Sub title

Level	Evaluation	Time	Memory used	Solution length	Nodes explored
SAD1	A*	0.01 s	5.12 MB	19	35
SAD1	WA*	0.01 s	5.12 MB	19	19
SAD1	Greed	0.01 s	5.12 MB	19	19
SAD2	A*	0.01 s	5.12 MB	19	19
SAD2	WA*	0.01 s	5.12 MB	19	19
SAD2	Greedy	0.01 s	5.12 MB	19	19
custom	A*	0.01 s	5.12 MB	19	19
custom	WA*	0.01 s	5.12 MB	19	19
custom	Greedy	0.01 s	5.12 MB	19	19
FireFly	A*	DNF	DNF	DNF	DNF
FireFly	WA*	DNF	DNF	DNF	DNF
FireFly	Greedy	DNF	DNF	DNF	DNF
Crunch	A*	DNF	DNF	DNF	DNF
Crunch	WA*	DNF	DNF	DNF	DNF
Crunch	Greedy	DNF	DNF	DNF	DNF

a) “Write a best-first search client by implementing the StrategyBestFirst class. The Heuristic argument in the constructor must be used to order nodes. As it implements the Comparator<Node> interface it integrates well with the Java Collections library. Make sure you use an appropriate data structure for the frontier, and state which one you have used.”

The frontier is a List. When a node (the one with the lowest f(n)) is poped from the frontier, the frontier is cleared in order to make a new frontier based on the chosen node. The remaining nodes in the frontier, before clearing it, is stored as an old frontier - in order to go back to that, if the new one proves to be a dead end.

Actually I use two frontiers.

b) Heuristic function The heuristic functions functions calculates the distance from the agent to the box and from the box to the goal. The goal and box coordinates are found in the pre-processing.

Issue when multiple of same box or goal - because it is stored in a map with character as key, where keys are unique.

Benchmark:

c) Admissibility:

The heuristic function is admissible, as it calculates the Pythagorean distance between agent, box and goal without regarding any possible obstacles.

d) Competition levels: