CS24307 • MODULE 2

Module 2: Search Algorithms & Game Playing

Topics 8 to 13 • BFS/DFS/IDDFS, A* Admissibility Proofs & Alpha-Beta Pruning

---

Module 2 Table of Contents (Topics 8 to 13)

- **Topic 8:** Problem Formulation & State Space
- **Topic 9:** Uninformed Search (BFS, DFS, UCS, DLS, IDDFS)
- **Topic 10:** Informed Heuristic Search (A^*, Greedy, IDA*)
- **Topic 11:** Mathematical Admissibility & Consistency Proofs
- **Topic 12:** Adversarial Game Playing (Minimax Algorithm)
- **Topic 13:** Alpha-Beta Pruning Traces & Solved Numerical Bank

Topic 8: Formal Problem Formulation & State Space Representation

A classical search problem is formally defined by a 5-tuple:

$$\mathcal{P} = \langle \mathbf{S_0}, \text{Actions}(\mathbf{s}), \text{Result}(\mathbf{s}, \mathbf{a}), \text{GoalTest}(\mathbf{s}), \mathbf{c}(\mathbf{s}, \mathbf{a}, \mathbf{s'}) \rangle$$

1. **Initial State (S_0):** The starting state of the agent.
2. **Actions ($Actions(s)$):** The set of legal actions applicable in state s.
3. **Transition Model ($Result(s, a)$):** Returns the successor state s' produced by action a in state s.
4. **Goal Test ($GoalTest(s)$):** Boolean function determining if state s is a goal state.
5. **Path Cost ($c(s, a, s')$):** Step cost of taking action a from state s to s'. Total path cost is $g(n) = \sum \text{step costs}$.

🏛️ Step-by-Step Problem Formulation: The 8-Puzzle Problem

- **State:** 3×3 grid configuration containing numbers $\{1, 2, \ldots, 8\}$ and one blank space ($\text{Total States} = 9!/2 = 181,440$).
- **Initial State:** Any given permutation of the 8 tiles and blank.
- **Actions:** Move Blank $\{\text{Left}, \text{Right}, \text{Up}, \text{Down}\}$.
- **Goal Test:** Matches goal matrix `[[1,2,3],[8,blank,4],[7,6,5]]` (or standard sequential `[[1,2,3],[4,5,6],[7,8,blank]]`).
- **Path Cost:** Each tile move costs 1 unit ($g(n) = \text{number of moves}$).

Topic 9: Exhaustive Uninformed (Blind) Search Algorithms

Algorithm	Data Structure	Time Complexity	Space Complexity	Complete?	Optimal?
Breadth-First Search (BFS)	FIFO Queue	$O(b^d)$	$O(b^d)$ (Severe memory bottle-neck)	**Yes** (if $b < \infty$)	**Yes** (if step costs equal)
Uniform-Cost Search (UCS)	Priority Queue by $g(n)$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	**Yes** (if step cost $\geq \epsilon > 0$)	**Yes** (Cost optimal)
Depth-First Search (DFS)	LIFO Stack	$O(b^m)$ ($m = \max \text{depth}$)	$O(b \cdot m)$ (Linear memory!)	No (Infinite trees)	No
Depth-Limited Search (DLS)	LIFO Stack with limit l	$O(b^l)$	$O(b \cdot l)$	No (if $l < d$)	No
Iterative Deepening DFS (IDDFS)	LIFO Stack with increasing l	$O(b^d)$	$\mathbf{O(b \cdot d)}$ (Best of both worlds!)	**Yes**	**Yes** (if step costs equal)
Bidirectional Search	Two simultaneous Queues	$O(b^{d/2})$	$O(b^{d/2})$	**Yes**	**Yes** (if step costs equal)

Topic 10 & 11: Informed Heuristic Search (A^*) & Mathematical Proofs

A^* Search evaluates frontier nodes by the total estimated path cost function:

$$\mathbf{f(n) = g(n) + h(n)}$$

Where $g(n)$ is the exact cost from initial state to node n, and $h(n)$ is the estimated cost from node n to the closest goal state.

