火力网的搜索优化方案

搜索是万能算法，如果加上剪枝，它将变的更加完美，必要的剪枝可以使程序的效率成几何倍数的增长，所以可以说剪枝是搜索的生命。

问题：火力网（见附录一）

初解：由于最大节点数只有 10*10=100 个，因此考虑用深度优先搜索解决。在最初的棋盘上对所有的空地放上或拆除碉堡，用类似八皇后问题的回溯法很容易编出程序（见附录二）。但发现到当 n>=8 时，程序已经不能很快出解了，剪枝势在必行。

剪枝一：改变数据结构，使数据结构为算法服务。

原来用的是二维数组，在搜索时不免因为大量的非空节点（有墙或被控制）使找空节点的效率十分低下。方案：可以将所有的空节点放在一个数组中，记录 x,y 坐标。

剪枝二：避免重复的搜索，使搜索有序。（下界剪枝）

有时可能出现第 I 次搜索先选了 A 格子，后选了 B 格子，而又在第 J 次搜索先选了 B 格子，后选了 A 格子

为了避免重复的搜索，可以在每次选择一个空地放之后在编号大于它的空地中找下一个可放的位置，这样减去了大量的重复。

剪枝三：对不可能大于 Max 的进行剪枝。（上界剪枝）

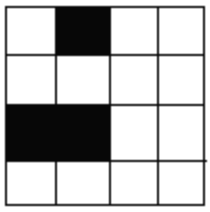
比如现在已经放了 n 个碉堡，还剩 m 块空地，若 n+m<=Max，那么就不可能大于 Max 了。

所以，需要对全部未扩展节点+已扩展节点 仍不能大于 Max 的进行剪枝。

***以上只是对于搜索的最一般剪枝方法，与题目关系不大，以下是针对题目的剪枝。

剪枝四：对横路径数&竖路径数 剪枝。

对于一个题目中的图，如果可以看成由空地组成的路径，就可以计算出有多少个横路径和竖路径。比如



1		2	2
3	3	3	3
		4	4
5	5	5	5

有 5 条横路径

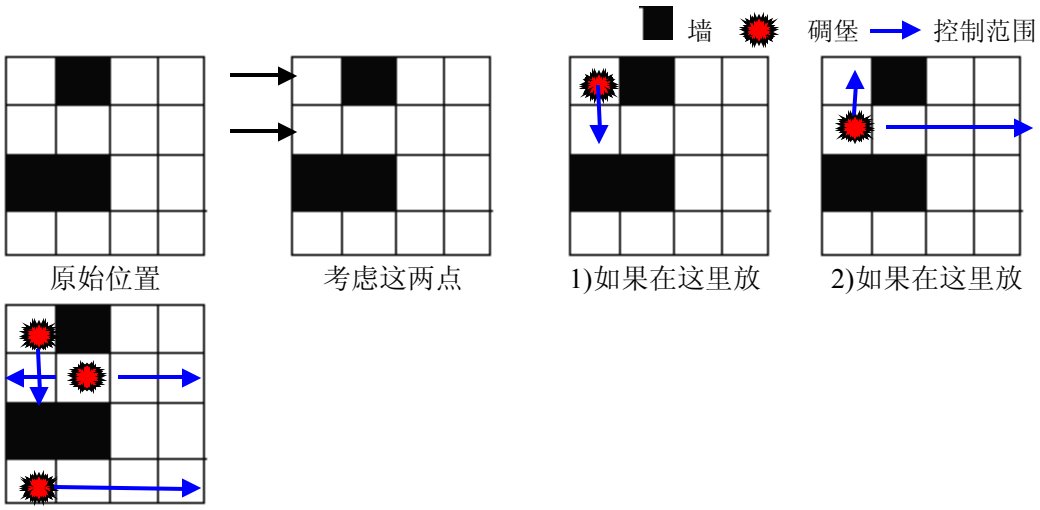
1		5	6
1	3	5	6
		5	6
2	4	5	6

6 条竖路径。

把题目中的空地抽象成横路径和竖路径后，由于每条路径最多放一个碉堡。根据抽屉原理，碉堡数不超过横路径数或者竖路径数。如果找到方案可以达到最大，那么就输出结束。（这样比 n*n/2 更加优秀）。

