导航

- 首页
- 社区主页
- 当前事件
- 最近更改
- 随机页面
- 使用帮助
- NOCOW地图
- 新手试练场

搜索

进入 搜索

工具箱

- 链入页面
- 链出更改
- 特殊页面
- 可打印版
- 永久链接

深度优先搜索

目录 [隐藏]

深度优先搜索(Depth-first search) [编辑]

介绍 [编辑]

深度优先搜索(Depth-first search)正如算法名称那样，深度优先搜索所遵循的搜索策略是尽可能“深”地搜索图。在深度优先搜索中，对于最新发现的顶点，如果它还有以此为起点而未探测到的边，就沿此边继续下去。当结点v的所有边都已被探寻过，搜索将回溯到发现结点v有那条边的始结点。这一过程一直进行到已发现从源结点可达的所有结点为止。如果还存在未被发现的结点，则选择其中一个作为源结点并重复以上过程，整个进程反复进行直到所有结点都被发现为止。

深度优先搜索的递归实现过程:

```
procedure dfs(i);
begin
   for i:=1 to r do
   begin
      if 子节点 mr符合条件
         then 产生的子节点mr入栈;
      if 子节点mr是目标节点
         then  输出
         else  dfs(i+1);
      栈顶元素出栈（即删去mr）;
   end;
end;
```

算法实例: [编辑]

基本搜索 [编辑]

深度优先搜索的一个经典范例是8皇后问题，但由于8皇后问题涉及搜索的剪枝优化问题，所以这里就举个简单一点的例子。

问题描述:

在一个4X4的棋盘上，放置4个皇后，求一种方案，使得4个皇后不互相攻击。

```
program DFS;
const
   n=4;
var
   a:array[1..n,1..n] of integer;
   i,j,x,y:longint;

procedure init;
var
   i,j:longint;
```

```
begin
   fillchar(a,sizeof(a),0);
end;

function check(x,y:longint):boolean;
var
   i,j:longint;
begin
   check:=false;
   for i:=1 to n do
      if a[i,y]+a[x,i]>0 then exit;
   i:=x;
   j:=y;
   repeat
      dec(i);
      dec(j);
   until (i=0)or(j=0);
   repeat
      inc(i);
      inc(j);
      if a[i,j]>0 then exit;
   until (i=n)or(j=n);
   i:=x;
   j:=y;
   repeat
      inc(i);
      dec(j);
   until (i=n+1)or(j=0);
   repeat
      dec(i);
      inc(j);
      if a[i,j]>0 then exit;
   until (i=1)or(j=n);
   check:=true;
end;

procedure pri;
var
   i,j:longint;
begin
   for i:=1 to n do
   begin
      for j:=1 to n do if a[i,j]=1 then write('X')
                                   else write('O');
      writeln;
   end;
end;

procedure search(x,y,k:longint);
var
   i,j:longint;
begin
   if k=n+1 then begin
                    pri;
                    halt;
                 end;  //当程序将搜索第5只时，表示已经完成了任务，输出。
   i:=x;
   j:=y;
   repeat
   //检查（I，J）是否可以放置皇后。
      if check(i,j) then begin
                            a[i,j]:=1;
                            if j=n then search(i+1,1,k+1)
                                   else search(i,j+1,k+1);  //搜索下一层。
                            a[i,j]:=0;  //回溯，既栈顶元素出栈。
                         end;
      inc(j);
      if j>n then begin
                     j:=1;
                     inc(i);
                  end;
   until (i=n+1);
end;

begin
   init;
   search(1,1,1);  //表示在坐标为（1，1）的地方搜索第1个皇后
end.
```

以下提供一种基础算法，这种方法不需要栈的操作，这种算法拥有其得天独厚的优势，需要学习，但较为繁琐，不建议使用。

--SepHiRoTH 23:14 2009年5月28日 (CST)

```
var
```

```
  a:array[0..100] of integer;
  n,i,j,p:integer;
  f:boolean;
begin
  readln(n);
  fillchar(a,sizeof(a),0);
  p:=1;
  a[p]:=0;
  while a[0]=0 do            //结束条件
    begin
      a[p]:=a[p]+1;
      if a[p]<=n then
        begin
          f:=true;             //判断是否属同列
          for i:=1 to p-1 do
            if a[i]=a[p] then
              begin
                f:=false;
                break;
              end;
          if f then            //判对角*2
            for i:=1 to p-1 do
              if a[i]+p-i=a[p] then
                begin
                  f:=false;
                  break;
                end;
          if f then
            for i:=1 to p-1 do
              if a[i]-p+i=a[p] then
                begin
                  f:=false;
                  break;
                end;
          if f then
            begin
              p:=p+1;
                if p=n+1 then
                  begin
                    for i:=1 to n do
                      begin
                        for j:=1 to n do
                          if a[i]=j then write('X') else write('O');
                        writeln;
                      end;
                    writeln;
                    p:=p-1;
                  end;
            end;
        end
      else          //出栈
        begin
          a[p]:=0;
          p:=p-1;
        end;
    end;
end.
```

优势:

这种算法对栈的要求低，很多的“古董语言”没有足够深的栈，这种算法既能派上大用场。

这是一种初级、简单的方法，可以解决一切DFS问题。

这种算法能让人更好的理解DFS的本质，发现它的魅力。

这种算法有利于BFS的学习。

引用 [编辑]

维库

[1]

图论及图论算法 [编辑]

图 - 有向图 - 无向图 - 连通图 - 强连通图 - 完全图 - 稀疏图 - 零图 - 树 - 网络

基本遍历算法：宽度优先搜索 - 深度优先搜索 - A* - 并查集求连通分支 - Flood Fill

最短路：Dijkstra - Bellman-Ford（SPFA） - Floyd-Warshall - Johnson算法

最小生成树：Prim - Kruskal

强连通分支：Kosaraju - Gabow - Tarjan

网络流：增广路法（Ford-Fulkerson，Edmonds-Karp，Dinic） - 预流推进 - Relabel-to-front

图匹配 - 二分图匹配：匈牙利算法 - Kuhn-Munkres - Edmonds' Blossom-Contraction

1个分类：图论