**Informed Search Algorithms – Research-Based Tasks**

**🔰 Level 1: Understanding Core Concepts**

**Task 1: What is a Heuristic?**

**Objective:**  
Research and define the term *heuristic* in the context of AI and search algorithms.

* What does a heuristic function do?
* How does it differ from path cost?
* Why is it used in informed search?

**Deliverable:**  
A one-page written explanation with real-world examples.

**Task 2: Admissible vs Inadmissible Heuristics**

**Objective:**

* Research what it means for a heuristic to be admissible.
* Find at least **two examples** of admissible heuristics and **one inadmissible** heuristic.
* Explain the consequences of using an inadmissible heuristic in A\*.

**Deliverable:**  
Table comparing admissible/inadmissible heuristics with examples.

**Task 3: Consistent (Monotonic) Heuristic**

**Objective:**

* Define a consistent heuristic.
* What is the mathematical condition for consistency?
* Find a situation where a heuristic is admissible but not consistent.

**Deliverable:**  
Summary of findings + example scenario.

**Task 4: Differences Between Tree Search and Graph Search**

**Objective:**

* Research the difference between **tree search** and **graph search** in the context of search algorithms.
* What are the advantages and limitations of each?
* When would each be preferred?

**Deliverable:**  
Comparison table + written explanation.

**🧭 Level 2: Exploring Specific Algorithms**

**Task 5: Greedy Best-First Search (GBFS)**

**Objective:**

* Research how GBFS works.
* How does it select the next node to expand?
* What are its strengths and weaknesses?

**Deliverable:**  
Summary report + diagram of node selection.

***Task 6: A Search Algorithm*\***

**Objective:**

* Explain how A\* combines g(n) and h(n) to compute f(n).
* What is the role of each component?
* In what scenarios is A\* considered optimal?

**Deliverable:**  
Written explanation + one example problem (e.g., path in a map) with annotated steps (no coding).

***Task 7: Why A is Optimal (with Admissible Heuristic)*\***

**Objective:**

* Explore the reasoning or proof sketch of why A\* is guaranteed to find an optimal path if h(n) is admissible.
* Why is consistency even better?

**Deliverable:**  
Write-up in bullet points + cited source.

**Task 8: Comparison Table – BFS, DFS, GBFS, A\***

**Objective:**  
Create a table comparing the following algorithms:

* Breadth-First Search
* Depth-First Search
* Greedy Best-First Search
* A\* Search

**Compare on:**

* Use of heuristic
* Completeness
* Optimality
* Time complexity
* Space complexity

**Deliverable:**  
Clean comparison table.

**🌍 Level 3: Real-World and Heuristic Design**

**Task 9: Heuristics in Real Life**

**Objective:**  
Find examples of heuristics used in **real-world applications**:

* GPS navigation
* Game AI (e.g., Chess, Pacman)
* Robotics

**Deliverable:**  
One-slide explanation for each real-world example (include images if possible).

**Task 10: Designing a Heuristic Function (Conceptual)**

**Objective:**  
Choose one of the following:

* Maze solving
* Food delivery path planning
* Robot vacuum cleaning

Design a heuristic *conceptually*:

* What would you estimate as the remaining cost to goal?
* What factors would your heuristic consider?

***TASK1:-***

**Heuristic Functions in AI and Search Algorithms**

**Definition**

**A heuristic function is a method or algorithm used to estimate the solution to a problem when exact solutions are computationally infeasible. It's a vital component of artificial intelligence that enables AI systems to make informed decisions and find efficient solutions to complex problems ¹.**

**What Does a Heuristic Function Do?**

**A heuristic function estimates the cost or distance from a given node or state to the goal node. It provides a "rule of thumb" or an educated guess that guides the search algorithm towards the most promising areas of the search space. This estimation helps AI systems prioritize paths and reduce computational effort ².**

**How Does it Differ from Path Cost?**

**Path Cost vs. Heuristic Function:**

**- Path Cost: The actual cost incurred to reach a node from the starting node, based on the cumulative cost of actions taken so far.**

**- Heuristic Function: An estimate of the future cost from the current node to the goal node, providing guidance on which paths are most likely to lead to the solution.**

**Why is it Used in Informed Search?**

**Heuristic functions are used in informed search algorithms, such as A\* search, to make the search process more efficient. By estimating the distance to the goal, the algorithm can prioritize nodes that are likely to lead to a solution more quickly, reducing the number of nodes that need to be explored.**

**Real-World Examples:**

**- Pathfinding Algorithm: A navigation system uses a heuristic function to predict the quickest path between two points, without computing all possibilities.**

**- Autonomous Vehicles: A self-driving car's AI system applies a pathfinding algorithm, like A\*, to calculate the best route, using heuristics to estimate the cost or distance to the destination.**

**- Robotics: Heuristic functions can guide robots in complex environments, helping them make decisions and find efficient paths to achieve their goals ³ ¹.**

