信奥算法

位运算

bitwise operation

输入正整数p,p<=10^{18}
输出p的二进制里哪些位上有1

枚举每一位
再判断

```
5   ll p;
6   cin>>p;
7   for(ll i=0;i<=62;i++)
8       if(                )
9           cout<<i<<" ";
```

快速访问
有1的位置

```
5   map<ll,ll> LOG2;
6   for(ll i=0;i<=62;++i)LOG2[1LL<<i]=i;
7   ll p;
8   cin>>p;
9   for(ll s=p;s;s&=s-1){
10      ll LSB_s=s&(-s);
11      ll i=LOG2[LSB_s];
12      cout<<i<<" ";
13  }
```

删末尾1

取末尾1

旅行商人问题

Traveling Salesman Problem

中国邮递员问题

哈密尔顿回路

TSP

共n个地点,编号1到n。其中i号到j号有道路,长度为w(i,j)。有个人希望从1号地点开始,遍历其他所有地点**恰好1次**，最终再次回到1号地点。求最短路程是多少?

以上问题难度较大,被分类为NPC问题
时间复杂度为n的多项式的解答未知
n较小时才能较快求解

Non-deterministic Polynomial Complete problem

现场挑战
快快编程1102

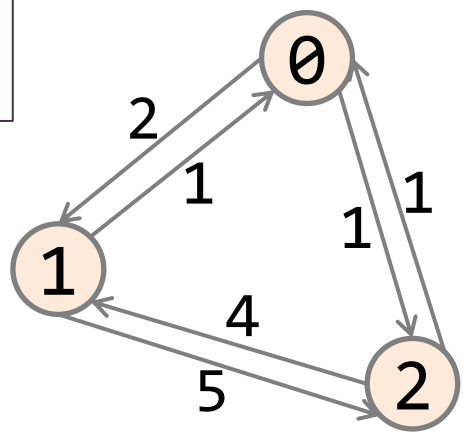
请写出第1102题和TSP的相同点和不同点

第1102题和TSP都需要访问所有节点

TSP里，除了起点以外，每个节点恰好访问1次

第1102题里，每个节点都可以多次访问

起点
0号



TSP: 0->2->1->0, 总费用6

1102: 0->2->0->1->0, 总费用5

解法思路

预先计算	两两地点间最短路 用Floyd-Warshall算法
方法1	暴力枚举全排列
方法2	状压DP

方法1 暴力枚举全排列

```
16      cin>>n;
17      for(ll i=0;i<=n;++i)
18          for(ll j=0;j<=n;++j)
19              cin>>d[i][j];
20      for(ll k=0;k<=n;++k)
21          for(ll i=0;i<=n;++i)
22              for(ll j=0;j<=n;++j)
23                  d[i][j]=min(d[i][j],d[i][k]+d[k][j]);
```

d[i][j]最终值表示i号地点到j号地点的最短路

枚举全排列

总复杂度
$O(n^3+n!*n)$

```
ll dist(){
    ll res=
    for(ll i=1;i<=n;++i)
        res+=d[p[i-1]][p[i]];
    return res;
}
```

```
    for(ll i=1;i<=n;++i)p[i]=i;
    ll ans=INF;
    do{
        ans=min(ans,dist());
    }while(next_permutation(p+1,p+1+n));
    cout<<ans<<endl;
```

方法2 状压DP

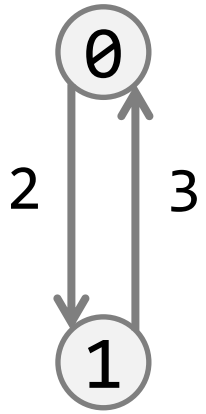
f[i][p]恰首次到达i号
且恰去过p模式里各地点的最少时间

递推的阶段性：
已访问1个地点
已访问2个地点
……
已访问n个地点

每个新阶段都会首次访问某个新地点

状态定义后该干什么？

```
/*
f[i][p]恰首次到达i号
且恰去过p模式里各地点的最少时间
     p= 00, 01, 10, 11
i=0    INF   0   INF INF
i=1    INF INF  INF   2
*/
```



$f[0][(00)_2]$	首次到0号且没到过任何地点	不可能
$f[0][(01)_2]$	首次到0号且恰到过0号节点没到过1号	原地不动
$f[0][(10)_2]$	首次到0号且恰到过1号节点没到过0号	不可能
$f[0][(11)_2]$	首次到0号且恰到过1号和0号	不可能

