Tutorial for Nov 3/7

See Russell and Norvig, chapters 5 and 6

1. Show how a single constraint on 3 variables such as $A + B = C$ can be turned into 3 binary constraints by using an extra variable. You can assume domains are finite. (You want to consider variables corresponding to pairs of values, with constraints to express that.)

   Does this work also for constraints with n variables, $n > 3$?

2. Consider this 2 person game.

   There are four squares in a row; each player has a counter. The starting position is:

   

   Play is in turns, starting with A. Each move is to an adjacent space in either direction. If the opponent is in an adjacent square, the player may jump over the opponent to the next open square, if there is one. The game ends when one player reaches the opposite end on the board. If A wins, the value to A is +1; If B wins, the value to A is -1.

   (a) Sketch the game graph (i.e. represent looping paths in the game tree as edges back to earlier nodes).

   (b) Assign game values to the terminal states.

   (c) Mark each node with the backed-up minimax value. You will have to work out how to deal appropriately with looping paths.

   (d) Standard minimax will not work on this example. Suggest how to adapt it in general to looping situations.

3. It is the case that α–β pruning returns the same strategy as does MINIMAX. Show why this is the case.

   (Hint: first look at the following case, where the tree is explored only as far as is shown, and say why the value for the unexplored node is irrelevant to the final outcome. There is a third branch from the top node still to be explored. Now try to generalise to trees explored to greater depth.)

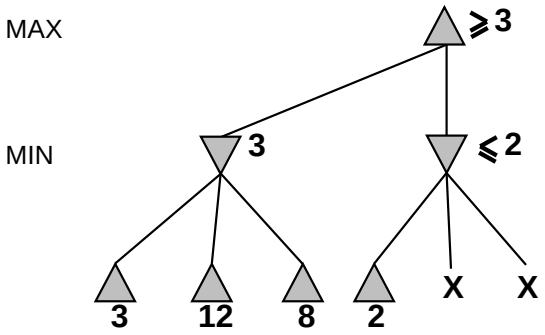

Figure 1: Alpha-beta example

function ALPHA-BETA-SEARCH(*state*, *game*) **returns** an action
  action, *state* ← the *a*, *s* **in** SUCCESSORS[*game*](*state*)
    such that MIN-VALUE(*s*, *game*, $-\infty$, $+\infty$) is maximized
  return *action*

function MAX-VALUE(*state, game,* α, β) **returns** the minimax value of *state*
  if CUTOFF-TEST(*state*) **then return** EVAL(*state*)
  for each *s* **in** SUCCESSORS(*state*) **do**
    $\alpha \leftarrow \max(\alpha,$ MIN-VALUE(*s*, *game*, α, β))
    if $\alpha \geq \beta$ **then return** β
  return α

---

function MIN-VALUE(*state, game,* α, β) **returns** the minimax value of *state*
  if CUTOFF-TEST(*state*) **then return** EVAL(*state*)
  for each *s* **in** SUCCESSORS(*state*) **do**
    $\beta \leftarrow \min(\beta,$ MAX-VALUE(*s*, *game*, α, β))
    if $\beta \leq \alpha$ **then return** α
  return β