

# Problem & Solving

## Unit 5

# You Will Know

- Problem and solving
- Production system
- Search
- Game Theory

# Problem and Solving

- In general
  - Problem : what goals do you want to reach?
  - Solving: how do you reach the goals?
- We want to build problem-solving agents for our AI problems
- Problem formulation
  - the process of deciding what actions and states to consider, given a goal

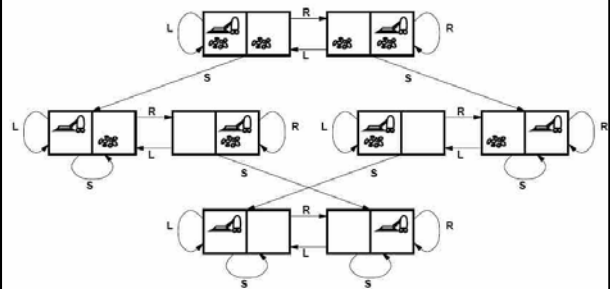
# Problem Formulation

- Four components in a problem
  - Initial state the agent starts in it.
  - Actions: possible actions available to the agent
  - Goal test: determine whether a given state is the goal
  - Path cost: assign a numeric cost to each path
- Optimal solution
  - The lowest path cost among all solutions

# Example: Vacuum Cleaner

1		2		<p><b>Goal:</b> clean the house in state 7 or 8</p> <p><b>Initial:</b> in state 5</p> <p><b>Actions:</b> {left, right, suck}</p> <p><b>Solution path:</b> {right, suck}</p>
3		4		
5		6		
7		8		

# Vacuum Cleaner Space



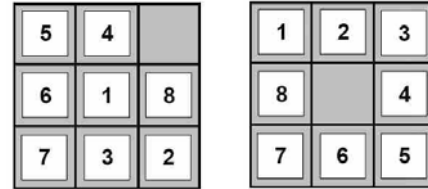
## Example: Vacuum Cleaner

- Problem formulation
  - States:  $2 \times 2^2 = 8$  possible world states
  - Initial state: any state can be designated as the initial state
  - Successor function (or actions): (left, right, suck)
  - Goal test: This check whether all the square are clean
  - Path cost: Each step costs 1, so the path cost is the number of steps in the path

## Example: 8-puzzle

**Goal:** goal state  
**Actions:** black move {left, right, up, down}

**Initial:** The start state  
**Path cost:** unit cost



Start State

Goal State

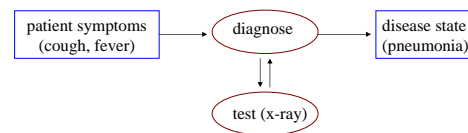
## Example: 8-puzzle

- Problem formulation
  - States: the **location** of each of the eight tiles and the **black** in one of the nine square
  - Initial state: any state can be designated as the initial state
  - Successor function (or actions): (left, right, up, down)
  - Goal test: This check whether the state matches the goal
  - Path cost: Each step costs 1, so the path cost is the number of steps in the path

## Example: Diagnosis

**Goal:** disease state  
**Actions:** test, diagnose

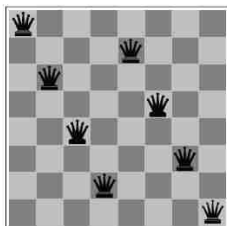
**Initial:** symptom  
**Path cost:** test cost



## N-Queen Problem

**Goal:** No two queens attacking  
**Actions:** Place a queen on square (i, j)

**State:** Placement of n queens  
**Path cost:** zero cost



## N-Queen Problem

- Problem formulation (please fill it)
  - States:
  - Initial state:
  - Successor function (or actions):
  - Goal test:
  - Path cost:

## Production System

- One type of problem-solving systems
- Three components in production system
  - Global database
  - Set of rules
  - Control system
- Data driven and data adaptive
- Take an example: **chess**

## Global Database

- Information of the problem
  - Environment description
  - Facts
  - Parameters of rule inferences
- In practice, global database means designing data structure for environment
- **What's the global database for chess?**

## Set of Rules

- Rule
  - Cover the ability and possibility of the agent
  - Rule inference can produce new facts
- The general form of a rule
  - Conditions => Actions
  - Facts => Results
  - If Condition then Action A
  - else Action B

**What's the rule for chess?**

## Control System

- Inference engine for rules
  - Select proper rules for observing facts
  - Enable rules
  - Disable rules
  - Goal: looking for an optimal path for solutions

**What's the control system for chess?**

## Search

- Why do we need search
  - Many many possible solutions for a problem
  - An optimal solution: minimum cost
- It is often impossible to test all possible solutions in difficult problems
  - Question: Who is the tallest in the world?
  - We need smarter strategies for search
  - How smart? What's smart?

