ME5413 - AUTONOMOUS MOBILE ROBOTICS

NATIONAL UNIVERSITY OF SINGAPORE

COLLEGE OF DESIGN AND ENGINEERING

Homework 3: Planning

Authors:
Tang Ziyue (ID: A0304036U)

Date: April 3, 2025

1 Task 1: Global Planning

1.1 Map Preprocessing

Firstly, load the original floor plan and its grayscale freespace map, where each 0.2m $\times$ 0.2m grid cell is either free (255) or occupied (0). Then, apply a circular kernel (Figure 2) to inflate obstacles, considering the human's 0.3m radius. The inflated map result is shown in Figure 3. An 8-connected neighbor system (Figure 1) is used, with step costs of 0.2m (straight) and 0.282m (diagonal), forming the basis for A* and other planners.

1.2 Heuristic Function Comparison for A*

`A*` is a graph-based search algorithm that finds the shortest path by minimizing a cost function `f(n)=g(n)+h(n)`, where `g(n)` is the actual cost from the start to node `n`, and `h(n)` is a heuristic estimate of the cost from `n` to the goal. By balancing exploration and goal-directed search, `A*` guarantees optimality when the heuristic is admissible.

1.2.1 Task 1.1: A* with Euclidean distance

The A* algorithm with a Euclidean-distance-based heuristic generates geometrically direct paths consistently, which ensures optimal and efficient search behavior. This is particularly evident in Figure 4, where the planned trajectories remain relatively straight when passing through unobstructed regions.

Table 1: Distance table for A* with Euclidean distance (unit: meters)

From\To	start	snacks	store	movie	food
start	0.00	143.28	155.26	178.91	224.70
snacks	143.28	0.00	115.21	108.11	134.31
store	155.26	115.21	0.00	210.02	111.12
movie	178.91	108.11	210.02	0.00	113.71
food	224.70	134.31	111.12	113.71	0.00

1.2.2 Task 1.2: A* with Manhattan distance

Compared to the Euclidean heuristic, the Manhattan-distance-based heuristic estimates cost by summing the absolute horizontal and vertical differences, without considering diagonal movements. Although it remains admissible and consistent, it tends to favor axis-aligned paths, causing trajectories to follow grid lines. This behavior is evident in Figure 5 '4: from start to food', '5: from snacks to store', and '10: from movie to food', where the planned routes exhibit more angular turns, particularly in open areas. These three cases also correspond to the bolded entries in Table 2, where the path lengths are notably longer than those plans using the Euclidean heuristic.

Table 2: Distance table for A* with Manhattan distance (unit: meters)

From\To	start	snacks	store	movie	food
start	0.00	143.28	155.26	178.91	**224.81**
snacks	143.28	0.00	**119.26**	108.11	134.31
store	155.26	**119.26**	0.00	210.02	111.12
movie	178.91	108.11	210.02	0.00	**114.30**
food	**224.81**	134.31	111.12	**114.30**	0.00

1.3 Alternative Algorithms

1.3.1 Greedy Best First Search (GBFS)

`Greedy Best First Search` (GBFS) is a heuristic-driven search algorithm that selects nodes based solely on the estimated cost to the goal, `h(n)`, ignoring the actual cost from the start, `g(n)`. Unlike A*, which balances exploration and optimality, GBFS prioritizes speed but may yield suboptimal or incomplete paths, as evidenced by Table 3 and Table **??**.

Table 3: Distance table using Greedy Best First Search (unit: meters)

From\To	start	snacks	store	movie	food
start	0.00	146.29	165.60	182.48	240.13
snacks	146.29	0.00	124.00	144.19	171.65
store	165.60	124.00	0.00	257.13	119.03
movie	182.48	144.19	257.13	0.00	208.22
food	240.13	171.65	119.03	208.22	0.00

1.3.2 Rapidly-exploring Random Tree (RRT) and Rapidly-exploring Random Tree Star (RRT*)

Rapidly-exploring Random Tree (RRT) is a sampling-based planning algorithm that incrementally builds a tree by randomly sampling the configuration space and connecting each new sample to the nearest existing node. Unlike A*, RRT does not rely on a cost function, and instead focuses on exploration through random expansion, making it well-suited for high-dimensional or complex spaces.

RRT* is an asymptotically optimal variant of RRT that incorporates path cost `g(n)` to improve the tree by rewiring nodes for lower-cost connections iteratively. Therefore, RRT* can gradually converge to the optimal path and make the path smoother.

