Heuristic Analysis

Part 1 - Planning Problems

➢ **TODO: Experiment and document metrics for non-heuristic planning solution searches**

- Run uninformed planning searches for `air_cargo_p1`, `air_cargo_p2`, and `air_cargo_p3`; provide metrics on number of node expansions required, number of goal tests, time elapsed, and optimality of solution for each search algorithm. Include the result of at least three of these searches, including breadth-first and depth-first, in your write-up (`breadth_first_search` and `depth_first_graph_search`).
- If depth-first takes longer than 10 minutes for Problem 3 on your system, stop the search and provide this information in your report. **DNR** = **Did Not Return** (in the allotted period of time)

NOTE: The results of the above requirements are listed below in summary and the actual runs are listed below for posterity sake.

PROBLEM	ALGORITHM	EXPANSIONS	GOAL TESTS	NEW NODES	PLAN LENGTH	TIME ELAPSED	COMMAND
1	**1**	43	56	180	6	0.0302	python run_search.py -p 1 -s 1
2	**1**	3343	4609	30509	9	12.9477	python run_search.py -p 2 -s 1
3	**1**	14663	18098	129631	12	110.1266	python run_search.py -p 3 -s 1
1	**2**	1458	1459	5960	6	0.9341	python run_search.py -p 1 -s 2
2	**2**	DNR	DNR	DNR	DNR	DNR	python run_search.py -p 2 -s 2
3	**2**	DNR	DNR	DNR	DNR	DNR	python run_search.py -p 3 -s 2
1	**3**	21	22	84	20	0.0155	python run_search.py -p 1 -s 3
2	**3**	624	625	5602	619	3.7341	python run_search.py -p 2 -s 3
3	**3**	408	409	3364	392	1.8528	python run_search.py -p 3 -s 3
1	**4**	101	271	414	50	0.10409	python run_search.py -p 1 -s 4
2	**4**	DNR	DNR	DNR	DNR	DNR	python run_search.py -p 2 -s 4
3	**4**	DNR	DNR	DNR	DNR	DNR	python run_search.py -p 3 -s 4
1	**5**	55	57	224	6	0.3689	python run_search.py -p 1 -s 5
2	**5**	4853	4855	44041	9	13.2434	python run_search.py -p 2 -s 5
3	**5**	18223	18225	159618	12	57.4901	python run_search.py -p 3 -s 5

> **python run_search.py -p 1 -s 1**

```
Solving Air Cargo Problem 1 using breadth_first_search...

Expansions   Goal Tests   New Nodes
   43         56         180

Plan length: 6  Time elapsed in seconds: 0.030201747807790807

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

> **python run_search.py -p 2 -s 1**

```
Solving Air Cargo Problem 2 using breadth_first_search...

