传染病控制解题报告

题意简述：

给定一棵树，其根节点 1 是已经被感染的患者。在一个疾病传播周期内，只能切断树的一条边，然后所有被感染的节点的孩子节点也会被感染。要求一个最佳的切断边的方案，使被感染的节点数最小。

解法分析：

将被感染节点中深度最大的节点集合称之为最新传染源，设为 S。当经过 X 个疾病传播周期，传染病传播的最新传染源的深度是一定的，即为 X（根节点 1 的深度为 0）。因此可以得到以下动态规划算法：设 $f(X,S)$ 表示经过 X 个疾病传播周期且最新传染源集合为 S 被感染节点的最小值，那么由 $f(X,S)$ 可以推得 $f(X+1,son(S)-v)$，其中 $son(S)$ 表示 S 集合中所有的孩子节点，v 是被切断边的一个深度较大的节点。似乎问题已经得到了很好的解决，但是考察一下算法的空间复杂度，达到了 $O(N\times 2^N)$，这是完全不可能实现的。

鉴于空间上的问题，我们只好放弃动态规划。尝试先从简单的角度思考问题：搜索的效果会怎么样呢？分析一下问题的状态空间：题目给定的是一棵树。把题目中的树称为原树，搜索树的层数就是原树的深度，而搜索树的分叉数多少与原树的宽度有关。综合起来看，原树深度较大，宽度会较小；而原树宽度较大，深度又会较小，所以搜索树的规模没有想象中那么大。

证明了搜索是“可能”出解的，剩下的就是怎么搜的问题：在 X 个传播周期内疾病传播达到的深度是一定的，为 X。如果在深度小于 X 的位置切断边，由于已经通过它进行了传播，切断是没有意义的；如果在深度大于 X 的位置切断边，效果肯定不如把它的祖先切断。因此最优的切断方法必须是切断与 S 相连的边中的一条。这样搜索已经可以通过大部分数据，但是对于极限大数据还是无法出解，需要优化。

优化一：初看这道题目的时候会想到贪心。也就是每次选节点数目最大的子树删除，虽然很容易找到反例，但是我们仍然可以认为选节点数目最大的子树删除更“可能”得到最优解，因此可以按照子树节点数目从大到小删除。

优化二：最优性剪枝。设当前已经搜到了第 X 层，有 T 个节点已被感染，那么如果有 $T+|son(S)|-1\geq best$ 即可剪枝。这是因为在一个疾病传播周期内，最多只能切断树的一条边，所以在下一周期至少还有 $T+|son(S)|-1$ 个新感染的节点。

优化三：搜索最后一两层时，由于子树的结构都相同，随便选择一个删除就可以退出了。

加入这三个优化后，程序的效率大大提高。最大官方数据在 0.2*s* 内解出。

参考程序：

```
{$R-,Q-,S-,I-}
```

```
{$M 65521,0,655360}
{Author:Zhou Gelin}
{Date:2005.1.11}
{QQ:379688236}
{MSN:zhougelin@hotmail.com}
{Test Report}
{P4 1.7G}
{Windows Xp SP1}
{Turbo Pascal 7.0}
{
EPIDEMIC.in1 = 10.0 (0.06s)
EPIDEMIC.in2 = 10.0 (0.08s)
EPIDEMIC.in3 = 10.0 (0.05s)
EPIDEMIC.in4 = 10.0 (0.03s)
EPIDEMIC.in5 = 10.0 (0.03s)
EPIDEMIC.in6 = 10.0 (0.03s)
EPIDEMIC.in7 = 10.0 (0.06s)
EPIDEMIC.in8 = 10.0 (0.03s)
EPIDEMIC.in9 = 10.0 (0.20s)
EPIDEMIC.in0 = 10.0 (0.11s)
}
program epidemic;
const inputfilename='epidemic.in';
      outputfilename='epidemic.out';
      maxn=300;
type pnode=^node;
     node=record
       data:integer;
       next:pnode;
         end;
     list=array[1..maxn]of integer;
var n,p,maxdep,ans:integer;
    g:array[1..maxn]of pnode;
    father:array[1..maxn]of integer;
    dep:array[1..maxn]of integer;
    sum:array[1..maxn]of integer;
    deg:array[1..maxn]of integer;
    son:array[1..maxn]of ^list;
    a:array[1..maxn]of boolean;
procedure insert(a,b:integer);
var q:pnode;
begin
      new(q);
      q^.data:=b;
```

```
    q^.next:=g[a];
    g[a]:=q;
end;
procedure read_data;
var i,a,b:integer;
begin
    fillchar(deg,sizeof(deg),0);
    assign(input,inputfilename);
    reset(input);
    readln(n,p);
    for i:=1 to p do
    begin
      readln(a,b);
      insert(a,b);
      insert(b,a);
      deg[a]:=deg[a]+1;
      deg[b]:=deg[b]+1;
    end;
    close(input);
end;
procedure dfs(v:integer);
var q:pnode;
   i,j,k:integer;
begin
    if father[v]>0 then
    begin
     deg[v]:=deg[v]-1;
     dep[v]:=dep[father[v]]+1;
     if dep[v]>maxdep then maxdep:=dep[v];
    end
    else dep[v]:=1;
    getmem(son[v],2*deg[v]);
    sum[v]:=1;q:=g[v];i:=0;
    while q<>nil do
    begin
     if q^.data<>father[v] then
     begin
       i:=i+1;
       son[v]^[i]:=q^.data;
       father[q^.data]:=v;
       dfs(q^.data);
       sum[v]:=sum[v]+sum[q^.data];
     end;
     q:=q^.next;
```

```
    end;
    for i:=1 to deg[v]-1 do
    begin
      k:=i;
      for j:=i+1 to deg[v] do
        if sum[son[v]^[j]]>sum[son[v]^[k]] then k:=j;
      j:=son[v]^[i];
      son[v]^[i]:=son[v]^[k];
      son[v]^[k]:=j;
    end;
end;
procedure search(deep,sick:integer);
var i,j,tmp:integer;
begin
    if deep=maxdep then
    begin
     if ans>sick then ans:=sick;
     exit;
    end;
    tmp:=0;
    for i:=1 to n do
      if a[i] and (dep[i]=deep) then
        tmp:=tmp+deg[i];
    if tmp=0 then begin search(maxdep,sick); exit; end;
    if sick+tmp-1>=ans then exit;
    for i:=1 to n do
      if a[i] and (dep[i]=deep) then
        for j:=1 to deg[i] do
          a[son[i]^[j]]:=true;
    for i:=1 to n do
      if a[i] and (dep[i]=deep) then
        for j:=1 to deg[i] do
        begin
          a[son[i]^[j]]:=false;
          search(deep+1,sick+tmp-1);
          a[son[i]^[j]]:=true;
          if deep=maxdep-1 then break;
        end;
    for i:=1 to n do
      if a[i] and (dep[i]=deep) then
        for j:=1 to deg[i] do
          a[son[i]^[j]]:=false;
end;
procedure answer;
```

```
begin
    assign(output,outputfilename);
    rewrite(output);
    writeln(ans);
    close(output);
end;
var time:longint;
begin
    time:=meml[64:108];
    read_data;
    dfs(1);
    ans:=1000;
    fillchar(a,sizeof(a),0);
    a[1]:=true;
    search(1,1);
    time:=meml[64:108]-time;
    writeln(time);
    answer;
end.
```