results

problem1

air cargo problems	search functions	Expansions	Goal Tests	New Nodes	Plan length	Time elapse in secon
1	BFS	43	56	180	6	0.04
1	DFS	21	22	84	20	0.02
1	Depth Limited	101	271	414	50	0.11
1	A*	55	57	224	6	0.05
1	A* ignore precond.	41	43	170	6	0.05
1	A* levelsum	11	13	50	6	1.04

Although it is a slight difference, the best algorithm for this problem is BFS. Although it is slightly different, A * and A * ignore precond also have good results. Although DFS finishes early, it has not been able to find an optimal solution. A * levelsum has been able to find an optimal solution, but it has a long time.

problem2

air cargo problems	search functions	Expansions	Goal Tests	New Nodes	Plan length	Time elapse in secon
2	BFS	3343	4609	30509	9	14.86
2	DFS	624	625	5602	619	3.50

2	Depth Limited	-	-	-	-	TIMEO
2	A*	4853	4855	44041	9	13.89
2	A* ignore precond.	1450	1452	13303	9	4.99
2	A* levelsum	86	88	841	9	185.21

Unlike problem 1, A * ignore precond. was the earliest and it was the algorithm that found the optimal solution. A * levelsum takes quite a while, but I will pay attention that the search functions, Expansions and Goal Tests are quite small.Also, in this issue Depth Limited has timed out.

problem3

air cargo problems	search functions	Expansions	Goal Tests	New Nodes	Plan length	Time elaps in secor
3	BFS	14663	18098	129631	12	107.3
3	DFS	408	409	3364	392	1.95
3	Depth Limited	-	-	-	-	TIMEC
3	A*	18234	18236	149707	12	62.13
3	A* ignore precond.	5040	5042	44944	12	19.81
3	A* levelsum	-	-	-	-	TIMEC

Also this time A * ignore precond. Was able to find the optimum solution quickly. Although DFS is fast, it has not been able to find an optimal solution. In this issue, in addition to Depth Limited, A * levelsum also timed out.

Comparison of non-heuristic searches

I compared BFS, DFS, Depth Limited Search.

> https://en.wikipedia.org/wiki/Breadth-first_search

BFS searches for the next depth after searching for the same depth.
As a result, more search will be required, and it will take some time to find a solution.
Such a trend was seen also this time.

> https://en.wikipedia.org/wiki/Depth-first_search

DFS is searching for depth with priority.
For this reason, the number of searches is small in the problem like this time, and we arrived at the solution quickly.
However, the solution found first from the complexity of the problem is rarely the optimal solution. As a result, I have not found an optimal solution.

> https://classroom.udacity.com/nanodegrees/nd889/parts/6be67fd1-9725-4d14-b36e-ae2b5b20804c/modules/f719d723-7ee0-472c-80c1-663f02de94f3/lessons/9b1a742a-fa2d-4940-922c-ed426b44f81b/concepts/53956591910923#

Depth Limited search is valid if you know the effective depth of the problem.
In this case, it seems that a good result was not born because the effective depth was not known beforehand.

Comparison of heuristic searches

Heuristic searches, in particular A * ignore precond., Seems to work well for this problem. A * ignore precond. Will find a solution soon. but potentially at the cost of not being executable.

Conclusion

In conclusion, I think that it is better to use A * ignore precond. In this problem.
As you can see in the results, I think that we can find an optimal solution and we will be satisfied with speed.