A. Search

1. reflex: based on current (or memory) IS
· planning agent: consequences of action (world model + goal) Would be

2. Search problem: state space + successor functions (action & cost) + start state + goal test

3. Each node in a search tree is an entire PATH in the state space graph, corresponding to a PLAN that achieve the state. 4. Complete 有就能找着 Optimal least cost path.

5. ① Uninformed search

	property	graph	optimal	complete	Time	space
DFS	LIFO stack (deepest)	b nodes, m tiers	X. leftmost	$m\to\infty$ if cycle then no!	$O(b^m)$	$O(bm)$
BFS	FIFO queue (shallowest) min edges.	s tiers	X. cost=1 ✓, other X.	✓	$O(b^s)$	$O(b^s)$
Iterative Deepening	DFS with limit 1,2,3… DFS space + BFS time & 最优		cost=1 ✓, other X		$O(b^s)$	$O(bm)$
Uniform Cost Search	priority queue (cheapest 累积). g(n) backward cost.	solution cost C^*, arc cost $\geq \varepsilon$.	✓	✓ if $C^*<\infty$ and $\varepsilon>0$	$O(b^{C^*/\varepsilon})$	$O(b^{C^*/\varepsilon})$

6. Heuristics (manhattan, euclidean): admissible: $0 \le h(n) \le h^*(n)$ ← true cost to a nearest goal ($h(A) \le$ cost (A to G))
consistent: $h(A)-h(C) \le$ cost (A to C). admissible ⇍ consistency (⇐)

7. Informed Search

	property	optimal	graph
Greedy	h(n). forward cost	X	may be badly guided
A^*	$h(n)+g(n)=f(n)$	✓ 若 h(n) admissible (tree) ⇐ consistent (tree). ✓ 若 h(n) consistent (graph).	start, goal

8. Graph Search: tree search + set of expanded states. (close set) tree: 一个点可多次出现 graph: 一个点只一次

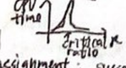
B. CSP

1. model state: $X_i \in D_i$. goal test: a set of constraint (explicit/implicit)
2. Binary CSP: each constraint relates (at most) 2 variables (binary + unary 单元约束) can't solve: DFS/BFS X
3. BackTracking: DFS + variable ordering + fail on violation (回溯 if violate) (一次 assign 一个) 只检查与之前 assignment 是否一致
4. Filtering: keep track of domains for unassigned variables and cross off bad.
• Forward checking: 每 assign 一个元素, 检查全部 unassigned, 去除不一致. ∅ 即回溯
• Constraint Propagation: 用 constraint 去删除某 domain 元素, 再基于元素 domain 变化去除其他 X_i.
① arc consistency: X→Y consistent ⇔ ∀x∈X (tail), ∃y 满足约束. delete from X.
② Forward checking: enforce [new assignment] → □, □, 未但没 propagate, arc 不全.
③ AC-3: Q = initial arcs (all arcs). 每删除 Y 中(多)一个元素, X→Y 都要被 recheck.
④ k-consistency: For each k nodes, any consistent assignment to k-1 can be extended to the kth node. (1-consistency = node c…, each node's domain 都有满足单元约束的值) 2 = arc, 3 = path.
⑤ strong k-consistency: also k-1, k-2, … 1 consistent. ⇒ can solve without backtrack.

5. Ordering • Minimum Remaing Values (MRV): 选 domain 最小的 assign. (= most constraint value)
• Least Constraint Value (LCV): 选了一个 assign 之后, 从 domain 里选 the one that rules out the fewest values in the remaing values. (找着选了它, 和别人冲突最少的那个). 计算++

6. structure 拆成 independent subproblem (连通分量) / 其他容易算的形式.
• Tree-structured CSP: $O(nd^2)$ [general CSP 最差 $O(d^n)$] 因为 ∀X→Y 都被 enforced arc consistent, 且 Y 不会被在之后删除.
(1) Order: parent 在 children 前面.
(2) 从后往前, 删 i=n:2, remove Inconsistency(Parent(X_i), X_i) Backward pass 后全部根→叶 consistent.
(3) 从前往后赋值: i=1:n, assign X_i consistently with parent(X_i) No backTrack
• CutSet: $O(d^c(n-c)d^2)$ 把个小 set 拿出来, 剩下成树, 小 set enumerate (instantiate) 分别算
* Finding the smallest cutset is NP-hard, 但存在 efficient approximation algorithm

