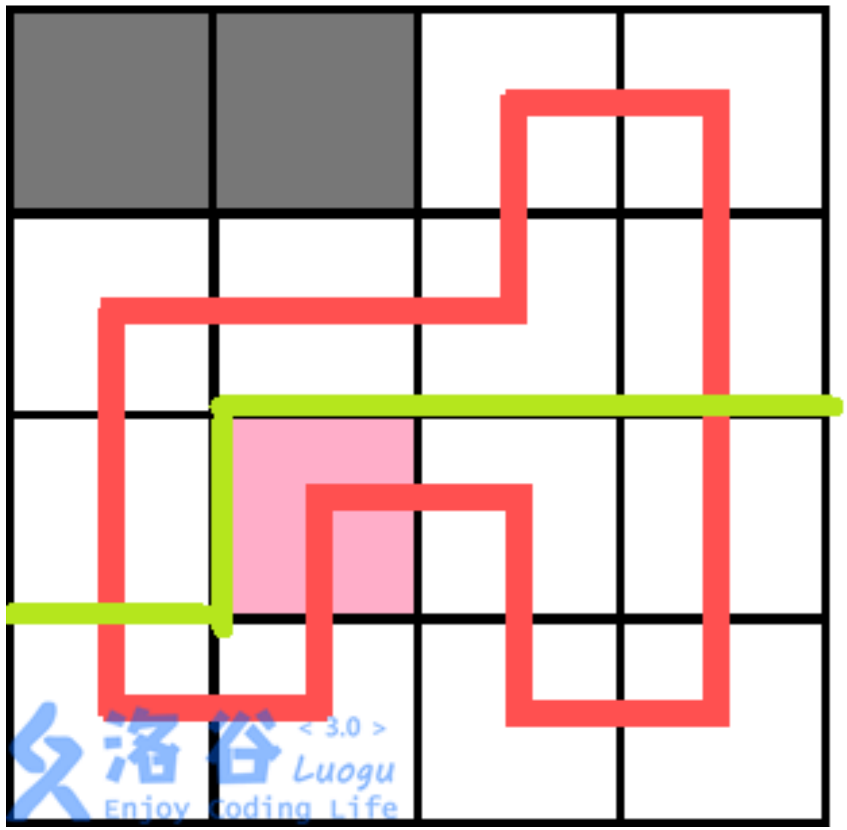
六种决策

接下来做什么?

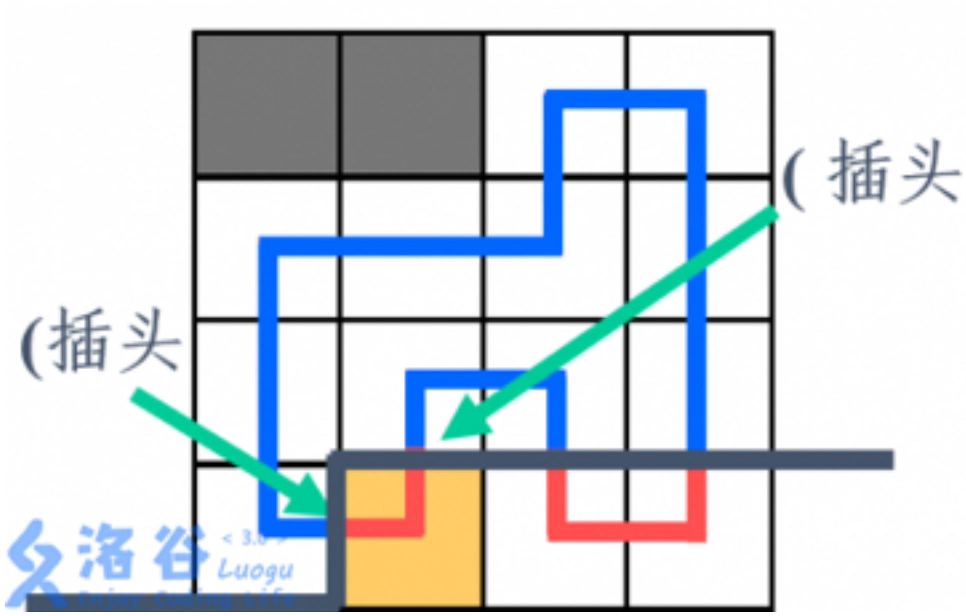
决策过程→S的转移



情况1:右+下

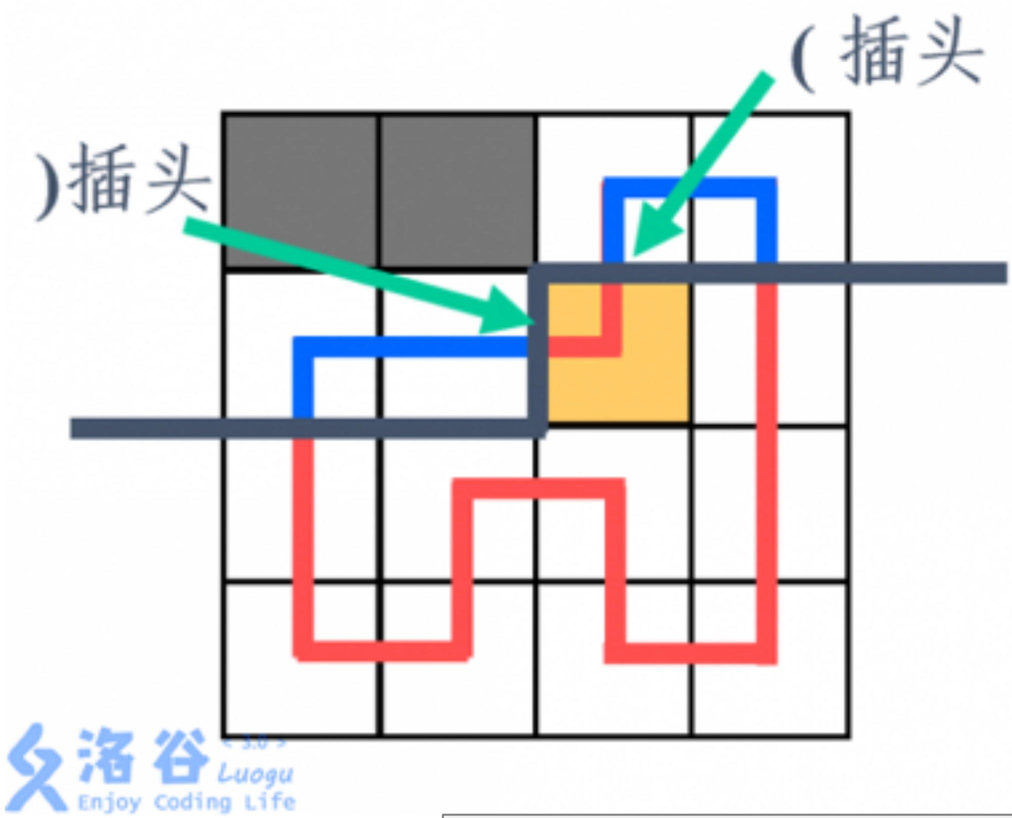


情况2:上+左