***以下是利用初始化大量减少空地的方法来剪枝。

剪枝五：在最优点放碉堡



所有最优位置（注意最底下两个位置，它们是等效的，任选其一即可）

很明显的看到情况 1 优于情况 2。因为同样是放一个碉堡，但情况 1 所覆盖的空位属于情况 2 的一部分（相当与情况 2 在情况 1 的基础上白白多控制了一部分）。

图中由于最左上角的点在放完所有的碉堡后一定被控制，而控制它的碉堡只能是上面两种情况（碉堡放 [1，1]或[2，1]），所以在左上角放碉堡一定最优，一定不会减少最多碉堡数。在搜索（或随机化之前）可以先在这种地方先放上碉堡（最优点，不影响结果）。

具体怎么找呢？有如下决策：

对于一个空地，如果在它放碉堡只能攻击到一条直线上不被控制的空地（或攻击不到其他空地），则它是最优点（放碉堡不影响结果）。

即：如果某空地放碉堡后在横 &竖都可以控制到空点（空地且不被控制），则它不是最优点，否则就是。

证明：设某一空地 A 满足只能攻击横或竖的空点（仍可以放碉堡的空地）或不能攻击空点。

1. A 不能攻击其余空点。

a.可能 A 周围都是墙; (那么放这里就一定了，相当于白白送一碉堡)

b.可能 A 周围都被其余碉堡控制; (由于不会减少其余空地的个数，相当于直接变一空地为碉堡)

c.可能 A 周围又有墙又被其余碉堡控制; (综上所述，放这里绝对没错)

2. A 只能控制一条直线上的空点。

首先，在放完所有的碉堡后 A 一定被控制（如果没被控制说明 A 还可以放），而且控制 A 的点只能是 A 所能控制直线上的点（ 若 A 控制不了 B，则 B 一定控制不了 A；若 A 能控制 B，则 B 一定能控制 A）。

就是说在 A 和同一条直线上的其他点之中要放一个。

注意到 不管是放哪一个都要控制这条直线，而 A 只控制这条直线。如果放其余位置则可能控制更多的地方（也有可能与 A 等效，但不会比 A 更好）。

这种情况放 A 绝对不错。

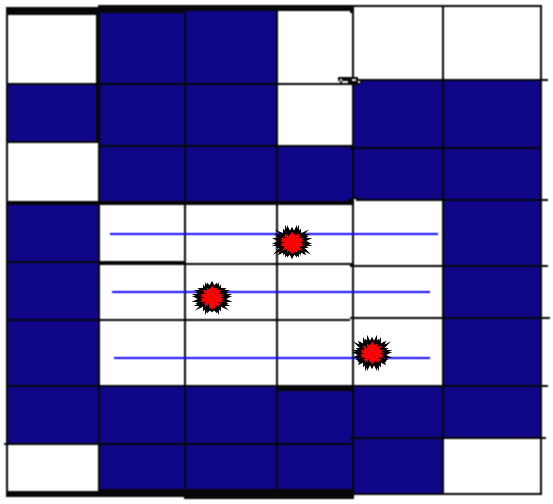
在最开始对地图扫描几遍，找出所有的最优点放上碉堡，这样可以使待搜索的节点大量减少，从而飞速提高程序效率，而不会漏解。（节点越多，搜索量越大）

