信奥算法

位运算验收测试

根据要求写出位运算表达式

```
if (____) //x是偶数
```

(x&1)==0

x&1^1

保留x右起第0位和第2位，其他位清零

x&=5

```
if (____) //x的二进制中有连续的1
```

x&(x>>1)

```
if (____) //a,b,c的二进制没有任何一位都是1
```

(a&b&c)==0

状压DP

状态压缩	动态规划

一类状态较特(fu)殊(za)的**DP**

DP经典场景

基础场景	高级场景	其他
序列切割	树形DP	DP优化
整数拆分	状压DP	DP+XXX。
背包问题	数位DP	
棋盘格行走	插头DP	
区间DP		

货郎问题（集合状压）

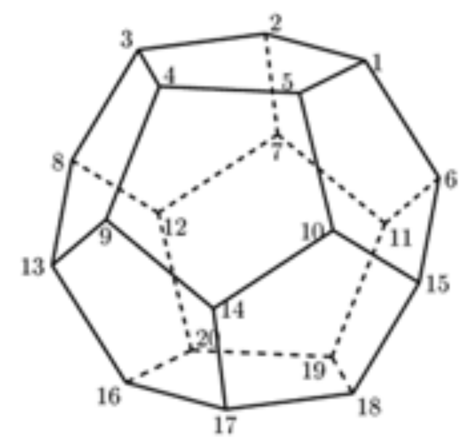
Traveling Salesman Problem (TSP)是一个著名问题

一个旅行商人要拜访n个城市，**每个城市只能拜访一次，并且最后回到出发城市**。已知两两城市间的距离，求总距离的最小值

红色部分也称**Hamilton回路**

故该问题也称为

最短Hamilton回路问题



咋做？暴力呗

枚举所有路径（DFS）
不可重复访问的条件作为可行性剪枝

枚举第i步，目前在x，目前总长=L

```
void dfs(int i,int x,int L){
    if (L+d[x][0]>=ans) return;
    if (i==n+1) ans=L+d[x][0];
    v[x]=1;
    for (int y=1;y<=n;y++)
        if (!v[y]) dfs(i+1,y,L+d[x][y]);
    v[x]=0;
}
```