**Benefits of Heuristic Functions**

**- Efficiency: Heuristic functions reduce the search space and computing time by guiding the algorithm towards the most promising paths.**

**- Scalability: Heuristics enable AI systems to tackle large and complex problems that would be impractical to solve exhaustively.**

**- Informed Decision-Making: Heuristics provide a basis for AI systems to make intelligent decisions, even with limited information ¹.**

**TASK 2:-**

**What Does Admissible Heuristic Mean?**

* **A heuristic is a "smart guess" of how far it is from a point to the goal.**
* **A heuristic is admissible if it never overestimates the real cost to reach the goal.**
* **It always gives a value that is equal to or less than the actual cost.**

**✅ Example:  
If the real cost to the goal is 10, an admissible heuristic might guess 6, 8, or even 10 — but never 11 or more.**

**❗ What Happens if Heuristic is Inadmissible?**

* **If the heuristic overestimates (gives a number too high), it is inadmissible.**
* **This can confuse A\* — it might miss the shortest path and take a wrong one.**
* **Result: The path found may not be the best.**

**📊 Comparison Table: Admissible vs Inadmissible Heuristics**

| **Feature** | **Admissible Heuristic** | **Inadmissible Heuristic** |
| --- | --- | --- |
| **Definition** | **Never overestimates the cost to the goal** | **May give a guess higher than real cost** |
| **Guarantees best path?** | **✅ Yes, A\* finds the shortest path** | **❌ No, A\* might choose a longer path** |
| **Used in A\*** | **Ideal for correct and optimal results** | **Risky — may work faster but give wrong path** |
| **Example 1** | **Manhattan Distance (grid moves in straight lines)** | **—** |
| **Example 2** | **Euclidean Distance (as-the-crow-flies distance)** | **—** |
| **Bad Example** | **—** | **Heuristic = 2 × real cost (too high)** |

**🧠 Simple Examples**

**✅ Admissible Heuristics:**

1. **Manhattan Distance**
   * **Used in grid maps where movement is only up, down, left, or right.**
   * **Example: From point A to B, it takes 3 steps right and 2 steps up → total = 5.**
   * **Always gives a safe guess.**
2. **Euclidean Distance**
   * **Straight-line distance between two points (like with a ruler).**
   * **Common in open maps or games with free movement.**

**❌ Inadmissible Heuristic:**

* **Let’s say the real cost from a point to the goal is 6.  
  But the heuristic wrongly says it's 10.  
  ➤ A\* might avoid that path, thinking it's too long — and miss the best solution**

**TASK 3:-**

**What is a Consistent Heuristic?**

**A consistent heuristic is a smart guess that follows a rule:**

**“The cost from one step to another + the guess from there should always be at least as much as your guess from the start.”**

**This makes sure that:**

* **The A\* algorithm works fast and smooth**
* **It doesn’t go back and check the same paths again**

**🟡 What’s the Rule for a Heuristic to Be Consistent?**

**A consistent heuristic must:**

* **Never guess too high**
* **Increase step by step in a reasonable way (like moving forward without shortcuts)**

**✅ What is an Admissible Heuristic?**

**An admissible heuristic is just a safe guess — it never guesses more than the real cost to reach the goal.**

**❓ Can a Heuristic Be Admissible but Not Consistent?**

**Yes! A guess can be safe (not too high) but still jump around in strange ways. That means:**

* **It’s admissible ✅**
* **But not consistent ❌**

**🔍 Easy Example:**

**Imagine a small map with 3 places:**

**A → B → Goal**

* **Going from A to B costs 1**
* **B to Goal costs 1**
* **So real cost A to Goal is 2**

**Heuristic guesses:**

* **Goal: 0**
* **B: 0**
* **A: 2**

**This is:**

* **Admissible (no guess is more than real cost) ✅**
* **Not consistent (A says 2, but B says 0, and A→B only costs 1 — the jump is too big) ❌**

**📌 SUMMARY TABLE**

|  |  |  |
| --- | --- | --- |
| **Term** | **Meaning** | **Notes** |
| **Admissible** | **Never guesses too high** | **Safe but not always smooth** |
| **Consistent** | **Step-by-step guesses increase smoothly** | **A\* works faster** |
| **Can be Admissible but not Consistent?** | **Yes** | **It may guess safely but not smoothly** |

**This violates the consistency condition, even though the heuristic is admissible.**

|  |  |  |
| --- | --- | --- |
| **Aspect** | **Tree Search** | **Graph Search** |
| **What it does** | **Explores all paths, even repeated ones** | **Explores paths and avoids repeated states** |
| **Cycle handling** | **May get stuck in loops** | **Avoids loops by tracking visited states** |
| **Memory use** | **Uses less memory** | **Uses more memory to store visited nodes** |
| **Time taken** | **Can be slower due to repeated work** | **Faster by skipping already seen paths** |
| **Finds best path** | **Not always guaranteed** | **More likely, especially with A\* algorithm** |
| **When to use** | **For small or simple problems** | **For large or complex problems with loops** |