$f[1][(00)_2]$	首次到1号且没到过任何地点	不可能
$f[1][(01)_2]$	首次到1号且恰到过0号节点没到过1号	不可能
$f[1][(10)_2]$	首次到1号且恰到过1号节点没到过0号	不可能
$f[1][(11)_2]$	首次到1号且恰到过1号和0号	2

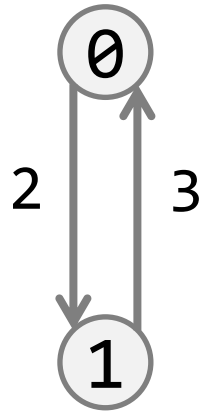
手算表格

输入

```
1
0 2
3 0
```

输出

```
5
```



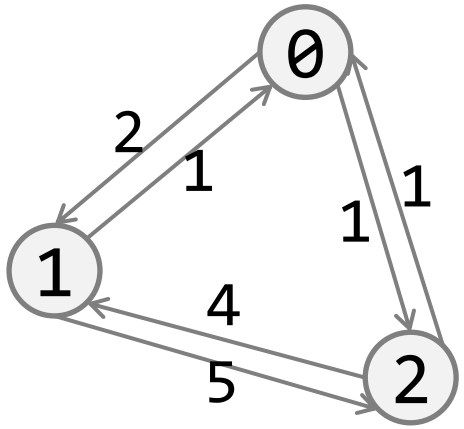
二进制p	00	01	10	11
十进制p	0	1	2	3
f[0][p]	INF	0	INF	INF
f[1][p]	INF	INF	INF	2

f[1][11]	依赖	f[0][01]

Floyd-Warshall预计算

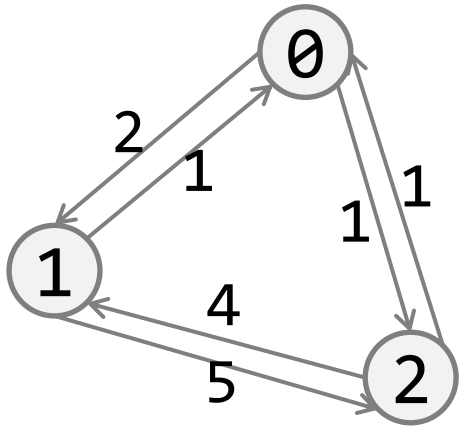
输入

```
2
0 2 1
1 0 5
1 4 0
```



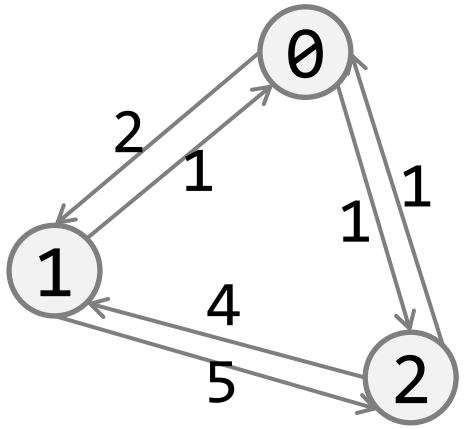
d[i][j]	j=0	j=1	j=2
i=0	0	2	1
i=1	1	0	2
i=2	1	3	0

f[2][111]	依赖	f[0][011]
		f[1][011]



二进制p	000	001	010	011	100	101	110	111
十进制p	0	1	2	3	4	5	6	7
f[0][p]	INF	0	INF	INF	INF	INF	INF	INF
f[1][p]	INF	INF	INF	2	INF	INF	INF	4
f[2][p]	INF	INF	INF	INF	INF	1	INF	4

f[1][111]	依赖	f[0][101]
		f[2][101]



二进制p	000	001	010	011	100	101	110	111
十进制p	0	1	2	3	4	5	6	7
f[0][p]	INF	0	INF	INF	INF	INF	INF	INF
f[1][p]	INF	INF	INF	2	INF	INF	INF	4
f[2][p]	INF	INF	INF	INF	INF	1	INF	4

每格依赖左侧格子(二进制较小的状态)