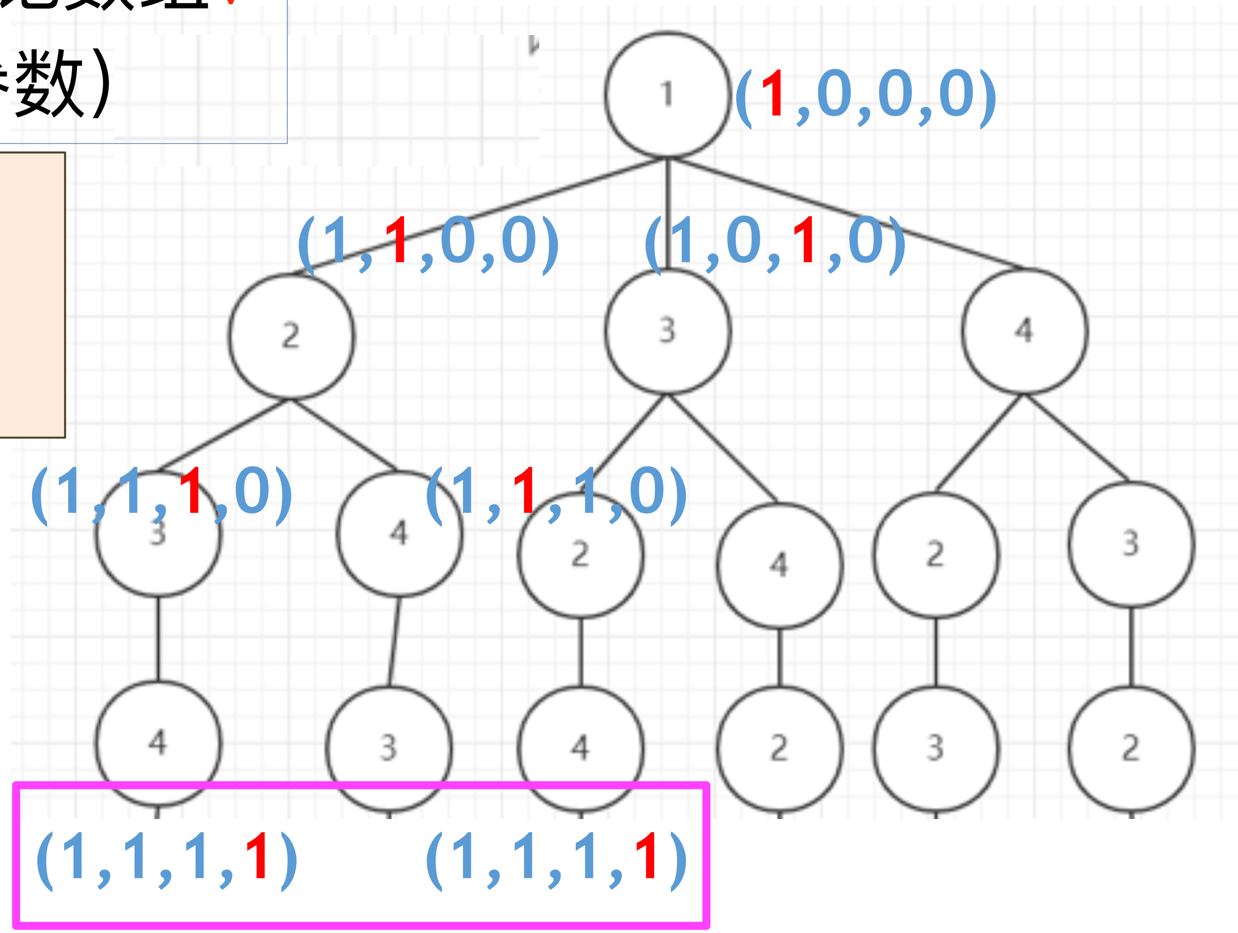
复杂度：O(n!)
其实就是枚举**全排列**啦

分析DFS枚举过程

下一步选择只依赖
当前位置**x**+访问标记数组**v**
(i,L只是辅助参数)

如**x和v一模一样**
则后续枚举完全相同
(没必要重复进行)

记忆化搜索!

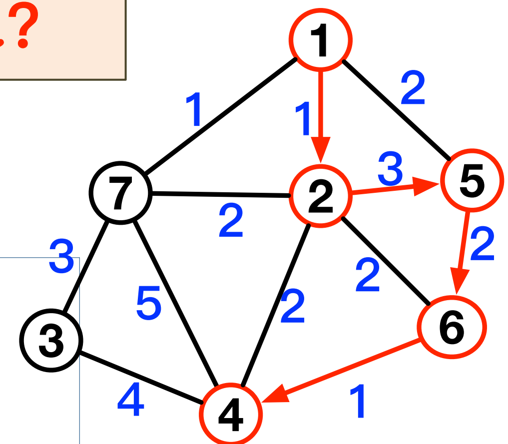


记忆化搜索（DP）

f[x,V]=从1无重复走到x，已访问**点集**为V的最小代价

状态的1个维度**V是个集合...SoWhat?**

Exp. f[4,{1,2,4,5,6}] **=7**



决策： 上一步从y走到x

转移方程：

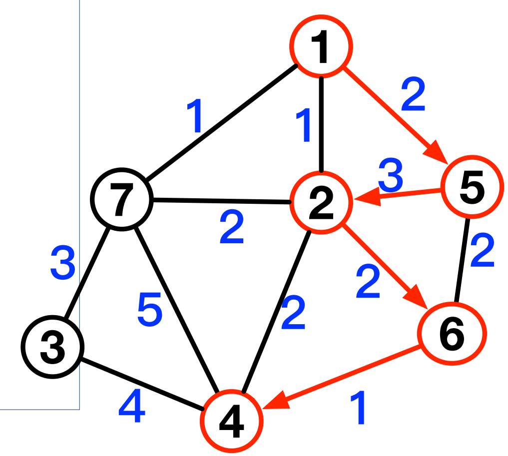
$f[x,V]=\min\{f[y,V-\{x\}]+d[y,x]\}$， $x\in V$

$f[x,V]=INF$， $x\notin V$（非法状态）

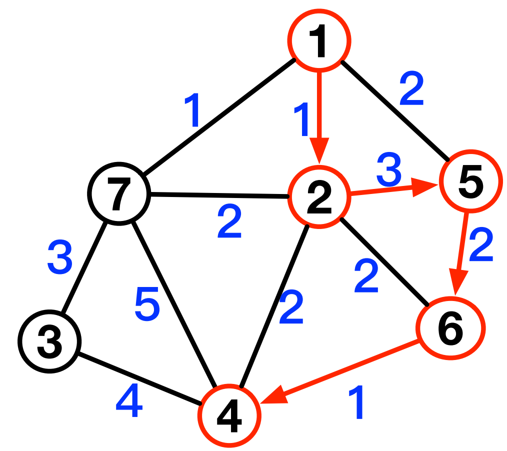
答案： $ans=\min\{ f[x,\{1,2,\ldots n\}]+d[x,1] \}$

计算顺序： |V|递增顺序（1~n）

边界： $f[1,\{1\}]=0$, $f[x,\{\}]=f[2..n,\{1\}]=INF$



压缩存储



语法上，数组不能以集合为下标，咋拌

将V看做**n位二进制数**
第x位表示x是否在V内

Exp.V={1,2,4,5,6} → $(0111011)_2$=59

f[4,{1,2,4,5,6}]=7 → f[4,59]=7

维度v的范围：$(00..0)_2$~$(11..1)_2$即**$0\sim2^n-1$**

多个变量，bool数组
（但不限于bool数组）
以某种规则用1个变量存储
此计数称为**压缩存储**（不限用于DP）

本质上
是一种
散列(Hash)