***但如果大量的节点都是空点，最优点少之又少怎么办呢？虽然前五个剪枝足以应付题目，但如果棋盘在大怎么办呢？对此，我们有如下初始化剪枝（矩形剪枝系列）。

注：以下矩形内部均不被事先控制。

剪枝六：在矩形中放碉堡。

如果在某一地区由墙所围成了一个 m*n 的矩形（设 m<=n）,则矩形中最多可放 m 个碉堡。（证明从略）

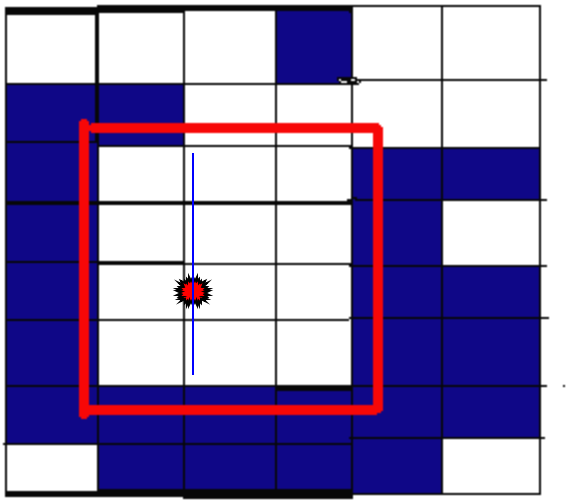


如上图是一个 3*4 的矩形，很明显最多只能放短边数，既 3 个碉堡。

若 m=n(正方形)，放对角线就可以了。

***当然，这种刚好矩形情况少之又少，说出来只是为了进一步解释以下剪枝

剪枝七：三面墙一面开口且开口边是宽（或是正方形）的矩形。



如图，注意红匡内的矩形，就是满足定义的矩形。

决策：m*n（m<=n）的矩形中放 m 个碉堡一定最优。

证明：

首先，在这种 m*n（m<=n）的矩形中最多可以放 m 个碉堡

其次，注意蓝线所标出的列上的所有节点，在最后一定被控制。那么就有两种控制情况。（不可能有些被横着控制，有些被竖着控制）

1．全被竖着控制;（也就是说在这一列上必有一碉堡）

2．全被横着控制;（在矩形中如果不在这一列放碉堡，那么最多只可以放 m-1 个碉堡，小于长，那么就有一行不能被控制（不可能））。

综上所述，每一列都要放一碉堡。

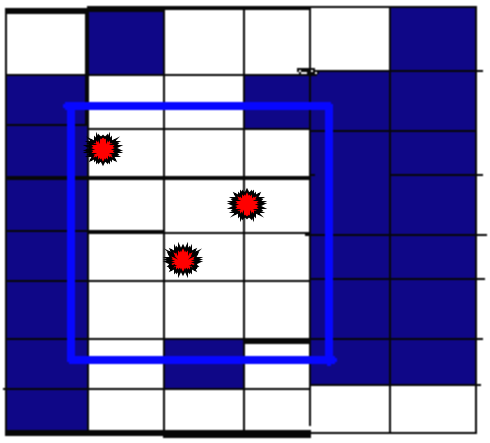
a.如果放矩形内部，则除了控制这一列还控制矩形中的一行（事实是， m 个碉堡已经把矩形控制完了），这其实不会影响什么。

b.而如果放矩形外部，则可能控制更多的节点。（不会比放内部更好）

这样，一批一批的节点被剪掉，程序效率突飞猛进，事实上还有更为优秀的策略。

剪枝八：对面开口，且开口边不大于另一边的管状矩形。

如图中的矩形（3*4），两面（对面）开口，另两面是墙。
决策：m*n（m<=n）的矩形中放 m 个碉堡一定最优。



证明：（改编于剪枝七的证明）

首先，在这种 m*n（m<=n）的矩形中最多可以放 m 个碉堡

其次，矩形内任意一行的所有节点，在最后一定被控制。那么就有两种控制情况。（因为不可能有些被横着控制，有些被竖着控制）

1．全被横着控制;（也就是说在这一横行上必有一碉堡）

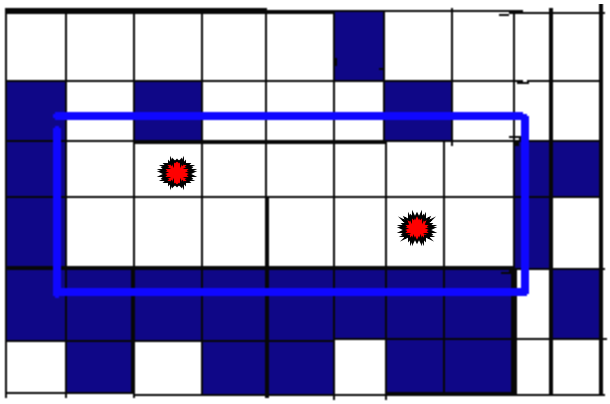
2．全被竖着控制;（在矩形中如果不在这一行放碉堡，那么最多只可以放 m-1 个碉堡，小于长 ，那么就有一列不能被控制（不可能））。