Expansions   Goal Tests   New Nodes
   3343       4609       30509
```

Plan length: 9 Time elapsed in seconds: 13.947722039296705

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

> **python run_search.py -p 3 -s 1**

Solving Air Cargo Problem 3 using breadth_first_search...

Expansions	Goal Tests	New Nodes
14663	18098	129631

Plan length: 12 Time elapsed in seconds: 110.12662220169483

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

> **python run_search.py -p 1 -s 2**

Solving Air Cargo Problem 1 using breadth_first_tree_search...

Expansions	Goal Tests	New Nodes
1458	1459	5960

Plan length: 6 Time elapsed in seconds: 0.9341666790824257

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

- **python run_search.py -p 2 -s 2 DNR**
- **python run_search.py -p 3 -s 2 DNR**

> **python run_search.py -p 1 -s 3**

Solving Air Cargo Problem 1 using depth_first_graph_search...

Expansions	Goal Tests	New Nodes
21	22	84

Plan length: 20 Time elapsed in seconds: 0.015549496665127113
Fly(P1, SFO, JFK)
Fly(P2, JFK, SFO)

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

> **python run_search.py -p 2 -s 3**

Solving Air Cargo Problem 2 using depth_first_graph_search...

Expansions	Goal Tests	New Nodes
624	625	5602

Plan length: 619 Time elapsed in seconds: 3.734117347133036
Fly(P3, ATL, SFO)
Fly(P1, SFO, ATL)
Fly(P3, SFO, JFK)
Fly(P1, ATL, JFK)
...

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

> python run_search.py -p 3 -s 3

Solving Air Cargo Problem 3 using depth_first_graph_search...

Expansions	Goal Tests	New Nodes
408	409	3364

Plan length: 392 Time elapsed in seconds: 1.8850886733513113
Fly(P1, SFO, ORD)
Fly(P2, JFK, ORD)
Fly(P1, ORD, ATL)
Fly(P2, ORD, ATL)
Fly(P1, ATL, JFK)
Fly(P2, ATL, SFO)
...
Fly(P2, ORD, ATL)
Fly(P1, ORD, ATL)
Fly(P2, ATL, JFK)
Fly(P1, ATL, JFK)
Unload(C3, P1, JFK)

> **python run_search.py -p 1 -s 4**

Solving Air Cargo Problem 1 using depth_limited_search...

Expansions	Goal Tests	New Nodes
101	271	414

Plan length: 50 Time elapsed in seconds: 0.10408875777847305

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

- **python run_search.py -p 2 -s 4**
- **python run_search.py -p 3 -s 4**

> python run_search.py -p 1 -s 5

Solving Air Cargo Problem 1 using uniform_cost_search...

Expansions Goal Tests New Nodes
55 57 224

Plan length: 6 Time elapsed in seconds: 0.036887385754323584
Load(C1, P1, SFO)
Load(C2, P2, JFK)
Fly(P1, SFO, JFK)
Fly(P2, JFK, SFO)
Unload(C1, P1, JFK)
Unload(C2, P2, SFO)

> python run_search.py -p 2 -s 5

Solving Air Cargo Problem 2 using uniform_cost_search...

Expansions Goal Tests New Nodes
4853 4855 44041

Plan length: 9 Time elapsed in seconds: 13.243421054809781
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(C3, P3, SFO)
Unload(C2, P2, SFO)
Unload(C1, P1, JFK)

> python run_search.py -p 3 -s 5

Solving Air Cargo Problem 3 using uniform_cost_search...

Expansions Goal Tests New Nodes
18223 18225 159618

Plan length: 12 Time elapsed in seconds: 57.490074511108794
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(P2, ORD, SFO)
Fly(P1, ATL, JFK)
Unload(C4, P2, SFO)
Unload(C3, P1, JFK)
Unload(C2, P2, SFO)
Unload(C1, P1, JFK)

Part 2 - Domain-independent heuristics

➢ **TODO: Experiment and document metrics for non-heuristic planning solution searches**

- Run A* planning searches using the heuristics you have implemented on air_cargo_p1, air_cargo_p2 and air_cargo_p3. Provide metrics on number of node expansions required, number of goal tests, time elapsed, and optimality of solution for each search algorithm and include the results in your report.

NOTE: The results of the above requirements are listed below in summary and the actual runs are listed below for posterity sake.

PROBLEM	ALGORITHM	EXPANSIONS	GOAL TESTS	NEW NODES	PLAN LENGTH	TIME ELAPSED	COMMAND
1	**8**	55	57	224	6	0.3899	python run_search.py -p 1 -s 8
2	**8**	4853	4855	44041	9	12.829	python run_search.py -p 2 -s 8
3	**8**	18223	18225	159618	12	58.2203	python run_search.py -p 3 -s 8
1	**9**	41	43	170	6	0.0392	python run_search.py -p 1 -s 9
2	**9**	1450	1452	13303	9	4.5943	python run_search.py -p 2 -s 9
3	**9**	5040	5042	44944	12	18.1442	python run_search.py -p 3 -s 9
1	**10**	11	13	50	6	1.1687	python run_search.py -p 1 -s 10
2	**10**	86	88	841	9	216.4841	python run_search.py -p 2 -s 10
3	**10**	325	327	3002	12	1323.4798	python run_search.py -p 3 -s 10

> python run_search.py -p 1 -s 8

```
Solving Air Cargo Problem 1 using astar_search with h_1...