位运算实现集合运算

现在，将之前的方程等用位运算重写即可

转移方程（j是V的二进制表示）：

$f[x,V]=\min\{f[y,V-\{x\}]+d[y,x]\}$，$x\in V$

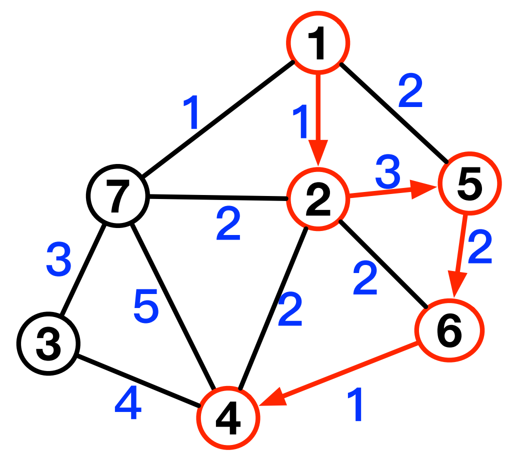
$f[x,j]=\min\{f[y,j\text{\textasciicircum}(1<<(x-1))]+d[y,x]\}$，$j\&(1<<(x-1))>0$

$f[x,V]=INF$，$x\notin V$（非法状态）

$f[x,j]=INF$，$j\&(1<<(x-1))==0$

Exp. $f[4,\{1,2,4,5,6\}]=7$

$f[4,59]=7$



边界： $f[1,\{1\}]=0$，$f[x,\{\}]=f[2..n,\{1\}]=INF$

$f[1,1]=0$，$f[x,0]=f[2..n,1]=INF$

答案： $ans=\min\{\ f[x,\{1,2,\dots n\}]+d[x,1]\ \}$

$ans=\min\{\ f[x,2^n-1]+d[x,1]\ \}$

计算顺序： |V|递增顺序（1~n）

j递增顺序（为什么？）

状态某维度是压缩存储形式
此种DP就叫**状压DP**
其实这定义怪无聊的...

$f[x,j]=\min\{f[y,j\wedge(1<<(x-1))]+d[y,x]\}$, $j\&(1<<(x-1))>0$

$f[x,j]=INF$, $j\&(1<<(x-1))==0$

$f[1,1]=0$, $f[x,0]=f[2..n,1]=INF$

$ans=\min\{ f[x,2^n-1]+d[x,1] \}$

计算顺序：j递增

```
int M=1<<n;
for (int x=1;x<=n;++x) f[x][0]=INF;
for (int x=2;x<=n;++x) f[x][1]=INF;
for (int j=2;j<M;++j)
    for (int x=1;x<=n;++x){
        f[x][j]=INF;
        if (j&(1<<(x-1)))
            for (int y=1;y<=n;++y)
                f[x][j]=min(f[x][j],
                            f[y][j^(1<<(x-1))]+d[y][x]);
    }
for (int x=2;x<=n;++x)
    ans=min(ans,f[x][M-1]+d[x][1]);
```

复杂度：O(n^2*2^n)

友情提示：不要无脑抄，有坑🙃

深度思考：优化在哪里?

后续步骤只与“已访问了哪些点”有关
而与“**以什么顺序**访问这些点”无关

枚举组合

枚举排列

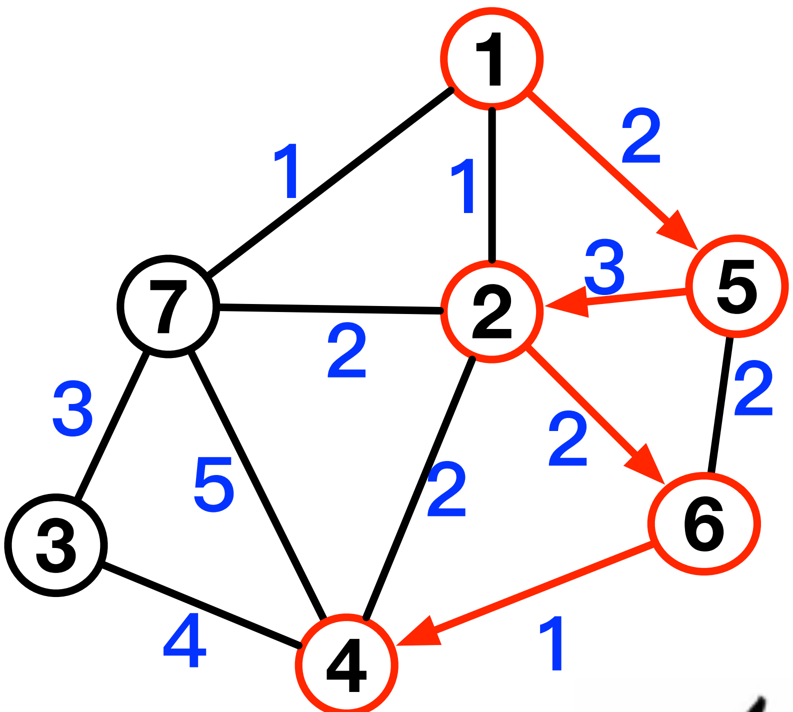
事实上，所有DP本质上都是在做这样的事而已
在暴力枚举过程中，**后续步骤只与之前步骤某些信息有关**
（而不依赖之前步骤的完整方案）