7. Iterative Improvement [重新赋值以减少冲突]
① Algorithm: While not solved, a. variable selection: randomly select any conflicted vars;
b. value selection: 根据 min-value heuristic 选 violate 最少 constraint 的值.
② performance: $R = \frac{\#\text{ constraints}}{\#\text{ vars}}$



8. Local Search [取得 local optimal]
① Model: state: a complete assignment; successor function: how local changes.
② Hill climbing: best neighboring state (greedy).
③ Beam search: greedy hill climbing + keeping K states.
④ Simulated Anneding: 允许下坡. a. pick a random move b. always accept uphill move
c. accept a downhill move with $P = e^{-\Delta E/T}$. d. T↓ 随时间. [更难往下坡]
* T↓ 的够慢, 收敛至 optimal.

C. Adversariel Search

1. zero-sum game: agents have opposite utilities (一个 max, 另一个 min)
2. Adversariel Game Trees (Minimax): Agent 于 $V(s) = \max_{s'\in successors(s)} V(s')$ opponent: $V(s') = \min_{s\in successors(s')} V(s)$ (V(s) known ↑ terminal)

```
def value(state):
    if state is terminal state: return the state's utility
    if next agent is Max: return max-value(state)
    if next agent is Min: return min-value(state)
```

Time: $O(b^m)$ Space: $O(bm)$ Just like exhausted DFS

```
def max-value(state): △
    v = -∞
    for each successor of state:
        v = max(v, value(successor))
    return v
```

```
def min-value(state): ▽
    v = +∞
    for each successor of state:
        v = min(v, value(successor))
    return v
```

3. Resource Limit ① Depth-limited search: terminal utility → evaluation function 限制深度。
Eval(s): weighted linear sum of features. ② (CNN, Monte Carlo)
② Branching factor: only consider good moves. (CNN) e.g. 距离终点的 d 的倒数 … to root

4. α-β Pruning α = MAX's best option on path to root β: MIN's best option on path

```
def max-value(state, α, β):
    v = -∞
    for each successor of state
        v = max(v, value(succ, α, β))
        if v ≥ β return v
    return v  α = max(α, v)
```

```
def min-value(state, α, β):
    v = +∞
    for each succ of state:
        v = min(v, value(succ, α, β))
        if v ≤ α return v
        β = min(β, v) return v
```

$O(b^{\frac{m}{2}})$ Good child ordering, effectiveness ++.

5. Expectimax Search MIN→EXP ○ No pruning! $V(s') = \sum_{s\in successors(s')} P(s) \times V(s)$ ← state under opponent's control
Use expectimax search if opponent's action is known with informed probability of each move. To figure out each chance node's probability, run a simulation of 对手.

6. Multi-agent utilities utility tuple, each player maximize its own component.

D. Propositional Logic

1. Logic AI

Inference engine ← domain-independent algorithm
Knowledge Base ← domain-specific content