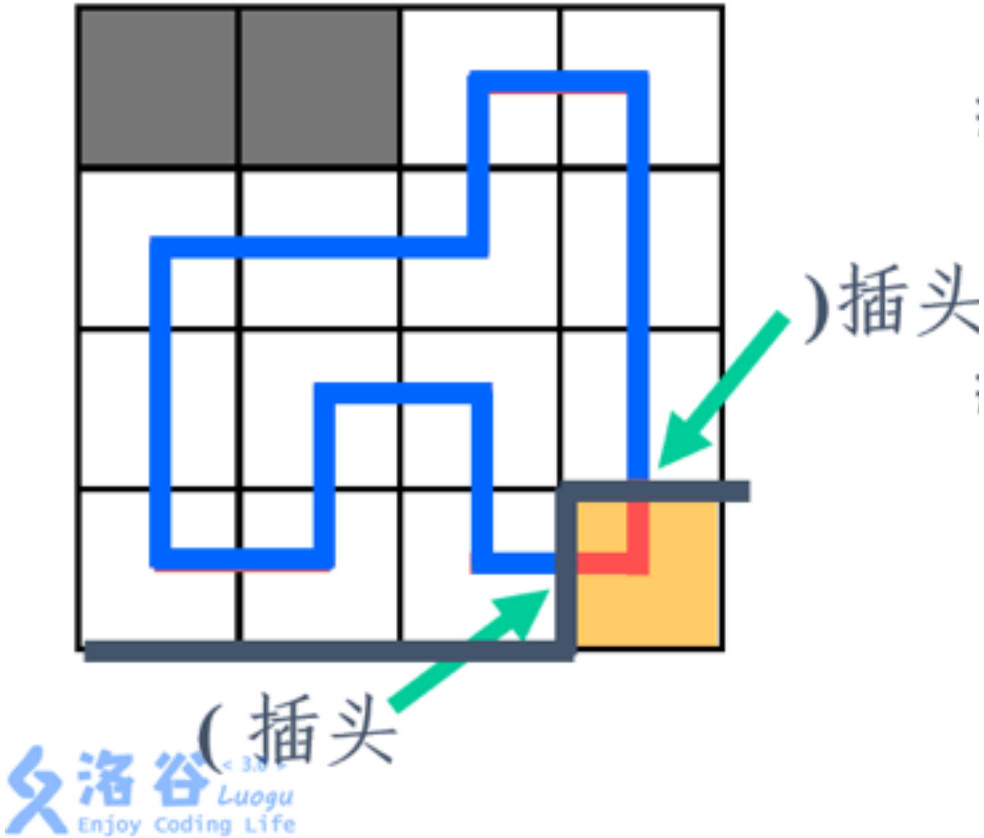
情况2.1:上=(左=(

情况2:上+左



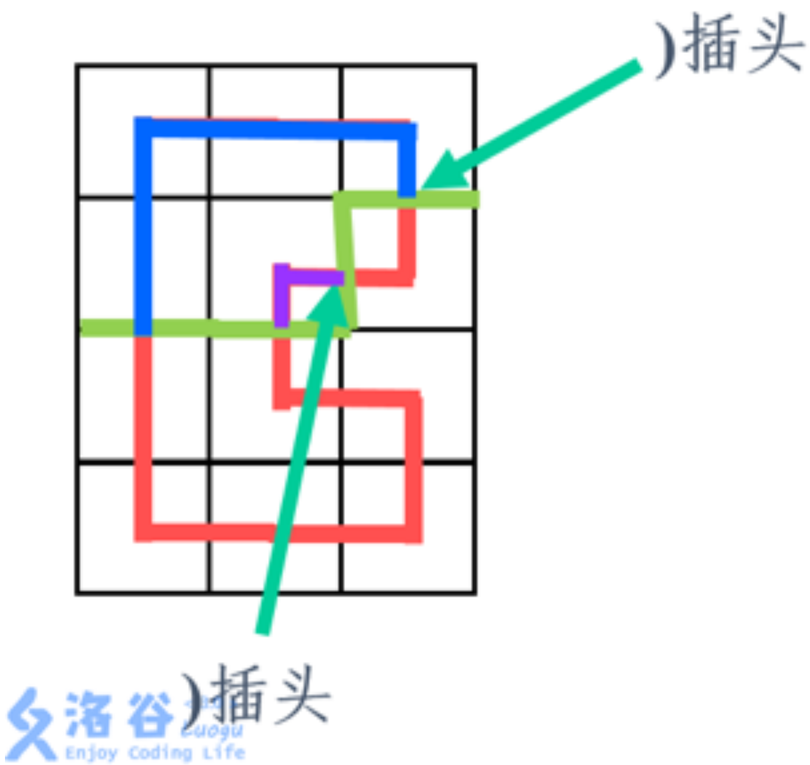
情况2.2:上=(左=)

