AI Experimental Course

School of Data and Computer Science
Sun Yat-sen University

Overview

Uninformed Search

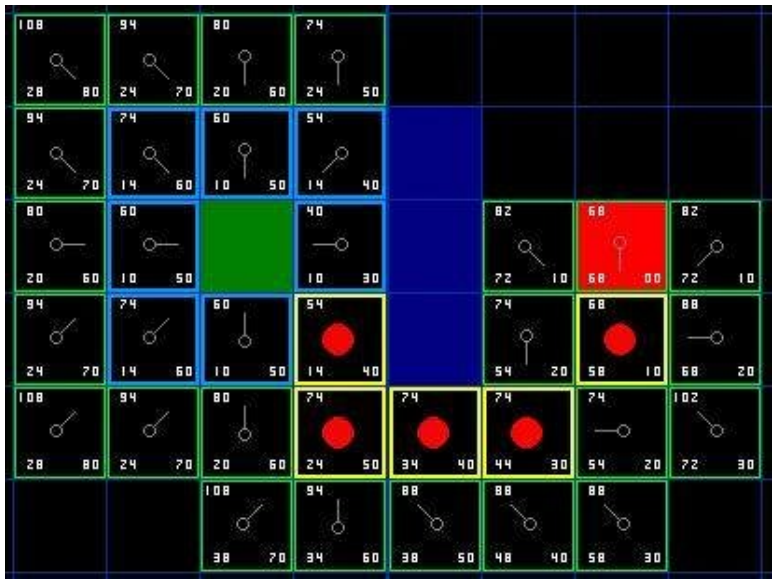
- DFS
- BFS
- Uniform-cost search
- Depth-limited search
- Iterative-Deepening search

function ITERATIVE-DEEPENING-SEARCH(*problem*) **returns** a solution, or failure
  for *depth* = 0 **to** ∞ **do**
    result $\leftarrow$ DEPTH-LIMITED-SEARCH(*problem*, *depth*)
    if *result* $\neq$ cutoff **then return** *result*

- Bidirectional search

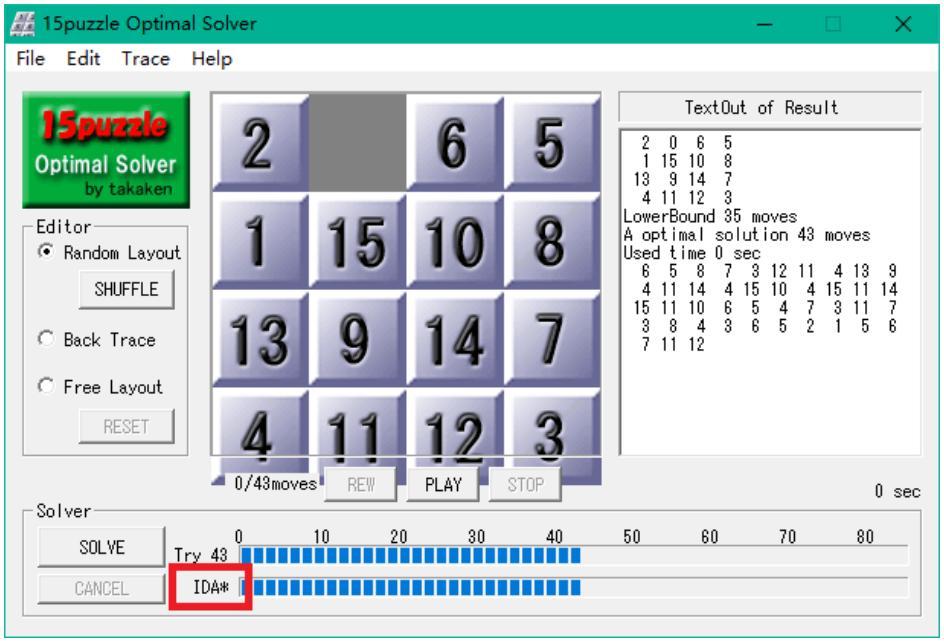
Informed (Heuristic) Search

- Greedy best-first search
- A* (https://www.redblobgames.com/pathfinding/a-star/introduction.html)
  - $f(n) = g(n) + h(n)$
  - $f(n)$ is the estimated cost of the cheapest solution through n
  - $g(n)$ is the path cost from the start node to node n
  - $h(n)$ is the estimated cost of the cheapest path from n to the goal



Informed (Heuristic) Search (cont'd)

- Iterative-deepening A* (IDA*)