However, RRT often performs poorly in environments with narrow scenarios because randomly sampled points are likely to fall within obstacles, leading to frequent rejections and sparse exploration in constrained areas. RRT*, by considering path cost and rewiring locally, can better utilize feasible samples and gradually refine paths within such narrow regions. But even so, neither of them has succeeded in providing complete path planning in Vivo scenario, as given in Figure 7, Figure 8, Table 4, and Table 5.

Table 4: Distance table using RRT (unit: meters)

From\To	start	snacks	store	movie	food
start	0.00	inf	inf	191.99	288.52
snacks	inf	0.00	127.91	inf	inf
store	inf	127.91	0.00	231.86	142.71
movie	191.99	inf	231.86	0.00	125.42
food	288.52	inf	142.71	125.42	0.00

Table 5: Distance table using RRT* (unit: meters)

From\To	start	snacks	store	movie	food
start	0.00	inf	inf	173.76	inf
snacks	inf	0.00	113.36	123.59	inf
store	inf	113.36	0.00	242.20	inf
movie	173.76	123.59	242.20	0.00	121.29
food	inf	inf	inf	121.29	0.00

1.4 Summary and Discussion

This task compared several global planning algorithms based on trajectory quality and completeness. A* with Euclidean distance yielded the most direct and shortest paths overall, while the Manhattan heuristic introduced axis-aligned bias, resulting in longer paths in some cases. GBFS, relying solely on `h(n)`, produced suboptimal results with increased distances. Sampling-based methods such as RRT and RRT* struggled in narrow environments due to random sampling falling into obstacles, often failing to generate complete paths. RRT* slightly improves path smoothness. However, both methods are highly random, and the results are quite unstable. A conceptual summary of all methods is given in Table 6.
In conclusion, A* remains the most effective approach in structured indoor settings, while sampling-based methods require further refinement for constrained environments.

Table 6: Comparison of Path Planning Algorithms

Algorithm	Uses $g(n)$?	Uses $h(n)$?	Core Concept
Dijkstra	Yes	No	Finds the shortest path (uninformed)
A*	Yes	Yes	Shortest path with heuristic acceleration
GBFS	No	Yes	Focuses solely on proximity to goal
RRT	No	No	Rapidly expands to explore the space
RRT*	Yes	No	Incrementally optimizes path via tree structure

2 Task 2: The "Travelling Shopper" Problem (TSP)

2.1 Brute-force Search

A brute-force solution iterates through all permutations of the four intermediate locations (snacks, store, movie, food), evaluating every possible route. Although computationally expensive, this method guarantees the optimal route due to exhaustive search. The best path

and its total length are visualized, along with step-by-step metrics such as grid visits, segment distances, and runtime.

2.2 Held-Karp Algorithm

To improve efficiency, the Held-Karp algorithm is used as a dynamic programming solution for TSP. It encodes visited nodes using bitmasks and recursively computes the shortest subpaths. While still exact, it reduces computation from factorial to exponential time complexity. The final optimal path and metrics are presented similarly to the brute-force method.

2.3 Comparison and Discussion

Both the brute-force and Held-Karp algorithms consistently produced the same optimal visiting order:
`start → snacks → movie → food → store → start`.
This route remained unchanged regardless of whether the distance matrix was derived from A* (Euclidean or Manhattan) or GBFS, confirming the correctness and robustness of the two TSP solvers.
To further analyze performance, the selected route was evaluated under three different planners. As shown in the following three performance tables, there is little difference, 0.59 meters, between A* with Euclidean distance and A* with Manhattan distance, with the former being better. In contrast, GBFS has significantly longer routes, which is due to the fact that it only uses a heuristic search strategy and often ignores the global optimal path.
In terms of computational cost, A* (ED) required more visited nodes per segment, reflecting its thorough search and higher path optimality. Manhattan-based A* visited fewer nodes in some segments but took longer runtime in certain cases due to the heuristic's directional bias. GBFS achieved the fastest runtimes with minimal node expansion, but this came at the expense of overall path quality.
These results illustrate the trade-offs between optimality, efficiency, and search completeness across planning methods. A* remains the most reliable for structured environments, while GBFS may be suitable when speed is prioritized over path quality.