情况2:上+左



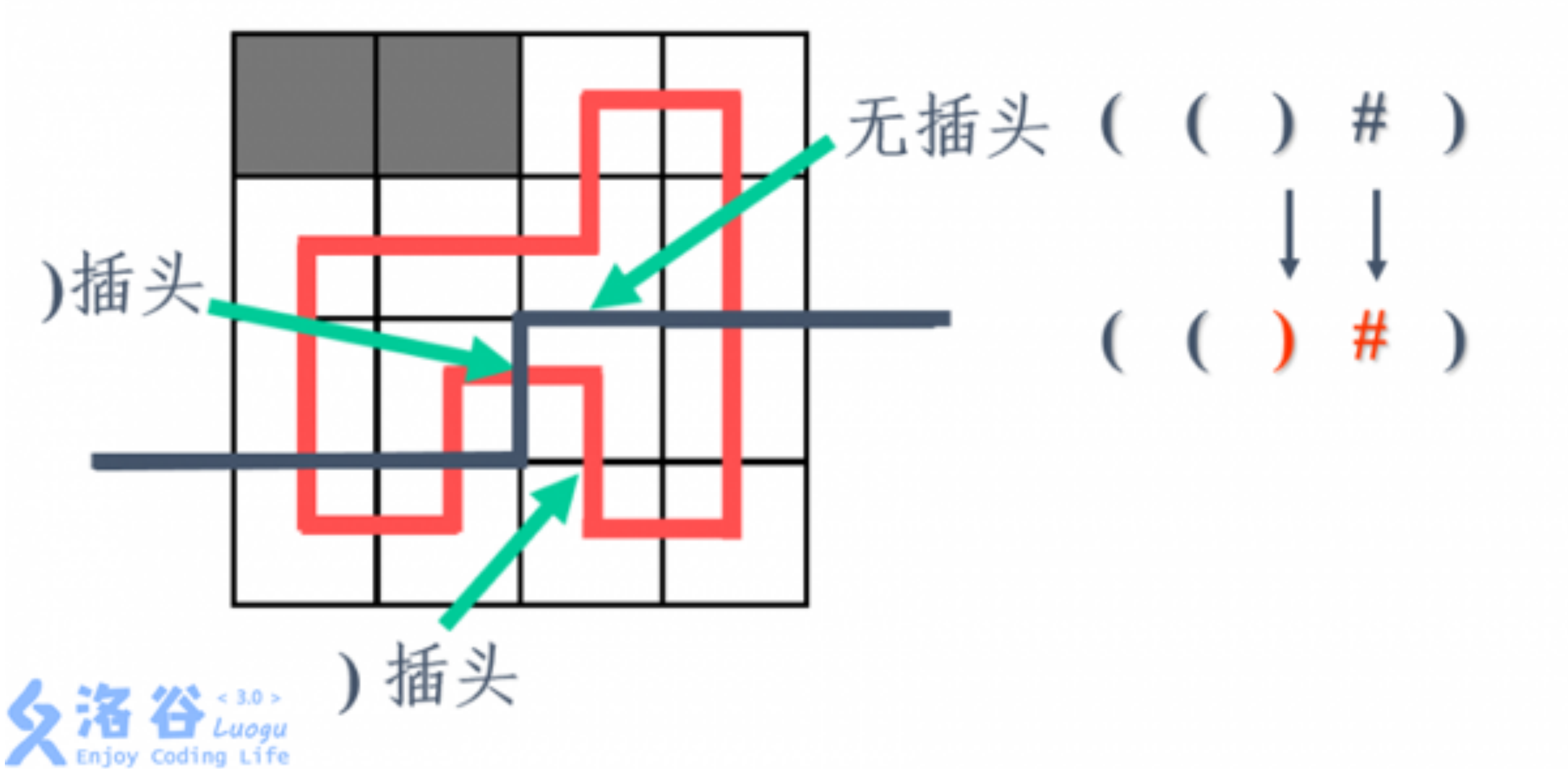
情况2.3:上=) 左=(

情况2:上+左

