Heuristic Analysis

Definitions and key points from [Artificial Intelligence: A Modern Approach]: (they are helpful in explaining the test results we see for problems 1, 2 and 3.)

- **Breadth-first search** is a simple strategy in which the root node is expanded first, then all the successors of the root node are expanded next, then their successors, and so on. Breadth-first search is optimal if the path cost is a non-decreasing function of the depth of the node. The most common such scenario is that all actions have the same cost.

- Instead of expanding the shallowest node, **uniform-cost search** expands the node n with the lowest path cost g(n). This is done by storing the frontier as a priority queue ordered by g. In addition to the ordering of the queue by path cost, there are two other significant differences from breadth-first search. The first is that the goal test is applied to a node when it is selected for expansion rather than when it is first generated. The reason is that the first goal node that is generated may be on a suboptimal path. The second difference is that a test is added in case a better path is found to a node currently on the frontier.

- **Depth-first graph search** always expands the deepest node in the current frontier of the search tree. The time complexity of depth-first graph search is bounded by the size of the state space (which may be infinite, of course). The depth limit search solves the infinite-path problem. Unfortunately, it also introduces an additional source of incompleteness if we choose $l < d$, that is, the shallowest goal is beyond the depth limit.

- **Greedy best-first search** tries to expand the node that is closest to the goal, on the grounds that this is likely to lead to a solution quickly. Thus, it evaluates nodes by using just the heuristic function; that is, $f(n) = h(n)$.

- The most widely known form of best-first search is called **A∗ search**. It evaluates nodes by combining g(n), the cost to reach the node, and h(n), the cost to get from the node to the goal: $f(n) = g(n) + h(n)$. Since g(n) gives the path cost from the start node to node n, and h(n) is the estimated cost of the cheapest path from n to the goal, we have $f(n)$ = estimated cost of the cheapest solution through n. The algorithm is identical to **uniform-cost search** except that A∗ uses g + h instead of g.

- [Artificial Intelligence A Modern Approach] page 91 figure 3.21:

Criterion	Breadth-First	Uniform-Cost	Depth-First	Depth-Limited	Iterative Deepening	Bidirectional (if applicable)
Complete?	Yes[a]	Yes[a,b]	No	No	Yes[a]	Yes[a,d]
Time	$O(b^d)$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(b^m)$	$O(b^\ell)$	$O(b^d)$	$O(b^{d/2})$
Space	$O(b^d)$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(bm)$	$O(b\ell)$	$O(bd)$	$O(b^{d/2})$
Optimal?	Yes[c]	Yes	No	No	Yes[c]	Yes[c,d]

Figure 3.21 Evaluation of tree-search strategies. b is the branching factor; d is the depth of the shallowest solution; m is the maximum depth of the search tree; l is the depth limit. Superscript caveats are as follows: [a] complete if b is finite; [b] complete if step costs $\geq \epsilon$ for positive ϵ; [c] optimal if step costs are all identical; [d] if both directions use breadth-first search.

Heuristics tested:

- **H_1**: this is set to constant 1 – not a true heuristic.
- **H_Ignore_Preconditions**: this heuristic estimates the minimum number of actions that must be carried out from the current state in order to satisfy all of the goal conditions by ignoring the preconditions required for an action to be executed.
- **H_PG_Levelsum**: this heuristic uses a planning graph representation of the problem state space to estimate the sum of all actions that must be carried out from the current state in order to satisfy each individual goal condition.

Air Cargo Problem 1 -

Search	Expansions	Goal Tests	Time Elapsed
Breadth First Search	43	56	0.0307
Breadth First Tree Search	1458	1459	0.8700
Depth First Graph Search	21	22	0.0133
Depth Limited Search	101	271	0.0818
Uniform Cost Search	55	57	0.0357
Recursive Best First Search with H_1	4229	4230	2.6739
Greedy Best First Graph Search with H_1	7	9	0.0048
A* Search with H_1 (identical to UCS)	55	57	0.0367
A* Search with H_Ignore_Preconditions	41	43	0.0420
A* Search with H_PG_Levelsum	11	13	2.3389

- Optimal plan: (Greedy Best First Graph Search)
  Load(C1, P1, SFO)
  Load(C2, P2, JFK)
  Fly(P1, SFO, JFK)

Fly(P2, JFK, SFO)
Unload(C1, P1, JFK)
Unload(C2, P2, SFO)

- Compare and contrast non-heuristic search result metrics:
  Uniform Cost Search, Breadth First, Breadth First Tree, Recursive Best First Search has the same plan as Greedy Best First Graph Search, with more nodes expansions and longer time elapsed. This makes sense since they are optimal in this case given step costs are all identical. That also means there's no material difference between Uniform Cost and Breadth First Search. Depth First Graph (plan length 20) and Depth Limited (plan length 50) Search output much longer plans which are not optimal. "The time complexity of depth-first graph search is bounded by the size of the state space" which may be infinite in this problem – imagine the planes flying around indefinitely without ever arrive at the right airport with the right cargo.

