Prolog lecture 4

- Playing Countdown
- Iterative deepening
- Search

Countdown Numbers

Select 6 of 24 numbers tiles

- large numbers: 25,50,75,100
- small numbers: 1,2,3,...,10 (two of each)

Contestant chooses how many large and small

Randomly chosen 3-digit target number

Get as close as possible using each of the 6 numbers at most once and the operations of addition, subtraction, multiplication and division

- No floats or fractions allowed

Countdown Numbers

Strategy – generate and test

Maintain a list of symbolic arithmetic terms

- initially this list consists of ground terms e.g.:
  `[gnd(25),gnd(6),gnd(3),gnd(3),gnd(7),gnd(50)]`
- if the head of the list evaluates to the total then succeed
- otherwise pick two of the elements, combine them using one of the available arithmetic operations, put the result on the head of the list, and repeat

Countdown Numbers

Prerequisite predicates:

`eval(A,B)`

- true if the symbolic expression A evaluates to B

`choose(N,L,R,S)`

- true if R is the result of choosing N items from L and S is the remaining items left in L

`arithop(A,B,C)`

- true if C is a valid combination of A and B in the context of the game
  - e.g. arithop(A,B,plus(A,B)).

Countdown Numbers

```
%%% arith_op(+A, +B, -C)
%%% unify C with a valid binary operation
%%% of expressions A and B
arithop(A,B,plus(A,B)).
% minus is not commutative
arithop(A,B,minus(A,B)) :- eval(A,D), eval(B,E), D>E.
arithop(B,A,minus(A,B)) :- eval(A,D), eval(B,E), D>E.
% don't allow mult by 1
arithop(A,B,mult(A,B)) :- eval(A,D), D \== 1,
                          eval(B,E), E \== 1.
% div is not commutative and don't allow div by 0 or 1
arithop(A,B,div(A,B)) :- eval(B,E), E \== 1, E \== 0,
                         eval(A,D), 0 is D rem E.
arithop(B,A,div(A,B)) :- eval(B,E), E \== 1, E \== 0,
                         eval(A,D), 0 is D rem E.
```

Countdown Numbers

The code almost explains itself!

```
% Soln evaluates to the target number
countdown([Soln|_],Target,Soln) :-
     eval(Soln,Target).

% Combine from L to form new experiment
countdown(L,Target,Soln) :-
     choose(2,L,[A,B],R),
     arithop(A,B,C),
     countdown([C|R],Target,Soln).
```

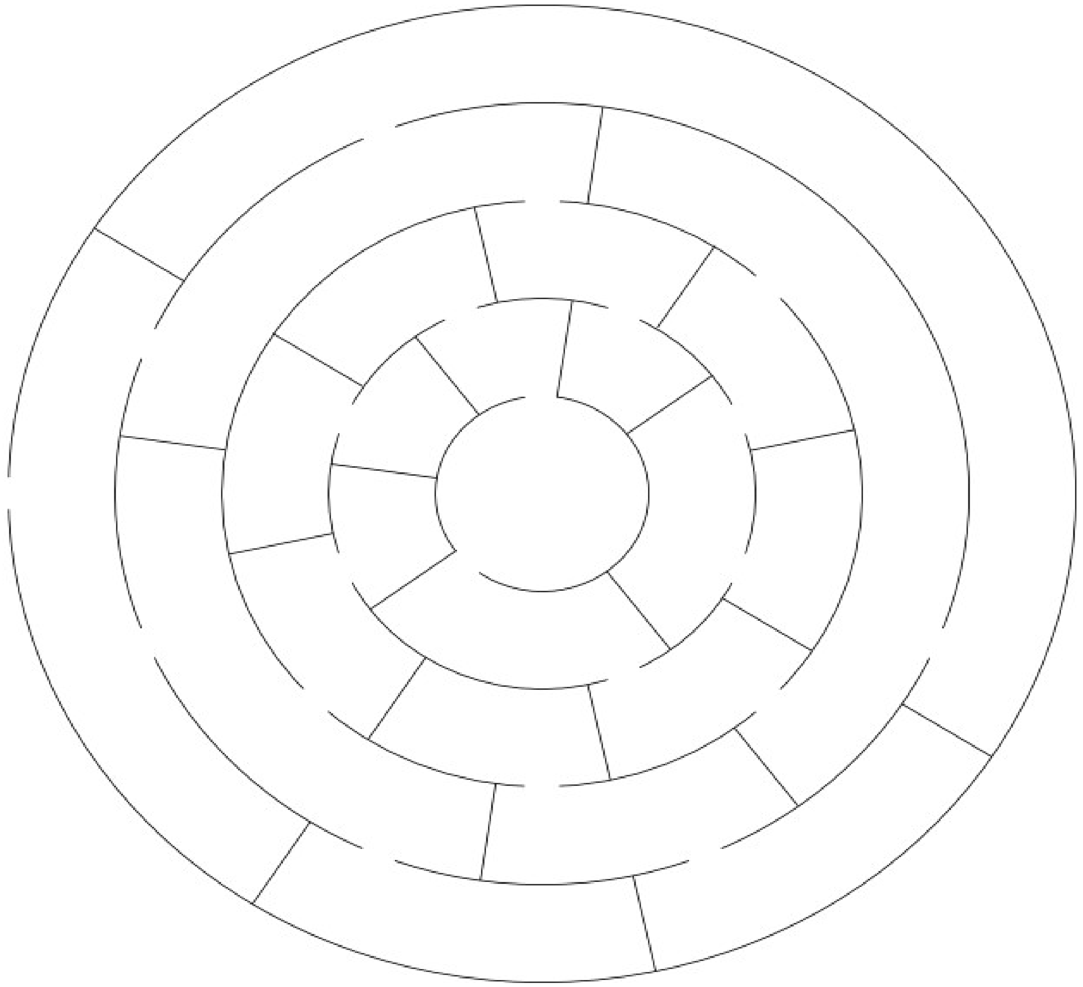
Closest Solution

No exact solutions? Find the closest solution instead.

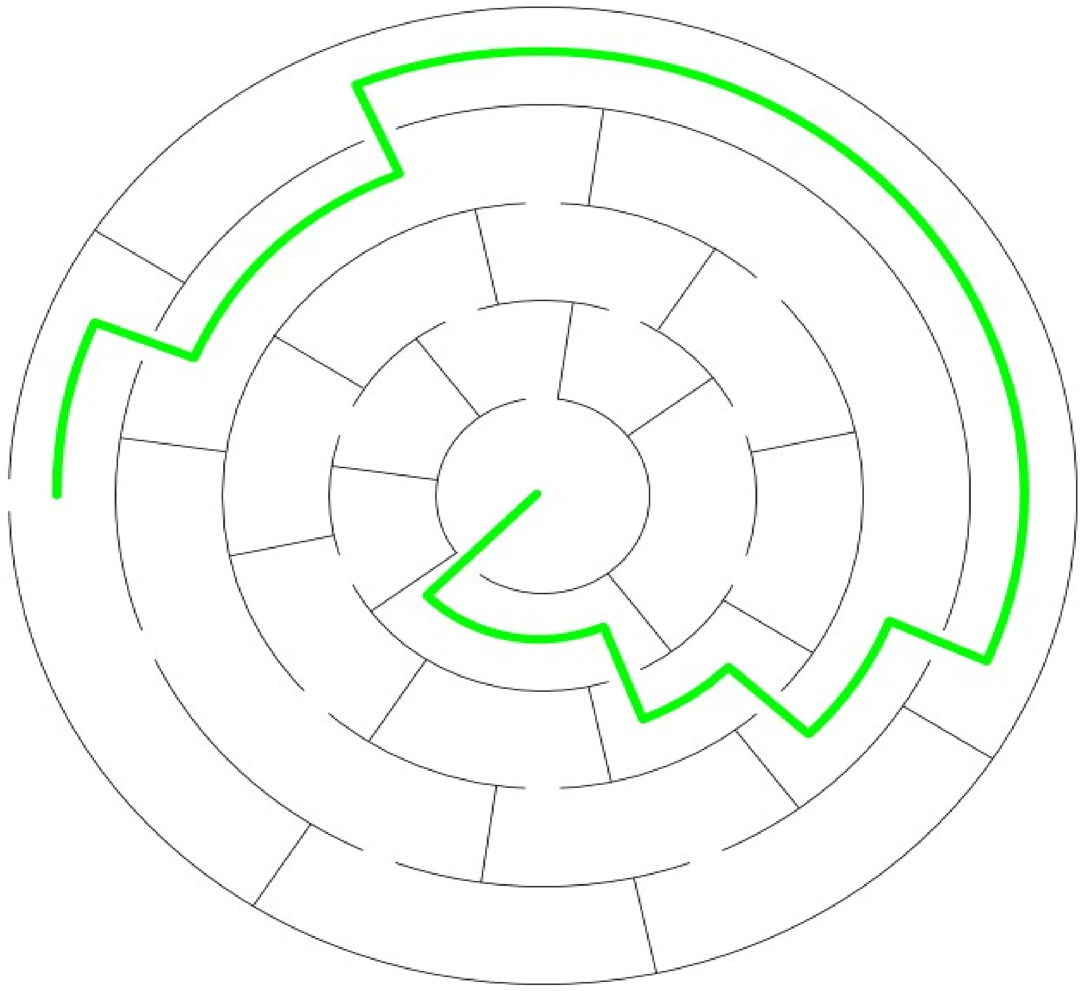
- This is iterative deepening and will be covered in your Artificial Intelligence course (p244/265)

```
% our result value R is D different from the Target
solve([Soln|_],Target,Soln,D) :- eval(Soln,R),
                                 diff(Target,R,D).
% recursive case is akin to the equivalent countdown/3
solve(L,Target,Soln,D) :- choose(2,L,[A,B],R),
                          arithop(A,B,C),
                          solve([C|R],Target,Soln,D).
% search for a solution decreasingly close to the target
solve(L,Target,Soln) :- range(0,100,D),
                        solve(L,Target,Soln,D).
```