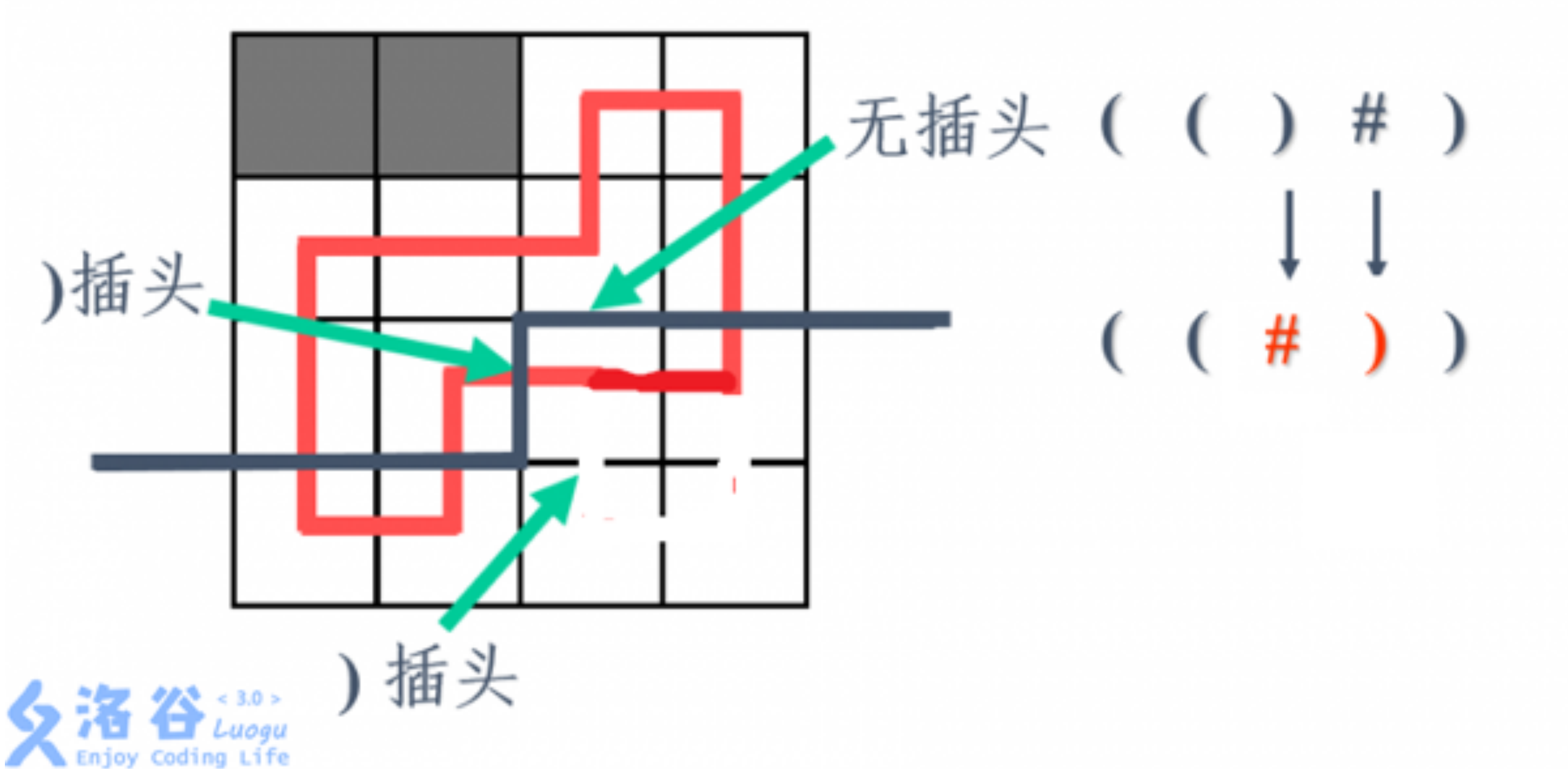


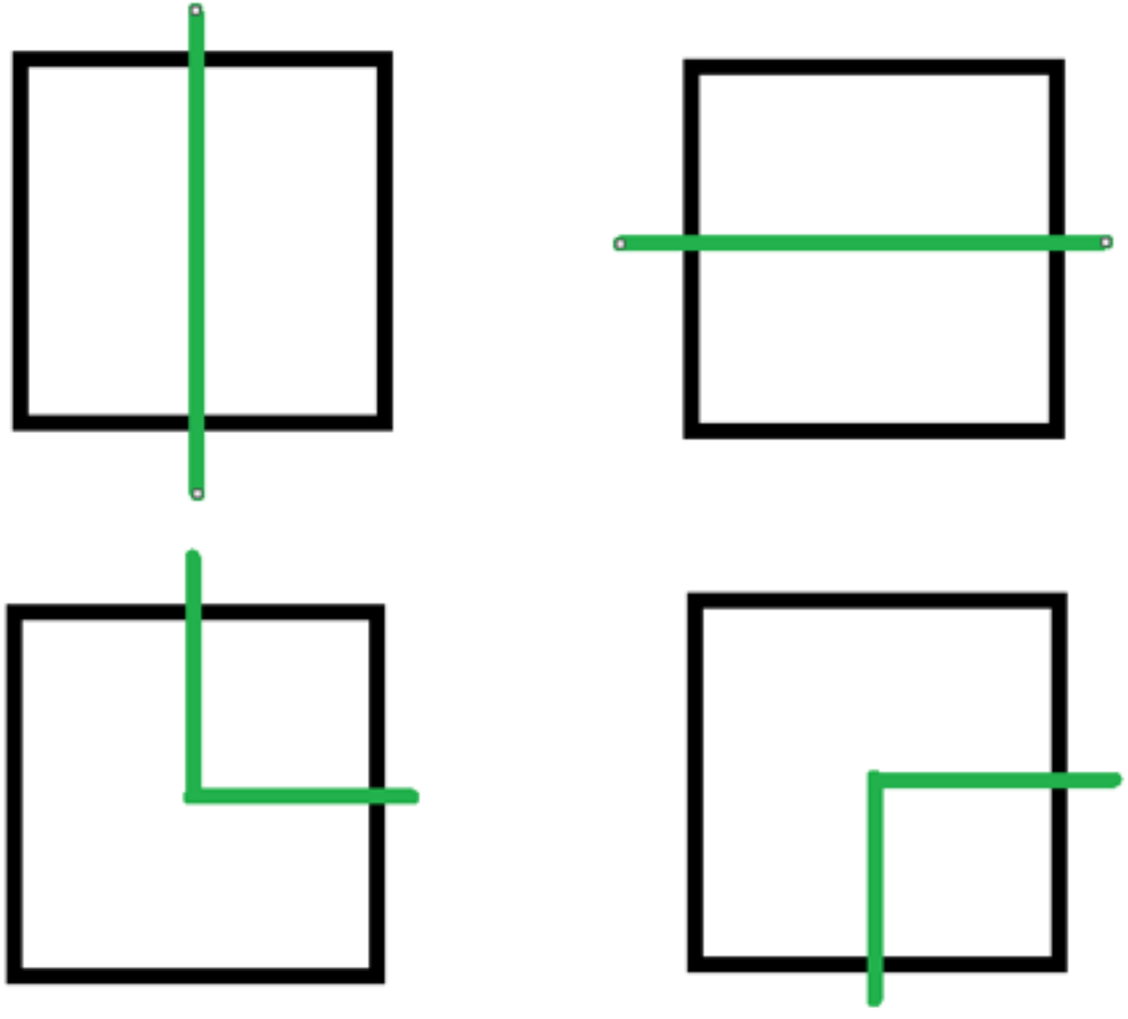
情况2.4:上=) 左=)