综上所述，每一行都要放一碉堡。

a.如果放矩形内部，则除了控制这一行还控制矩形中的一列（事实是， m 个碉堡已经把矩形控制完了），这其实不会影响什么。

b.而如果放矩形外部，则可能控制更多的节点。（不会比放内部更好）

剪枝九：三面墙一面开口 且 开口边是长的矩形。



如图，是个 2*7 的矩形，最多可放 2 个碉堡。（图中两个碉堡只攻击矩形内部，所以最优）

决策：m*n（m<=n）的矩形中在开口一边若与墙相邻，则在这一列放碉堡一定最优，最多放 m 个碉堡。（我只能证明到这里了）

证明：

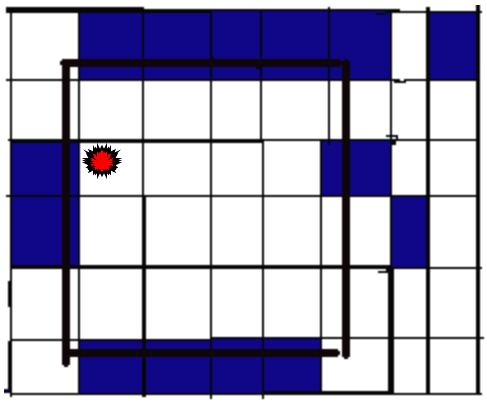
首先此矩形开口一边一定有墙相邻（否则就没法优化了）。

因此如果不在矩形内放 m 个碉堡，就会使有些空地得不到控制。（因为不能来自同一行和外界的控制）

所以一定有 m 个碉堡（既每一行都有碉堡，也就是矩形内全被控制）。

那么在决策的地方放碉堡就等于是白送了。

剪枝十：对面开口，且开口边大于另一边的管状矩形。



决策：m*n（m<=n）的矩形中在开口边相邻位置有对称的墙出现，则在这一行放碉堡一定最优，最多放 m 个碉堡。

（证明可以仿照剪枝八与剪枝九的证明）

事实上，由于所给的地图的外面一层可以看做是墙，矩形剪枝大有用武之地。

事实是，用前 5 个剪枝就足以解决题目。

但更为事实的是由于编码的复杂性，矩形剪枝的理论性大于实践性。写在这里仅做抛砖引玉之用而已。

除此之外，还可以搜索时对每一个格子设一个估价函数，对优秀性进行评估，选出可能是最好的格子放，进行启发式搜索，这样再大的地图也可以瞬间出界。

最后奉劝一句，不要看我写的代码（冗长啊）。

附录一：火力网(Fire)

【问题描述】 在一个 n*n 的阵地中，有若干炮火不可摧毁的石墙，现在要在这个阵地中

的空地上布置一些碉堡。假设碉堡只能向上下左右四个方向开火，由于建筑碉堡的材料挡不住炮火，所以任意一个碉堡都不能落在其他碉堡的火力范围内，请问至多可建造几座碉堡？

输入：

输入文件名为 FIRE.IN，第一行是一个整数 n(n<=10)，下面 n 行每行为一个由 n 个字符构成的字符串，表示阵地的布局，包括空地（’.’），和石墙（’X’）。

输出：

输出文件名为 FIRE.OUT，一个整数，即最多可建造的碉堡数。

【样例输入输出】

FIRE.IN

```
4
.X..
....
XX..
....
```

FIRE.OUT

```
5
```

附录二：不带任何优化的搜索代码

```
Program Fire;
```

```
Var
  n,                                  //大小
  Max:Longint;                        //最多
  x:Array[0..11,0..11]of Boolean;     //地图
  f:Array[1..10,1..10]of Longint;     //火力覆盖

Procedure Init;{读入}
  Var Q1,Q2:Longint; ch:Char;
  Begin
    Fillchar(x,sizeof(x),false);
    Max:=0;
    readln(n);
    For Q1:=1 To n Do
      Begin
        For Q2:=1 To n Do
          Begin
            read(ch);
            if ch='.' then x[Q1,Q2]:=True else x[Q1,Q2]:=False;
            if ch='.' then f[Q1,Q2]:=0    else f[Q1,Q2]:=1;
          End;
        readln;
      End;
  End;

Procedure Put(a,b:Longint);{在[a,b]放碉堡}
  Var ww:Longint;
  Begin
    inc(f[a,b]);
    ww:=a+1;
    while x[ww,b] do Begin  inc(f[ww,b]); inc(ww); End;
    ww:=a-1;
    while x[ww,b] do Begin  inc(f[ww,b]); dec(ww); End;
    ww:=b+1;
    while x[a,ww] do Begin  inc(f[a,ww]); inc(ww); End;
```

```
    ww:=b-1;
    while x[a,ww] do Begin  inc(f[a,ww]); dec(ww); End;
  End;

Procedure Out(c,d:Longint);{去掉[c,d]的碉堡}
  Var ee:Longint;
  Begin
    dec(f[c,d]);
    ee:=c+1;
    while x[ee,d] do Begin  dec(f[ee,d]); inc(ee); End;
    ee:=c-1;
    while x[ee,d] do Begin  dec(f[ee,d]); dec(ee); End;
    ee:=d+1;
    while x[c,ee] do Begin  dec(f[c,ee]); inc(ee); End;
    ee:=d-1;
    while x[c,ee] do Begin  dec(f[c,ee]); dec(ee); End;
  End;

Procedure Find(k:longint);{找第 k 个碉堡}
  Var i,j,it:Longint;
  Begin
    If k-1>Max then Max:=k-1;
    For it:=1 to sqr(n) do
      Begin
        i:=(it-1) div n+1;
        j:=(it-1) mod n+1;
        If F[i,j]=0 then
          Begin
            Put(i,j);
              Find(k+1,(i-1)*n+j+1);
            Out(i,j);
          End;
      End;
  End;
```

```
Begin
  Assign(Input,'fire.in');
  Assign(Output,'fire.out');
  Reset(Input);
  Rewrite(Output);
  Init;
  Find(1);
  writeln(Max);
  Close(Input);
  Close(Output);
       End.
```