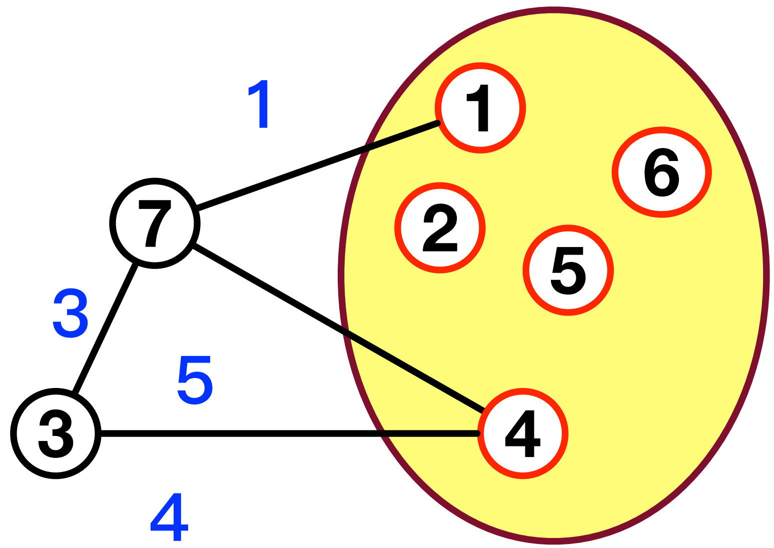
Exp.01背包：
后续步骤只与**还可选哪些物品+剩余空间**有关
与已选哪些物品/顺序无关

DP建模关键在于能否精确“剥离”出影响后续决策的部分信息

演示：从BF到DP

暴力枚举眼中的中间状态	DP眼中的中间状态

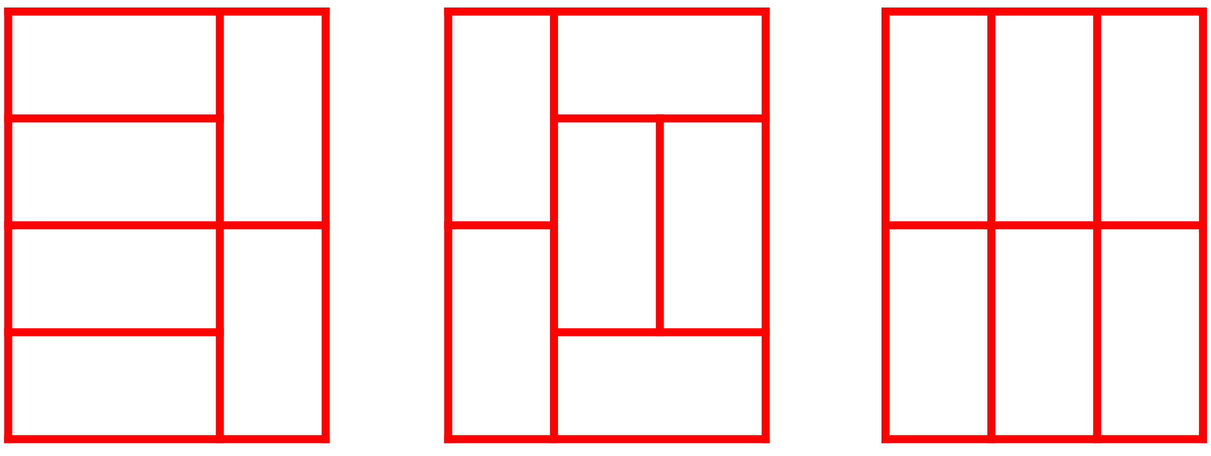




P1101铺地砖（切面状压）

用1*2的地砖铺满n*m的区域，不可重叠，求方案数

n,m<=10



暴力枚举

纯暴力的话，每次放1块地砖

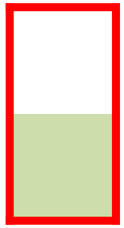
　　复杂度： 约O((nm)!)

