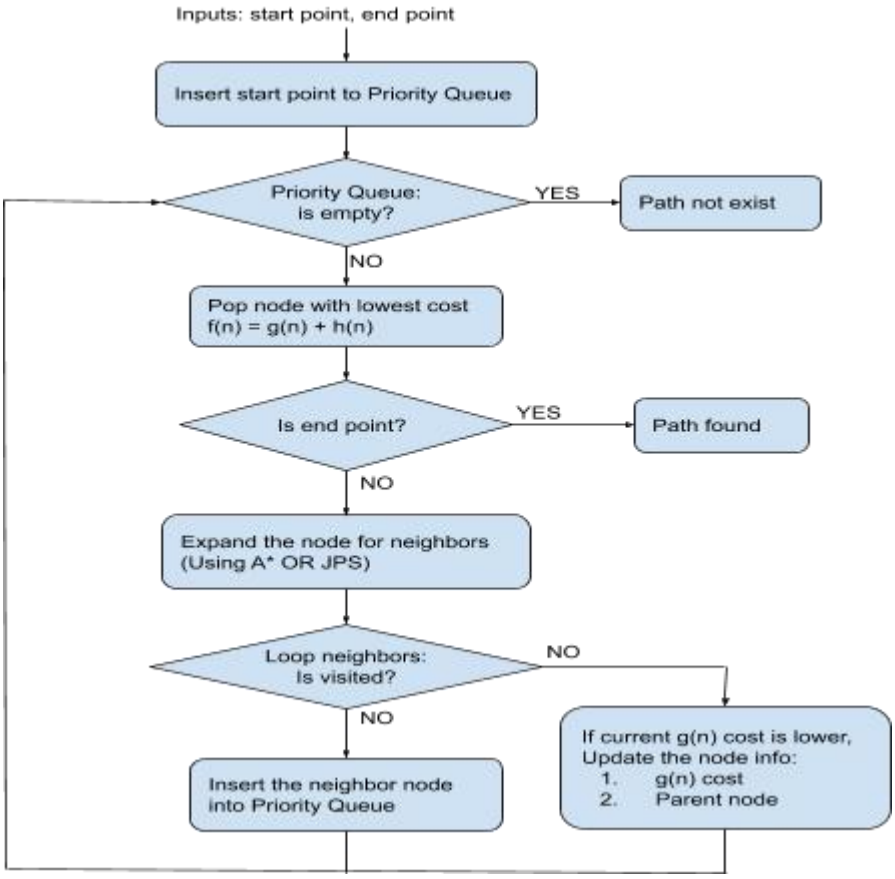
Search-Based Path Finding

Description: experiment results with two different algorithms A* search and JPS are analysed in this report. Also, results infected by other elements (tie break, heuristic function) are compared.

Algorithm Workflow



First study is in A* Searching, to exquisite the effluence of a variety of Heuristic function and tie breaker, 4 study cases that have same end point and in the same obstacles map created.

A* Searching	Path Cost (m)				Visited Node Numbers			
	Case 1	Case 2	Case 3	Case 4	Case 1	Case 2	Case 3	Case 4
0 (Dijkstra)	7.340377	7.774691	7.774691	9.858214	57224	58032	58033	58237
Manhattan Distance	7.593967	7.774691	8.009006	10.092529	30	49	33	5089
Manhattan Distance via Tie Breaker	7.476809	7.774691	8.009006	10.219664	28	45	32	4874
Euclidean Distance	7.340377	7.774691	7.774691	9.858214	1044	2475	1948	12629
Euclidean Distance via Tie Breaker	7.340377	7.774691	7.774691	9.858214	828	2257	1782	12362
Diagonal Distance	7.340377	7.774691	7.774691	9.858214	237	1261	958	10124
Diagonal Distance via Tie Breaker	7.340377	7.774691	7.774691	9.858214	91	1072	828	10001

Analysis:
We know Dijkstra searching is always has the optimal path cost in the end. By comparing the path cost from different searching methods with Dijkstra, the path cost of the heuristic function by Manhattan distance is longer, even with tie breaker or not. So Manhattan distance is not admissible. Euclidean distance and Diagonal distance are admissible in test cases.
On the other hand, it is quite intuitive Dijkstra searching has the greatest number of visited nodes. Even though the strategy utilizing the Manhattan distance may not the minimum path cost, it visited the minimum number of nodes. Than is the Diagonal distance, which is tight and better than Eulidean distance.
The effect by tie break, in sometimes (in the Case1 by Diagonal distance), can be obvious. Theoretically, tie break may affect the admissibility of the heuristic function. This is not happened in the experiment.

Second study is focused on JPS, which is compared by A* searching. Both of them are implemented with the same heuristical function, diagonal distance with tie breaker. 10 cases with same end point and obstacle map were examinated.

	A* Searching			JPS (Jump Point Searching)		
	Time (ms)	Path Cost (m)	Visited Node Size	Time (ms)	Path Cost (m)	Visited Node Size
Case 1	0.176366	5.429107	24	0.334539	5.429107	20
Case 2	0.188567	3.414894	13	0.154184	3.414894	11
Case 3	0.195953	4.672719	19	0.172513	4.672719	24
Case 4	0.274263	4.476955	24	0.137092	4.476955	17
Case 5	0.200535	3.701972	13	0.218759	3.701972	15
Case 6	0.192085	3.107034	13	0.09715	3.107034	7
Case 7	0.239933	6.029107	24	0.160971	6.029107	14
Case 8	0.181725	5.31195	20	0.153721	5.31195	16
Case 9	0.222723	4.277781	26	0.100061	4.277781	18
Case 10	0.187293	3.746264	13	0.199646	3.746264	11

Analysis:
From the comparision, both A* searching and JPS had the same path cost result in all test cases. In the most complicated occasions, JPS has a better performance. The visited node numbers is less significantly, while for the cases, if there is a large free space behind the expected path and the straight path is blocked, the JPS is getting worse.