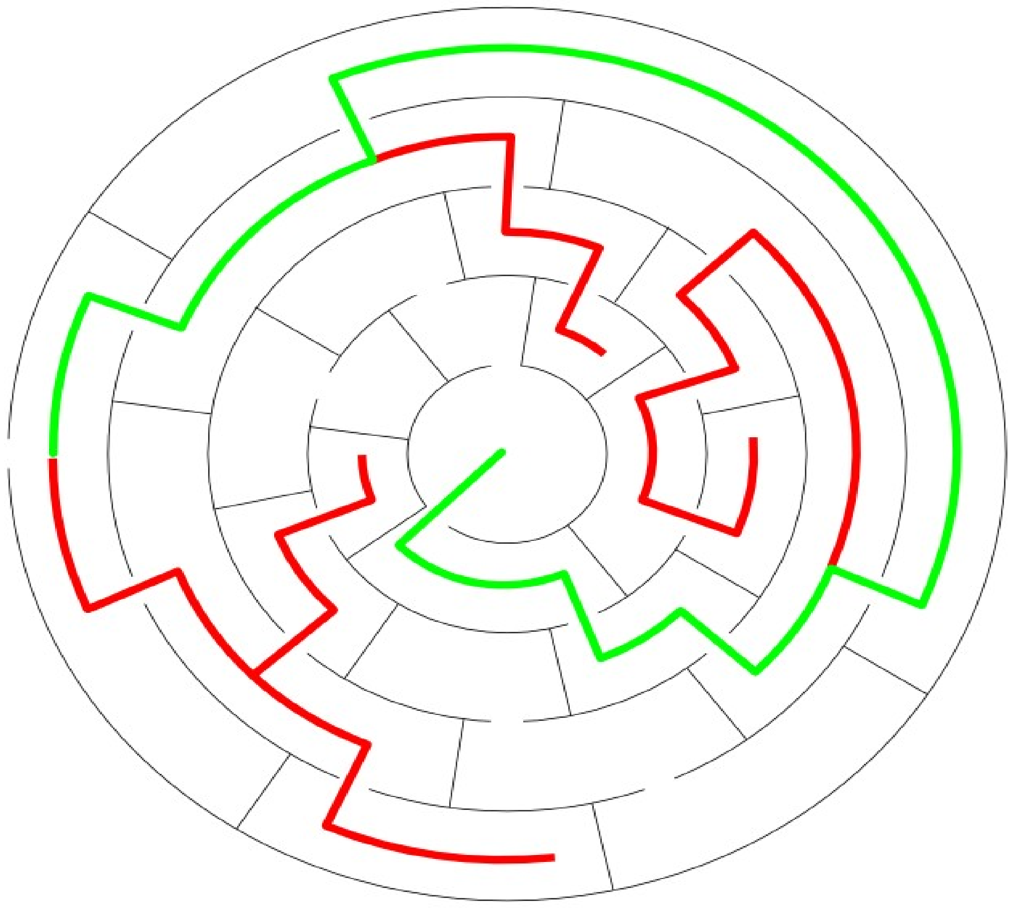
Searching



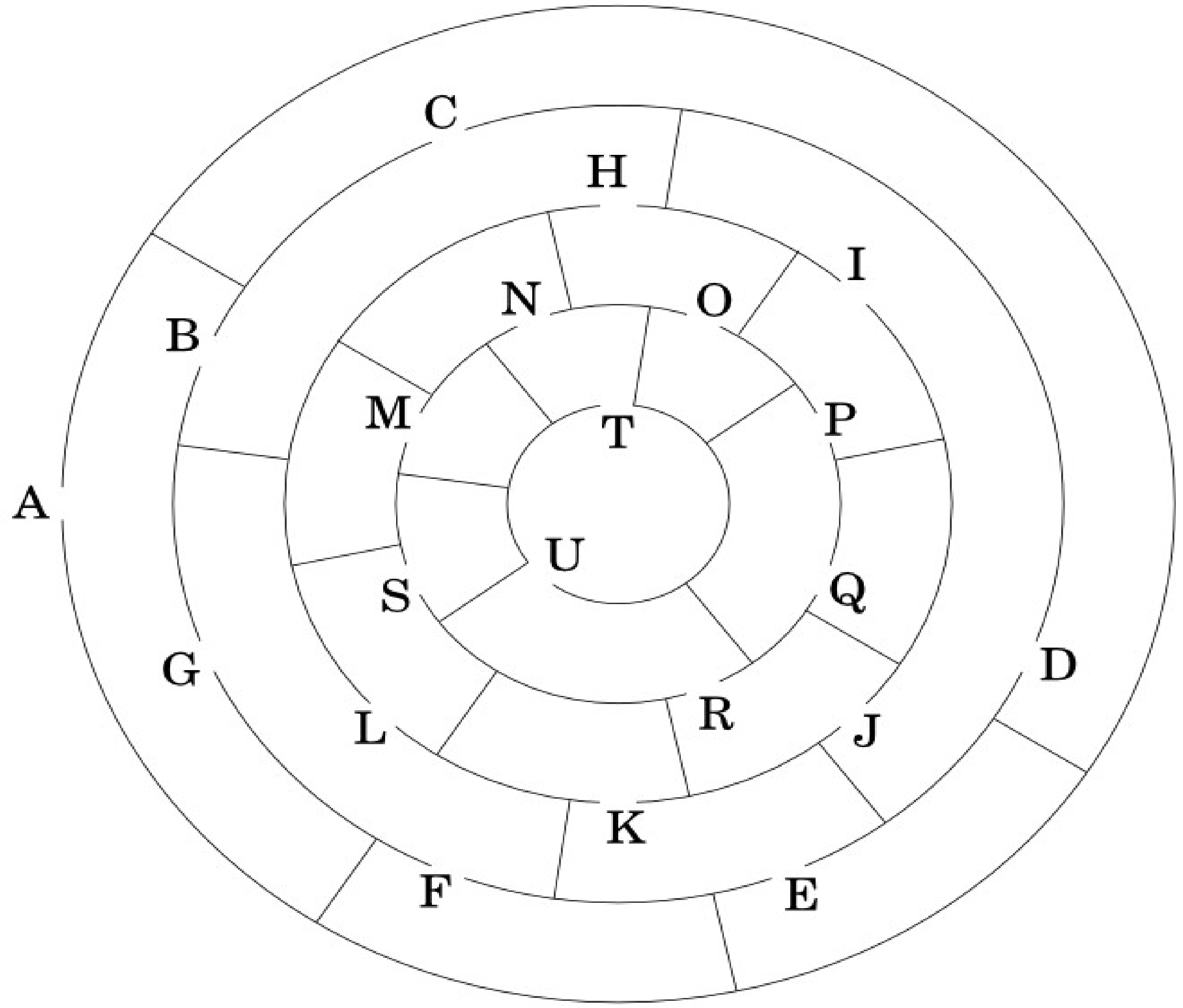
Searching: maze solution



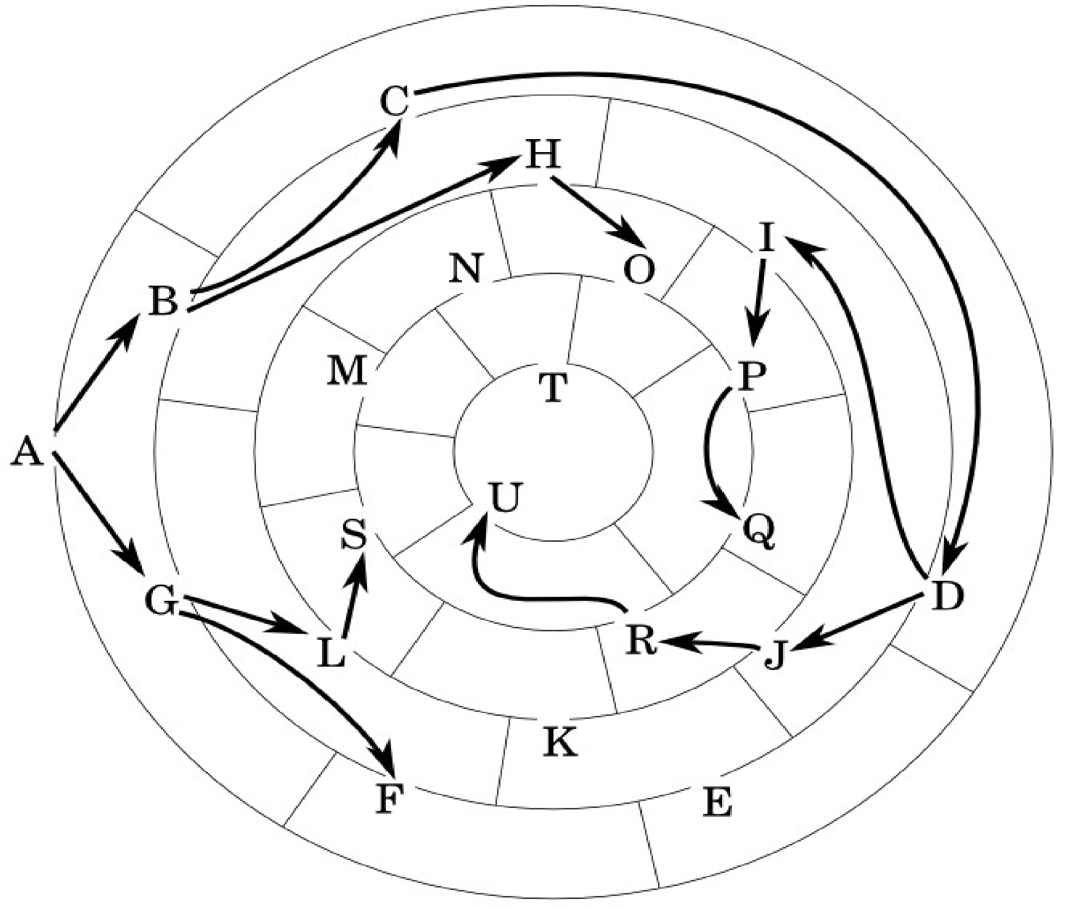