1. Admissibility Property (Tree-Search Optimality): A heuristic $h(n)$ is **admissible** if it never overestimates the true minimal cost $h^*(n)$ to reach the goal:

$$\mathbf{0 \leq h(n) \leq h^*(n) \quad \forall n}$$

2. Consistency / Monotonicity Property (Graph-Search Optimality): A heuristic $h(n)$ is **consistent** if for every node n and every successor n' generated by action a:

$$\mathbf{h(n) \leq c(n, a, n') + h(n')}$$

$$\text{Triangle Inequality: } \mathbf{f(n') = g(n') + h(n') = g(n) + c(n, a, n') + h(n') \geq g(n) + h(n) = f(n)}$$

Crucial Consequence: Monotonicity guarantees that $f(n)$ is non-decreasing along any path, ensuring that the first time a node is expanded in Graph-Search, the path is globally optimal!

🏛️ **Step-by-Step Solved Problem: Trace of A^* Search on Romania Map**

Find shortest path from **Arad to Bucharest** given straight-line distance heuristics h_{SLD}:

- $h(\text{Arad}) = 366, h(\text{Zerind}) = 374, h(\text{Timisoara}) = 329, h(\text{Sibiu}) = 253, h(\text{Fagaras}) = 176, h(\text{Rimnicu}) = 193, h(\text{Pitesti}) = 100, h(\text{Bucharest}) = 0$.

A^* Priority Queue Execution Trace:

1. Expand **Arad**: Front = $\{\text{Sibiu: } g = 140, f = 140 + 253 = 393;\ \text{Timisoara: } g = 118, f = 118 + 329 = 447;\ \text{Zerind: } g = 75, f = 75 + 374 = 449\}$.
2. Select lowest f: **Sibiu ($f = 393$)** → Expand Sibiu: Successors: $\{\text{Fagaras: } g = 140 + 99 = 239, f = 239 + 176 = 415;\ \text{Rimnicu: } g = 140 + 80 = 220, f = 220 + 193 = 413\}$.
3. Select lowest f: **Rimnicu Vilcea ($f = 413$)** → Expand Rimnicu: Successors: $\{\text{Pitesti: } g = 220 + 97 = 317, f = 317 + 100 = 417;\ \text{Craiova: } g = 220 + 146 = 366, f = 366 + 160 = 526\}$.
4. Select lowest f: **Fagaras ($f = 415$)** → Expand Fagaras: Successor: $\{\text{Bucharest: } g = 239 + 211 = 450, f = 450 + 0 = 450\}$.
5. Select lowest f: **Pitesti ($f = 417$)** → Expand Pitesti: Successor: $\{\text{Bucharest: } g = 317 + 101 = \mathbf{418}, f = 418 + 0 = \mathbf{418}\}$.
6. Select lowest f: $\text{Bucharest } (\mathbf{f = 418})$ → Goal reached!

$$\text{Optimal Path: Arad} \to \text{Sibiu} \to \text{Rimnicu Vilcea} \to \text{Pitesti} \to \text{Bucharest} \quad (\text{Total Cost } = \mathbf{418})$$

Topic 12 & 13: Adversarial Search, Minimax & Alpha-Beta Pruning

In a zero-sum two-player game, $\mathbf{MAX}$ seeks to maximize the utility score while $\mathbf{MIN}$ seeks to minimize it:

$$\text{Minimax}(\mathbf{s}) = \begin{cases} \text{Utility}(s) & \text{if Terminal}(s) \\ \max_{a \in Actions(s)} \text{Minimax}(\text{Result}(s,a)) & \text{if Player}(s) = \text{MAX} \\ \min_{a \in Actions(s)} \text{Minimax}(\text{Result}(s,a)) & \text{if Player}(s) = \text{MIN} \end{cases}$$

The Alpha-Beta Pruning Invariant: - α: The highest-value choice found so far at any choice point along the path for MAX (initialized to $-\infty$). - β: The lowest-value choice found so far at any choice point along the path for MIN (initialized to $+\infty$).

$$\text{Pruning Rule: Prune remaining subtree whenever } \alpha \geq \beta$$

🏛️ **Step-by-Step Solved Problem: Alpha-Beta Pruning Tree Execution**

Consider a 2-ply game tree with root MAX, children B, C, D (MIN), each having 2 terminal leaves:

- Node B: Leaves are $[3, 5] \implies \text{MIN}$ evaluates 3, updates $\beta = 3$. Evaluates $5 \implies \text{Node } B = 3$. Root MAX updates $\alpha = 3$.
- Node C: Left leaf is 2. MIN updates $\beta = 2$. Now $\alpha = \mathbf{3} \geq \beta = \mathbf{2} \implies \text{PRUNE RIGHT CHILD OF } \mathbf{C}$! (No need to evaluate right child).
- Node D: Left leaf is 0. MIN updates $\beta = 0$. Now $\alpha = \mathbf{3} \geq \beta = \mathbf{0} \implies \text{PRUNE RIGHT CHILD OF } \mathbf{D}$!
- $\text{Root Value } = \mathbf{3}$ (Move to B is optimal).

Topic 13.2: Master University Examination Solved Question Bank (10 Solved Questions)

Q1. Prove that A^* Search using tree-search is guaranteed to be optimal if the heuristic function $h(n)$ is admissible. (10 Marks)

Let G_2 be a sub-optimal goal state on the frontier, and $C^* = f(G^*) = g(G^*)$ be optimal goal cost. Suppose n is an unexpanded node on the path to optimal goal G^*. By admissibility, $h(n) \leq h^*(n)$, so $f(n) = g(n) + h(n) \leq g(n) + h^*(n) = C^*$. Since G_2 is sub-optimal, $f(G_2) = g(G_2) > C^* \geq f(n)$. Therefore, $f(n) \leq f(G_2)$ holds for all ancestors of optimal goal G^*, meaning A^* will always select and expand n before ever expanding sub-optimal goal G_2!

Q2. Compare Iterative Deepening Search (IDDFS) with Breadth-First Search (BFS) in terms of time and space complexity. (8 Marks)

BFS has time complexity $O(b^d)$ and space complexity $O(b^d)$, causing catastrophic out-of-memory errors on large trees. IDDFS combines the completeness and shallowest-goal optimality of BFS with the linear space complexity $O(bd)$ of DFS. In IDDFS, nodes at depth d are generated only once, depth $d-1$ twice, and the root $d+1$ times. The overhead factor is $\frac{b}{b-1}$, which is negligible for $b \geq 3$!

Q3. What is the Horizon Effect in Adversarial Game Playing? How is Quiescence Search used to mitigate it? (8 Marks)

The Horizon Effect occurs when an unavoidable catastrophic opponent move (e.g. losing a Queen) is pushed just beyond the search tree depth limit by executing stalling delaying moves (spurious checks). **Quiescence Search** mitigates this by extending the search beyond the fixed depth limit for "noisy" / unstable positions (e.g., capture sequences or checks) until the board reaches a quiescent (stable) state before evaluating the heuristic evaluation function.

Q4. Explain Constraint Satisfaction Problems (CSP). Trace the Backtracking Search with MRV and Forward Checking. (10 Marks)

A CSP is defined by Variables $X = \{X_1, \ldots, X_n\}$, Domains $D = \{D_1, \ldots, D_n\}$, and Constraints C. Standard backtracking is augmented by:
1. **Minimum Remaining Values (MRV / Most Constrained Variable):** Choose the variable with the fewest legal values remaining.
2. **Degree Heuristic:** Tie-breaker choosing variable with most constraints on remaining unassigned variables.
3. **Least Constraining Value (LCV):** Prefers the value that rules out the fewest choices for neighboring variables.
4. **Forward Checking:** Keeps track of remaining legal values for unassigned variables and terminates immediately if any domain becomes empty.

Q5. Explain Hill Climbing Search and its three primary failure modes. How does Simulated Annealing overcome them? (8 Marks)

Hill Climbing is a local search algorithm that iteratively moves in the direction of increasing elevation (greedy gradient ascent).
Failure Modes:
1. **Local Maxima:** A peak that is higher than neighboring states but lower than the global maximum.
2. **Ridges:** A sequence of local maxima where uphill moves are impossible in single coordinate directions.
3. **Plateaux / Shoulders:** Flat areas where the evaluation function is constant.
Simulated Annealing: Allows downhill moves with probability $P = e^{\Delta E/T}$. At high initial temperature T, it explores broadly; as $T \to 0$ (annealing schedule), it converges to the global maximum.