2. Syntax {¬, ∨ (disjunction), ∧, ⇒, ⇔} + propositional symbols
3. Semantics $S_1 \Rightarrow S_2 \equiv \neg S_1 \vee S_2$ S valid ⇔ ¬S unsatisfiable
4. Valid true in all models, satisfiable true in some models, unsatisfiable true in no model
5. Entailment $\alpha \models \beta$ (α entails β / β follows from α) ⇔ in every world α = true, β is also true
model(α) ⊆ model(β)
6. Proof $\alpha \vdash \beta$ proof for $\alpha \models \beta$. ① Model checking (truth table enumeration) (exponential)
② Inference rules e.g. resolution NP-complete
Sound: everything that can be proved is in fact entailed ①✓ ②?
Complete: everything that is entailed can be proved. ①✓ ②? ⇐ search complete
can formulate $\alpha \vdash \beta$ as a search problem KB ⊇ α, inference rule, ⇒ KB ⊇ β
7. Resolution for CNF. 一次消一个, 不可同时消多个. sound & complete for propositional logic.
Prove KB ⊨ α: contradiction i.e. show KB ∧ ¬α is unsatisfiable.
① convert KB ∧ ¬α to CNF
② Repeatedly apply the resolution rule to add new clauses, until one of the 2 happens
a. two clauses yield to the empty clause ⇒ KB ⊨ α.
b. No new clause to be added anymore ⇒ KB does not entail α.
8. Horn Logic: a subset of propositional logic that supports efficient inference
Horn clause: $P_1 \wedge P_2 \wedge \dots \wedge P_n \Rightarrow Q$ / $\neg P_1 \vee \neg P_2 \vee \dots \neg P_n \vee Q$ P_i, Q 非负. n 可以 = 0 (single atom).
Inference: Modus Ponens sound and complete for Horn Logic
$\frac{\alpha_1, \dots, \alpha_n,\ \alpha_1 \wedge \dots \wedge \alpha_n \Rightarrow \beta}{\beta}$
Forward chaining add new clause into KB until Q is added
Backward chaining check if q is known to be true already, or prove by BC all premises of some rule concluding q. ① Avoid Loop: check if new subgoal is already on the goal stack ② Avoid repeat: check if new subgoal has already proven true or has already failed by caching old result.
FC: data-driven, unconscious processing; BC: goal-driven, complexity much less than (linear in size of KB)
9. Logic Programming Encode info in KB + problem instance $\xrightarrow[\text{query}]{\text{SAT solver}}$ result / t++.

E. First Order Logic

~~fact~~ objects + relations (predicate) + functions. = 1 order predicate logic
1. Syntax: Connective {¬, ∧, ∨, ⇒, ⇔}, Quantifier {∀, ∃}, variables {x, y, z}, Equality {=}, Constants, Predicate {Brother, >}, Function {sqrt, leftleg of}.
Atomic sentence = predicate(term1, … termn) / term1 = term2
Term = constant / variable / function(term1, … termn) Atomic + connective → complex sentence
2. Semantic: objects + relations + interpretation. truth respect to a model
interpretation (constant symbol → objects; predicate symbols → relations; function symbol → functional relation).
3. Quantifier: ∀x A⇒B. ∃x A∧B. (否则会导致问题) ∀x∀y, ∃x∃y 可交. ∃x∀y ≠ ∀y∃x.
∀x A ≡ ¬∃x¬A, ∃x B ≡ ¬∀¬B. free: ∀x P(x, y). FOL 中, 每个 var 都应 bound
4. 优先级: ¬ > ∧ > ∨ > ⇒ 和 ⇔
5. Inference: Universal Instantiation (UI) $\frac{\forall v\ \alpha}{Subst(\{v/g\}, \alpha)}$ g: ground term, term without vars. e.g. (Father(A))
Every instantiation of a universally quantified sentence is entailed by it.
Existential Instantiation (EI) k: constant symbol that 没出现在 KB 的任何地方
$\frac{\exists \alpha}{Subst(\{v/k\}, \alpha)}$ k: skolem constant. EI can be applied once to replace an existential sentence.
6. Propositionalization FOL → PL → resolution → result. FOL → PL: UI everything
Problem: with function, ∞ gound terms: Father(Father(A)); generate irrelevant sentence.
Theorem: ~~If sentence~~ α FOL KB ⊨ α ⇒ a finite subset of FOL KB ⊨ α.
Idea: For n = 0 to n = ∞ do: ① create a Prop… KB by Instantiating with depth-n terms. function nesting levels.
② If KB ⊨ α, return true. ∞ loop if KB ⊭ α.
Theorem: Entailment for FOL is semi-decidable. (∃ algorithm say yes to every KB ⊨ α; ¬∃ algorithm says no to every non-entailed sentence)
7. Unification Find substitutions that make different expressions identical.
Unify(α, β) if αθ = βθ; Standardizing apart: eliminate overlap of vars.
Most General Unifier: ∃ 一个 MGU
8. Horn Logic $p_1 \wedge p_2 \wedge \dots p_n \Rightarrow Q$. p_i: atomic sentence, all vars 假设是 universally quantified
Generalized Modus Ponens (GMP)
$\frac{p_1', p_2', \dots p_n', (p_1 \wedge p_2 \wedge \dots \wedge p_n \Rightarrow q)}{q\theta}$ $p_i'\theta = p_i\theta$ for all i.
e.g. King(J), Greedy(y), King(x) ∧ Greedy(x) ⇒ Evil(x)
θ = {x/John, y/John}
qθ = Evil(John)
Forward Chaining 从事实出发开始, 向 Q 推.
Sound and complete for 1st order Horn clauses. FC terminate with no functions in finite # of iterations. 但不 entail 可能停不下来。Entailment for Horn Clauses is semi-decidable.
Backward Chaining 从结论开推, + unification. space linear (DFS).
9. Resolution $\frac{l_1 \vee \dots l_k,\ m_1 \vee \dots \vee m_n}{(l_1 \vee \dots l_{i} \vee \dots)\theta}$ unify(l_i, ¬m_j) = θ. m… 和 l… 应该 standardized apart.
sound and complete for FOL
Conversion to CNF: ① ⇔, ⇒ 去除 ② 把 ¬ move inwards ¬∀x p ≡ ∃x ¬p, ¬∃x p ≡ ∀x ¬p.
③ Standardize vars: 每个量词 var 不同, ④ Skolemize: existential var → function G(x)
⑤ 去除 ∀ ⑥ 用分配律转化成 CNF. (或, 把全部量词提到前面) (把量词去除) ∀…