## Search

- Complexity
  - The cost of the search approach
  - In most cases, we use "complexity" to measure the cost of an algorithm
  - $O(n)$ ,  $O(n^2)$ ,  $O(n \log n)$ , ...
  - Two kinds of complexity
    - Time complexity
    - Space complexity

## Complexity Example

- Given a data sequence with size 1000000 items, does it contain the number "1205"?
  - sequential search:
    - It may be success at the first time if you are so lucky
    - $O(n)$ : 1000000 times at the worst case
  - binary search:
    - It may be success at the first time if you are so lucky
    - $O(n \log n)$ : 1000 times at the worst case
    - But the data sequence need to be sorted

## Complexity

- Consider the example at the previous page
  - If the data size is 100 items?
  - If the data size is 10000 items?
  - If the data size is 100000000 items?
- Scalability issue
  - The performance, either time or space, of an algorithm depends on the size of the data

## Complexity

- The search problem is
  - P problem
    - There exists a polynomial-time algorithm to solve the search problem
    - This is not very difficult
  - NP-Complete
    - No polynomial-time solution can be found
    - That means that we can solve the problem if the data size is not small
    - We need to find a non-optimal solution
    - e.g., 0-1 knapsack

## Search Space

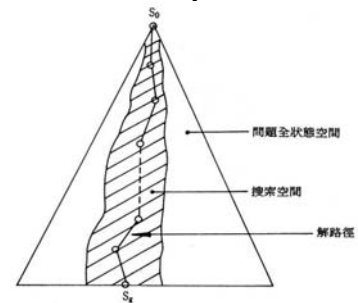


圖 2.1 搜索空間示意圖

## Quality of A Search Algorithm

- Complete
  - Is it a correct solution?
- Optimality
  - Is it high quality?
- Time complexity
  - How long to find a solution?
- Space complexity
  - How much memory is required to find a solution?

## 4-Queen Problem

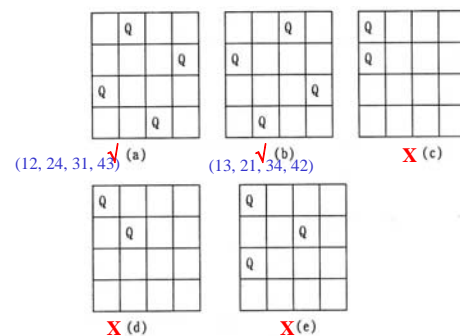
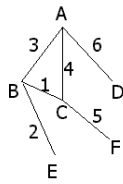


圖 2.2 四皇后問題棋盤的幾個態勢



## Exercise

- Start from A
  - Travel by DFS
  - Travel by BFS



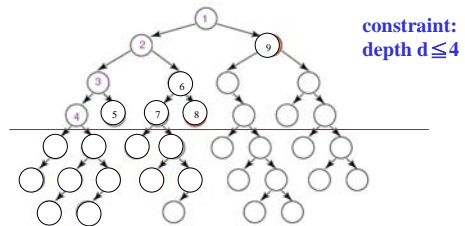
## Which Strategy to Use

- Depends on your problem.
- If there are infinite paths
  - DFS is bad
- If goal is at know depth
  - DFS is good almost
- If there exist large branching factor
  - BFS is bad
- Consider data structure in your program.

## Improvement

- Either in DFS or BFS
  - These are basic approaches for graph traversal
  - Cost is very high if the graph is huge
  - It is necessary to cost down
- Set constraints on search
  - Take some limited criteria, not optimal solution
  - Example: looking for a pair of shoes at shopping
    - Optimal: looking for the cheapest one
    - Constraint: looking for the one less than NT\$1000
  - Depth-limited Depth-first Search