Q6. What is Genetic Algorithm (GA)? Explain Selection, Crossover, and Mutation with bit-string representations. (8 Marks)

A Genetic Algorithm is a stochastic local search inspired by natural evolution:
1. **Initial Population:** Set of candidate state bit-strings.
2. **Fitness Function:** Assigns higher numeric reproduction probability to better states.
3. **Selection:** Roulette wheel or tournament selection picks parent chromosomes proportional to fitness.
4. **Crossover:** Recombines parent bit-strings at random crossover points to produce offspring.
5. **Mutation:** Randomly flips individual bits with small probability p_m to maintain genetic diversity and prevent premature convergence.

Q7. Formulate the 8-Queens Problem as a State Space Search Problem. (6 Marks)

- **States:** Any arrangement of 0 to 8 queens on an 8×8 chessboard.
- **Initial State:** No queens on the board.
- **Actions:** Add a queen to any empty square in the next empty column such that no queen attacks another.
- **Goal Test:** 8 queens placed on the board with no two queens attacking each other (same row, column, or diagonal).
- **Path Cost:** Zero (only the final state matters).

Q8. Explain Iterative Deepening A^* (IDA*) and compare it with Memory-Bounded A^* (SMA*). (8 Marks)

IDA* is a memory-bounded heuristic search that uses depth-first search with an f-cost limit. Each iteration expands nodes until $f(n) > \text{limit}$. The next iteration's limit is the minimum f-value among all pruned nodes. It consumes only $O(bd)$ memory while retaining the cost-optimality of A^*! SMA* uses all available memory and prunes worst-performing subtrees while backing up values to parents.

Q9. Explain the Minimax Algorithm for Game Trees with Chance Nodes (Expectiminimax). (8 Marks)

In games with stochastic elements (e.g., Backgammon dice rolls), **Expectiminimax** adds *Chance Nodes* between MAX and MIN layers. The value of a Chance Node is the mathematical expected value: $\text{Expectivalue}(s) = \sum_s P(s) \cdot \text{Expectiminimax}(s)$. Alpha-Beta pruning can be adapted if bounds on evaluation functions are strictly known.

Q10. Calculate the Effective Branching Factor (b^*) if A^* search finds a solution at depth $d = 5$ after expanding $N = 150$ nodes. (6 Marks)

Using the polynomial formula $N + 1 = 1 + b^* + (b^*)^2 + (b^*)^3 + (b^*)^4 + (b^*)^5 = 151$:
For $b^* = 2$: $1 + 2 + 4 + 8 + 16 + 32 = 63 < 151$.
For $b^* = 2.5$: $1 + 2.5 + 6.25 + 15.625 + 39.0625 + 97.65625 = 162.09 > 151$.
Interpolating: $\mathbf{b^* \approx 2.45}$.

Topic 13.5: Local Search, Continuous Optimization & Genetic Algorithms

When the path to the goal is irrelevant and only the final state configuration matters (e.g. 8-Queens, VLSI circuit layout, protein folding), **Local Search Algorithms** operate on a single current state rather than maintaining extensive search tree paths.

Algorithm	Core Operational Mechanism	Space Complexity & Global Optimality
Hill-Climbing (Greedy Local Search)	Continuously moves in the direction of increasing elevation/objective value ($s' = \arg\max_{n \in \text{Neighbors}(s)} f(n)$). Terminates when no neighbor has higher value.	$\mathbf{O(1)}$ space; prone to getting trapped in local maxima, ridges, and plateau shoulders.
Simulated Annealing	Accepts uphill moves unconditionally ($\Delta E > 0$). Downhill moves are accepted with Boltzmann probability $\mathbf{P(\text{accept}) = e^{\Delta E / T}}$, where temperature T decreases according to cooling schedule $T(t)$.	$\mathbf{O(1)}$ space; asymptotically guaranteed to find global optimum if temperature decreases sufficiently slowly ($\sum_k T_k = \infty$).
Local Beam Search	Maintains k states in parallel. At each step, all successors of all k states are generated, and only the top k best successors across the entire pool are retained.	$\mathbf{O(k)}$ space; facilitates information sharing among parallel exploratory search trajectories.
Genetic Algorithms (GA)	Populations of candidate state chromosomes undergo fitness-proportional Selection (Roulette Wheel / Tournament), Crossover recombination, and point Mutation.	$\mathbf{O(\text{PopSize} \cdot L)}$; powerful global explorer across complex rugged discontinuous fitness landscapes.

