Planning Problem Representation

One of the optimal sequences for each problem is below. Some parts might be different a little among each algorithm. (For example, the Load orders of C1 and C2 might be different in Problem 1.) This sample was derived from breadth_first_search.

Problem1

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

Problem2

- Load(C2, P2, JFK)
- Load(C1, P1, SFO)
- 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)

Problem3

- Load(C2, P2, JFK)
- Load(C1, P1, SFO)
- 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)

Performance Comparison

non-heuristic search

|---------------| ------------- | ------------ | ------------- | ----------------- | | Problem | Algorithm | Path Length | Expansions | Elapsed Time (sec) | |---------------| ------------- | ------------ | ------------- | ----------------- | | Problem1 | breadth-first | 6 | 43 | 0.150 | | Problem1 | depth-first | 12 | 12 | 0.037 | | Problem1 | uniform-cost | 6 | 55 | 0.182 | | ------------- | ------------- | ------------ | ------------- | ----------------- | | Problem2 | breadth-first | 9 | 3343 | 69 | | Problem2 | depth-first | 582 | 582 | 11.4 | | Problem2 | uniform-cost | 9 | 4852 | 123.6 | | ------------- | ------------- | ------------ | ------------- | ----------------- | | Problem3 | breadth-first | 12 | 14663 | 475 | | Problem3 | depth-first | 596 | 627 | 15.1 | | Problem3 | uniform-cost | 12 | 18235 | 1255 | | ------------- | ------------- | ------------ | ------------- | ----------------- |

- abount problems
    - the number of problem1, problem2, and problem3 initial states are 12, 27 and 32. The larger the number of state is, The large the number of expansions is. As AIND 10.2.1 says, the total cost for naieve algorithm estimates (average possible actions)**(path length). So, the cost order tends to be exponential for the number of cargo in goal states.
- breadth-first search(BFS)
    - BFS could find the optimal path for each problem. This algorithm can becomes a benchmark to compare to other algorithms because of the simple strategy.
- depth-first search(DFS)
    - DFS could not find optimal path. This is because this search algorithm finished when goal is found, regardless of the goal is optimal. And therefore, this algorithm is the fastes(minimum elapsed time) among I choose.
- uniform-cost search(UCS)
    - UCS could find the optimal path for each problem as lesson videos said. This algorithm's expansions and elaplsed time were larger than breadth-first. However, the path cost of the UCS is same as the number of steps(uniform_cost_search's path_cost uses Problem.path_cost). So, essentially, UCS cost is same order as BFS cost in this implements.

heuristic search

|---------------| -------------------- | ------------ | ------------- | ---------- | | Problem | Algorithm | Path Length | Expansions | Elapsed Time (sec) | |---------------| -------------------- | ------------ | ------------- | ---------- | | Problem1 | ignore preconditions | 6 | 41 | 0.188 | | Problem1 | level-sum | 6 | 11 | 2.246 | |---------------| -------------------- | ------------ | ------------- | ---------- | | Problem2 | ignore preconditions | 9 | 1506 | 49.46 | | Problem2 | level-sum | 9 | 86 | 301 | |---------------| -------------------- | ------------ | ------------- | ---------- | | Problem3 | ignore preconditions | 12 | 5118 | 421.81 | | Problem3 | level-sum | 12 | 404 | 2692 | |---------------| -------------------- | ------------ | ------------- | ---------- |

- ignore preconditions
  - This algorithm is the minimum elapsed time in my environment and expansions is lower than BFS. This algorithm is similar to A* search(estimate minimum distance ignoring wall). The decreased expansions designates the efficient path cut-off.
- level-sum
  - This algorithm is the minimum expansions. However the elapsed time is larger than BFS and ignore preconditions. I think this is because look-ahead search in heuristic function is higher cost. In other words, expansion is done in heuristic function. According to AIND 10.3.1, this algorithm is effective for largely decomposable problem. So this algorithm seems not be suitable for this problem (this problem's actions are largely constrained by each cargo and plain can be exists same places, not so decomposable).