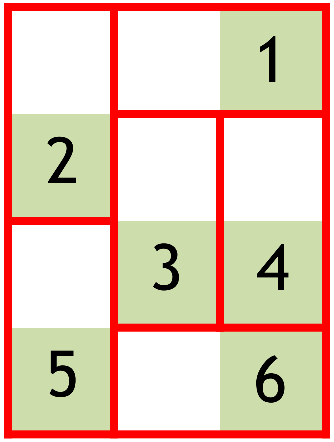
这里必须解决的是**去重问题（方案与放置顺序无关）**

解决方法： 指定一种顺序

　　首先规定地砖以右部/下部的那个为**基准**

　　以基准所在位置（先行后列）从小到规定放置顺序





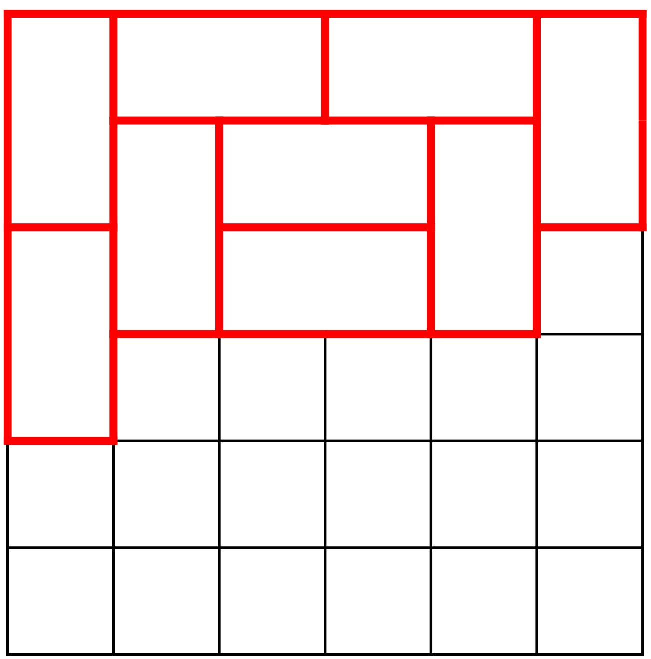
剥离有用信息

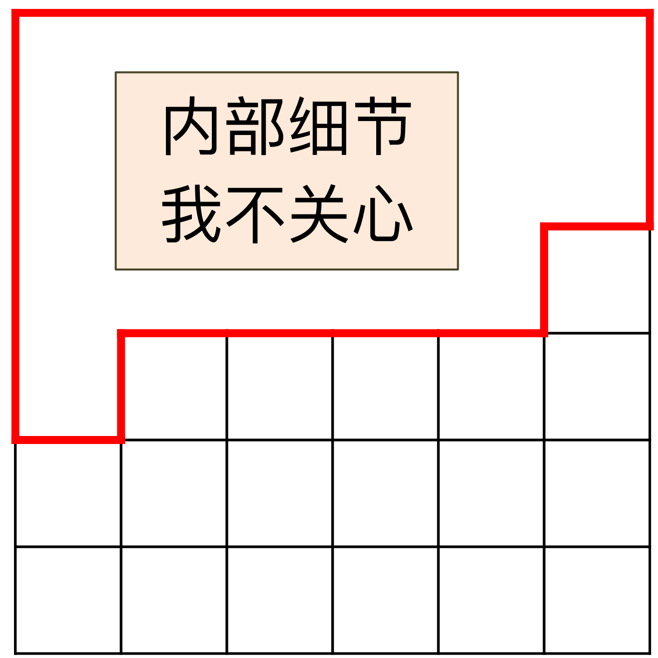
枚举到中途，**什么信息会影响后续步骤?**

答：**已被覆盖区域的“轮廓”**

（与内部具体方案无关）

“剥离”出影响
后续决策的信息



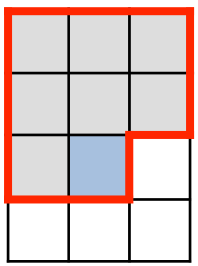
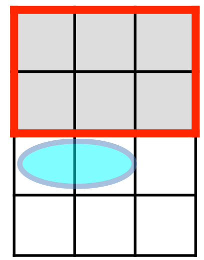
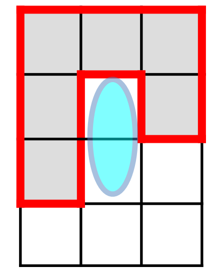


构造DP

状态： f(G)=已覆盖**位置集合G**的方案数

决策： G中**最右下位置**是横着还是竖着盖的

转移方程：

f()=f()+f()

