状压DP

状态压缩，是利用二进制数的性质对问题进行优化的一种算法，经常与搜索和DP结合。

状态压缩动态规划，俗称状压DP。在动态转移的过程中用二进制数表示状态，并利用位运算的性质可使得枚举方案大大减少。

前置知识

位运算基础

判断数字x二进制下第i位是不是1。

```
if(((1<<(i-1))&x)>0)
```

将数字x二进制下第i位改成1

```
x=x|(1<<(i-1)) //改成0的情况改成x=x&~(1<<(i-1))
```

将数字x最右边的1去掉

```
x=x&(x-1) //也是我们熟知的lowbit
```

数字x的二进制下1的个数

```
int count(int x){int res=0;while(x){res++;x&=(x-1);}return res;}
```

图论应用

状压DP+图论状态转移举例

```
for(int S=0;S<(1<<n);S++) //可以理解为n个点的选或不选
    for(int i=0;i<n;i++) if(S&(1<<i))
        for(int j=0;j<n;j++) //转移点
            if(!(S&(1<<j))&&G[i][j]) //j不在点集中且i到j有路径
                dp[j][S|(1<<i)]=min(dp[j][S|(1<<i)],dp[i][S]+G[i][j]);//S表示点集，j是转移结点，G[i][j]是转移代价
```

生成子集

```
for(int subs=s&(s-1);subs;subs=s&(subs-1)) //子集，可生成子图
//subs=(subs-1)&s 可以枚举s的子集
```

有兴趣的同学可以搜一下斯坦纳树。

例题

P1896 [SCOI2005]互不侵犯

模板题：n*n的格子里面放k个国王，国王附近八个格子内没有国王，可以有多少种方案?

```
#include<bits/stdc++.h>
#define inf 0x3f3f3f3f
#define int long long
using namespace std;
const int N=10;
const int mod=1e9+7;

int count(int x){int res=0;while(x){res++;x&=(x-1);}return res;}
int lb(int x){return x&(-x);}

int n,K,ans=0,dp[N][(1<<10)][N*N];

signed main(){
    ios::sync_with_stdio(0);
    cin.tie(0);cout.tie(0);
    cin>>n>>K;
    for(int j=0;j<(1<<n);j++){
        if(j&(j<<1)) continue;
        dp[1][j][count(j)]=1;
    }
    for(int i=2;i<=n;i++){
        for(int j=0;j<(1<<n);j++){ //枚举子集
            if(j&(j<<1)) continue; //左右互不侵犯
            for(int k=0;k<(1<<n);k++){ //前一行的状态
                if(k&(k<<1)) continue;
                if(!((j&k)||(j&(k<<1))||((j<<1)&k))){ //当前状态满足和上一行的状态互不侵犯
                    int num=count(j);//,num2=count(k);
                    for(int p=0;p<=K;p++){
                        dp[i][j][p+num]+=dp[i-1][k][p];
```

```
                    }
                }
            }
        }
    }
    for(int j=0;j<(1<<n);j++){
        if(j&(j<<1)) continue;
        ans+=dp[n][j][K];
    }
    cout<<ans<<"\n";
    return 0;
}
```

luoguP3694 邦邦的大合唱站队

给定n个人m种颜色排成一列，可以抽出k个人将他们以任意顺序插回，问最少k的最小值使得每种颜色的人站在一起。

状态转移方程$f[S]=min(f[S\ xor\ (1<<j)]+num[j]-s[r][j]+s[l][j])$

其中l,r表示最后一个乐队所对应的区间。

```
#include<bits/stdc++.h>
#define inf 0x3f3f3f3f
#define int long long
using namespace std;
const int N=1e5+7;

int n,m,a[N],num[25],sm[1<<21];
int sum[N][22],dp[1<<21];

void dfs(int x,int s,int b){ //dp初始状态--每种状态中的颜色顺序排好的花费
    if(x==m) return;
    if(b) sm[s|(1<<x)]=sm[s]+num[x+1],dfs(x+1,s|(1<<x),1),dfs(x+1,s|(1<<x),0);
    else dfs(x+1,s,1),dfs(x+1,s,0);
}

signed main(){
    ios::sync_with_stdio(0);
    cin.tie(0);cout.tie(0);
    cin>>n>>m;
    for(int i=1;i<=n;i++){
        cin>>a[i];
        for(int j=1;j<=m;j++) sum[i][j]=sum[i-1][j];
        sum[i][a[i]]++; //每种颜色的前缀和
        num[a[i]]++;
    }
    dfs(0,0,0);dfs(0,0,1);
    memset(dp,inf,sizeof(dp));
    dp[0]=0;
    for(int i=1;i<(1<<m);i++){ //枚举状态
        for(int j=1;j<=m;j++){ //枚举颜色
            if(i&(1<<(j-1))){
                int l=sm[i^(1<<j-1)],r=sm[i];
                dp[i]=min(dp[i],dp[i^(1<<(j-1))]+(r-l)-(sum[r][j]-sum[l][j]));
```

```
            }
        }
    }
    cout<<dp[(1<<m)-1]<<"\n";
    return 0;
}
```