🏛️ **Step-by-Step Solved Problem: Simulated Annealing Acceptance Probability Calculations**

An optimization problem minimizing cost E considers a neighbor with cost change $\Delta E = E_{\text{new}} - E_{\text{current}} = +15$ (worse by 15 units).

1. Calculate the acceptance probability at initial high temperature $T = 100$.
2. Calculate the acceptance probability at medium temperature $T = 20$.
3. Calculate the acceptance probability at freezing temperature $T = 1.0$.

Solution:

$$\mathbf{P = e^{-\Delta E / T}}$$

- At $T = 100$: $P = e^{-15/100} = e^{-0.15} \approx \mathbf{0.8607 = 86.07\%}$ (Very high exploratory freedom!).
- At $T = 20$: $P = e^{-15/20} = e^{-0.75} \approx \mathbf{0.4724 = 47.24\%}$ (Balanced exploration/exploitation).
- At $T = 1.0$: $P = e^{-15/1.0} = e^{-15} \approx \mathbf{3.059 \times 10^{-7} \approx 0.00003\%}$ (Pure greedy hill climbing!).

🏛️ **Step-by-Step Solved Problem: Constraint Propagation via the AC-3 Algorithm**

Let variables $X_1, X_2, X_3 \in \{1, 2, 3, 4\}$ be constrained by $X_1 < X_2$ and $X_2 < X_3$. Trace the **AC-3 (Arc Consistency 3) Algorithm**:

1. Initial domains: $D_1 = \{1, 2, 3, 4\}, D_2 = \{1, 2, 3, 4\}, D_3 = \{1, 2, 3, 4\}$.
2. Queue of Arcs = $\{(X_1, X_2), (X_2, X_1), (X_2, X_3), (X_3, X_2)\}$.
3. Pop (X_1, X_2): For $X_1 = 4$, no value in D_2 satisfies $4 < X_2$. Remove 4 from $D_1 \implies \mathbf{D_1 = \{1, 2, 3\}}$.
4. Pop (X_2, X_3): For $X_2 = 4$, no value in D_3 satisfies $4 < X_3$. Remove 4 from $D_2 \implies \mathbf{D_2 = \{1, 2, 3\}}$. Arc (X_1, X_2) re-added.
5. Pop (X_3, X_2): For $X_3 = 1$, no value in D_2 satisfies $X_2 < 1$. Remove 1 from $D_3 \implies \mathbf{D_3 = \{2, 3, 4\}}$.
6. Pop (X_1, X_2): For $X_1 = 3$, since $D_2 = \{1, 2, 3\}$, only $X_2 = 4$ could satisfy $3 < X_2$, but $4 \notin D_2$. Remove 3 from $D_1 \implies \mathbf{D_1 = \{1, 2\}}$.
7. Pop (X_2, X_1): For $X_2 = 1$, no value in $D_1 = \{1, 2\}$ satisfies $X_1 < 1$. Remove 1 from $D_2 \implies \mathbf{D_2 = \{2, 3\}}$.
8. Pop (X_3, X_2): For $X_3 = 2$, since $D_2 = \{2, 3\}$, no value satisfies $X_2 < 2$. Remove 2 from $D_3 \implies \mathbf{D_3 = \{3, 4\}}$.
9. Pop (X_1, X_2): For $X_1 = 2, X_2 = 3 \in D_2$ satisfies $2 < 3$. Valid!

$$\mathbf{\text{Final Arc-Consistent Domains: } D_1 = \{1, 2\}, \quad D_2 = \{2, 3\}, \quad D_3 = \{3, 4\}}$$

Q12. Prove that Depth-First Search with Graph Search is NOT optimal even for uniform step costs. (6 Marks)

Consider a graph where Start S has two edges: $S \xrightarrow{1} G$ (direct goal path, depth 1) and $S \xrightarrow{1} A \xrightarrow{1} G$ (depth 2). If DFS expands A first because of arbitrary node ordering, it adds A to the explored set and then reaches G via path $S \to A \to G$ with cost 2. When it backtracks to S, G is already in the explored set, so the direct optimal path $S \to G$ (cost 1) is never explored! Hence DFS is not cost-optimal.