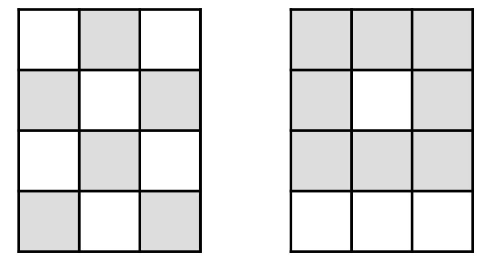
状态的一个维度是个**图形...SoWhat?**

无非是确定**编码如何存这么个图形**而已

最简单的就是n*m位压缩存储表示

复杂度： $O(2^{nm})$

优化（去掉无效状态）



→ **无脑状压的话，有很多无效状态**

→ **分析有效状态的性质（必要条件）**

因“基准”先行后列递增枚举

而一块砖只占1*2格

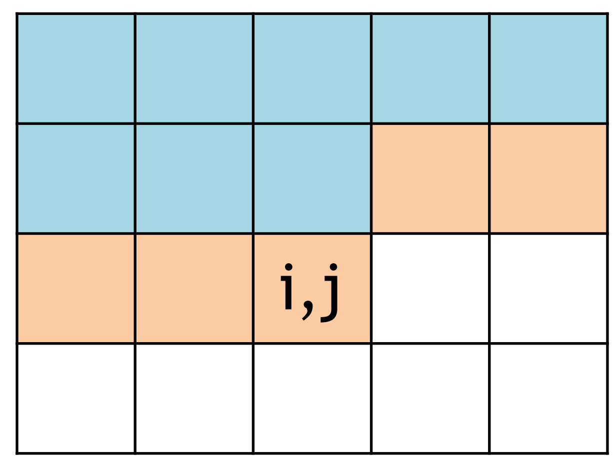
如**基准已经枚举到(i,j)**

则**与基准相差超过1行的位置**必须填满

→ **描述有效状态不需要2^{nm}**

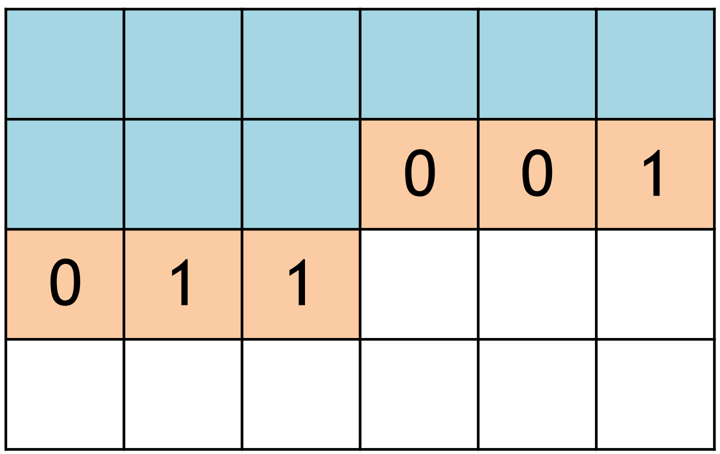
只需i,j和**这一行**填充情况即可

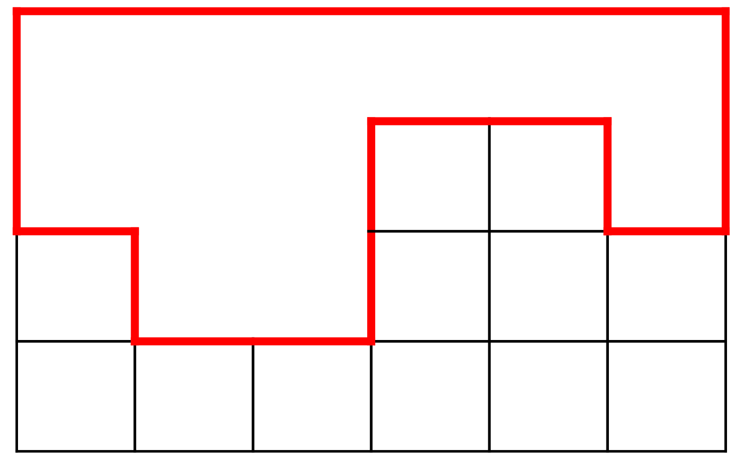
该行我们称其为**“切面”**



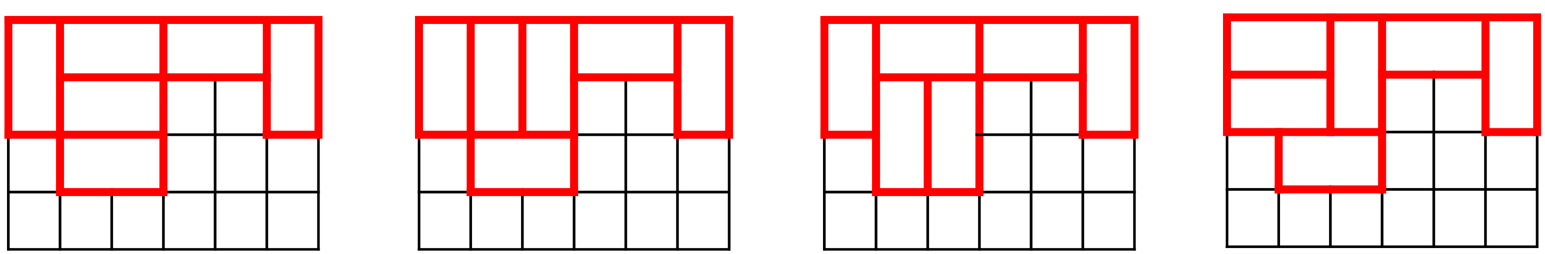
