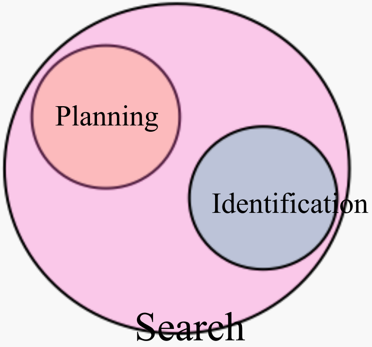
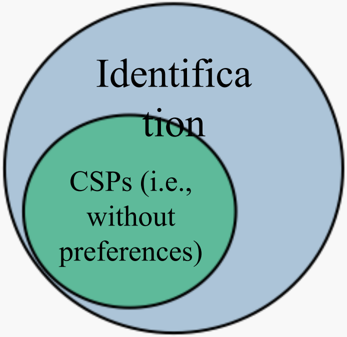
Revision (search)

Dr. Miqing
Li

Constraint Satisfaction Problem (CSP)

- Search is a process of navigating from a start state to a goal state by transitioning through intermediate states.
- CSP is a special class of search problems, which can be formally represented and allows useful general-purpose search strategies.



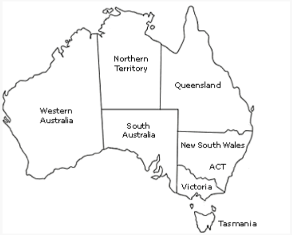


CSP cont.

- A CSP consists of
  - A set of variables
  - A domain for each variable
  - A set of constraints
- In a CSP, an assignment is complete if every variable has a value, otherwise it is partial. Solutions are complete assignments satisfying all the constraints.
- Constraint graph is used to represent relations among constraints in a CSP, where nodes correspond to the variables and arcs reflect the constraints.

CSP example: Map Colouring Problem

Map colouring problem: Paint the Australian map with three colours (red, green and blue) in such a way that none of adjacent regions can have the same colour.



Variables:

WA, NT, Q, NSW, V, SA, T

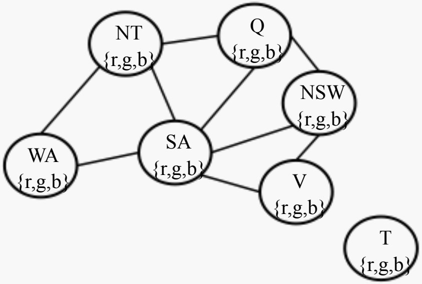
Domain:

{red, green, blue}

Constraints:

WA ≠ NT, WA ≠ SA, NT ≠ SA, NT ≠ Q, SA ≠ Q, SA ≠ NSW, SA ≠ V, Q ≠ NSW, NSW ≠ V

Constraint graph:



Strategies to solve CSPs

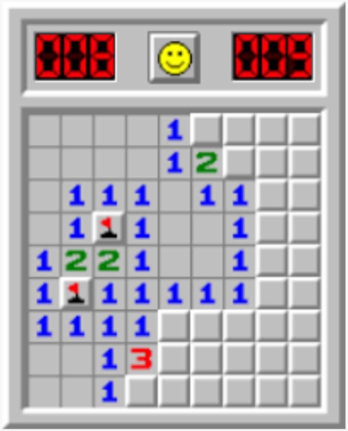
- Systematic search:
  - Backtracking: A depth first search method with two additional features: 1) Check constraints as we go and 2) Consider one variable at one layer in the search tree.
  - Forward checking: when assigning a variable, cross off anything that is now violated on its neighbours' domains.
  - Ordering: choose the variable with the fewest legal values left in its domain.

- Local search: not systematically search the space, start with a (constraint-violated) complete assignment and improve it iteratively.

CSP example: Minesweeper

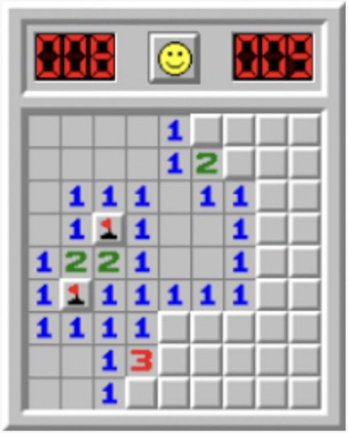
- Minesweeper is a single-player puzzle game. The goal is to not uncover a square that contains a mine; if you've identified a square that you think it is a mine, then you flag it.