附录三：剪枝 12345 的搜索代码

```
Program Fire;
```

```
Const
  MaxN=10;
Var
  n,                                    {大小}
  MinPath,                              {横路径数&竖路径数中最小的，也是可能放碉堡最多的数目}
  Max,                                  {最多放的碉堡数}
  Npoint:Longint;                       {空点总数}
  k:Array[0..MaxN+1,0..MaxN+1]of Boolean;{是否是空地}
  f:Array[0..MaxN+1,0..MaxN+1]of Longint;{火力覆盖数}
  p:array[1..MaxN*MaxN]of Record x,y:Longint; End;{空地的坐标}


Procedure Put(a,b:Longint);{在[a,b]放碉堡}
  Var ww:Longint;
  Begin
    Inc(f[a,b]);
    ww:=a+1;
    While k[ww,b] Do Begin  Inc(f[ww,b]); Inc(ww); End;
    ww:=a-1;
    While k[ww,b] Do Begin  Inc(f[ww,b]); Dec(ww); End;
    ww:=b+1;
    While k[a,ww] Do Begin  Inc(f[a,ww]); Inc(ww); End;
    ww:=b-1;
    While k[a,ww] Do Begin  Inc(f[a,ww]); Dec(ww); End;
  End;


Procedure Out(c,d:Longint);{去掉[c,d]的碉堡}
  Var ee:Longint;
  Begin
    Dec(f[c,d]);
    ee:=c+1;
    While k[ee,d] Do Begin  Dec(f[ee,d]); Inc(ee); End;
```

```
    ee:=c-1;
    While k[ee,d] Do Begin  Dec(f[ee,d]); Dec(ee); End;
    ee:=d+1;
    While k[c,ee] Do Begin  Dec(f[c,ee]); Inc(ee); End;
    ee:=d-1;
    While k[c,ee] Do Begin  Dec(f[c,ee]); Dec(ee); End;
  End;

Procedure Init;        {读入}
  Var
    Q1,Q2:Longint; ch:Char;
  Begin
    Fillchar(k,Sizeof(k),False);
    Readln(n);
    For Q1:=1 To n Do
      Begin
        For Q2:=1 To n Do
          Begin
            Read(ch);
            If ch='.' Then k[Q1,Q2]:=True;
          End;
        Readln;
      End;
  End;

Procedure Beginning;          {主要是算 横&竖路径}
  Var
    W1,W2,HHH,SSS:Longint;
  Begin
    Max:=0;
    Fillchar(f,Sizeof(f),0);
    For W1:=0 To n+1 Do
      For W2:=0 To n+1 Do
        If Not k[W1,W2] Then f[W1,W2]:=1{MaxLongint};  {有墙，火力覆盖置为 MaxLongint}
```

```
    HHH:=0;{算横路径数}
    For W1:=1 To n Do {行}
      Begin
        W2:=1;
        While W2<=n Do
          Begin
            While (W2<=n) And (Not k[W1,W2]) Do Inc(W2);     {去掉墙}
            If k[W1,W2] Then Begin Inc(HHH); While k[W1,W2] Do Inc(W2); End;  {算空地}
          End;
      End;
    SSS:=0;{算竖路径数}
    For W2:=1 To n Do
      Begin
        W1:=1;
        while W1<=n Do
          Begin
            While (W1<=n) And (Not k[W1,W2]) Do Inc(W1);
            If k[W1,W2] then Begin Inc(SSS); While k[W1,W2] Do Inc(w1); End;
          End;
      End;
    If SSS>HHH then MinPath:=HHH Else MinPath:=SSS;{横路径数&竖路径数中最小的赋给 MinPath}
  End;

Procedure PutMust;     {放最优点}
  Var
    Over,HH,SS:Boolean;
    E1,E2,E3,E4:Longint;
  Begin
    Repeat
      Over:=True;     {是否找完最优点}
      For E1:=1 To n Do
        For E2:=1 To n Do
          If f[E1,E2]=0 Then
```