- Recursive best-first search (RBFS)

Adversarial Search

- The minimax algorithm
- $\alpha - \beta$ pruning

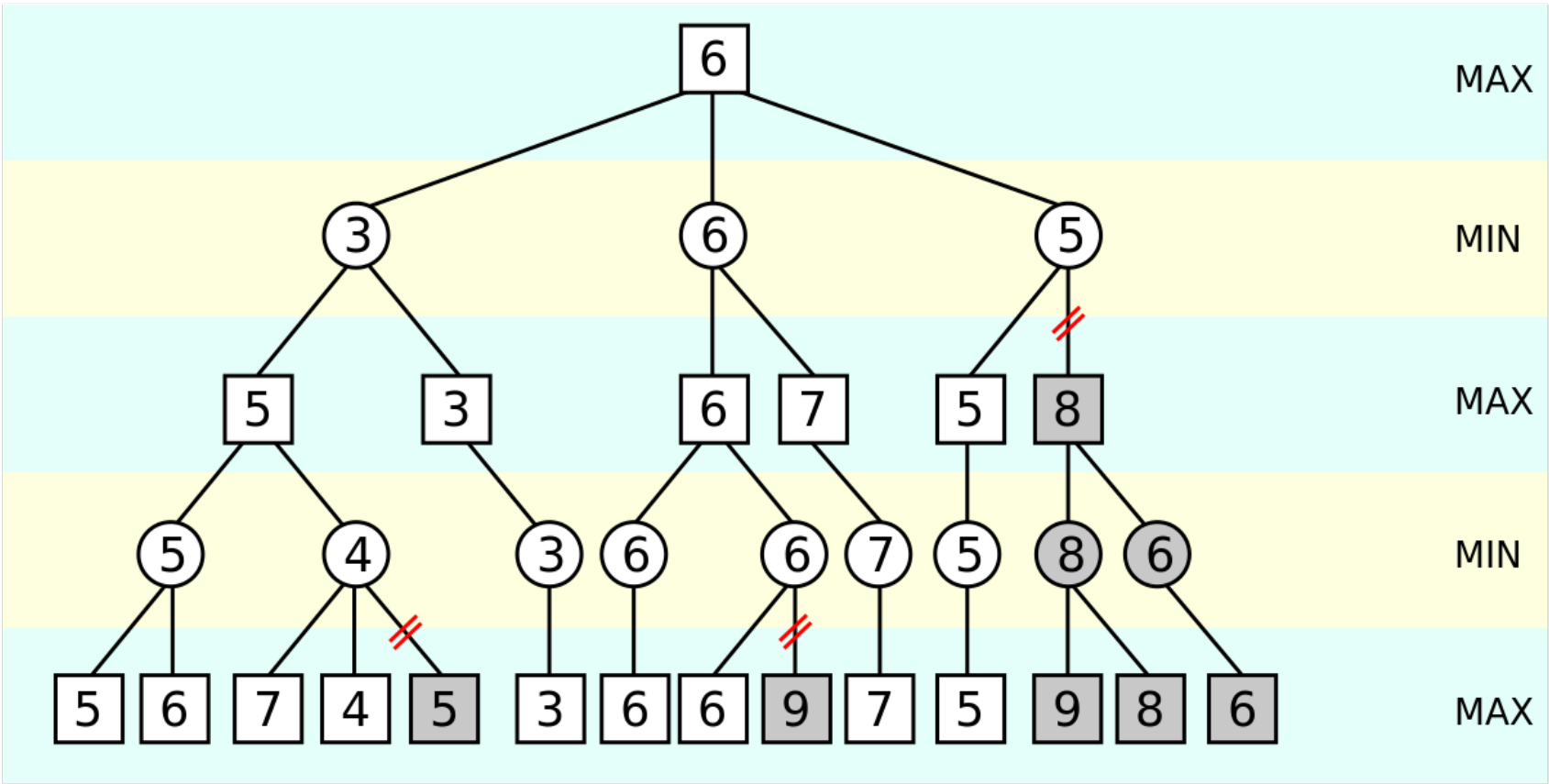


Figure 1: $\alpha - \beta$ pruning

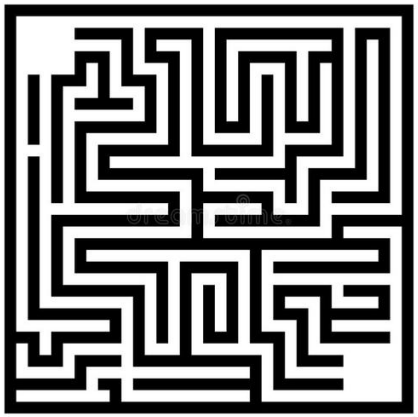
Constraint Satisfaction Problems (CSPs)