Expansions   Goal Tests   New Nodes
   55         57         224

Plan length: 6  Time elapsed in seconds: 0.038997402623201574
Load(C1, P1, SFO)
Load(C2, P2, JFK)
Fly(P1, SFO, JFK)
Fly(P2, JFK, SFO)
Unload(C1, P1, JFK)
Unload(C2, P2, SFO)
```

> python run_search.py -p 2 -s 8

```
Solving Air Cargo Problem 2 using astar_search with h_1...

Expansions   Goal Tests   New Nodes
   4853       4855       44041

Plan length: 9  Time elapsed in seconds: 12.829036652976223
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(C3, P3, SFO)
Unload(C2, P2, SFO)
Unload(C1, P1, JFK)
```

> python run_search.py -p 3 -s 8

Solving Air Cargo Problem 3 using astar_search with h_1...

Expansions   Goal Tests   New Nodes
 18223   18225   159618

Plan length: 12 Time elapsed in seconds: 58.2202943031531
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(P2, ORD, SFO)
Fly(P1, ATL, JFK)
Unload(C4, P2, SFO)
Unload(C3, P1, JFK)
Unload(C2, P2, SFO)
Unload(C1, P1, JFK)

> python run_search.py -p 1 -s 9

Solving Air Cargo Problem 1 using astar_search with h_ignore_preconditions...

Expansions   Goal Tests   New Nodes
   41   43   170

Plan length: 6 Time elapsed in seconds: 0.03928743475518606
Load(C1, P1, SFO)
Fly(P1, SFO, JFK)
Unload(C1, P1, JFK)
Load(C2, P2, JFK)
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)

> python run_search.py -p 2 -s 9

Solving Air Cargo Problem 2 using astar_search with h_ignore_preconditions...

Expansions   Goal Tests   New Nodes
  1450   1452   13303

Plan length: 9 Time elapsed in seconds: 4.594375495595716
Load(C3, P3, ATL)
Fly(P3, ATL, SFO)
Unload(C3, P3, SFO)
Load(C2, P2, JFK)
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)
Load(C1, P1, SFO)
Fly(P1, SFO, JFK)
Unload(C1, P1, JFK)

> python run_search.py -p 3 -s 9

Solving Air Cargo Problem 3 using astar_search with h_ignore_preconditions...

Expansions   Goal Tests   New Nodes
  5040   5042   44944

Plan length: 12 Time elapsed in seconds: 18.144223443932916
Load(C2, P2, JFK)
Fly(P2, JFK, ORD)
Load(C4, P2, ORD)
Fly(P2, ORD, SFO)
Unload(C4, P2, SFO)

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

> python run_search.py -p 1 -s 10

Solving Air Cargo Problem 1 using astar_search with h_pg_levelsum...

Expansions	Goal Tests	New Nodes
11	13	50

Plan length: 6 Time elapsed in seconds: 1.1686493806191658
Load(C1, P1, SFO)
Fly(P1, SFO, JFK)
Load(C2, P2, JFK)
Fly(P2, JFK, SFO)
Unload(C1, P1, JFK)
Unload(C2, P2, SFO)

> python run_search.py -p 2 -s 10

Solving Air Cargo Problem 2 using astar_search with h_pg_levelsum...

Expansions	Goal Tests	New Nodes
86	88	841

Plan length: 9 Time elapsed in seconds: 216.48416229218088
Load(C1, P1, SFO)
Fly(P1, SFO, JFK)
Load(C2, P2, JFK)
Fly(P2, JFK, SFO)
Load(C3, P3, ATL)
Fly(P3, ATL, SFO)
Unload(C3, P3, SFO)
Unload(C2, P2, SFO)
Unload(C1, P1, JFK)

> python run_search.py -p 3 -s 10

Solving Air Cargo Problem 3 using astar_search with h_pg_levelsum...

Expansions	Goal Tests	New Nodes
325	327	3002

Plan length: 12 Time elapsed in seconds: 1323.4797856543353
Load(C2, P2, JFK)
Fly(P2, JFK, ORD)
Load(C4, P2, ORD)
Fly(P2, ORD, SFO)
Load(C1, P1, SFO)