Table 7: Performance of A* using Euclidean Distance (ED)

From → To	Visited Grids	Path Length (m)	Run Time (s)
start → snacks	34394	143.28	0.4524
snacks → movie	11010	108.11	0.1153
movie → food	25765	113.71	0.3951
food → store	16645	111.12	0.1693
store → start	27953	155.26	0.4093
		Total Travel Length: 631.48 meters	

Table 8: Performance of A* using Manhattan Distance (MD)

From → To	Visited Grids	Path Length (m)	Run Time (s)
start → snacks	30026	143.28	0.8563
snacks → movie	8414	108.11	0.4543
movie → food	1361	114.30	0.0182
food → store	6290	111.12	0.6461
store → start	2260	155.26	0.0343
		Total Travel Length: 632.07 meters	

Table 9: Performance of Greedy Best First Search (GBFS)

From → To	Visited Grids	Path Length (m)	Run Time (s)
start → movie	2952	182.48	0.0183
movie → food	4150	208.22	0.0285
food → store	1459	119.03	0.0065
store → snacks	1461	124.00	0.0066
snacks → start	4746	146.29	0.0407
		Total Travel Length: 780.01 meters	

3 Task 3: GitHub

Link: git@github.com:YukiKuma111/NUS_ME5413_Homework3_Planning.git
Code style: PEP 8

A Appendix

A.1 Task 1

Figure 1 shows the 8-connected neighbor system with direction and cost labels.

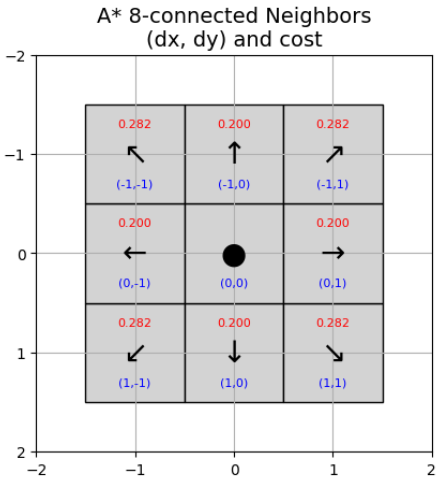

Figure 1: 8-connected neighbor system.

Figure 2 shows the elliptical kernel used for dilation.

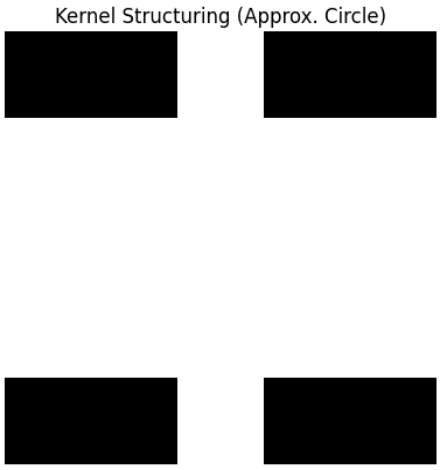

Figure 2: Kernel for dilation.

The resulting inflated map is shown in Figure 3, where obstacles expand to account for the human's footprint.