情况3:上和左中有一个



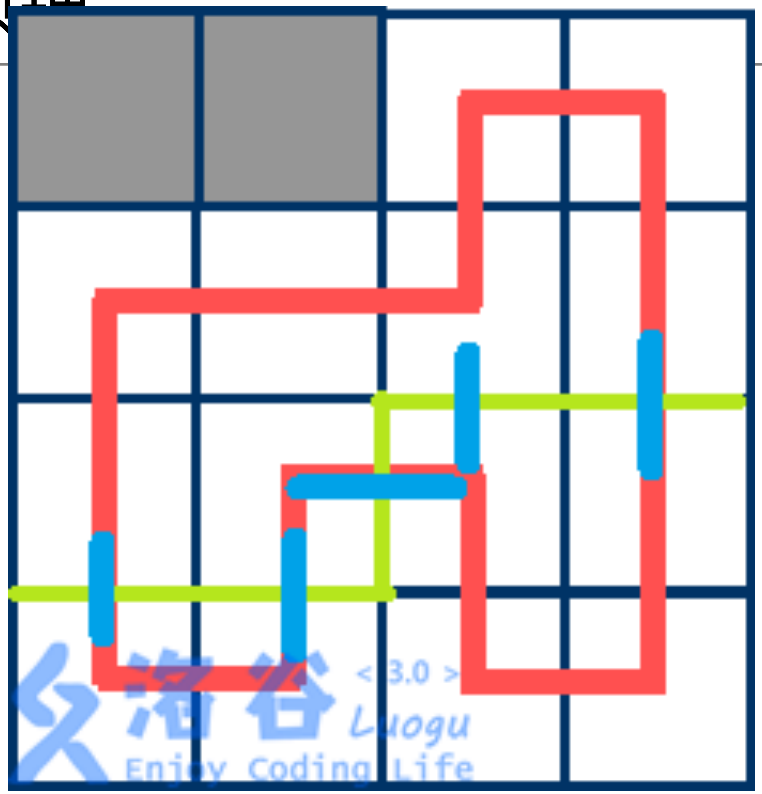
情况3:上和左中有一个





回路计数

给你一个n×m的棋盘(有的格子存在障碍)，求经过所有非障碍格子的哈密顿回路个数。 (n,m<=12)$(n,m<=12)$
*表示障碍 .表示必须铺



技巧：把上一个点可达的所有状态存在a里
转移时可以转移有效状态

```
void work() {
    tot[now]=1,f[now][1]=1,a[now][1]=0;
    for(int i=1;i<=n;++i) {
        for(int j=1;j<=tot[now];++j) a[now][j]<<=2;
        for(int j=1;j<=m;++j) {
            las=now,now^=1;
            memset(h,0,sizeof(h)),tot[now]=0;
            for(int k=1;k<=tot[las];++k) {
                int S=a[las][k],b1=(S>>(j*2-2))&3,b2=(S>>(j*2))&3;
```

S是当前状态　　b1:左插头　　b2:上插头

```
void ins(int S,LL num) {
    int tmp=S%Has+1;
    for(int i=h[tmp];i;i=ne[i])
        if(a[now][i]==S) {f[now][i]+=num;return;}
    ne[++tot[now]]=h[tmp],h[tmp]=tot[now];
    a[now][tot[now]]=S,f[now][tot[now]]=num;
}
```

用hash存储状态
如果状态存在则累加

解读这个函数

```
int S=a[las][k],b1=(S>>(j*2-2))&3,b2=(S>>(j*2))&3;
LL num=f[las][k];
if(!mp[i][j]) {if(!b1&&!b2) ins(S,num);}
else if(!b1&&!b2)
    {if(mp[i+1][j]&&mp[i][j+1]) ins(S+bin[j-1]+2*bin[j],num);}
else if(!b1&&b2) {
    if(mp[i][j+1]) ins(S,num);
    if(mp[i+1][j]) ins(S-bin[j]*b2+bin[j-1]*b2,num);
}
else if(b1&&!b2) {
    if(mp[i][j+1]) ins(S-bin[j-1]*b1+bin[j]*b1,num);
    if(mp[i+1][j]) ins(S,num);
}
else if(b1==1&&b2==1) {
    int kl=1;
    for(int t=j+1;t<=m;++t) {
        if((S>>(t*2))%4==1) ++kl;
        if((S>>(t*2))%4==2) --kl;
        if(!kl) {ins(S-bin[j]-bin[j-1]-bin[t],num);break;}
    }
}
```

不能到达，一定不能有插头

右下插头

增加()