F. Bayesian Network = DAG + CPTs $a^* = \arg\max_a \sum_s P(s|e)U(s)$ $O(d^n)$ space (d: domain size, n: var)

1. Math: $P(t)=\sum_w P(t,w)$ $P(a|b)=\frac{P(a,b)}{P(b)} \Rightarrow P(y)P(x|y)=P(x,y)$ product rule

Chain rules: $P(x_1,x_2,\cdots x_n)=\prod_i P(X_i|x_1,\cdots x_{i-1})=\prod_i P(X_i|Parents(X_i))$

$P(A|B)=\frac{P(B|A)P(A)}{P(B)}$ 条件独立 $P(x,y|z)=P(x|z)P(y|z)$ for $\forall x,y,z \in X,Y,Z$ 或 $P(x|y,z)=P(x|z)$

2. Inference by enumeration Goal: $P(Q|e_1,\cdots e_k)$ ① select entries consistent with e_i's.
② $P(Q,e_1,\cdots e_k)=\sum_{h_1\cdots h_k} P(Q,h_1\cdots h_r,e_1,\cdots e_k)$ ③ $P(Q|e_1,\cdots e_k)=\frac{P(Q|e_1,\cdots e_k)}{\sum_q P(Q,e_1,\cdots e_k)}$ normalize.
worstcase time $O(d^n)$ space $O(d^n)$.

3. General syntax, formula for BN. max domain size d, max parent # = k, $O(n\cdot d^{k+1})$ (linear scaling)

4. Semantics - conditional independent
① 给定 parents, 它与它的非 descendents 节点间独立. ⇔ global semantics.
② 给定 Markov Blanket, 它与其它节点间条件独立. (父,子,子的其他父).

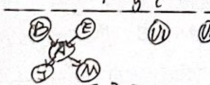
5. D-separation ⇔ X→Y 的全部路径(无向)都被堵住. ⇒ $X \perp\!\!\!\perp Y|Z$.
被堵住: ∃ 一个 inactive triple O→◎→O, O←◎→O, O→O←O ← 要注意和 active 区分!
Active triple: O→O→O, O←O→O, O→◎←O, (O→O←O, 子孙被观测) ☆