请选择填表顺序

一列一列,从左往右

二进制p	000	001	010	011	100	101	110	111
十进制p	0	1	2	3	4	5	6	7
f[0][p]	INF	0	INF	INF	INF	INF	INF	INF
f[1][p]	INF	INF	INF	2	INF	INF	INF	4
f[2][p]	INF	INF	INF	INF	INF	1	INF	4

f[i][p]恰首次到达i号
且恰去过p模式里各地点的最少时间

```
25    ll nPtn=(1LL<<(n+1));


36    ll ans=INF;
37    for(int i=1;i<=n;++i)
38        ans=min(ans,                              );
```

注意d[i][0]是
预计算的最短路

f[i][p]恰首次到达i号
且恰去过p模式里各地点的最少时间

```
25      ll nPtn=(1LL<<(n+1));
26      for(ll p=0;p<nPtn;++p)
27          for(ll i=0;i<=n;++i)
28              f[i][p]=INF;
29      f[0][1]=0;
30      for(ll p=1;p<nPtn;++p)
31          for(ll i=1;i<=n;++i)if((1LL<<i)&p){
32              ll q=(1LL<<i)^p;
33              for(ll j=0;j<=n;++j)if(          )
34                  f[i][p]=min(f[i][p],                    );
35          }
```

只枚举满足条件的i,j

复杂度?
$O(2^{n+1}*n^2)$
能否加速?

注意d[j][i]是
预计算的最短路

加速枚举

手算:体验枚举量的区别

$p=(1010)_2$

朴素枚举		
	i=0	j=0,1,2,3
	i=1	j=0,1,2,3
	i=2	j=0,1,2,3
	i=3	j=0,1,2,3

加速枚举		
	i=1	j=3
	i=3	j=1

加速枚举

f[i][p]恰首次到达i号
且恰去过p模式里各地点的最少时间

```
30      for(ll i=0;i<=n;++i)LOG2[1LL<<i]=i;
31      f[0][1]=0;
32      for(ll p=1;p<nPtn;++p)
33          for(ll s=p;s;s&=s-1){
34              ll LSB_s=s&(-s);
35              ll i=LOG2[LSB_s];
36              ll q=LSB_s^p;
37              for(ll t=q;t;t&=t-1){
38                  ll LSB_t=t&(-t);
39                  ll j=LOG2[LSB_t];
40                  f[i][p]=min(f[i][p],f[j][q]+d[j][i]);
41              }
42          }
```

删末尾1

取末尾1

复杂度?

加速枚举

复杂度计算	m=n+1代表位数

二进制p	000	001	010	011	100	101	110	111
枚举i,j计算量	0	0	0	2	0	2	2	6

$$C(m,k)=\frac{m!}{k!\,(m-k)!}$$

$$\sum_{k=2}^{m} C(m,k)\times k\times(k-1)$$

$$=\sum_{k=2}^{m} m\times(m-1)\times C(m-2,k-2)$$

$$=m\times(m-1)\times\sum_{k=2}^{m} C(m-2,k-2)$$

$$=m\times(m-1)\times 2^{m-2}$$

现场挑战
快快编程1103

解法思路

方法1 | 暴力:逐格枚举+剪枝

方法2 | 状压DP

方法1 暴力:逐格枚举+剪枝

f[x][y]表示第x行第y列是否放置炮兵

```
 8 void dfs(int x,int y,int cnt){
 9     if(x>n){ans=max(ans,cnt);return;}
10     int nx=(y==m?x+1:x);
11     int ny=(y==m?1:y+1);
12     if(d[x][y]=='H'){dfs(nx,ny,cnt);return;}
13     if(x>1&&f[x-1][y]){dfs(nx,ny,cnt);return;}
14     if(x>2&&f[x-2][y]){dfs(nx,ny,cnt);return;}
15     if(y>1&&f[x][y-1]){dfs(nx,ny,cnt);return;}
16     if(y>2&&f[x][y-2]){dfs(nx,ny,cnt);return;}
17     f[x][y]=1;
18     dfs(nx,ny,cnt+1);
19     
20     
21 }
```

主函数调用什么参数

如何剪枝？

如何剪枝?