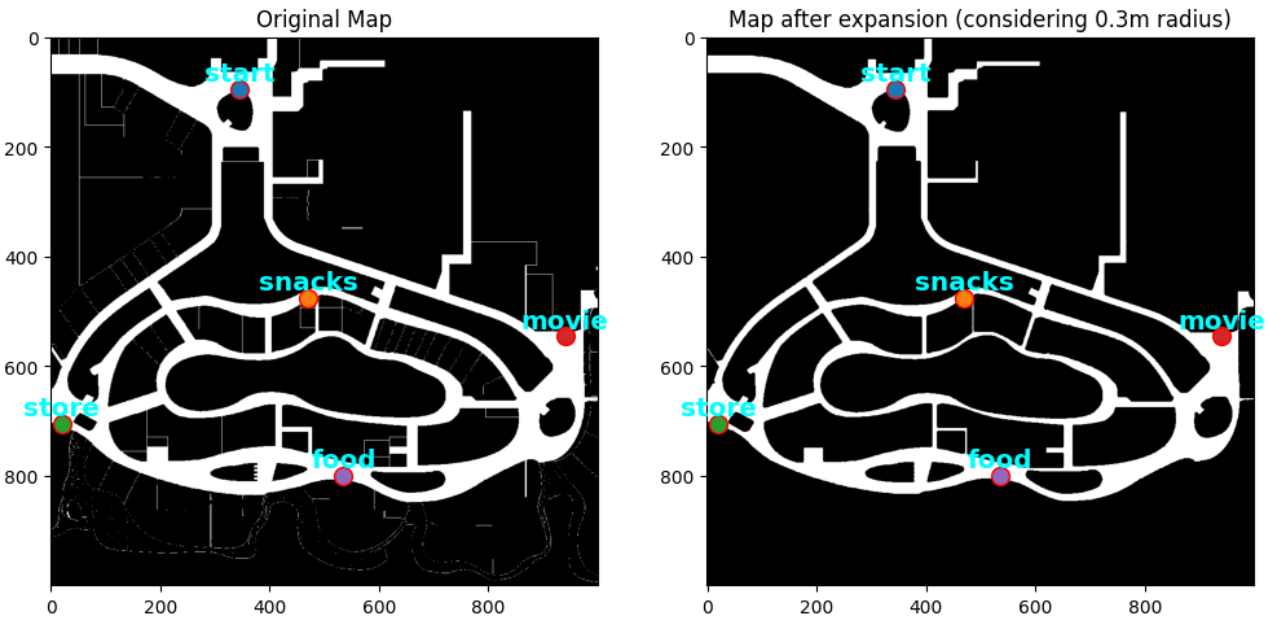

Figure 3: Comparison of original and inflated map.

A.2 Task 2

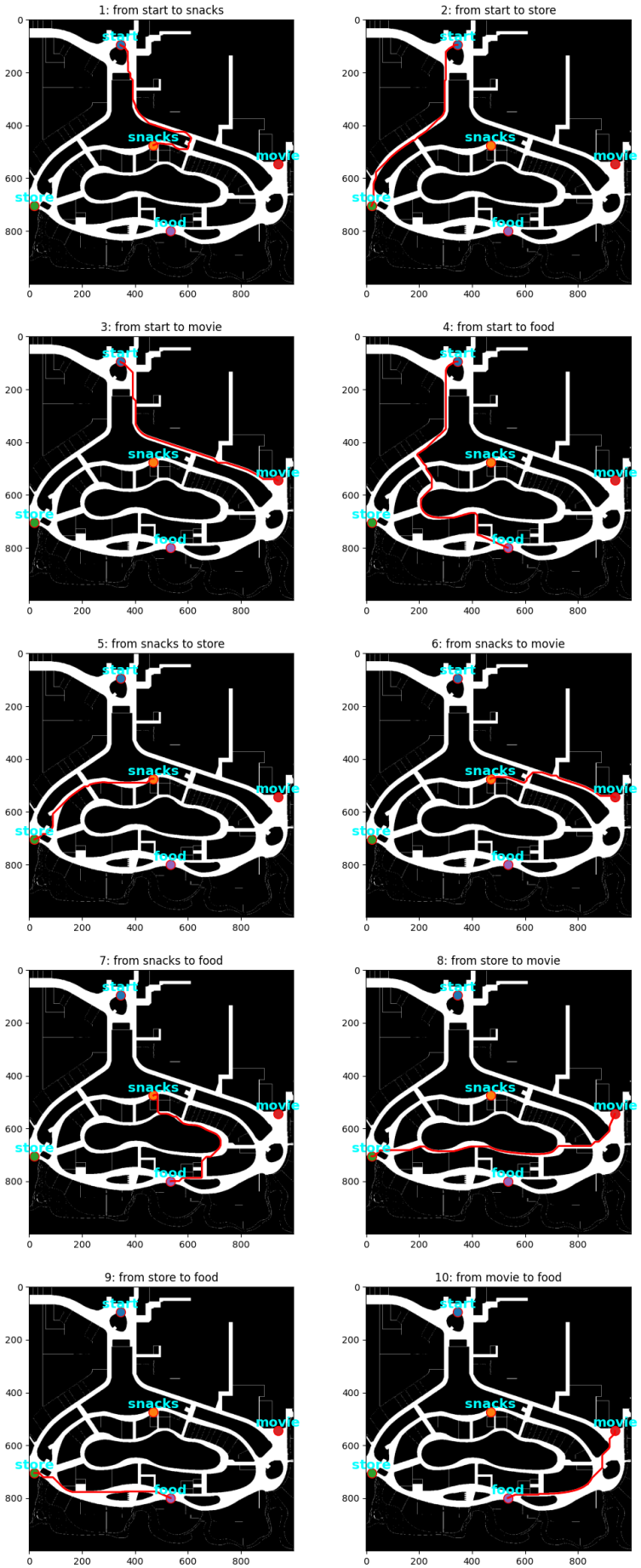

Figure 4: A* with euclidean distance.