[NOI2001] 炮兵阵地

题意：给定N×M的格子，有些地方不能放炮，炮之间不能相互攻击到，问最多能放多少个炮。（炮的攻击范围是上下左右两格以及自己所在格子）。

dp[i][j][k]表示两行状态分别为i和j时的答案，然后第三维用来滚动就ok了。

```
#include<bits/stdc++.h>
using namespace std;
const int N=1<<10;

int n,m,mp[105],f[N][N][2],sum[N],ans=0; //f[i][j]表示前一行是状态i,后一行是状态j的答案，第三位用来滚动
bool g[N];
char ch;

int getsum(int x){  //统计1的个数
    int res=0;
    while(x){
        res+=(x&1);
        x>>=1;
    }
    return res;
}

signed main(){
    cin>>n>>m;
    int M=(1<<m);
    for(int i=0;i<n;i++){
        for(int j=0;j<m;j++){
            ch=getchar();
            while(ch!='P'&&ch!='H') ch=getchar();
            if(ch=='H') mp[i]=mp[i]<<1|1;
            else mp[i]=mp[i]<<1;
        }
    }
    for(int i=0;i<M;i++) sum[i]=getsum(i);  //预处理
    for(int i=0;i<M;i++){
        for(int j=0;j<M;j++){  //枚举前两行的状态和答案
            if((i&j)||(i&mp[0])||(j&mp[1])||(i&(i<<1))||(j&(j<<1))||(i&(i<<2))||(j&(j<<2))) continue;
            f[i][j][1]=sum[i]+sum[j];
        }
    }
    for(int i=2;i<n;i++){  //枚举行
        for(int j=0;j<M;j++){  //枚举第i-1行状态
            if((j&mp[i-1])||(j&(j<<1))||(j&(j<<2))) continue;
            for(int k=0;k<M;k++){  //枚举第i行状态
```

```
                if((k&(k<<1))||(k&(k<<2))||(k&mp[i])||(k&j)) continue;
                for(int v=0;v<M;v++){  //枚举第i-2行状态
                    if(v&(v<<1)||(v&(v<<2))||(v&j)||(v&k)||(v&mp[i-2]))
continue;
                    f[j][k][i%2]=max(f[j][k][i%2],f[v][j][(i-1)%2]+sum[k]);
                }
            }
        }
    }
    for(int i=0;i<M;i++){
        for(int j=0;j<M;j++){
            ans=max(ans,f[i][j][(n-1)%2]);  //取最大值
        }
    }
    cout<<ans<<"\n";
    return 0;
}
```

[SCOI2008]奖励关

有n种宝物，k关游戏，每关游戏随机给出一种宝物，可捡可不捡。每种宝物有一个价值（有负数）。每个宝物有前提宝物列表，必须在前面的关卡取得列表宝物才能捡起这个宝物，求期望收益。

```
#include<bits/stdc++.h>
using namespace std;
const int N=105;

int k,n,v[N],pre[N];
double dp[N][1<<20];

signed main(){
    scanf("%d%d",&k,&n);
    for(int i=1,x;i<=n;++i){
        scanf("%d%d",&v[i],&x);
        while(x) pre[i]|=(1<<(x-1)),scanf("%d",&x);
    }
    for(int i=k;i>0;--i){
        for(int s=0;s<(1<<n);++s){
            for(int w=1;w<=n;++w){
                if((s&pre[w])==pre[w]) dp[i][s]+=max(dp[i+1][s],dp[i+1][s|
(1<<(w-1))]+v[w]);
                else dp[i][s]+=dp[i+1][s];
            }
            dp[i][s]/=n;
        }
    }
    printf("%.6lf\n",dp[1][0]);
    return 0;
}
```

参考文献

https://www.bilibili.com/video/BV1Z4411x7Kw

https://www.cnblogs.com/ljy-endl/p/11627018.html