Fly(P1, SFO, ATL)
Load(C3, P1, ATL)
Fly(P1, ATL, JFK)
Unload(C4, P2, SFO)
Unload(C3, P1, JFK)
Unload(C2, P2, SFO)
Unload(C1, P1, JFK)

Part 3: Written Analysis

TODO: Include the following in your written analysis.

- Provide an optimal plan for Problems 1, 2, and 3.
- Compare and contrast non-heuristic search result metrics (optimality, time elapsed, number of node expansions) for Problems 1,2, and 3. Include breadth-first, depth-first, and at least one other uninformed non-heuristic search in your comparison; Your third choice of non-heuristic search may be skipped for Problem 3 if it takes longer than 10 minutes to run, but a note in this case should be included.
- Compare and contrast heuristic search result metrics using A* with the "ignore preconditions" and "level-sum" heuristics for Problems 1, 2, and 3.
- 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?
- Provide tables or other visual aids as needed for clarity in your discussion.

Provide an optimal plan for Problems 1, 2, and 3.

- **Definition**: an optimal plan / optimal solution has the lowest path cost among all solutions, which would take into account the plan length, expansions, new nodes, and goal tests.

Problem One

PROBLEM	ALGORITHM	EXPANSIONS	GOAL TESTS	NEW NODES	PLAN LENGTH	TIME ELAPSED	COMMAND
1	**10**	11	13	50	6	1.1687	**python run_search.py -p 1 -s 10**

> python run_search.py -p 1 -s 10

Solving Air Cargo Problem 1 using astar_search with h_pg_levelsum...

Expansions Goal Tests New Nodes
11 13 50

Plan length: 6 Time elapsed in seconds: 1.1686493806191658

1. **Load(C1, P1, SFO)**
2. **Fly(P1, SFO, JFK)**
3. **Load(C2, P2, JFK)**
4. **Fly(P2, JFK, SFO)**
5. **Unload(C1, P1, JFK)**
6. **Unload(C2, P2, SFO)**

Problem Two

PROBLEM	ALGORITHM	EXPANSIONS	GOAL TESTS	NEW NODES	PLAN LENGTH	TIME ELAPSED	COMMAND
2	**10**	86	88	841	9	216.4841	**python run_search.py -p 2 -s 10**

> python run_search.py -p 2 -s 10

Solving Air Cargo Problem 2 using astar_search with h_pg_levelsum...

Expansions Goal Tests New Nodes
86 88 841

Plan length: 9 Time elapsed in seconds: 216.48416229218088

1. **Load(C1, P1, SFO)**
2. **Fly(P1, SFO, JFK)**
3. **Load(C2, P2, JFK)**
4. **Fly(P2, JFK, SFO)**
5. **Load(C3, P3, ATL)**
6. **Fly(P3, ATL, SFO)**
7. **Unload(C3, P3, SFO)**
8. **Unload(C2, P2, SFO)**
9. **Unload(C1, P1, JFK)**

Problem Three

PROBLEM	ALGORITHM	EXPANSIONS	GOAL TESTS	NEW NODES	PLAN LENGTH	TIME ELAPSED	COMMAND
3	**10**	325	327	3002	12	1323.4798	**python run_search.py -p 3 -s 10**

> python run_search.py -p 3 -s 10

Solving Air Cargo Problem 3 using astar_search with h_pg_levelsum...

Expansions Goal Tests New Nodes
325 327 3002

Plan length: 12 Time elapsed in seconds: 1323.4797856543353

1. **Load(C2, P2, JFK)**
2. **Fly(P2, JFK, ORD)**
3. **Load(C4, P2, ORD)**
4. **Fly(P2, ORD, SFO)**
5. **Load(C1, P1, SFO)**
6. **Fly(P1, SFO, ATL)**
7. **Load(C3, P1, ATL)**
8. **Fly(P1, ATL, JFK)**
9. **Unload(C4, P2, SFO)**
10. **Unload(C3, P1, JFK)**
11. **Unload(C2, P2, SFO)**
12. **Unload(C1, P1, JFK)**

Q: Compare and contrast non-heuristic search result metrics (optimality, time elapsed, number of node expansions) for Problems 1,2, and 3.