6. Topology encodes conditional independence. $P(X_i|X_1,\cdots X_{i-1})=P(X_i|Parents(X_i))$

7. Markov Network Undirected graph + potentials ← capture uncertainty of a clique.
① Joint distribution $p(\vec{x})=\frac{1}{Z}\prod_c \psi_c(x_c)$ ← potential over clique c $Z=\sum_x \prod_c \psi_c(x_c)$ ← partition function
Clique 两两之间有边连接. 每个 c 内部有 potentials, 但不 normalize.
② Conditional Independence in MN
A⫫B|C markov blanket (一圈). 就没有V型结构了, 条件独立只能通过链路去描述. (被堵住)
③ BN, MN conversion a. Moralization: CPT 中有的, 必须成团. 大团吃小团.
$p(\vec{x})=p(x_1)p(x_2)p(x_3)p(x_4|x_1,x_2,x_3)=\frac{1}{Z}\psi(x_1,x_2,x_3,x_4)$
b. construct potential function from CPT.
④ BN & MN encode same distribution, not encode same set of 条件独立.
④ Not able to convert (V-structure) 无MN. BN×→ A⫫B|∅, A⫫B|C∪D, A⫫B|∅, A⫫B|C. C⫫D|A∪B.
P: The set of distributions whose conditional independence can be exactly (no more, no less) represented by a 有向/无向图.
⑤ MN & constraint Graph constraint graph can be seen as Markov Networks with 0/1 potentials
⑥ BN/MN v.s. Logic BN/MN can be seen as a probabilistic extension of PL; PL can be seen as BN/MN with deterministic CPTs/potentials

8. Conditional Random Field - discriminative models. MN + input.
$P(y|x)=\frac{1}{Z(x)}\prod_c \psi_c(y_c,x)$ $Z(x)=\sum_y \prod_c \psi_c(y_c,x)$

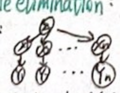
BN Exact Inference



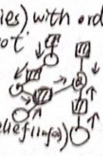
1. enumeration formula 其中 +e, -e, +a, -a 隐含 $P(+m|a)$
$P(B|+j,+m) \propto_B P(B,+j,+m)=\sum_{e,a}P(B,e,a,+j,+m)=\sum_{e,a}P(B)P(e)P(a|B,e)P(+j|a)P(+m|a)$

2. factor: 多维 array 表示 $P(Y_1,\cdots Y_n|X_1,\cdots X_m)$ joint / condition / specified family

3. variable elimination: 交叉 join, eliminate on hidden variables; 最后 joint + $\frac{1}{Z}$.
① ordering:
• e.g.



$P(X_n|y_1,\cdots y_n)$ 先消 Z: $f_1(X_1,\cdots X_n)\to f_2(y_1,X_2\cdots X_n)\to\cdots f_{n+1}(y_1,y_2\cdots X_n)$. 2^{n+1}
后消 Z: $f_1(Z,y_1), f_2(Z,y_2),\cdots f_{n-1}(Z,y_{n-1})$, $P(Z)$, $P(X_n|Z)\to f_n(X_n,y_1,\cdots y_{n-1})$ 空间 2^2
• 3SAT. #p-hard, 不存在好 factor.
② VE on Polytree: 有向的无无向环路的图.
• complexity of VE is linear in BN size (# of CPT entries) with ordering:
a. convert to a factor graph b. take Q as the root
c. eliminate from leaves to root.
• ⇔ sum-product message passing algorithm / belief propagation algorithm. (从可向根 pass belief $f_1(m_0)$)



4. Message Passing on General Graph
① Junction tree algorithm (exact): a. group individual nodes to form cluster nodes in a way that the resulting network is a polytree (junction/join tree). b. Run a sum-product like algorithm on JT. c. Intractable on 大图(团).
② Loopy Belief Propagation (approx): simply pass info on general graph. 有时不 terminate. 时间不收敛(不稳). tractable for 大图.

BN Approximate Inference

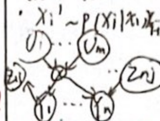
1. Prior Sampling For i=1,2, ... n (in topological order): sample x_i from $P(X_i|parents(X_i))$. Return $(x_1,\cdots x_n)$
渐进 consistent 求 $P(C|r,w)$. ① count $(c|r,w)$ $(\neg c,r,w)$ ② $P(C|r,w)=\langle c, \neg c\rangle \frac{1}{Z}$