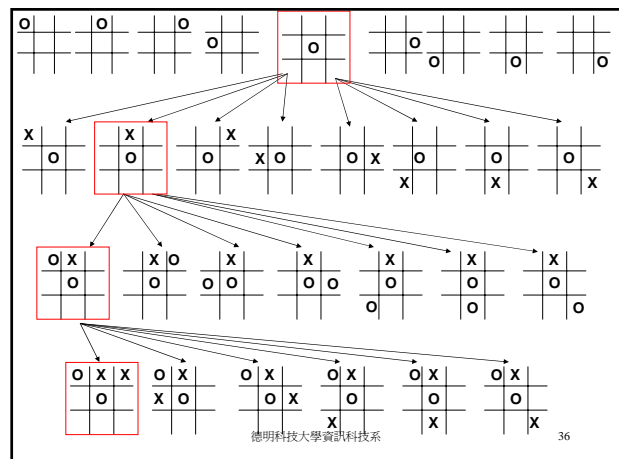
## Depth-limited Depth-first Search



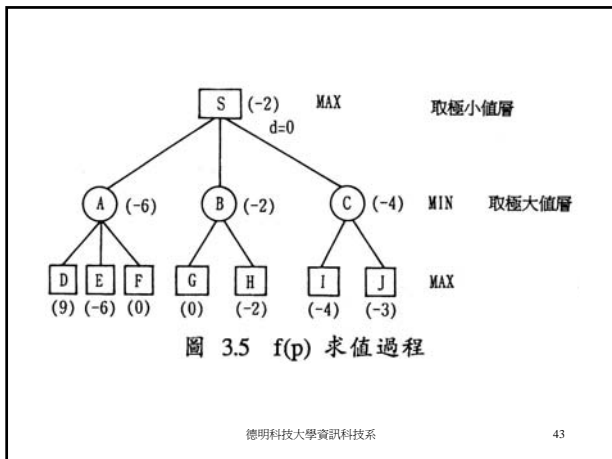
Complete: No, unless at depth  $\leq d$     Time:  $O(b^d)$   
 Optimal: No    Space:  $O(b^d)$

## Game Theory

- Game
  - A set of game rules
  - Players' decision to win the game
- 單人遊戲的問題
  - 找出最佳的結果
  - 可用一般的搜索技術進行求解
  - 8-puzzle; n-queen; 接龍
- 雙人遊戲
  - 對方同時也想取得最佳的結果
  - 可用與或樹(或圖)來描述，因而搜索就是尋找最佳解樹(或圖)的問題
  - Chess; Chinese chess; Go; Bridge







### Minimax algorithm

```

function Minimax-Decision(game) returns an operator
  for each op in Operators[game] do
    Value[op] ← Minimax-Value(Apply(op, game), game)
  end
  return the op with the highest Value[op]

function Minimax-Value(state, game) returns a utility value
  if Terminal-Test[game](state) then
    return Utility[game](state)
  else if max is to move in state then
    return the highest Minimax-Value of Successors(state)
  else
    return the lowest Minimax-Value of Successors(state)
  
```

德明科技大學資訊科技系

### Properties of MiniMax

**Complete:** Yes, if tree is finite  
[chess has specific rules for this]

**Optimal:** Yes, against an optimal opponent.  
Otherwise??

**Time complexity:**  $O(b^m)$

**Space complexity:**  $O(bm)$   
(depth-first exploration)


For chess:  
 $b \approx 35, m \approx 100$  for "reasonable" games  
⇒ exact solution completely infeasible

45

### Resource Limits

- Example: Chess(西洋棋)
  - a reasonable game:  $O(35^{100})$  nodes
  - suppose we have 10 seconds/move
  - If explore  $10^9$  nodes/second
  - ⇒  $10^{10}$  nodes per move
  - ⇒ not enough

Solutions: 1. cutoff test, eg: depth limit  
2. evaluation function  
or estimated desirability of position



46

### Cutoff Search

- Depth-limit in Minimax search
  - e.g. depth < 6
  - means the player can think less than 6 steps
- Does it work in practice?
  - $b^m = 10^6, b = 35, \Rightarrow m = 4$
  - $m=4 \Rightarrow$  the computer can think less than 4 steps
- In chess
  - $m=4$ , human novice
  - $m=8$ , typical pc, human master
  - $m=12$ , deep blue, kasparov

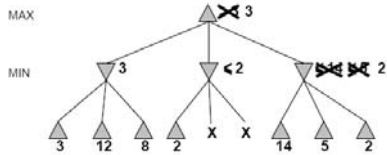
47

### $\alpha - \beta$ Pruning

MINIMAX 程序是把搜索樹的產生和格局估值這兩個程序分開來進行，即先產生全部搜索樹，然後再進行端節點靜態估值和倒推值計算，這顯然會導致低效率。如圖3.8中，其中一個 MIN 節點要全部產生 A、B、C、D 四個節點，然後還要逐個計算其靜態估值，最後在求倒推值階段，才賦給這個 MIN 節點的倒推值  $-\infty$ 。其實，如果產生節點 A 後，馬上進行靜態估值，得知  $f(A) = -\infty$  之後，就可以斷定再產生其餘節點及進行靜態計算是多餘的，可以馬上對 MIN 節點賦倒推值  $-\infty$ ，而絲毫不會影響 MAX 的最好優先走步的選擇。這是一種極端的情況，實際上把產生和倒推估值結合起來進行，再根據一定的條件判定，有可能儘早修剪掉一些無用的分枝，同樣可獲得類似的效果，這就是  $\alpha - \beta$  程序的基本思想。下面再用一字棋的例子來說明  $\alpha - \beta$  剪枝方法。