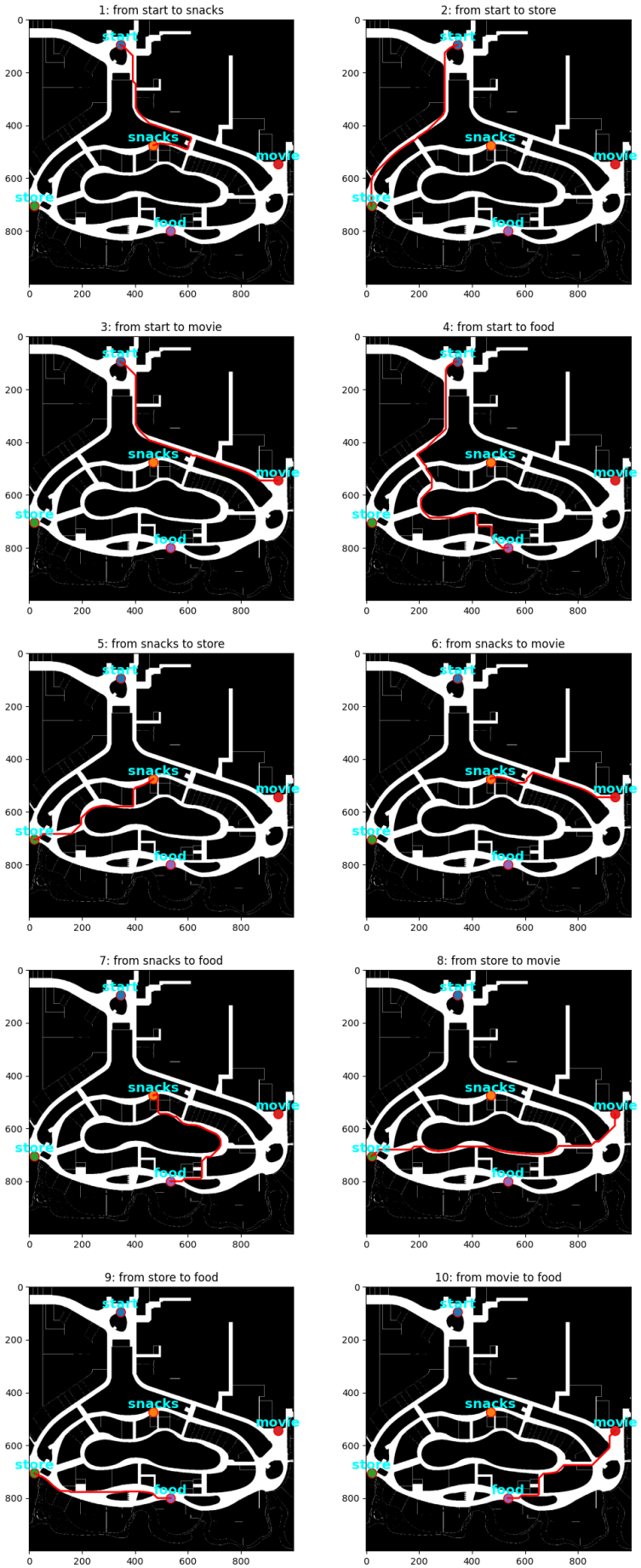

Figure 5: A* with manhattan distance.

