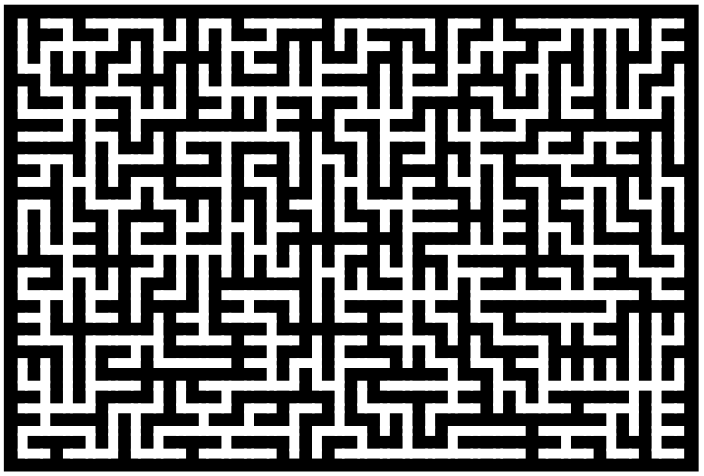
迷宫生成器

创建随机迷宫.

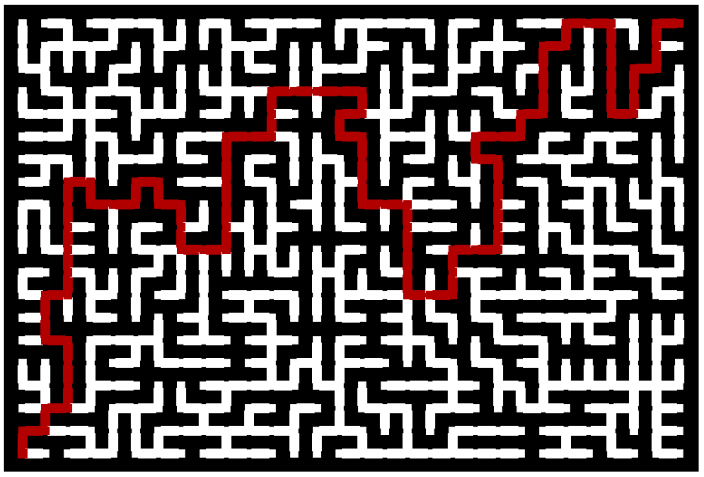
```
In[ ]:= (*custom styling*)style =
          {Background → GrayLevel[0], BaseStyle → {Directive[White, EdgeForm[], Opacity[1]]},
           背景色 灰度级 基本样式 指令 白色 边的格式 不透明度
           VertexShapeFunction → (Rectangle[#1 + .16, #1 - .16] &),
           顶点形状函数 矩形
           EdgeShapeFunction → (Rectangle[#1[[1]] + .16, #1[[2]] - .16] &)};
           边的形状函数 矩形
        embedding = GraphEmbedding[GridGraph[{20, 30}]];
                    嵌入法得出的绘… 网格图

In[ ]:= g = GridGraph[{20, 30}, EdgeWeight → RandomReal[10, 1150]];
          网格图 边的权值 伪随机实数
        tree = FindSpanningTree[{g, 1}];
               找到生成树
        maze = Graph[tree, VertexCoordinates → embedding, style]
               图 顶点坐标

Out[ ]=
```



解迷宫并突出显示路径.

In[ ]:=
```
HighlightGraph[maze, PathGraph[FindShortestPath[maze, 1, 600]]]
```
突出显示图 路径图 求指定顶点间的最短路径

Out[ ]=



使用循环生成迷宫图案

In[ ]:=
```
genLines[width_, height_, step_] :=
 Module[{lines = {}}, (For[i = 0, i < width, i += step,
```
模块 For循环
```
    For[j = 0, j < height, j += step,
```
For循环
```
     If[RandomInteger[] == 1,
```
… 伪随机整数
```
      AppendTo[lines, {{i, j}, {i + step, j + step}}],
```
附加
```
      AppendTo[lines, {{i + step, j}, {i, j + step}}]
```
附加
```
     ]
    ]];
   lines)]
```

In[ ]:=
```
lines = genLines[200, 150, 10];
g = Graphics[{Thickness[0.02], Purple, Line[lines]}, ImageSize → Large]
```
图形 粗细 紫色 线段 图像尺寸 大

Out[ ]=