- When you uncover a square, if the square is a mine, the game ends and you lost; otherwise it is a numbered square indicating how many mines are around it (between 0 and 8). If you uncover all of the squares except for any mines, you win the game.



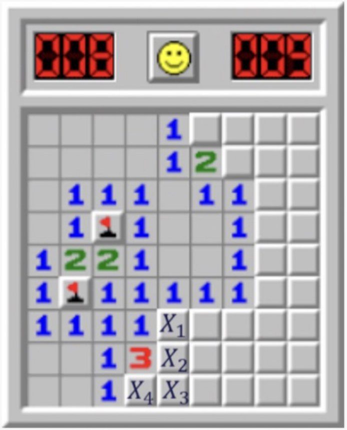
Minesweeper

- Variables:
  - All squares to be uncovered X_1, X_2,...
- Domain:
  - $D = \{0, 1\}$, where 0 denotes not a mine and 1 denotes a mine
- Constraint description:
  - The number on a square is the sum of its neighbour's values.



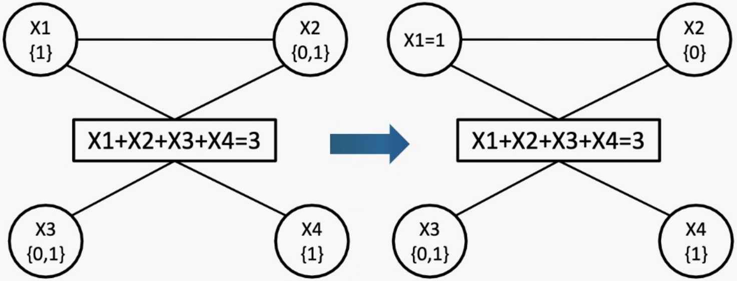
Find out the values of four squares X_1, X_2, X_3, X_4

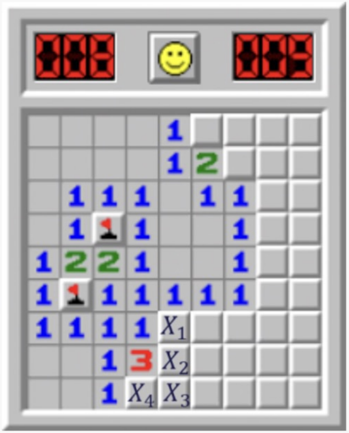
- Question: using systematical search to solve the problem, and also give the order of the four squares to be visited (the tie is broken numerically).
- Variables:
  - X_1, X_2, X_3, X_4
- Domain:
  - $D = \{0, 1\}$, where 0 denotes not a mine and 1 denotes a mine
- Constraints:
  - $X_1 = 1$
  - $X_1 + X_2 = 1$
  - $X_1 + X_2 + X_3 + X_4 = 3$
  - $X_4 = 1$
  - ...



Find out the values of four squares X$_1$, X$_2$, X$_3$, X$_4$

- Constraints:
  - $X_1 = 1$
  - $X_1 + X_2 = 1$
  - $X_1 + X_2 + X_3 + X_4 = 3$
  - $X_4 = 1$





The order of variables to be visited is (tie is broken numerically):

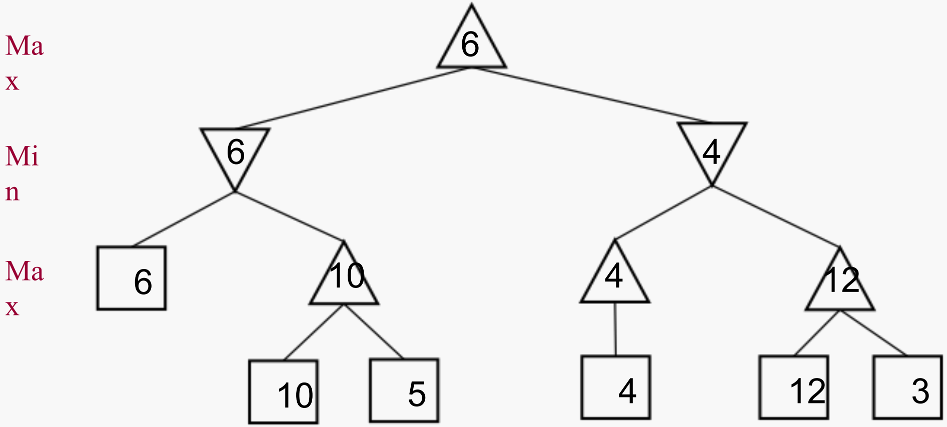
$X_1 \rightarrow X_2 \rightarrow X_4 \rightarrow X_3$

Games (Adversarial Search)

- Game is a special class of search problems, where there are usually multiple agents playing against each other.
- We want a search algorithm to find a strategy which recommends a move for each state.
- Utility is the final value for a game that ends in terminal state for a particular player.
- Minimax value of a node in a search tree is the utility of the terminal state to which both players play optimally from that node.
- To determine minimax value of a node, we may not need to visit all nodes of the search tree. We can use alpha-beta pruning to cut off some nodes/branches.