Searching: solution and failed paths



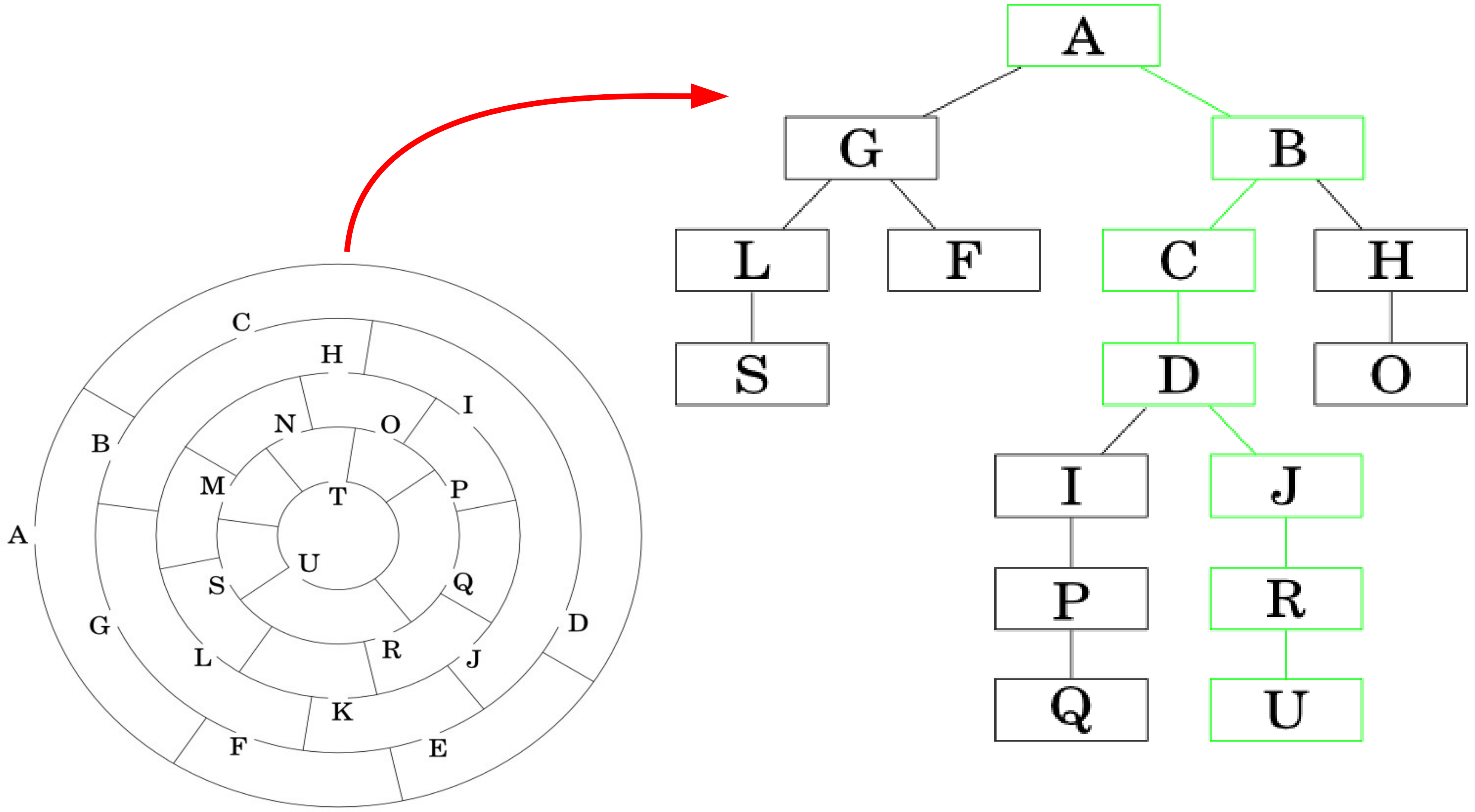
Searching: represent the problem



Searching: possible paths



Search space as a tree