状态优化

状态： $f(i,j,S)$=已枚举到基准(i,j)
切面覆盖状态为S的方案数





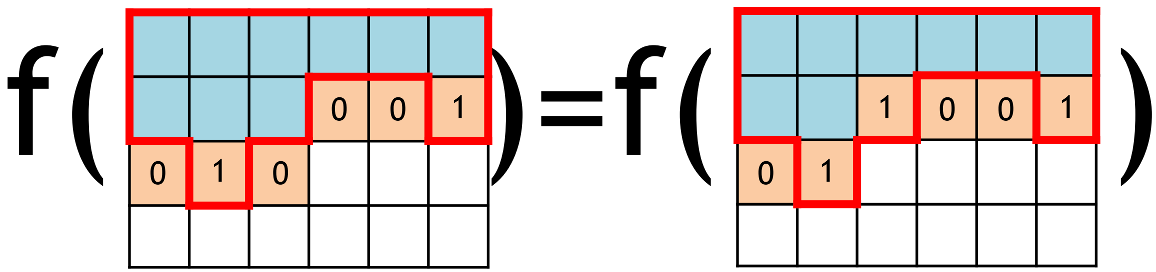
如上状态为：$f(3,3,(011001)_2)=f(3,3,25)=4$



决策/转移方程

决策：(i,j)位置如何填充

情况1：(i,j)位置值为0（未填充）

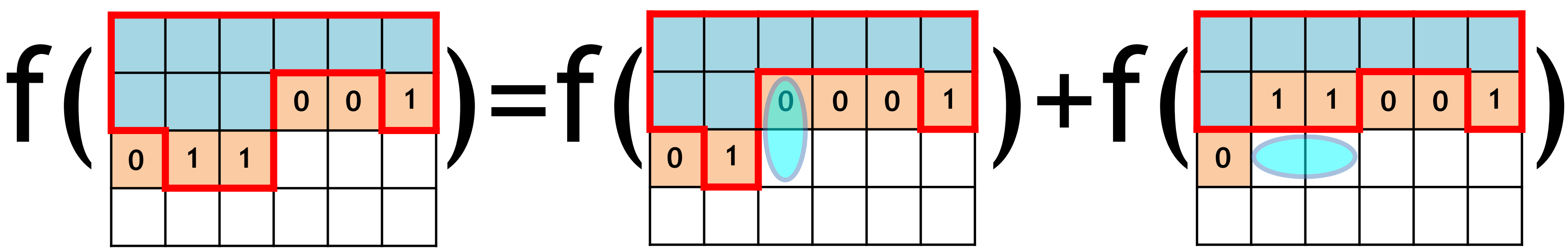


$f(3,3,(01\color{red}{0}001)_2)=f(3,2,(01\color{red}{1}001)_2)$

此项要求

(i,j-1)位置也为1

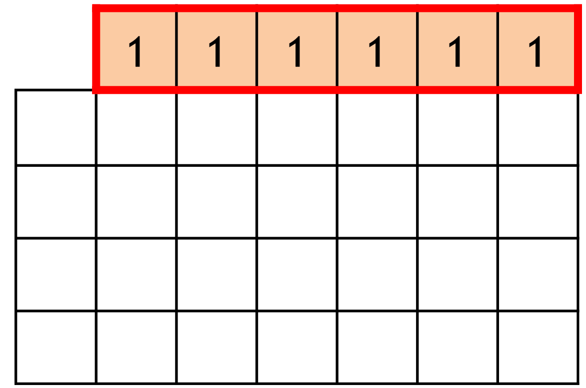
情况2：(i,j)位置为1（已填充），横向/纵向2种情况



$f(3,3,(0\color{red}{11}001)_2)=f(3,2,(01\color{red}{0}001)_2)+f(3,1,(0\color{red}{11}001)_2)$

边界/答案

第0行缓冲行



起点： $f[0,m,2^n-1]=1$

加缓冲行为了代码比较统一

写$f[1,1,2^{n-1}-1]=1$也可

换行处理（新起一行时）

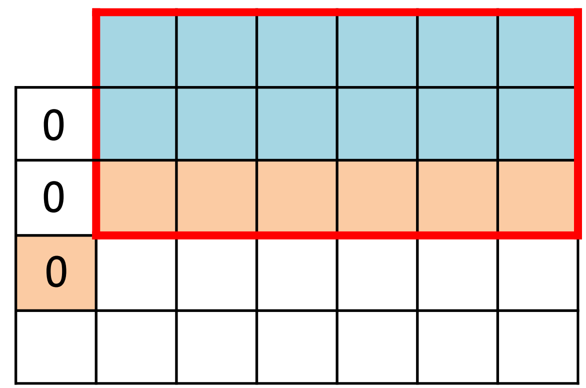
$f[i,0,S]$和$f[i-1,m,S]$其实是一回事

可视为i-1行处理了m列

也可视为i行处理了0列

加缓冲列的好处见后

第0列缓冲列

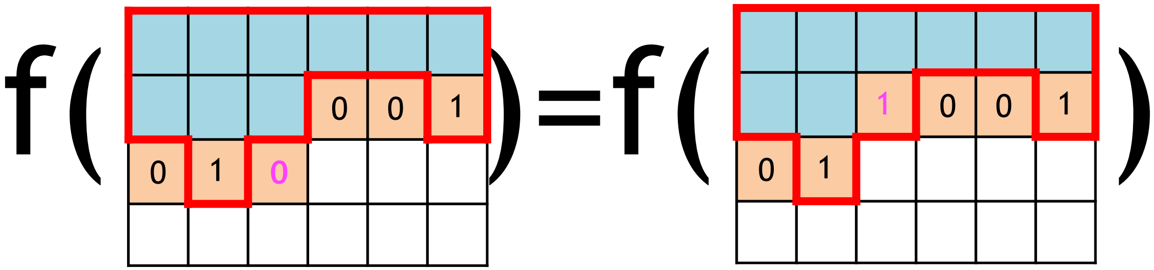


答案（终点）： $ans=f[n,m,2^m-1]$

使用位运算具体实现

情况1：(i,j)位置为0（未填充）

S&(1<<(m-j))==0



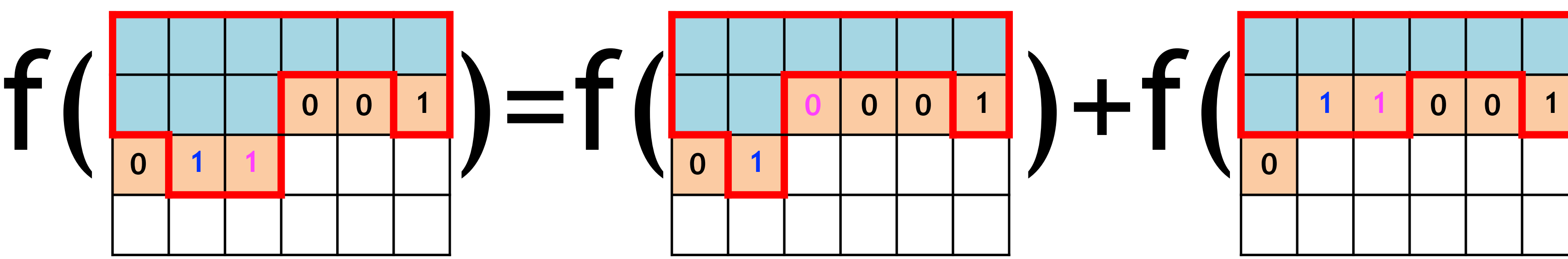
此项必须

(i,j-1)位置也为1

f(i,j,S)=f(i,j-1,S|(1<<(m-j))

S&(1<<(m-j+1))>0

情况2：(i,j)位置为1（已填充），横向/纵向2种情况



f(i,j,S) = f(i,j-1,S^(1<<(m-j)) + f(i,j-2,S)

万事俱备

```
p=(1<<m)-1;
f[0][m][p]=1;
for (ll i=1;i<=n;i++){
    for (ll S=0;S<=p;S++)
        f[i][0][S]=f[i-1][m][S];
    for (ll j=1;j<=m;j++){
        for (ll S=0;S<=p;S++){
            ll msk=1<<(m-j);
            if (S&msk){
                f[i][j][S]=f[i][j-1][S^msk];
                if (S&(msk<<1))
                    f[i][j][S]+=f[i][j-2][S];
            } else {
                f[i][j][S]=f[i][j-1][S|msk];
            }
    //      cout<<"f["<<i<<","<<j<<","
    //      <<toB(S)<<"]="<<f[i][j][S]<<endl;
        }
    }
}
cout<<f[n][m][p]<<endl;
```

起点

换行

(i,j)为1

缓冲列可使j-1不用判越界

(i,j-1)也为1

(i,j)为0

思考题：为何j-2也不用判越界

终点

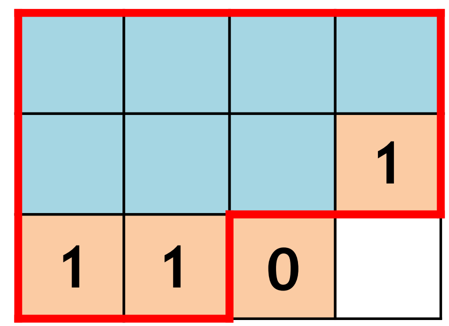
打印中间变量查错

```
string toB(ll S,string ret=""){
    for (int i=0;i<m;++i,S>>=1)
        ret=(S&1 ?'1':'0')+ret;
    return ret;
}
```

整数转定长二进制串

```
f[3,1,0000]=0
f[3,1,0001]=1
f[3,1,0010]=0
f[3,1,0011]=0
f[3,1,0100]=2
f[3,1,0101]=0
f[3,1,0110]=0
f[3,1,0111]=5
f[3,1,1000]=1
f[3,1,1001]=0
f[3,1,1010]=0
f[3,1,1011]=2
f[3,1,1100]=0
f[3,1,1101]=0
f[3,1,1110]=1
f[3,1,1111]=0
```

```
f[3,3,0000]=0
f[3,3,0001]=5
f[3,3,0010]=2
f[3,3,0011]=0
f[3,3,0100]=0
f[3,3,0101]=0
f[3,3,0110]=0
f[3,3,0111]=6
f[3,3,1000]=1
f[3,3,1001]=0
f[3,3,1010]=0
f[3,3,1011]=0
f[3,3,1100]=0
f[3,3,1101]=7
f[3,3,1110]=4
f[3,3,1111]=0
```



状态定义精准
每个都能手算验证

总结（哲学）

1.状压DP就是一种特殊的DP，没什么可怕的

除了有的维度压缩存储了，其他跟普通DP一样

基础的状压DP并不难，只是比较繁

2.从暴力枚举到DP建模（难得糊涂）

剥离出影响后续步骤的信息作为状态维度

其他信息作为状态的值

复杂对象以一定规则(Hash)存成整数（人有多大胆）

3.“状压”的概念不仅限于（二进制）压缩存储

凡多个维度存储在一个下标，都可视为压缩存储

具体也不仅限于集合、切面这两个场景

作业

1101铺地砖

1769排兵布阵1

拓展题 1102送外卖

TSP变种

拓展题 1759排兵布阵2

1769变种

瞻仰题

1103炮兵阵地（NOI2001D2T1）

471宝藏（NOIP2017D2T2）

828填数游戏（NOIP2018D2T2）