可行性剪枝	预先计算每行 行内合法状态
最优性剪枝	未来空位都摆放 都无法突破最优解

```
 9 void dfs(int x,int y,int cnt){
10     if(x>n){ans=max(ans,cnt);return;}
11     if(cnt+nP[x][y]<=ans)return;
```

```
31     for(int i=n;i>=1;--i)
32         for(int j=m;j>=1;--j){
33             nP[i][j]=(d[i][j]=='P');
34             if(j<m) nP[i][j]+=nP[i][j+1];
35             else if(i<n) nP[i][j]+=nP[i+1][1];
36         }
```

nP[i][j]表示该格及以后共几个空地P

方法2 状压DP

f[i][a][b]表示第i行恰为状态a
且第i-1行恰为状态b时前i行最多炮兵数

行数n<=100,列数m<=10

数组大小如何设定?

```
int f[109][1<<10][1<<10];
```

内存超出限制

需要滚动数组

```
int f[3][1<<10][1<<10];
```

答案表示

f[i][a][b]表示第i行恰为状态a
且第i-1行恰为状态b时前i行最多炮兵数

```
48      int ans=0;
49      for(int a=0;a<nPtn;++a)
50          for(int b=0;b<nPtn;++b)
51              ans=max(ans,                    );
```

需要滚动数组

```
int f[3][1<<10][1<<10];
```

```
19      cin>>n>>m;
20      for(int i=1;i<=n;++i)
21          for(int j=1;j<=m;++j){
22              char ch;
23              cin>>ch;
24              mp[i]=(mp[i]<<1)+(ch=='H');
25          }

26      int nPtn=(1<<m);
27      for(int a=1;a<nPtn;++a)
28          cntBit[a]=1+cntBit[a&(a-1)];
29      for(int a=0;a<nPtn;++a)if(OK(1,a))
30          f[1][a][0]=cntBit[a];
```

判断在第i行
状态x是否合法

地图山丘没有放炮兵

同一行炮兵不冲突

```
13 bool OK(int i,int x){
14     return !(mp[i]&x)
15         && !(x&(x>>1))
16         && !(x&(x>>2));
17 }
```

```
26     int nPtn=(1<<m);
27     for(int a=1;a<nPtn;++a)
28         cntBit[a]=1+cntBit[a&(a-1)];
29     for(int a=0;a<nPtn;++a)if(OK(1,a))
30         f[1][a][0]=cntBit[a];
```

```
29      for(int a=0;a<nPtn;++a)if(OK(1,a))
30          f[1][a][0]=cntBit[a];

31      for(int a=0;a<nPtn;++a)if(OK(2,a))
32          for(int b=0;b<nPtn;++b)if(OK(1,b)){
33              if(a&b)continue;
34              f[2][a][b]=
35          }
```

优化:只枚举合法状态a,b,c

```
36      for(int i=3;i<=n;++i)
37          for(int a=0;a<nPtn;++a)if(OK(i,a))
38              for(int b=0;b<nPtn;++b)if(OK(i-1,b)){
39                  if(a&b)continue;
40                  for(int c=0;c<nPtn;++c)if(OK(i-2,c)){
41                      if((a&c)||(b&c))continue;
42                      f[i%3][a][b]=max(
43                          f[i%3][a][b],
44                          f[(i-1)%3][b][c]+cntBit[a]
45                      );
46                  }
47              }
```

方法2

状压DP

预处理计算
每一行的合法状态

```
13 bool OK(int x){
14     return !(x&(x>>1)) && !(x&(x>>2));
15 }
16 bool OK(int i,int x){
17     return !(mp[i]&x);
18 }
```

```
32     int nOK=0;
33     for(int a=0;a<nPtn;++a)if(OK(a))
34         ptn[nOK++]=a;
```

```
50      for(int i=3;i<=n;++i)
51          for(int u=0;u<nOK;++u){
52              int a=ptn[u];
53              if(!OK(i,a))continue;
54              for(int v=0;v<nOK;++v){
55                  int b=ptn[v];
56                  if(!OK(i-1,b))continue;
57                  if(a&b)continue;
58                  for(int w=0;w<nOK;++w){
59                      int c=ptn[w];
60                      if(!OK(i-2,c))continue;
61                      if((a&c)||(b&c))continue;
62                      f[i%3][a][b]=max(
63                          f[i%3][a][b],
64                          f[(i-1)%3][b][c]+cntBit[a]
65                      );
66                  }
67              }
68          }
```

快快编程作业

1102

1103

拓展题	1101