Q13. What is the difference between Minimax with Alpha-Beta Pruning and SSS* (State Space Search) in Game Trees? (8 Marks)

• **Minimax with Alpha-Beta:** A depth-first traversal of the game tree that maintains single-path scalar bounds (α, β) and consumes only $O(bd)$ memory. It evaluates nodes in a fixed depth-first order.
• **SSS* (Stockman 1979):** A best-first branch-and-bound search that maintains an open list of game tree solution trees (sets of strategies). SSS* never expands any leaf that Alpha-Beta prunes, and can prune additional nodes that Alpha-Beta explores. However, SSS* requires exponential queue memory $O(b^{d/2})$, making it computationally impractical for deep game trees like Chess and Go!

Topic 13.6: Exhaustive Adversarial Game Theory & Alpha-Beta Deep Traces

Alpha-Beta Pruning Efficiency Theorems: - **Worst-Case Move Ordering (Worst-first):** Alpha-Beta examines every leaf node $\implies O(b^d)$ (No pruning advantage over standard Minimax). - **Ideal / Optimal Move Ordering (Best-first):** The effective branching factor is reduced from b to $\sqrt{b} \implies \mathbf{O(b^{d/2})}$. - *Profound Consequence:* With optimal move ordering, an Alpha-Beta search can search **twice as deep** in the same compute time as standard Minimax (e.g., searching 12 plies instead of 6 plies in Chess)!

🏛️ **Step-by-Step Solved Problem: Full 3-Ply Alpha-Beta Pruning Tree Trace**

Consider a 3-ply Minimax tree where Root is MAX, Level 1 is MIN (B, C), Level 2 is MAX (D, E, F, G), and Level 3 has 8 terminal leaves:

- Node D leaves: $[4, 6] \implies \text{Node } D = \max(4, 6) = 6$.
- Node E leaves: $[7, 9] \implies \text{Node } E = \max(7, 9) = 9$.
- Node B (MIN of D, E): $\beta = \min(6, 9) = 6$. Updates Root MAX: $\alpha = 6$.
- Node F left leaf: 1. Since Node F is MAX, $F \geq 1$. Node F right leaf: $2 \implies \text{Node } F = 2$.
- Node C (MIN): Updates $\beta = 2$.
- Now at Node C, we have $\alpha = \mathbf{6} \geq \beta = \mathbf{2} \implies$ PRUNE NODE **G** AND ALL ITS LEAVES COMPLETELY!
- Root Optimal Value $= \mathbf{6}$ (Optimal Move: $\mathbf{A} \to \mathbf{B}$).

Q14. Explain Transposition Tables in Game Tree Search. How do Zobrist Hashing keys operate? (8 Marks)

In games like Chess, different move orders can lead to the identical board position (Transposition: $1.e4\ e5\ 2.Nf3\ Nc6$ vs $1.Nf3\ Nc6\ 2.e4\ e5$). A **Transposition Table** is a high-speed hash map caching previously evaluated board states. **Zobrist Hashing:** Assigns a random 64-bit integer to each `(piece, square)` tuple. A board position's hash is computed by XORing ($\oplus$) all present piece keys. When a move occurs, updating the hash takes $O(1)$ XOR operations ($\text{Hash}' = \text{Hash} \oplus \text{Key}_{\text{old}} \oplus \text{Key}_{\text{new}}$), enabling millions of board lookups per second!

Topic 13.7: Advanced Adversarial Search & Monte Carlo Tree Search (MCTS)

In ultra-large state space games where evaluation functions cannot be hand-crafted (e.g. Go, Chess variants), **Monte Carlo Tree Search (MCTS)** uses stochastic rollouts to evaluate positions.