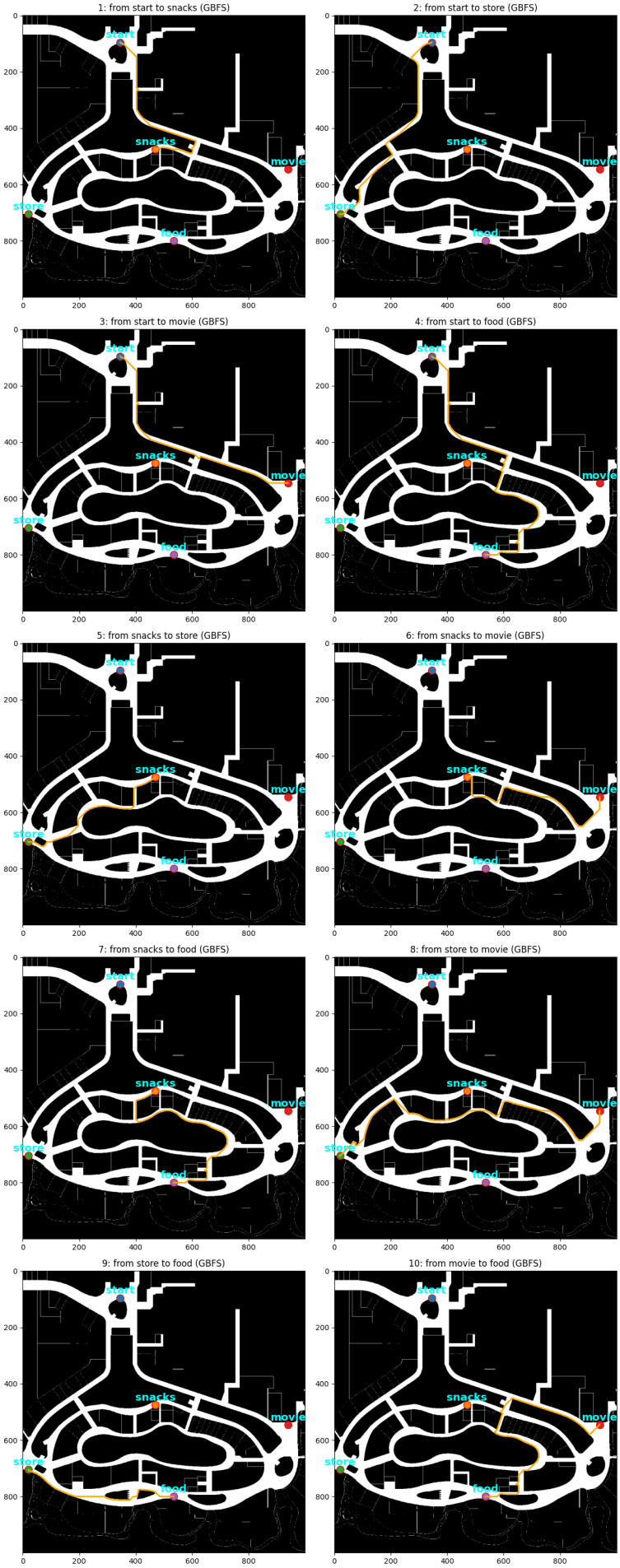

Figure 6: Greedy Best First Search.

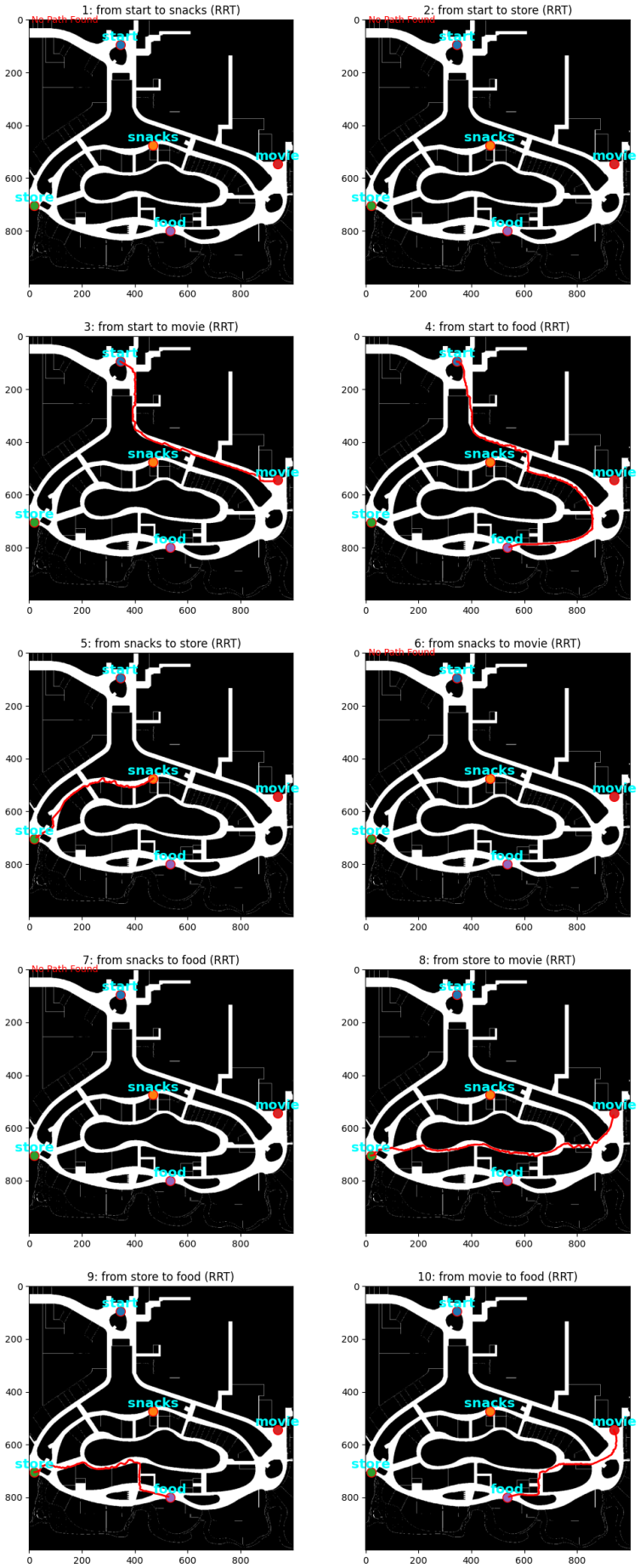
1: from start to snacks (RRT)
No Path Found
2: from start to store (RRT)
No Path Found
3: from start to movie (RRT)
4: from start to food (RRT)
5: from snacks to store (RRT)
6: from snacks to movie (RRT)
No Path Found
7: from snacks to food (RRT)
No Path Found
8: from store to movie (RRT)
9: from store to food (RRT)
10: from movie to food (RRT)
start
snacks
movie
store
food
0
200
400
600
800