- **Include breadth-first, depth-first, and at least one other uninformed non-heuristic search in your comparison; Your third choice of non-heuristic search may be skipped for Problem 3 if it takes longer than 10 minutes to run, but a note in this case should be included.**

The Raw Results Data is below:

PROBLEM	ALGORITHM	EXPANSIONS	GOAL TESTS	NEW NODES	PLAN LENGTH	TIME ELAPSED	COMMAND
1	**1**	43	56	180	6	0.0302	python run_search.py -p 1 -s 1
1	**2**	1458	1459	5960	6	0.9341	python run_search.py -p 1 -s 2
1	**3**	21	22	84	20	0.0155	python run_search.py -p 1 -s 3
1	**4**	101	271	414	50	0.10409	python run_search.py -p 1 -s 4
1	**5**	55	57	224	6	0.3689	python run_search.py -p 1 -s 5
2	**1**	3343	4609	30509	9	12.9477	python run_search.py -p 2 -s 1
2	**2**	DNR	DNR	DNR	DNR	DNR	python run_search.py -p 2 -s 2
2	**3**	624	625	5602	619	3.7341	python run_search.py -p 2 -s 3
2	**4**	DNR	DNR	DNR	DNR	DNR	python run_search.py -p 2 -s 4
2	**5**	4853	4855	44041	9	13.2434	python run_search.py -p 2 -s 5
3	**1**	14663	18098	129631	12	110.1266	python run_search.py -p 3 -s 1
3	**2**	DNR	DNR	DNR	DNR	DNR	python run_search.py -p 3 -s 2
3	**3**	408	409	3364	392	1.8528	python run_search.py -p 3 -s 3
3	**4**	DNR	DNR	DNR	DNR	DNR	python run_search.py -p 3 -s 4
3	**5**	18223	18225	159618	12	57.4901	python run_search.py -p 3 -s 5

LEGDENDS

Problems

---

1. Air Cargo Problem 1
2. Air Cargo Problem 2
3. Air Cargo Problem 3

Search Algorithms

---

1. breadth_first_search
2. breadth_first_tree_search
3. depth_first_graph_search
4. depth_limited_search
5. uniform_cost_search

Commentary:

We have to keep in mind that the relative sizes of the problems **seems** to coincide with small (p1), medium (p2) and large (p3), particularly with the elapsed time metric on average. Given that relationship, I have observed the following:

- The **breadth_first_tree_search**, and the **depth_limited_search** both **did not return** on p2 and p3 respectively in the allotted period of time, which tells me they are computationally expensive (slow) algorithms. The supporting evidence for that is that all of p1 did return in the allotted period of time and both **breadth_first_tree_search**, and the

depth_limited_search were **very costly** in terms of result metrics (optimality, time elapsed, number of node expansions) relative to the other metrics.

See below in red:

PROBLEM	ALGORITHM	EXPANSIONS	GOAL TESTS	NEW NODES	PLAN LENGTH	TIME ELAPSED	COMMAND
1	**1**	43	56	180	6	0.0302	python run_search.py -p 1 -s 1
1	**2**	1458	1459	5960	6	0.9341	python run_search.py -p 1 -s 2
1	**3**	21	22	84	20	0.0155	python run_search.py -p 1 -s 3
1	**4**	101	271	414	50	0.10409	python run_search.py -p 1 -s 4
1	**5**	55	57	224	6	0.3689	python run_search.py -p 1 -s 5

Search Algorithms

---

1. breadth_first_search
2. **breadth_first_tree_search**
3. depth_first_graph_search
4. **depth_limited_search**
5. uniform_cost_search

- The **depth_first_graph_search** performed very well in terms of **elapsed time** on all three problems but produced far longer solutions in terms of path length, expansions, new nodes and goal tests for a search strategy which returned in the allotted time period on all three problems.