- Backtracking Search for CSPs
- Forward Checking (FC)
- Generalized Arc Consistency (GAC)

Enforce GAC (prune all GAC inconsistent values)

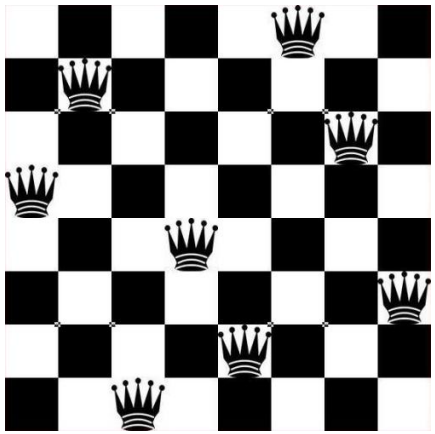
```
GAC_Enforce()
   // GAC-Queue contains all constraints one of whose variables has
   // had its domain reduced. At the root of the search tree
   // first we run GAC_Enforce with all constraints on GAC-Queue
   while GACQueue not empty
       C = GACQueue.extract()
       for V := each member of scope(C)
           for d := CurDom[V]
                Find an assignment A for all other
                variables in scope(C) such that
                C(A ∪ V=d) = True
                if A not found
                   CurDom[V] = CurDom[V] - d
                   if CurDom[V] = ∅
                        empty GACQueue
                        return DWO //return immediately
                   else
                        push all constraints C' such that
                        V ∈ scope(C') and C' ∉ GACQueue
                        on to GACQueue
   return TRUE //while loop exited without DWO
```

Typical Problems

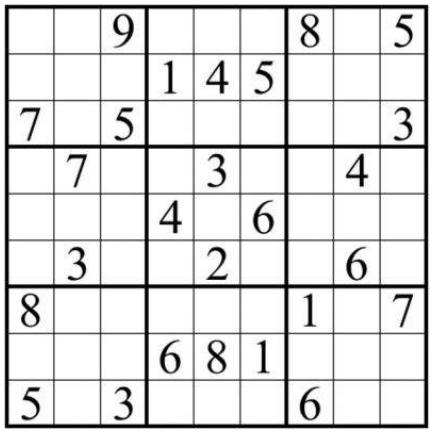


(a) Maze

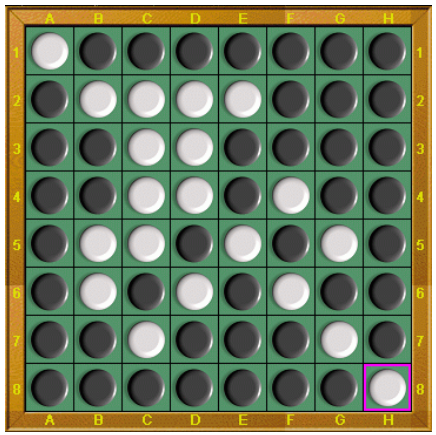
(b) 8 Puzzle



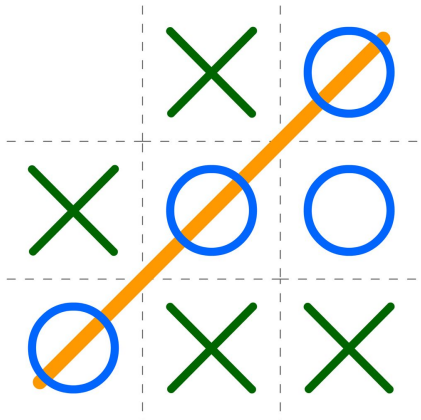
(c) 8 queens puzzle



(d) Sudoku



(e) Othello



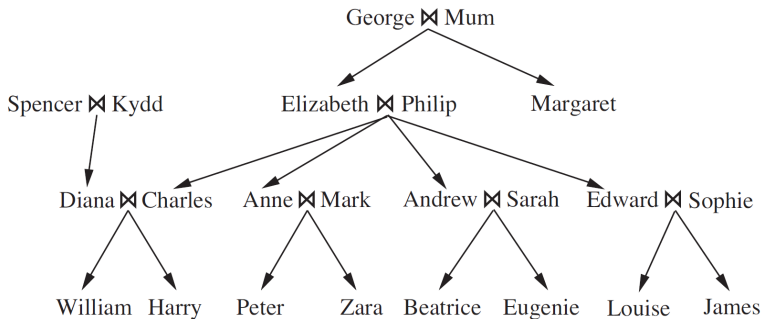
(f) Tic Tac Toe

Figure 2: Typical Problems

1. Search
   - Uninformed Search
   - Informed (Heuristic) Search
   - Adversarial Search
   - Constraint Satisfaction Problems (CSPs)
   - Typical Problems
2. Prolog
   - Typical Problems
3. FF Planning System
   - Typical Problems
4. Machine Learning
   - Typical Algorithms
5. Course Requirements