48

## $\alpha - \beta$ Pruning: Example



德明科技大學資訊科技系

49

## $\alpha - \beta$ Pruning: Example

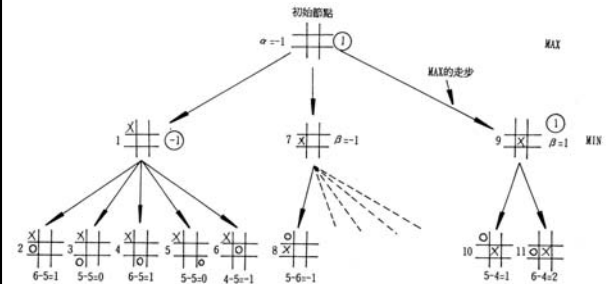


圖 3.9 一字棋第一階段  $\alpha - \beta$  剪枝方法

德明科技大學資訊科技系

50

## $\alpha - \beta$ Pruning: Example

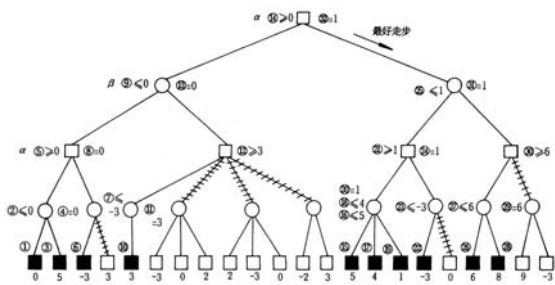


圖 3.10  $\alpha - \beta$  搜索程序的博奕樹

德明科技大學資訊科技系

51

## Properties of $\alpha - \beta$

- Pruning doesn't affect the final result
- Good move ordering can improve effectiveness of pruning
- With "perfect ordering"
  - Time complexity:  $O(b^{m/2})$
  - Double depth for search
  - Easily reach depth 8  $\Rightarrow$  human master

德明科技大學資訊科技系

52

## Why is it called $\alpha - \beta$

- $\alpha$ : the best value to Max
  - If  $v$  is worse than  $\alpha$ , Max will avoid it
  - i.e. prune that branch
- $\beta$ : the best value to Min
  - or the worst value to MAX
- $\alpha$  for Max, and  $\beta$  for Min

德明科技大學資訊科技系

53

## $\alpha - \beta$ Algorithm

```

function Max-Value(state, game,  $\alpha$ ,  $\beta$ ) returns the minimax value of state
inputs: state, current state in game
       game, game description
        $\alpha$ , the best score for max along the path to state
        $\beta$ , the best score for min along the path to state
if Cutoff-Test(state) then return Eval(state)
for each s in Successors(state) do
   $\alpha \leftarrow \text{Max}(\alpha, \text{Min-Value}(s, \text{game}, \alpha, \beta))$ 
  if  $\alpha \geq \beta$  then return  $\beta$ 
end
return  $\alpha$ 

function Min-Value(state, game,  $\alpha$ ,  $\beta$ ) returns the minimax value of state
if Cutoff-Test(state) then return Eval(state)
for each s in Successors(state) do
   $\beta \leftarrow \text{Min}(\beta, \text{Max-Value}(s, \text{game}, \alpha, \beta))$ 
  if  $\beta \leq \alpha$  then return  $\alpha$ 
end
return  $\beta$ 
    
```

Basic MiniMax + tracking of  $\alpha - \beta$  + pruning  
德明科技大學資訊科技系

54

## Now

**Chess:** *Deep Blue* defeated human world champion Gary Kasparov in 6-game match [1997].

- 200 million positions/sec
- very sophisticated evaluation
- undisclosed methods for extending some lines of search, up to 40 ply!

**Othello:** human champions refuse to compete against computers, who are too good!

**Go:** human champions refuse to compete against computers, who are too bad!

$b > 300 \Rightarrow$  most programs use *pattern knowledge bases* to suggest plausible moves

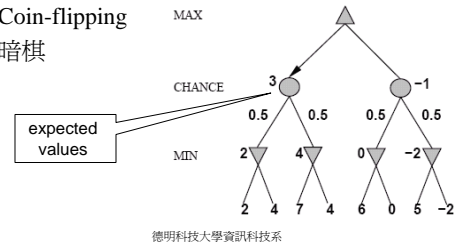
德明科技大學資訊科技系

55

## Non-determined Game

- You cannot exactly know which step you can play

- Backgammon: dice rolls determine legal moves
- Coin-flipping
- 暗棋



56