Finding a route through the maze

```prolog
start(a).
finish(u).

route(a,g).
route(g,l).
route(l,s).
% ...
travel(A,A).
travel(A,C) :- route(A,B),travel(B,C).

solve :- start(A),finish(B), travel(A,B).
```

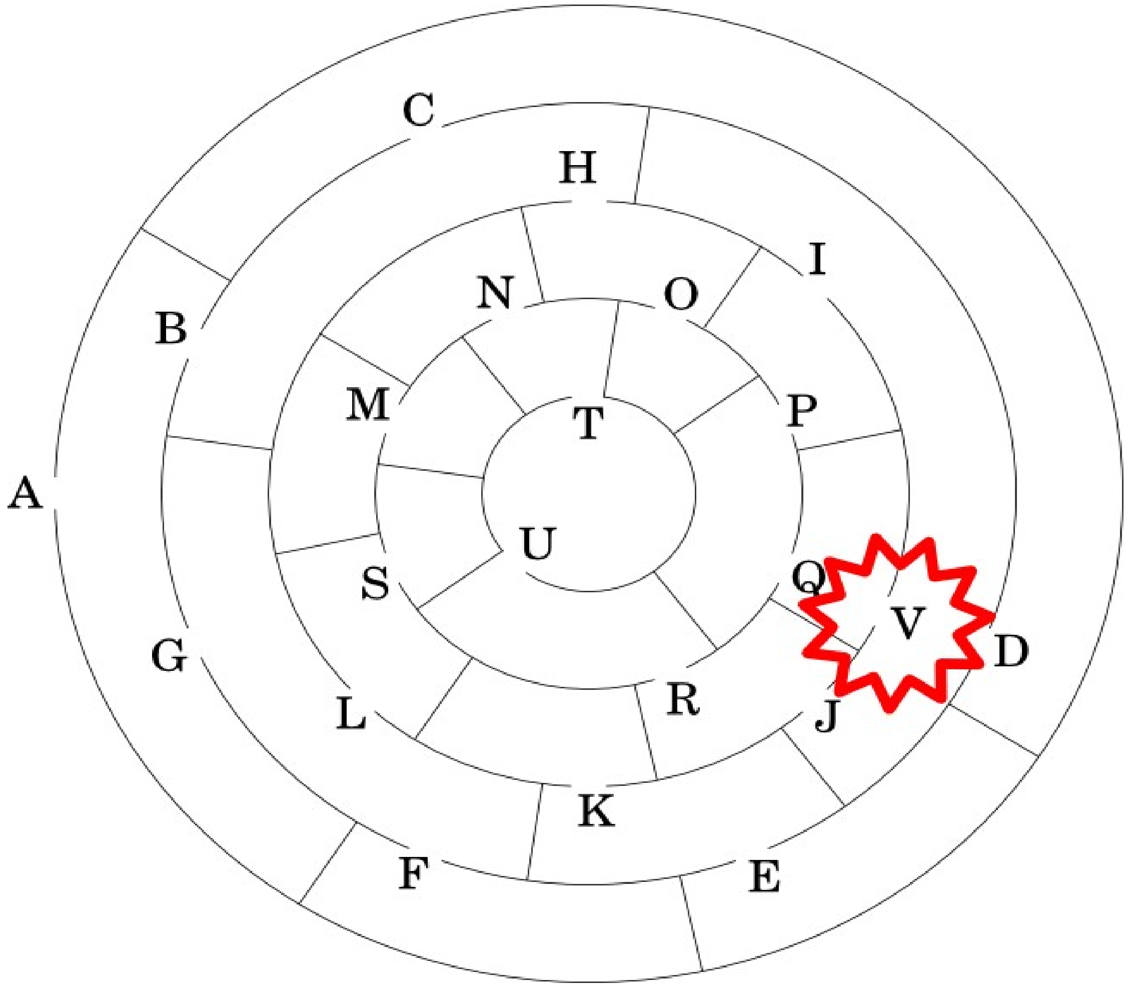
We need to remember the route

From “is there a route?” to “show us a route.”