Typical Problems

- Search Problems
- KR (e.g. Family Problem)
- Queries on KB (Similar to SQL)
- CSPs
  - Sudoku Problem, Eight Queens Problem and Other Games



Examples

Example 1 (Family Problem)

```
grandChild(A,B):-child(A,C),child(C,B).
aunt(A,B):-child(B,C),sister(A,C).
...
male('George').
child('Elizabeth','George').
...
```

Example 2 (Tower of Hanoi)

```
hanoi(N):-move(N,a,b,c).
move(1,A,_,C):-inform(A,C).
move(N,A,B,C):-N1 is N-1, move(N1,A,C,B),
               inform(A,C), move(N1,B,A,C).
inform(Loc1,Loc2):-nl, write('from '), write(Loc1),
               write(' to '), write(Loc2).
```

1. Search
   - Uninformed Search
   - Informed (Heuristic) Search
   - Adversarial Search
   - Constraint Satisfaction Problems (CSPs)
   - Typical Problems
2. Prolog
   - Typical Problems
3. FF Planning System
   - Typical Problems
4. Machine Learning
   - Typical Algorithms
5. Course Requirements

Typical Problems

- Blocks Problem
- Logistic Problem
- 8-puzzle Problem
- Freecell Game
- Boxman Game

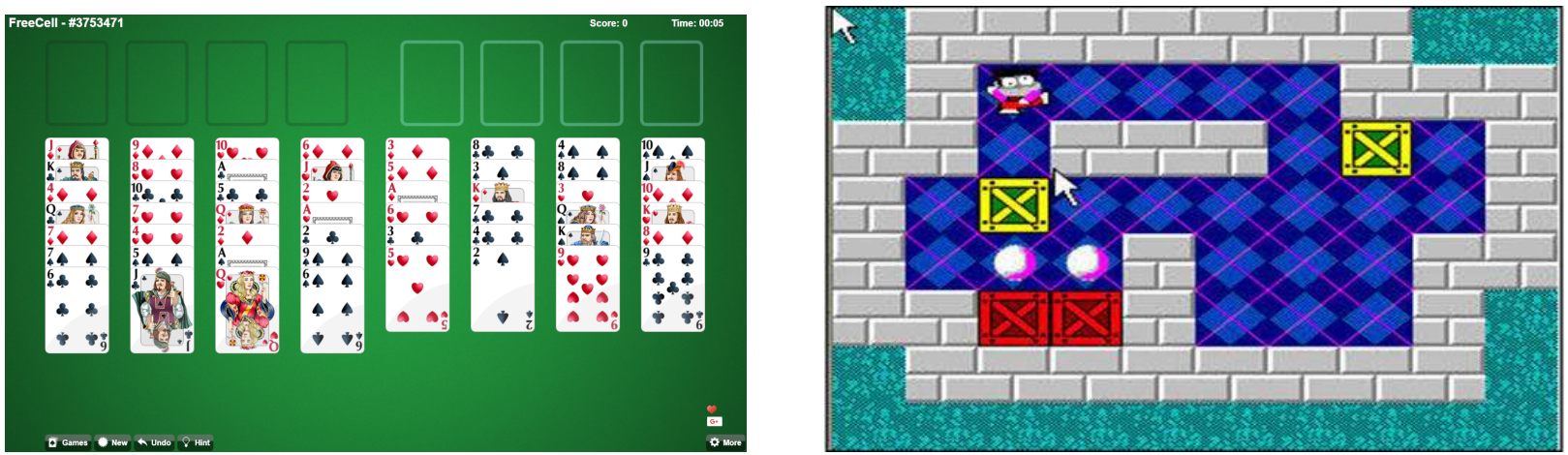


Figure 3: Freecell Game and Boxman Game

Examples

Spare Tire (domain)

```
(:action Remove
:parameters (?x - physob ?y - location)
:precondition (At ?x ?y)
:effect (and (not (At ?x ?y)) (At ?x Ground)))
(:action PutOn
:parameters (?x - physob)
:precondition (and (Tire ?x) (At ?x Ground)
(not (At Flat Axle)))
:effect (and (not (At ?x Ground)) (At ?x Axle)))
```