Figure 7: RRT.

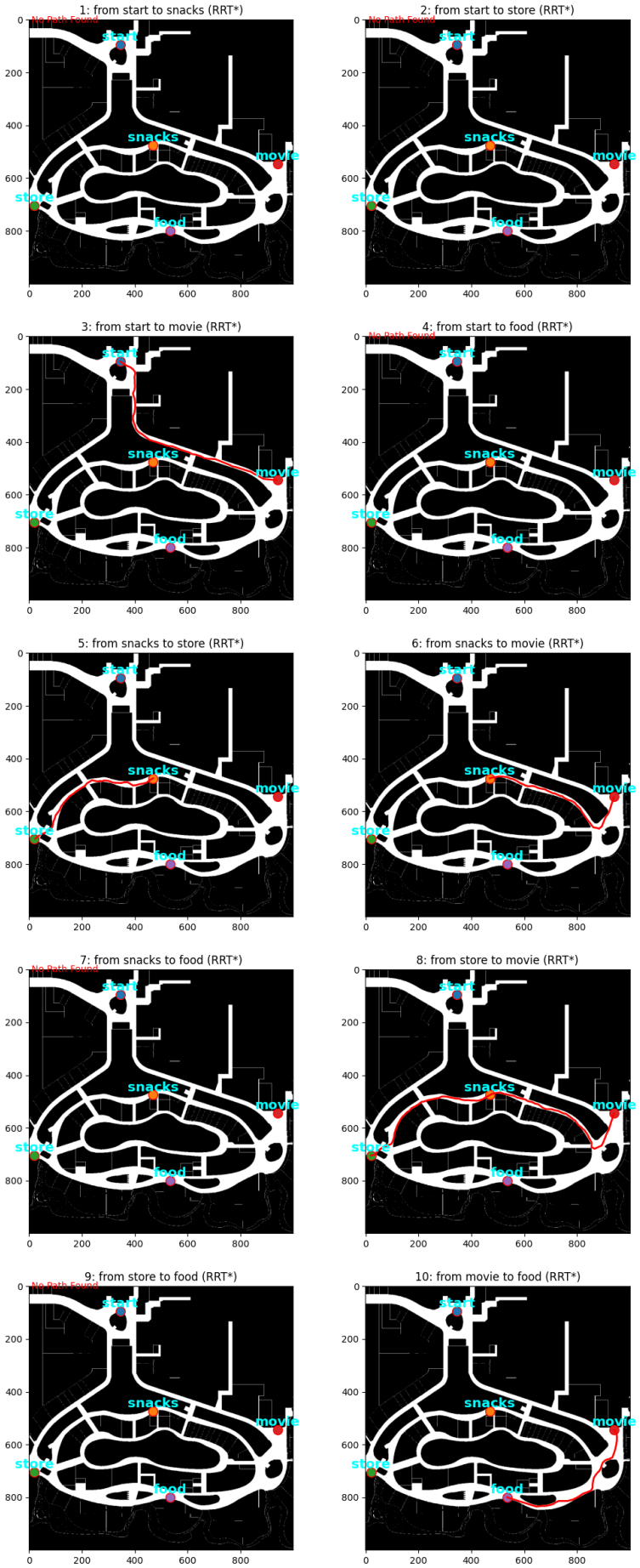
1: from start to snacks (RRT*)
2: from start to store (RRT*)
3: from start to movie (RRT*)
4: from start to food (RRT*)
5: from snacks to store (RRT*)
6: from snacks to movie (RRT*)
7: from snacks to food (RRT*)
8: from store to movie (RRT*)
9: from store to food (RRT*)
10: from movie to food (RRT*)
No Path Found
start
snacks
movie
store
food
0
200
400
600
800

Figure 8: RRT*.