继续写出每个转移的含义

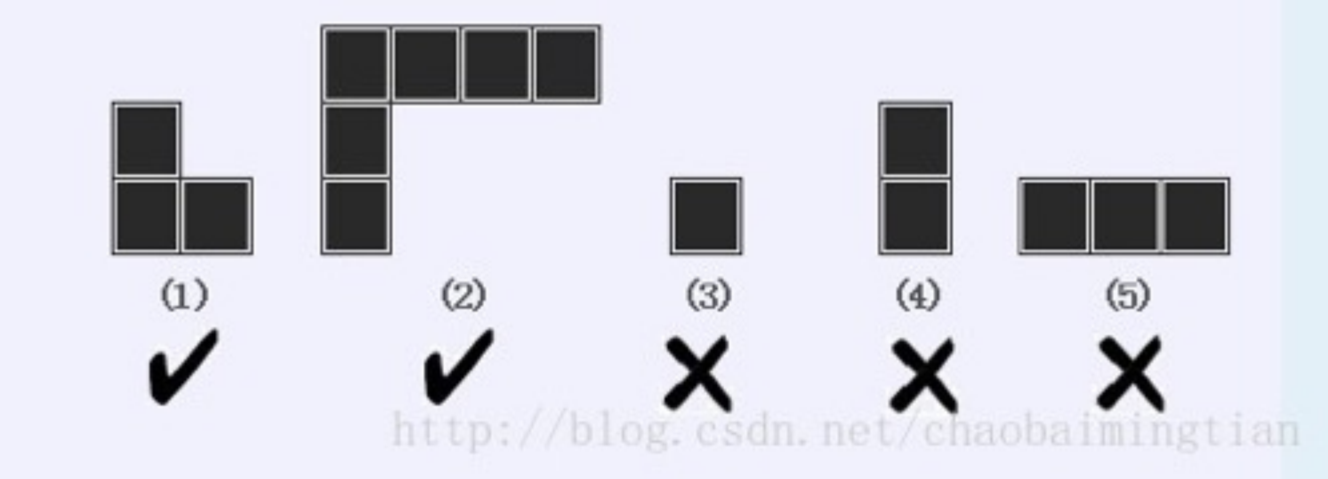
```
else if(b1==2&&b2==2) {
    int kl=1;
    for(int t=j-2;t>=0;--t) {
        if((S>>(t*2))%4==1) --kl;
        if((S>>(t*2))%4==2) ++kl;
        if(!kl) {ins(S+bin[t]-2*bin[j]-2*bin[j-1],num);break;}
    }
}
else if(b1==2&&b2==1) ins(S-2*bin[j-1]-bin[j],num);
else if(i==e1&&j==e2) ans+=num;
```

插头DP能解决哪些问题

单回路覆盖方案数
多回路覆盖方案数
联通块最大权
单回路最大权
最小联通块方案
特殊型覆盖
多种限制的最大权

铺地砖加强版2

给出一个n*m的地面。有些是障碍。用**L型**的地板砖铺满。有多少种方案。(n*m<=100)