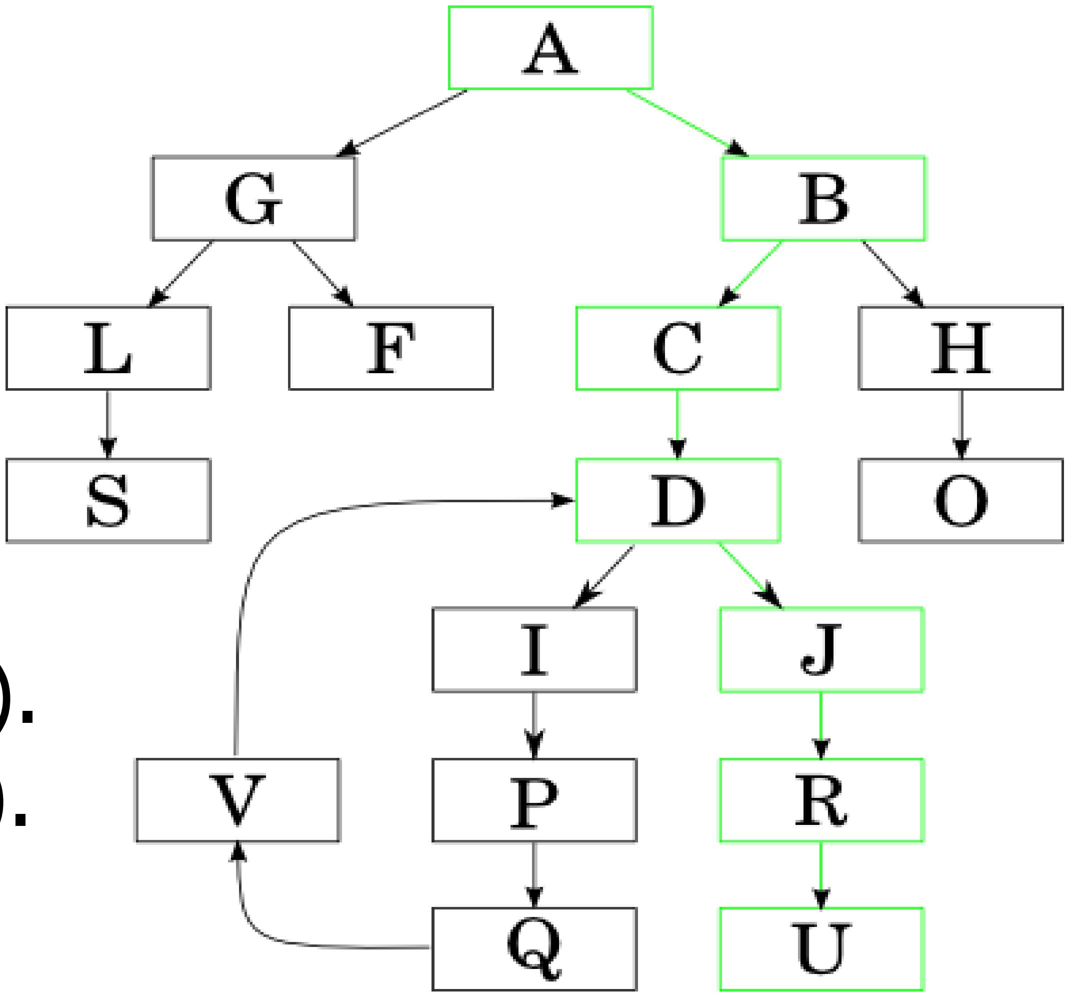
```
travellog(A,A,[]).
travellog(A,C,[A-B|Steps]) :-
    route(A,B), travellog(B,C,Steps).

solve(L) :- start(A), finish(B),
    travellog(A,B,L).
```

What if we have a cyclic graph?



Cyclic Graphs



```
route(q,v).
route(v,d).
```

Searching

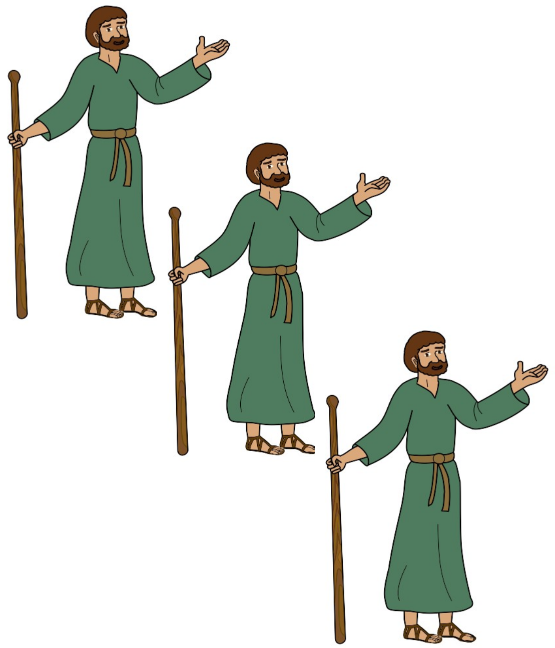
Solution: maintain a set of places we've already been – the closed set