- The **breadth_first_search** performed well in terms of **path length** as a general observation, but (as expected) it performed poorly on the expansions, new nodes, and goal tests per it's nature as an algorithm.
- The **uniform_cost_search** performed nearly identical the **breadth_first_search**, and given it's similarity as an algorithm, that is not unexpected.

Q: Compare and contrast heuristic search result metrics using A* with the "ignore preconditions" and "level-sum" heuristics for Problems 1, 2, and 3.

The Raw Results Data is below:

PROBLEM	ALGORITHM	EXPANSIONS	GOAL TESTS	NEW NODES	PLAN LENGTH	TIME ELAPSED	COMMAND
1	**9**	41	43	170	6	0.0392	python run_search.py -p 1 -s 9
2	**9**	1450	1452	13303	9	4.5943	python run_search.py -p 2 -s 9
3	**9**	5040	5042	44944	12	18.1442	python run_search.py -p 3 -s 9
1	**10**	11	13	50	6	1.1687	python run_search.py -p 1 -s 10
2	**10**	86	88	841	9	216.4841	python run_search.py -p 2 -s 10
3	**10**	325	327	3002	12	1323.4798	python run_search.py -p 3 -s 10

Search Algorithms

9. astar_search h_ignore_preconditions
10. astar_search h_pg_levelsum

Commentary:

- As the **A*** algorithms operated on the 3 problems sets, it is clear that the **h_ignore_preconditions** heuristic performed far better in terms of **elapsed time**.
- The two heuristics performed **identically** in terms of their **path length** on all three problem sets.
- The big difference is in the expansions, new nodes and goal tests where **h_pg_levelsum** performed orders of magnitudes better than **h_ignore_preconditions**heuristic.

Q: 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?

Commentary:

The **best** heuristic used to solve these three problems (p1,p2 and p3) **and overall** was (10) **astar_search h_pg_levelsum**, which incidentally performed better than the non-heuristic planning methods. The reason the heuristic approaches were better than the uninformed searches was that because of their inherent behavior as an algorithm, they were far more efficient at traversing the search space (particularly because they were directed by the capabilities of the heuristic) than the uninformed strategies.

The results are below.

Problem One

PROBLEM	ALGORITHM	EXPANSIONS	GOAL TESTS	NEW NODES	PLAN LENGTH	TIME ELAPSED	COMMAND
1	**10**	11	13	50	6	1.1687	**python run_search.py -p 1 -s 10**

Problem Two

PROBLEM	ALGORITHM	EXPANSIONS	GOAL TESTS	NEW NODES	PLAN LENGTH	TIME ELAPSED	COMMAND
2	**10**	86	88	841	9	216.4841	**python run_search.py -p 2 -s 10**

Problem Three

PROBLEM	ALGORITHM	EXPANSIONS	GOAL TESTS	NEW NODES	PLAN LENGTH	TIME ELAPSED	COMMAND
3	**10**	325	327	3002	12	1323.4798	**python run_search.py -p 3 -s 10**

Note:

Search Algorithms

---

1. breadth_first_search
2. breadth_first_tree_search
3. depth_first_graph_search
4. depth_limited_search
5. uniform_cost_search
6. recursive_best_first_search h_1
7. greedy_best_first_graph_search h_1
8. astar_search h_1
9. astar_search h_ignore_preconditions
10. **astar_search h_pg_levelsum**