Algorithm A : Best-First Search
uses a heuristic that estimates total path length —
$f(n)$ is an estimate of total path length from start to goal if going through node n

Algorithm A* : Algorithm A, but guarantees $h(n)$ never overestimates dist $n \to$ goal

$f(n) = \underline{\text{depth}(n)} + \underline{h(n)}$

actual dist start $\to$ n

estimate $n \to$ goal

Admissible Heuristics : one of these → which guarantees it will find a shortest-path solution.

More Informed Heuristics   if two heuristics are admissible
$h_1(n)$ + $h_2(n)$
and $h_2(n) \geq h_1(n)$ for all n   then h_2 is more informed than h_1

Note $h_0(n) = 0$ for all n — admissible
In 8-puzzle, $h_1(n)$ = # tiles out of place ; $h_1(n) \geq h_0(n)$

Suppose $h_1(n)$, $h_2(n)$ admissible

$h_1(n) \geq h_2(n)$ for some n

$h_2(n) \geq h_1(n)$ for most n

is $h_2(n)$ more informed? not exactly

Idea: Can we create a function $h_3(n)$ that is more informed than $h_1(n)$ and $h_2(n)$?

$$h_3(n) = \max\{h_1(n), h_2(n)\}$$

Adversarial Search

- one or more entities beyond your control which affect state transitions:
    - active opponent(s)
    - chance

- must cope with adversarial conditions – choose your actions to minimize adverse impact of others' actions
    - instead of choosing move with best immediate gain, determine what your adversaries will do in each case, and select best of these outcomes
    - assumes adversary will select options most beneficial to them / least beneficial to you.

Minimax Algorithm

Version 0: determine all possible moves you can make, select one at random

Version 1: determine all possible moves you can make, evaluate with a heuristic h(), choose one with highest value (assumes "killer heuristic")

Version 2: determine all possible moves you can make, and for each:

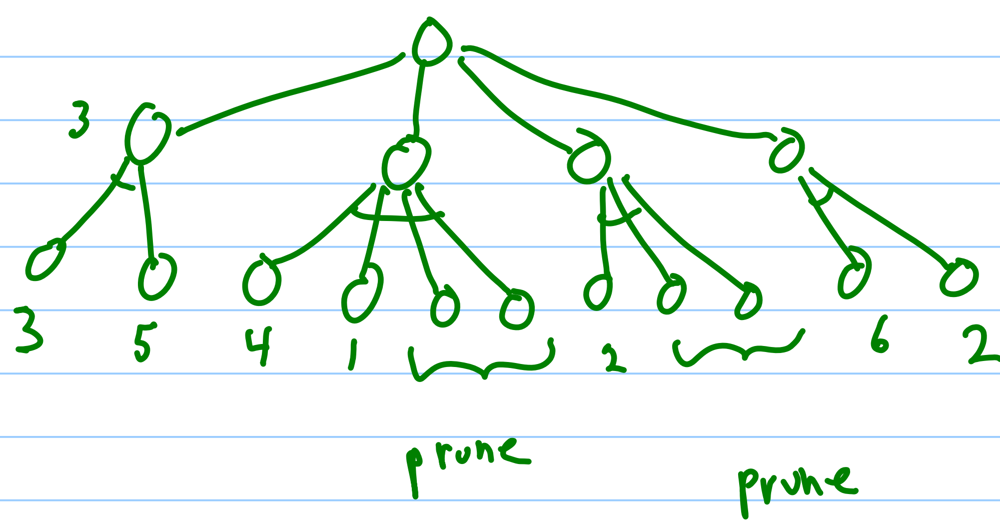
  determine all possible moves opponent can make
  choose one with lowest value of h().

For each of these "backed-up" values of opponent's moves, choose move which yields highest value
[1-ply lookahead - your move + opponent's move]

Version n: n-ply lookahead

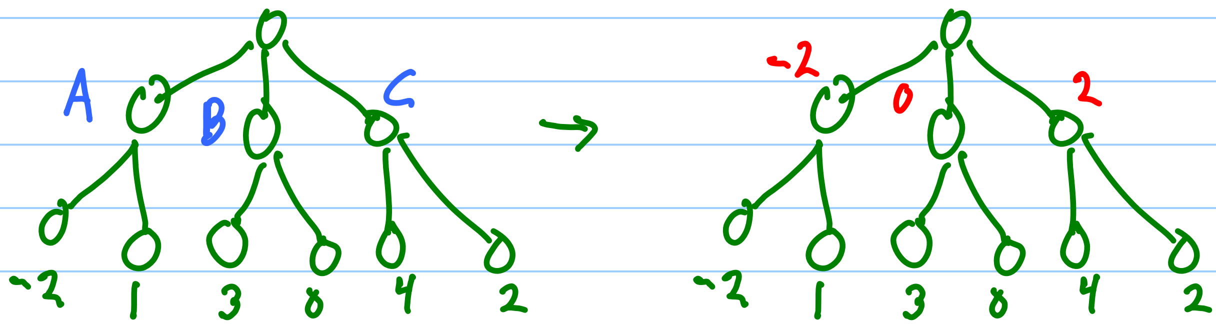
Alpha-Beta pruning:

Recognize when outcome of moves cannot have any impact on computations and prune them:

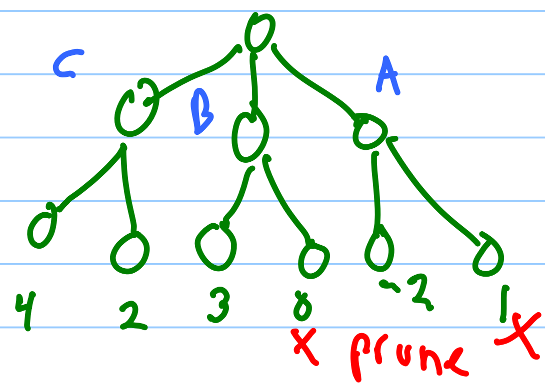


Enhance opportunities for pruning by using a static evaluation function that reorders moves from best to worst —

Instead of



Reorder



Natural Language Processing :

See Emily's notes at

https://learn.dcollege.net/bbcswebdav/courses/30328.201435/BABELFY2-2.pptx

(Must log in to Bb Learn)