2. Rejection Sampling 不符合 evidence: reject, i++ (进入下一轮)

3. Likelihood weighting fix evidence, sample the rest; weigh each sample by probability of evident vars given parents. 因权重而不是次数计算.
找 evidence var 的 parent.
w = 1.0
for i = 1,2,...n
  if X_i is an evidence var: x_i = observed value; for X_i, set $w = w * P(x_i|Parent(X_i))$
  else: sample x_i from $P(X_i|parent(X_i))$.
return $(x_1,\cdots x_n), w$

• $S_{ws}(\vec{z},\vec{e})=\prod_{i=1}^{l} P(z_i|Parents(Z_i))$ z: 被 sampled, e: evident. ← q(x) 好 sample
$W(z,e)=\prod_{i=1}^{m} P(e_i|Parent(E_i))$. ← p(x)/q(x)
$S_{ws}(z,e)\cdot W(z,e)=\prod_{i=1}^{l+m=n} P(x_i|Parent(X_i))=P(z,e)$. ← p(x), 难 sample
• an instance of importance sampling
q(x): proposal distribution, 它的选择很大程度影响收敛速度.
If want to estimate E(f(x)), q(x) should be close to |f(x)|p(x).
• 下游不能影响上游 ∴ all samples are used ☺

4. Gibbs Sampling ① fix evidence ② 其他值随便取 ③ 每个 non-evidence, 根据它 Markov Blanket 里的人的取值 sampling. (MB 里是 evidence)
• Generate each sample by making a random change to the preceding sample
• $X_i' \sim P(X_i|x_1,\cdots x_n)$. 即 $P(X_i|x_1,\cdots x_{i-1},x_{i+1},\cdots x_n)=P(X_i|Markov Blanket(X_i))$
$=\alpha P(X_i|u_1,\cdots u_m)\prod_j P(y_j|parents(Y_j))$



• Gibbs Sampling is consistent.
• Markov Chain Monte Carlo (MCMC): a family of 随机算法 for 估计某些值 of interest over a very large state space
Markov Chain = a sequence of random chosen states, $X_i|X_{i-1}$.
Monte Carlo ∴ MCMC = sampling by constructing a Markov Chain.
e.g. Gibbs, Metropolis-Hastings, Hamiltonian, Slice, etc.
• Metropolis-Hastings
Repeat ① Draw a sample from proposal distribution $g(x'|x)$. 易 sample.
② Accept this sample with $P=\min\left(1,\frac{P(x')g(x|x')}{P(x)g(x'|x)}\right)$
Gibbs 是 MH 的特例, P == 1.

错题

1. H_1 consistent, H_2 inconsistent, $\frac{H_1+H_2}{2}$, $\max(H_1,H_2)$, $\min(H_1,H_2)$ 都不是 consistant.
2. h_1 admissible, $h_2=2h_1$, the solution found by A* tree search with h_2 is 保证 have a cost at most 2 much as the optimal path. $\bar{g}=g+h_2=g+2h_1\le 2(g+h_1)$
3. prime: 2,3,5,7,11,13,17,19,23,29,31,37,41,43,47,53,59,61,67,71,73 $\le 2C^*$.
4. A is valid ⇔ True entails A
$A\equiv B \Leftrightarrow (A\Leftrightarrow B)$ is valid
A entails B ⇔ A⇒B is valid
A entails B ⇔ $A\wedge\neg B$ is unsatisfiable
A entails $(B\wedge C)$ ⇒ (A entails B) ∧ (A entails C)
5. Treesearch(problem, strategy) returns a solution, or failure

```
initialize the search tree using the initial state of the problem
loop do
    if there are no candidate for expansion the return failure
    choose a leaf node for expansion according to the strategy
    if the node contains a goalstate then return 相应sol.
    else expand the node and add the resulting nodes to the search tree
end.
```

6. GraphSearch(problem, fringe, strategy)

```
closed ← {}
fringe ← INSERT(MAKE-NODE(INITIAL-STATE[problem]), fringe)
loop do
    if fringe is ∅ then return failure
    NODE ← REMOVE-FRONT(fringe, strategy)
    if GOAL-TEST(problem, state[NODE]) then return NODE
    if state(node) is not in closed then
        add state[node] to closed
        for child-node in EXPAND(STATE(node), problem) do
            fringe ← Insert(child-node, fringe)
        end
←end
```