0：没有插头
1：插头没拐过弯
2：插头拐过弯

考虑所有插头和转移的情况！

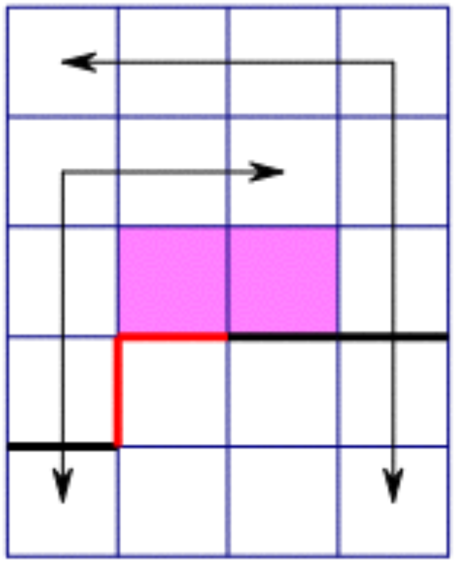


图1-0

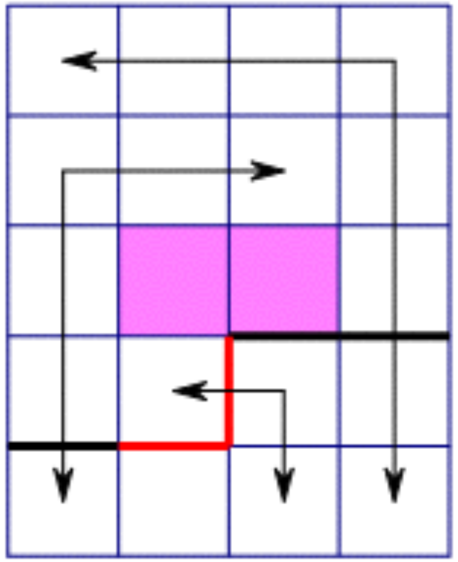


图1-1

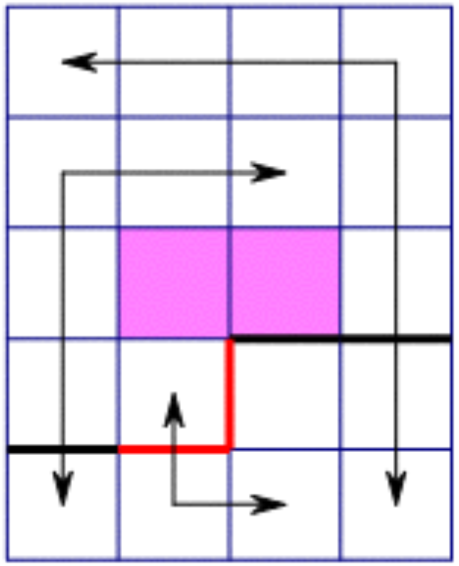


图1-2

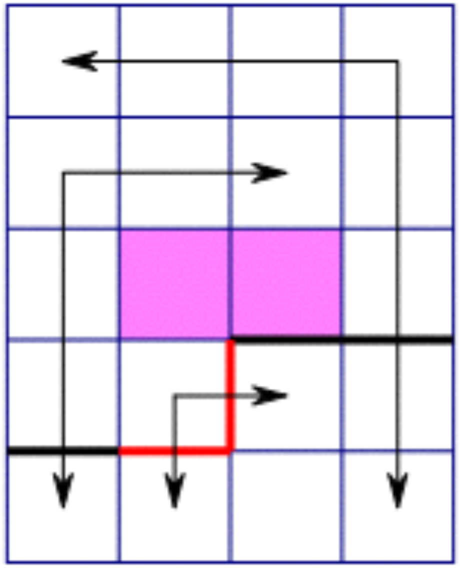


图1-3

1.当前格子上方和左方都没有插头

决策1：新建一个拐角
(2号下插头+2号右插头)

决策2：新建一个左端点
(1号右插头)

决策3：新建一个上端点
(1号下插头)

2.上方有一个一号下插头，左方没有插头

决策1：不拐弯
(1号下插头)

决策2：拐弯
(2号右插头)

3.上方有一个二号下插头，左方没有插头

决策1：继续延伸

（2号下插头）

决策2：停止

（没有插头）

4.左方有一个一号右插头，上方没有插头

同2

5.左方有一个二号右插头，上方没有插头

同3

6.上方和左方都有插头

如果都是1号插头
正好在这个点拐弯

如果不是1号插头：
不可能!

拓展：最小表示法

不用括号表示，还是连通块编号的形式(2,2,3,1,1,3)

最小表示→(1,1,2,3,3,2)
用来判重

猜一猜：这种表示法可以用来解决哪种问题?

最大连通块

在N*N的带权格点图中，找到一个四连通块，使它的权值和最大。(N<=9)

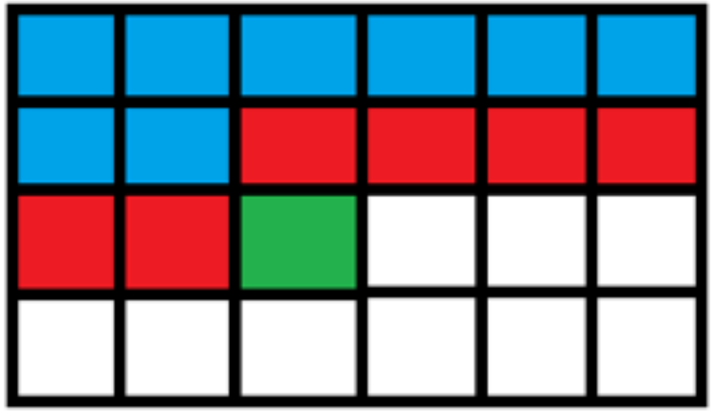
输入样例：

```
4
2 -1 -1 -1
5 -5 -1 -5
3  2 -1  3
2 -2 -3  2
```

输出样例：

```
18
```

```
2 -1 -1 -1
5 -5 -1 -5
3  2 -1  3
2 -2 -3  2
```



状态：轮廓线上连通块编号的最小表示

决策1：不选→考虑上面一格的连通块是否还有其他插头接口

决策2：选→考虑b1和b2

!b1&&!b2形成新块

!b1&&b2||b1&&!b2直接复制块的编号

b1&&b2合并两个块编号

```
void min_express(ll &u)
{
    memset(trans,0,sizeof(trans));
    int t=0;
    for(int i=0;i<n;i++)
        if((u>(3*i))&7){
            int current=1ll*(u>>(3*i))&7;
            if(!trans[current]) trans[current]=++t;
            u-=current<<(3*i);
            u+=trans[current]*1ll<<(3*i);
        }
}
```

总结

回路问题→括号表示法
连通块问题→最小表示法
其他问题→独立插头
三种的方法转移是类似的，只是状态表示不一样

对比轮廓线DP
单点决策+轮廓线转移
区别：
1.轮廓线（线段or格点）
2.轮廓线状态表示