**TASK 4:-**

**Tree Search vs Graph Search**

**Table:**

**Tree Search:**

* **Does not remember visited nodes**
* **Can explore the same state multiple times**
* **Simple and uses less memory**
* **Risk of going in circles**

**Graph Search:**

* **Remembers visited nodes**
* **Avoids repeating the same paths**
* **Uses more memory**
* **Works better for finding the best path**

**Use Tree Search for simple problems without cycles.  
Use Graph Search for large or complex problems with repeated states.**

**Ask ChatGPT**

**Task 5: How Greedy Best-First Search (GBFS) Works**

**GBFS is a search algorithm that always chooses the node that seems closest to the goal, using a guess called the heuristic.**

**It uses only h(n), the estimated cost from the current node to the goal. It does not care how far the node is from the start.**

**How GBFS selects nodes:**

1. **Keeps a list of nodes to explore.**
2. **Picks the one with the lowest h(n).**
3. **Expands it and adds its neighbors.**
4. **Repeats until goal is found or all nodes are tried.**

**Example:  
Start at A. If C looks closer to goal than B (lower h), it picks C. Even if B leads to a shorter path, GBFS may miss it.**

**Advantages:**

* **Fast in open areas**
* **Easy to code**
* **Focused on goal**

**Limitations:**

* **May miss the best path**
* **Can loop if states are repeated**
* **Depends on good heuristic**
* **Ignores real cost from start**

**Comparison with A\*:**

* **GBFS: f(n) = h(n)**
* **A\*: f(n) = g(n) + h(n)**
* **GBFS is faster but not always correct**
* **A\* is slower but gives best path if heuristic is good**

***Task 6: How A Search Works*\***

**A\* combines the real cost from start (g) and the estimated cost to goal (h) to decide which node to expand.**

**f(n) = g(n) + h(n)**

**Parts of the cost:**

* **g(n): Cost from start to current node**
* **h(n): Estimated cost from current node to goal**
* **f(n): Total cost guess**

***When A is correct:*\***

* **If h(n) never guesses too high (admissible)**
* **If h(n) follows triangle rule (consistent)**

**Example:  
Start at A  
A to C: cost = 4  
C to F: cost = 1  
F to G: cost = 3  
Total path A → C → F → G = 8 (lowest)**

**A\* picks nodes with lowest f(n) at each step.**

**Final summary:**

* **g(n): actual cost so far**
* **h(n): guess to goal**
* **f(n): total estimate**
* **A\* gives shortest path if heuristic is good**

***Task 6: A Search Algorithm*\***

**A\* chooses the best path by using:**

* **g(n): cost from start to current node**
* **h(n): estimated cost from current to goal**
* **f(n) = g(n) + h(n): total estimated cost**

**Roles:**

* **g(n): keeps track of real steps taken**
* **h(n): guesses how close the goal is**
* **f(n): helps pick the best node to expand**

***When is A optimal?*\***

* **When h(n) is admissible (never overestimates)**
* **Even better if h(n) is consistent (costs grow logically)**

**Example (Map):  
A → C → F → G  
Costs: A–C = 4, C–F = 1, F–G = 3  
Heuristics: A=7, C=2, F=1, G=0  
A\* chooses C (lowest f), then F, then G  
Final path cost = 8 (shortest)**

***Task 7: Why A is Optimal (if h is admissible)*\***

* **If h(n) never guesses higher than real cost, A\* won’t miss the best path**
* **It explores paths in increasing order of f(n)**
* **First time goal is reached, it’s the shortest path**

**Why consistency is better:**

* **Consistent h(n) means no backtracking**
* **A\* doesn't need to re-expand nodes**
* **Faster and more efficient**

**Key point:  
Admissible → optimal  
Consistent → optimal + efficient**

**Source:  
Russell & Norvig, *Artificial Intelligence: A Modern Approach***

**Task 8: Search Algorithm Comparison Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | **BFS** | **DFS** | **GBFS** |
| **Heuristic used** | **No** | **No** | **Yes (h)** |
| **Complete** | **Yes** | **No** | **No** |
| **Optimal** | **Yes** | **No** | **No** |
| **Time Complexity** | **High** | **Low** | **Depends on h** |
| **Space Complexity** | **High** | **Low** | **Medium** |

**Task 9: Heuristics in Real Life**

**GPS Navigation**

* **Heuristic: straight-line distance to destination**
* **Helps avoid traffic-heavy or long routes**

**Game AI (Chess)**

* **Heuristic: board score based on piece values and positions**
* **Helps the AI plan winning moves**

**Robotics (Vacuum cleaner)**

* **Heuristic: remaining dirty area distance**
* **Helps plan the next best move to clean efficiently**

**Task 10: Designing a Heuristic – Food Delivery**

**Goal: Shortest delivery time**

**Heuristic idea:  
Estimate time left = distance to customer / average speed**

**Factors to consider:**

* **Distance left**
* **Traffic on roads**
* **Type of road (highway vs. narrow)**
* **Time of day (rush hour or not)**