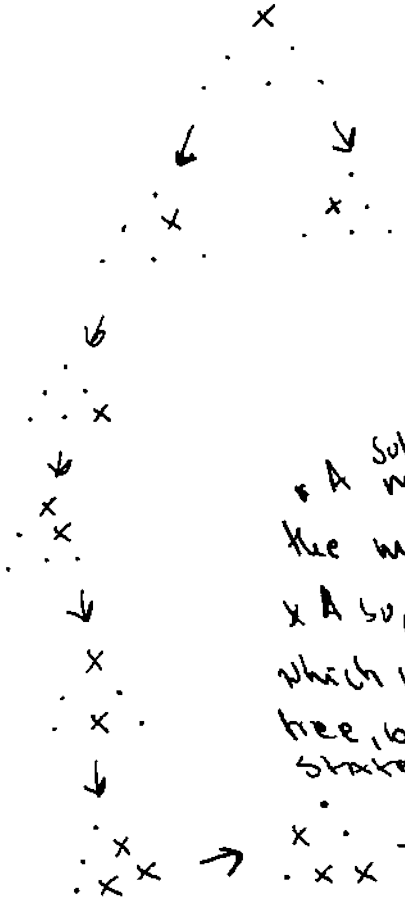
Peg Game Final Notes

Monday, September 14, 2015 02:48

* Make moves and demand; graph only upon } save space;

The states should be each move achieved,

* Regardless Algorithm will go through one option at a time and pursue said option until there is no more;

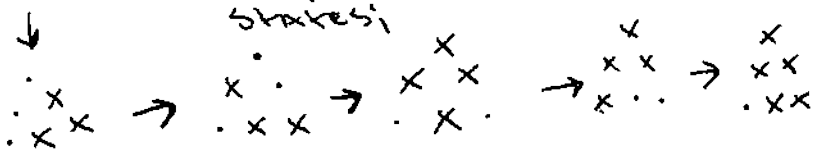


↓ Must then create → the tree is create. A state tree, which is a representation of the current situation of the game;

* A sub-module to compute the the moves and return state;

* A super module which builds another tree, based on current states;

{ If Final state then return ...;



DFS-Peg(G)

state

play scene

A graph of play

//initialize everything;

for each vertex $u \in STG$

vst.next = STG.Add(MakeMove())

v = vst.next;

if v.color == white;

visit-next(stG, v);

must try a different move everything so to work;

visit-next(stG, v)

v.color = gray;

time = time+1;

v.d = time,

.next = stG.add(v.makemove())

= v.next;

for each vertex $u \in adj[v]$

if u.color == white

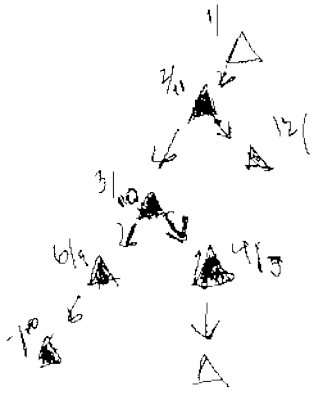
u.π = v;

visit-next(stG, u).

v.color = black;

time = time+1;

v.f = time;



working out the alg. to the right;