AI cheat sheet — GANESH V IYER

world = environment + actions

reflex agent: selects an action based solely on the current state of the world. < **planning agent**: maintain a model of the world to simulate actions.

	complete?	optimal?	time?	space?
DFS	NO	NO	$O(b^m)$	$O(bm)$
BFS	YES	YES if UCS, NO otherwise	$O(b^s)$	$O(b^s)$
UCS	YES	YES	$O(b^{C^*/\varepsilon})$	$O(b^{C^*/\varepsilon})$
Greedy	NO	NO	poorly guided DFS	
A*	YES	YES	BFS w/ bad heuristic	

Tree search + closed set = Graph search
A* optimal w/ admissible **h** only — closed set, not a list — A* optimal & complete w/ consistent **h** only

- A* tree search w/ admissible heuristics will yield an optimal solution.
- A* graph search w/ only consistent heuristics will yield an optimal solution.

SEARCH STATE

Start state
successor function
goal state / test
state space

path / plan: path from start state to goal state
strategy: order in which states are considered.

+ state space graph: each state is represented exactly once
+ search tree: each node is a path to current state

b = branching factor m = max depth
s = depth of shallowest goal node.
C* = optimal path cost
ε = minimal cost b/w two nodes

HEURISTICS - admissibility, consistency

Admissibility: $\forall n,\ 0 \le h(n) \le h^*(n)$
Consistency: $\forall A, C\ \ h(A) - h(C) \le cost(A, C)$
$f(C) \ge f(A)$ ∀ A* gs

- values of $f(n)$ ($g(n) + h(n)$) for nodes along any plan are non decreasing.
- **Dominance**: max function applied to multiple admissible heuristics → also admissible

CSP → identification problem, no regard to how we arrive at goal state. NP-HARD
- N variables, domain size $O(d)$ for each ⇒ total possible assignments $O(d^N)$

DFS < Backtracking search < Backtracking search w/ filtering < BTS + f w/ ordering

Backtracking search:
1. fix an order for variables
2. start assigning w/o conflict
3. if no value, return & repeat

forward checking < **arc consistency** (X → Y)

	when to use	how to use	result
forward checking	during the search, after assignment	unassigned neighbouring node	arc consistency only for the two nodes
arc consistency	during or before the search	enqueue & propagate → then enqueue those that point to X.	full arc consistency

ARC CONSISTENCY (X → Y)
are there elements here for which there are constraint violating variables — IF SO, prune — here

- A graph that is strong K-consistent possesses the property that any subset of K nodes is K, K-1, K-2, ... 1 consistent too.

Minimum remaining values: Next unassigned variable has the fewest valid remaining values
Least constraining value: choose that value that prunes the fewest from the remaining.

TREE STRUCTURED CSP

- no loops in its constraint graph
- reduce runtime $O(d^N) \to O(nd^2)$

LINEAR ORDERING STEPS
1. Pick an arbitrary node as root
2. Order the nodes so that the edges point right from root (DAG)
3. BACKWARD PASS of arc consistency
4. FORWARD assignment of values.

CUTSET - assign to cutset, prune neighbours, solve. $O(d^c(n-c)d^2)$

ITERATIVE IMPROVEMENT: while NOT SOLVED: 1. randomly choose conflicted variable 2. choose a value that violates fewest constraints

Critical ratio = $\frac{\#\text{ constraints}}{\#\text{ variables}}$ for which II is expensive af.

MINIMAX (deterministic zero sum games) ~ DFS traversal

$V(s) = \max V(s')$ for non-terminal states — or — Known for terminal } best possible outcome from here on
$= \min V(s')$ → if opponent controlled state

α-β PRUNING $O(b^{m/2})$

MAX node: prune if $v > \beta$
MIN node: prune if $v < \alpha$
} all finite

α, β = best explored path from current node to root for MAXIMIZER, MINIMIZER

EVAL FUNCTIONS

$Eval(s) = \sum_i w_i f_i(s)$

$V(s) = \sum_{s'} p(s') V(s')$ for all chance states

EXPECTIMAX

- Can never prune a chance node
- final utility no longer represents a node on a descendant

NON ZERO SUM GAMES

- nodes are tuples for various players
- Co-op, compete thru computation