Overall Outline of topics covered in the course:

Knowledge-based AI:

* Knowledge represented declaratively
* Using symbols to represent semantic meaning
* Based on facts
* Modular
* Interpretable – transparent
* Consists of basically three parts:
  + Knowledge sources
    - Vision
    - Language
  + Knowledge base
  + Actions
    - Manipulation
    - Natural language
  + Reasoning procedures act on and add to knowledge base

Corps-based Learning

* Task specific
* Corus of data
  + Labeled: instances labeled with the correct response/answer (supervised learning)
  + Unlabeled: data is not marked as correct or not (unsupervised learning)
  + Mixture: some of the data is labeled (semi-supervised learning)
* Learning program
* Task executor

Search

Games (chess, Go)

Logic and Inference

Plausible Reasoning

Probability

Machine Learning

Breadth-first search

N Queens Problem (also used for Sudoku solver):

Place N queens on a chess board such that none are in the same column, row, or diagonal

Use depth-first search: use a start state, search successor states until you find an invalid state, backtrack to a valid state and then try alternative successor

This creates a tree of states

Psedocode:

DFS-TOP(){

Return(dfs(start))

}

DFS(){

If (goal(S)){

Return(S)

}

Loop (for each child c of S)

Answer = DFS(c)

If (answer!=fail) return(fail)

Return(fail)

End loop

}

Complexity: (B^(D+1)-1)/(B-1), or asymptotically B^D, where B is the maximum branching factor, and D is the dimension of the state space.

Can also use to find Hamiltonian paths in a directed graph

This is an npr complete problem – can not be solved in polynomial time

Can also be applied to the 3-coloring problem of a planar map

However the branches alternate between deciding on which area to color, and which color it should be

Breadth-first search:

No recursive solution similar to the depth-first case

Pseudocode:

BFS(){

Fifo q = {start}

Loop (until Q is empty){

S= pop(Q)

For (each child c of S){

If (goal(c) return(c))

Add c to tail of Q

}

}

Return(fail)

}

Generally inferior to DFS, because it will probably search more of the tree, and memory requirements are much greater

However, it may be more efficient if the goal state is not necessarily at the end of the tree – i.e. we don’t know how many levels are necessary to find solution

Example: find the minimum number of queens necessary on the board such that all squares are being attacked. It is not so obvious how many queens are necessary, so we don’t know a priori the depth of the tree. This is a situation where BFS may be better.

There may also be situations in which the sate space in infinite, in which case the DFS may never terminate, but BFS will.

Example: Post Correspondence problem:

Given a collection of dominoes, each with a character string on top and bottom. Can have as many of these dominoes as you want – potentially infinite. Problem: line up the dominoes so that the strings concatenated on the top is the same as the strings concatenated on the bottom.