Examples (cont'd)

Spare Tire (data)

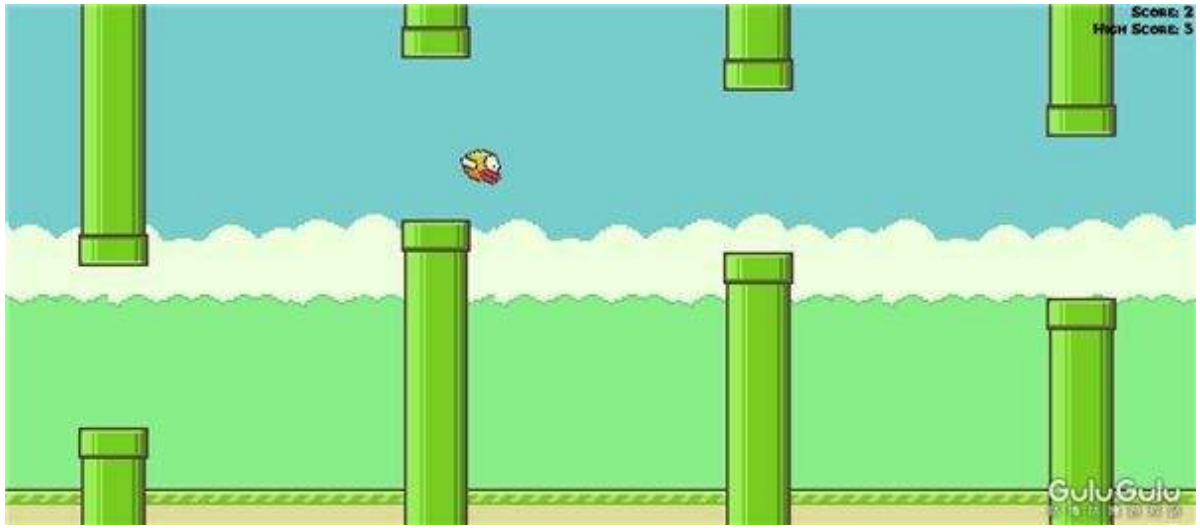
```
(define (problem prob)
(:init (Tire Flat)(Tire Spare)(At Flat Axle)(At Spare
Trunk))
(:goal (At Spare Axle))
)
```

Typical Algorithms

- Probabilistic Reasoning (Bayesian Network)
- Decision Tree (ID3, C4.5 and CART)
- Naive Bayes
- EM Clustering

Typical Algorithms (cont'd)

- BP-Neural Network
- Deep Learning
- Reinforcement Learning (Q learning and Sarsa)
- Deep Q-Learing (DQN)



1. Search
   - Uninformed Search
   - Informed (Heuristic) Search
   - Adversarial Search
   - Constraint Satisfaction Problems (CSPs)
   - Typical Problems
2. Prolog
   - Typical Problems
3. FF Planning System
   - Typical Problems
4. Machine Learning
   - Typical Algorithms
5. Course Requirements

Course Evaluation

Score: weekly exercises (30%) + 4 projects (70%).

Evaluation Criterion:

- Submission Time
- Correctness
- Efficiency
- Readability

Academic Honesty

- The work you submit must be your own
  - you are encouraged to discuss with each other
  - but write up and code independently
- If plagiarism is caught, all parties involved will receive 0 on the assignment/project
- If you have any questions, please turn to the instructor for help.

Reference

- Stuart Russell and Peter Norvig, Artificial Intelligence: A Modern Approach.
- Introduction to A*, https://www.redblobgames.com/pathfinding/a-star/introduction.html.
- A*, https://en.wikipedia.org/wiki/A*_search_algorithm
- A* Pathfinding for Beginners, https://www.gamedev.net/reference/articles/article2003.asp.

Reference (cont'd)

- "15puzzle Optimal solver" for windows, `http://www.ic-net.or.jp/home/takaken/e/15pz/`.
- Boxman Game, `http://www.pc6.com/softview/SoftView_56018.html`
- `https://github.com/chncyhn/flappybird-qlearning-bot`.
- slides from Sheila McIlraith, CSC384, University of Toronto, Winter 2016

The End