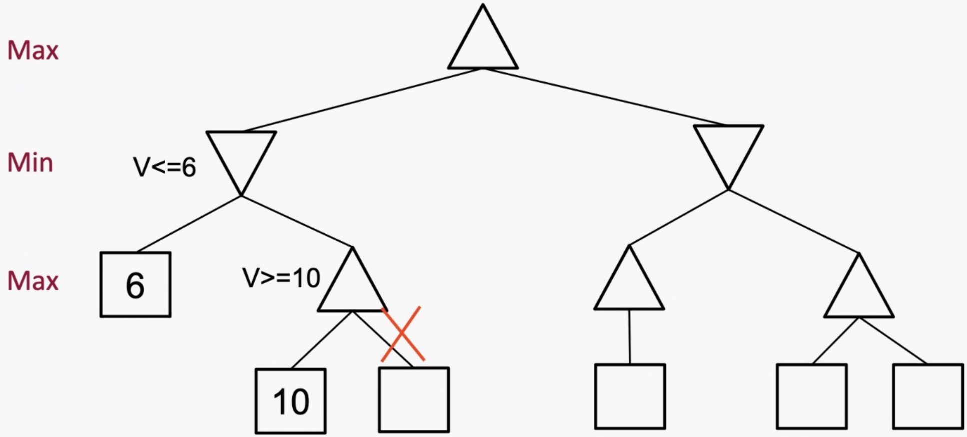
Example – find minimax value

- Question 1. Give the minimax value at each node for the game tree below.



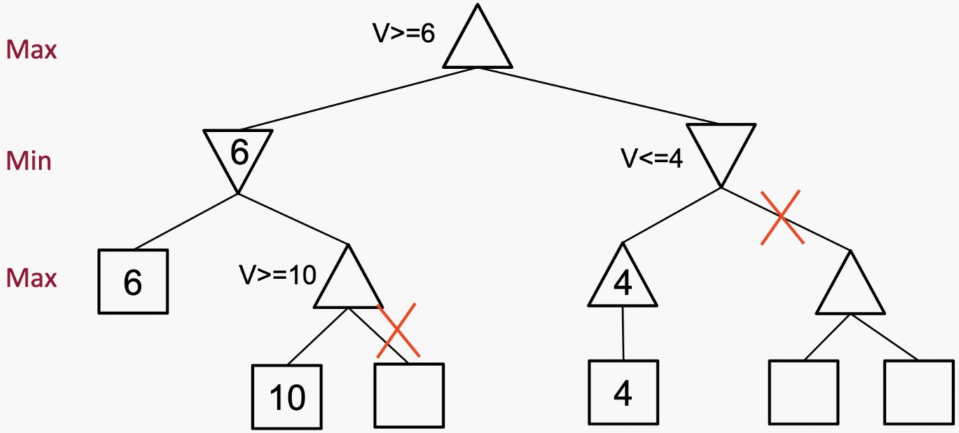
Example – Alpha-beta pruning

- Question 2. Find the nodes of the above tree pruned by alpha-beta pruning algorithm. Assuming child nodes are visited from left to right. There are four layers and you can use Lm-n to denote the nth node from left to right in the layer m, e.g., the first node (with value 10) at the bottom layer can be denoted by L4-1.