```
Begin
  HH:=False; SS:=False;

  E3:=E1+1;
  While k[E3,E2] do     {竖着往下走}
    Begin
      If f[E3,E2]=0 Then Begin SS:=True;Break;End;
      Inc(E3);
    End;
  If Not SS Then        {竖着往上走}
    Begin
     E3:=E1-1;
     While k[E3,E2] do
       Begin
         If f[E3,E2]=0 Then Begin SS:=True;Break;End;
         Dec(E3);
       End;
    End;

  E4:=E2+1;
  While k[E1,E4] do     {横着往右走}
    Begin
      If f[E1,E4]=0 Then Begin HH:=True;Break;End;
      Inc(E4);
    End;
  If Not HH Then        {横着往左走}
    Begin
     E4:=E2-1;
     While k[E1,E4] do
       Begin
         If f[E1,E4]=0 Then Begin HH:=True;Break;End;
         Dec(E4);
       End;
    End;
```

```
                    If HH And SS then Continue;
                    Put(E1,E2);     {放上}
                    Inc(Max);       {多了一个碉堡}
                    Over:=False; {找到则继续找}
                  End;
          Until Over;
          If Max=MinPath Then Begin Writeln(Max); Close(Input); Close(Output);Halt;End;  {运气好的话，哈哈哈~}
        End;

      Procedure DataSave_f_To_p;    {改变数据结构，剪枝 1}
        Var R1,R2:Longint;
        Begin
          Npoint:=0;
          For R1:=1 To n Do
            For R2:=1 To n Do
              If f[R1,R2]=0 then
                Begin
                  Inc(Npoint);
                  p[Npoint].x:=R1;
                  p[Npoint].y:=R2;
                End;
        End;

      Procedure Find(T,xy:Longint); {寻找第 T 个格子(p 中),第 xy 个碉堡}
        Var i:Longint;
        Begin
          If xy-1>Max Then Begin
    { 横 & 竖 路 径 判 断 →}If  xy-1=MinPath  Then  Begin  Writeln(MinPath);  Close(Input); Close(Output);Halt;End;
                              Max:=xy-1;
                            End;
          For i:=T To Npoint+xy-Max-1 Do      //剪枝 2，3 ：上下界剪枝
              If f[p[i].x,p[i].y]=0 then
```

```
          Begin
            Put(p[i].x,p[i].y);
              Find(i+1,xy+1);   {递归}
            Out(p[i].x,p[i].y)
          End;
  End;

Begin
  Assign(Input,'fire.in');
  Assign(Output,'fire.out');
  Reset(Input);
  Rewrite(Output);

  Init;
  Beginning;              {剪枝 4，算 横&竖路径}
  PutMust;                {剪枝 5，放最优点}
  DataSave_f_To_p;{改变数据结构，剪枝 1}
  Find(1,Max+1);    {DFS}
  Writeln(Max);

  Close(Input);
  Close(Output);
End.
```

附录四： 数据产生程序

{参数为空地的概率，越大表示空地越多}

```
Program MakeFire;
var i,j,n:longint; abc:extended;
begin
writeln('Input n');readln(n);
writeln('Input 参数(大于0小于1)');readln(abc);
assign(output,'fire.in');
rewrite(output);
randomize;
writeln(n);
for i:=1 to n do
  begin
    for j:=1 to n do if random<abc then write('.') else write('X');
    writeln;
  end;
  close(output);
end.
```