- In SWI Prolog you can (and should) write \+ to mean not()

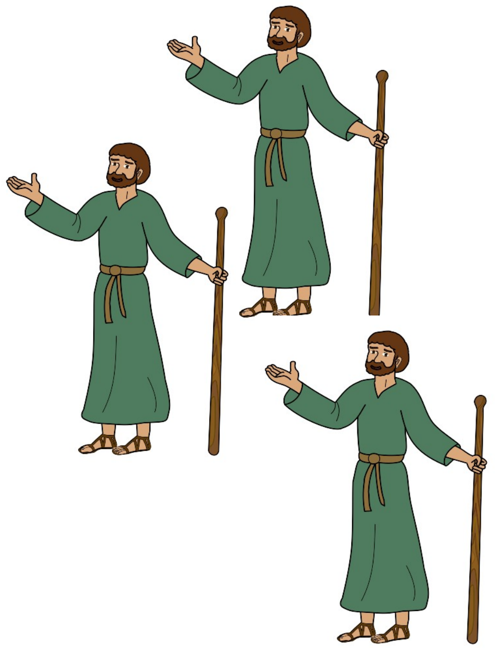
```
travelsafe(A,A,_).
travelsafe(A,C,Closed) :-
    route(A,B), \+ member(B,Closed),
    travelsafe(B,C,[B|Closed]).
```

Accumulate the list of nodes that we've visited

Missionaries and Cannibals



3 Missionaries

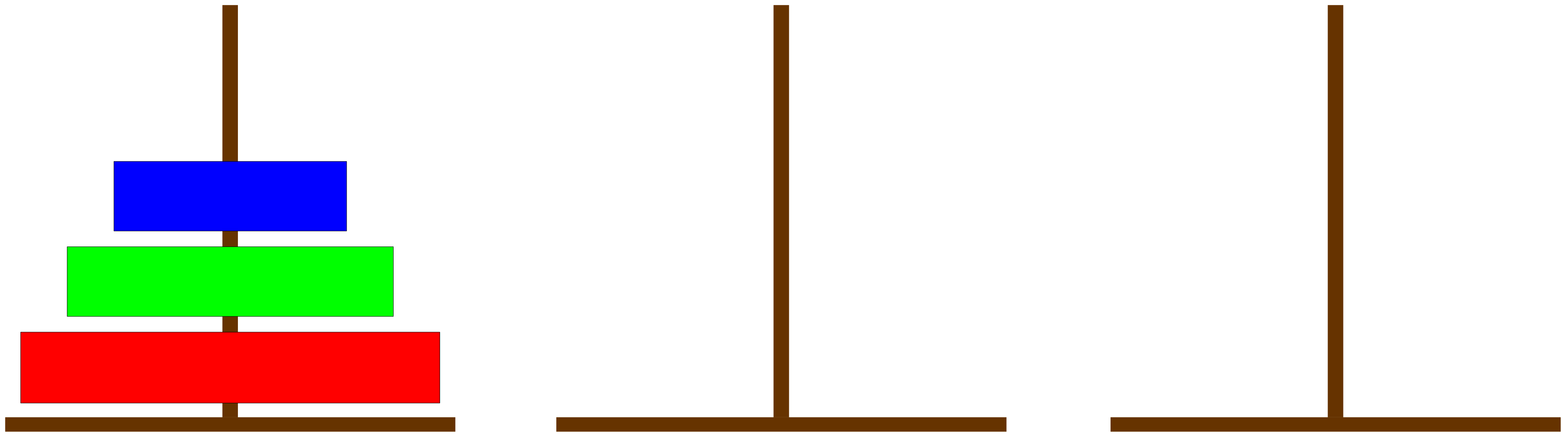


3 Cannibals

1 boat

- The boat carries two people
- If the Cannibals outnumber the Missionaries they will eat them
- Get them all from one side of the river to the other?

Towers of Hanoi



Umbrella problem

A group of 4 people, Andy, Brenda, Carl, & Dana, arrive in a car near a friend's house, who is having a large party. It is raining heavily, & the group was forced to park around the block from the house because of the lack of available parking spaces due to the large number of people at the party. The group has only 1 umbrella, & agrees to share it by having Andy, the fastest, walk with each person into the house, & then return each time. It takes Andy 1 minute to walk each way, 2 minutes for Brenda, 5 minutes for Carl, & 10 minutes for Dana. It thus appears that it will take a total of 19 minutes to get everyone into the house. However, Dana indicates that everyone can get into the house in 17 minutes by a different method. How? The individuals must use the umbrella to get to & from the house, & only 2 people can go at a time (& no funny stuff like riding on someone's back, throwing the umbrella, etc.).