Figure 2.1: The Four Discrete Phases of Monte Carlo Tree Search (MCTS / AlphaGo)

The Upper Confidence Bound for Trees (UCT) Formula:

$$\text{UCT}(\mathbf{v_i}, \mathbf{v}) = \frac{\mathbf{Q(v_i)}}{\mathbf{N(v_i)}} + \mathbf{c}\sqrt{\frac{\ln \mathbf{N(v)}}{\mathbf{N(v_i)}}} = \text{Exploitation Term} + \text{Exploration Term}$$

Where $Q(v_i)$ is total simulation reward, $N(v_i)$ is child visit count, $N(v)$ is parent visit count, and $c = \sqrt{2}$ is the exploration parameter balancing exploitation of winning moves with exploration of rarely visited branches.

🏛️ Step-by-Step Solved Problem: MCTS UCT Calculation

A parent node v has been visited $N(v) = 100$ times. It has two children A and B:

- Child A: Visited $N(A) = 60$ times with total wins $Q(A) = 45$.
- Child B: Visited $N(B) = 40$ times with total wins $Q(B) = 28$.

$$\text{UCT}(A) = \frac{45}{60} + \sqrt{2}\sqrt{\frac{\ln 100}{60}} = 0.75 + 1.414\sqrt{\frac{4.605}{60}} = 0.75 + 1.414(0.277) = 0.75 + 0.392 = \mathbf{1.142}$$

$$\text{UCT}(B) = \frac{28}{40} + \sqrt{2}\sqrt{\frac{\ln 100}{40}} = 0.70 + 1.414\sqrt{\frac{4.605}{40}} = 0.70 + 1.414(0.339) = 0.70 + 0.479 = \mathbf{1.179}$$

Selection Decision: Choose Child $\mathbf{B}$ because its exploration potential (1.179) outweighs Child $\mathbf{A}$ (1.142)!

Topic 13.8: Complete Step-by-Step Solved Problem Bank (Part III)

🏛️ Solved Numerical 3: IDA* (Iterative Deepening A*) Search Execution Trace

Consider a search tree with initial state S and goal G with branch costs and heuristics:

- $S \xrightarrow{2} A\ (h=5),\ S \xrightarrow{3} B\ (h=4)$.
- $A \xrightarrow{4} C\ (h=3),\ A \xrightarrow{5} D\ (h=2)$.
- $B \xrightarrow{2} E\ (h=3),\ B \xrightarrow{6} G\ (h=0)$.
- $C \xrightarrow{2} G\ (h=0)$.

IDA* Iterations:

1. **Iteration 1 (Cutoff = $f(S) = g(S) + h(S) = 0 + 6 = 6$):**
   - Expand S: $f(A) = 2 + 5 = 7 > 6$ (Pruned, min cutoff = 7); $f(B) = 3 + 4 = 7 > 6$ (Pruned, min cutoff = 7).
   - Threshold for Iteration 2 is set to **7**.
2. **Iteration 2 (Cutoff = 7):**
   - Expand S: $A\ (f = 7 \le 7), B\ (f = 7 \le 7)$.
   - Expand A: $f(C) = 2 + 4 + 3 = 9 > 7$ (Pruned, cutoff=9); $f(D) = 2 + 5 + 2 = 9 > 7$ (Pruned, cutoff=9).
   - Expand B: $f(E) = 3 + 2 + 3 = 8 > 7$ (Pruned, cutoff=8); $f(G) = 3 + 6 + 0 = 9 > 7$ (Pruned, cutoff=9).
   - Next threshold is $\min(9, 9, 8, 9) = \mathbf{8}$.
3. **Iteration 3 (Cutoff = 8):**
   - Expand $S \to B \to E$: Successors of $E \dots$
   - Next threshold = **8**. Goal G reached via path $S \to C \to G$ with cost **8**!

Q17. Explain Simulated Annealing Cooling Schedules and the Metropolis Criterion. (8 Marks)

The **Metropolis Criterion** dictates that downhill moves (cost improvements) are accepted with probability 1.0, while uphill moves ($\Delta E > 0$) are accepted with probability $P = e^{-\Delta E / T}$.

Cooling Schedules:

1. *Linear Cooling:* $T(t) = T_0 - \alpha t$.
2. *Geometric / Exponential Cooling:* $T(t) = T_0 \cdot \alpha^t$ (where $\alpha \in [0.80, 0.99]$).
3. *Logarithmic Cooling (Geman & Geman):* $T(t) = \frac{C}{\ln(1+t)}$. This is the only schedule mathematically proven to converge to the global optimum, but requires astronomically many iterations!