Example – Alpha-beta pruning

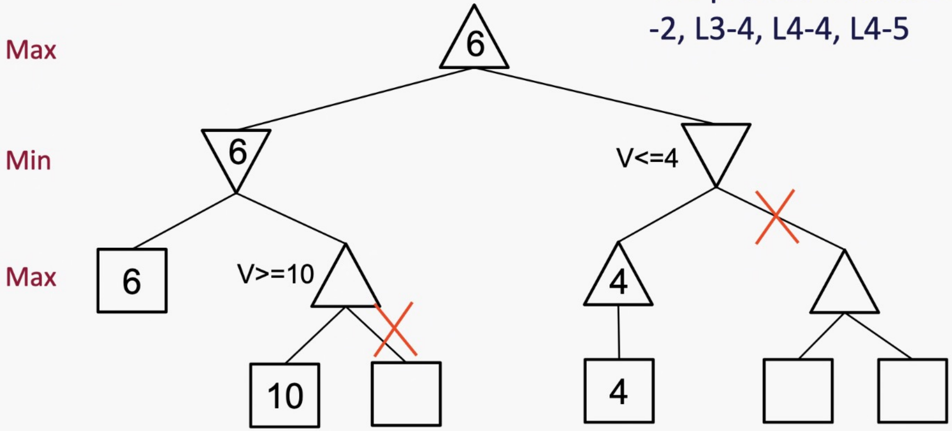
- Question 2. Find the nodes of the above tree pruned by alpha-beta pruning algorithm. Assuming child nodes are visited from left to right. There are four layers and you can use Lm-n to denote the nth node from left to right in the layer m, e.g., the first node (with value 10) at the bottom layer can be denoted by L4-1.



Example – Alpha-beta pruning

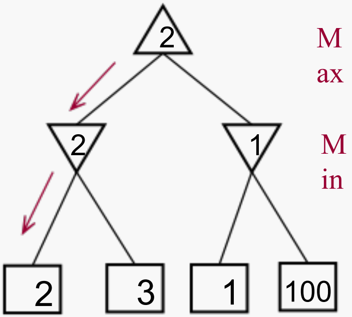
- Question 2. Find the nodes of the above tree pruned by alpha-beta pruning algorithm. Assuming child nodes are visited from left to right. There are four layers and you can use Lm-n to denote the nth node from left to right in the layer m, e.g., the first node (with value 10) at the bottom layer can be denoted by L4-1.

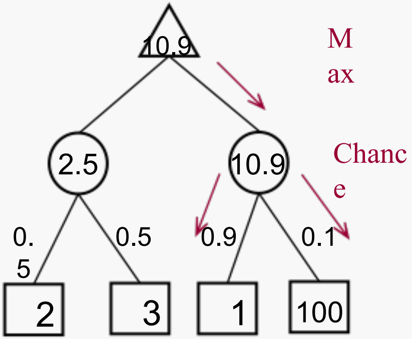
The pruned nodes: L4-2, L3-4, L4-4, L4-5



Games (Expectimax Search)

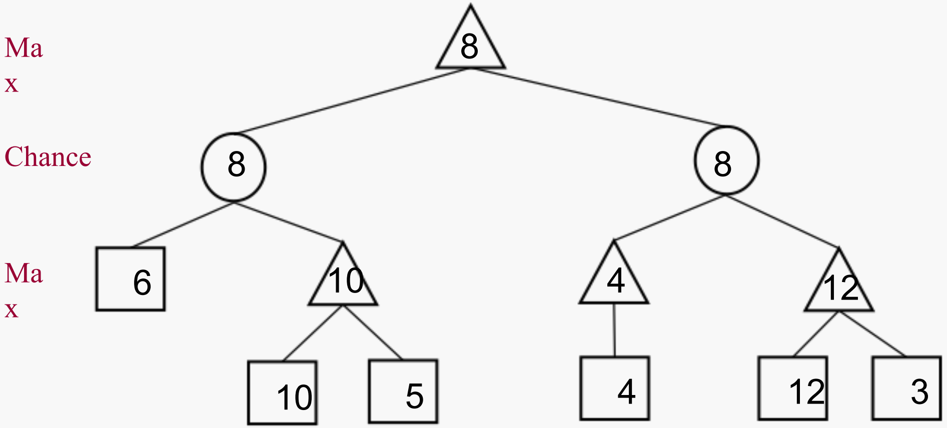
- In minimax search, we compute each node's minimax value, i.e. the best achievable utility in the worst case (i.e., against an adversarial, optimal opponent).
- But what if the opponent is not optimal? What if there are uncertain outcomes controlled by chance?
- Expectimax search: There is uncertainty when playing the game. There are chance nodes, whose utilities are the expected utilities, i.e. weighted average of their children's utilities.





Example – Expectimax search

- Now suppose that in the last question the Min player in the second layer is replaced by a Chance player. We then have the following game tree. Let us say that the chance nodes go their different children with the same probability. Find the value at each node of the tree.



- The content of Expectimax Search was taught in the on-campus lecture Week 9 Thursday.

- Office hours in Weeks 12-13: 10.00-11.00 and 12.00-13.00 Wed with the link: https://bham-ac-uk.zoom.us/j/2066382427