- Compare and contrast heuristic search result metrics using A* with the "ignore preconditions" and "level-sum" heuristics:
  Both search output the optimal plan. H_Ignore_Preconditions searches more nodes but arrives at the goal faster. This makes sense since more nodes need to be explored when preconditions for an action are ignored. But with a simpler heuristic, it was able to compute faster compare to level sum, which is a more "accurate" but complex heuristic.

- What was the best heuristic used in this problem? Was it better than non-heuristic search planning methods for all problems? Why or why not?
  As explained above, H_Ignore_Preconditions is better than H_PG_Levelsum for this problem since it arrives at the goal faster. However, Greedy Best First Graph Search without heuristic has the shortest execution time to an optimal solution.

Air Cargo Problem 2 -

Search	Expansions	Goal Tests	Time Elapsed
Breadth First Search	3343	4609	12.7710
Depth First Graph Search	624	625	3.2286
Depth Limited Search	222719	2053741	775.303
Uniform Cost Search	4853	4855	41.6690
Greedy Best First Graph Search with H_1	998	1000	6.7765
A* Search with H_1	4853	4855	41.9427
A* Search with H_Ignore_Preconditions	1506	1508	13.3721
A* Search with H_PG_Levelsum	86	88	259.7987

- Optimal plan: (H_Ignore_Preconditions)
  Load(C1, P1, SFO)
  Load(C2, P2, JFK)

Load(C3, P3, ATL)
Fly(P1, SFO, JFK)
Fly(P2, JFK, SFO)
Fly(P3, ATL, SFO)
Unload(C1, P1, JFK)
Unload(C2, P2, SFO)
Unload(C3, P3, SFO)

- Compare and contrast non-heuristic search result metrics:
  Similar to problem 1, though Depth First Graph Search has the shortest execution time, it has a plan length of 619. This is not optimal considering the real-life costs and logistics. Greedy best-first search tries to expand the node that is closest to the goal, on the grounds that this is likely to lead to a solution quickly – this does not guarantee an optimal plan. While Greedy Best First Graph Search was able to find the optimal plan for problem 1, it has a plan length of 15 here. Compare to Breadth First Search, Uniform Cost Search takes longer to arrive at the goal. This is because the step costs are assumed to be identical - see definitions above comparing the two search algorithms. Breadth First Search is the best without heuristic.

- Compare and contrast heuristic search result metrics using A* with the "ignore preconditions" and "level-sum" heuristics:
  Similar to problem 1, H_Ignore_Preconditions performs better in this case.

- What was the best heuristic used in these problems? Was it better than non-heuristic search planning methods for all problems? Why or why not?
  H_Ignore_Preconditions is better than Breadth First here since it allows a more targeted search that "estimates the minimum number of actions that must be carried out from the current state in order to satisfy all of the goal conditions by ignoring the preconditions required for an action to be executed". This resulted in less node expansions and goal tests, though the execution times were similar.

Air Cargo Problem 3 -

Search	Expansions	Goal Tests	Time Elapsed
Breadth First Search	14663	18098	95.4451
Depth First Graph Search	408	409	1.6652
Uniform Cost Search	18223	18225	362.1456
Greedy Best First Graph Search with H_1	5578	5580	97.6707
A* Search with H_1	18223	18225	362.1456
A* Search with H_Ignore_Preconditions	5118	5120	82.2090
A* Search with H_PG_Levelsum	414	416	1804.0072

- Optimal plan: (H_Ignore_Preconditions)

Load(C1, P1, SFO)
Load(C2, P2, JFK)
Fly(P1, SFO, ATL)
Load(C3, P1, ATL)
Fly(P2, JFK, ORD)
Load(C4, P2, ORD)
Fly(P1, ATL, JFK)
Unload(C1, P1, JFK)
Unload(C3, P1, JFK)
Fly(P2, ORD, SFO)
Unload(C2, P2, SFO)
Unload(C4, P2, SFO)

- Compare and contrast non-heuristic search result metrics: Same as Problem 2.
- Compare and contrast heuristic search result metrics using A* with the "ignore preconditions" and "level-sum" heuristics: Same as Problem 2.
- What was the best heuristic used in these problems? Was it better than non-heuristic search planning methods for all problems? Why or why not? Same as Problem 2.