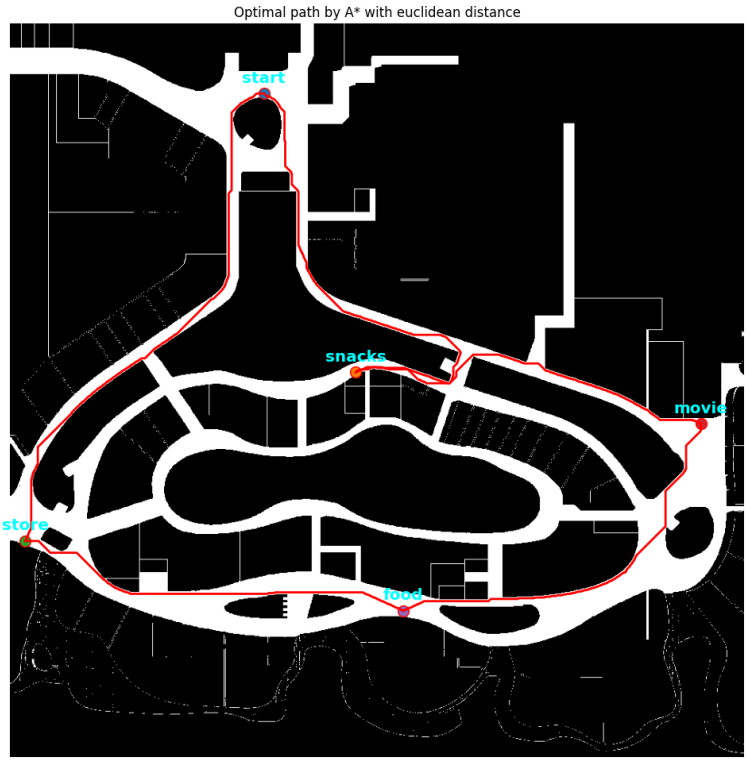

Figure 9: Optimal path by A* with euclidean distance.

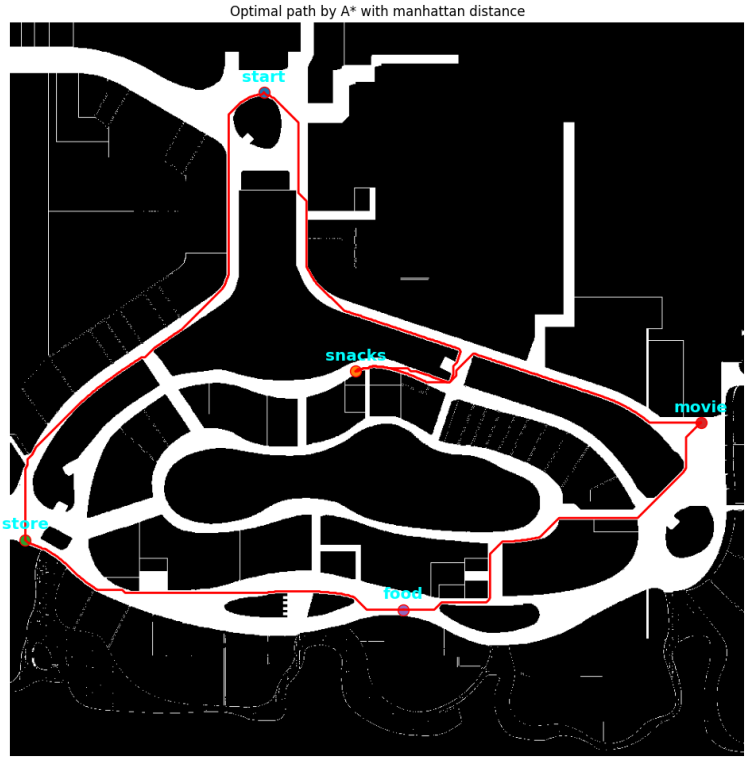

Figure 10: Optimal path by A* with manhattan distance.

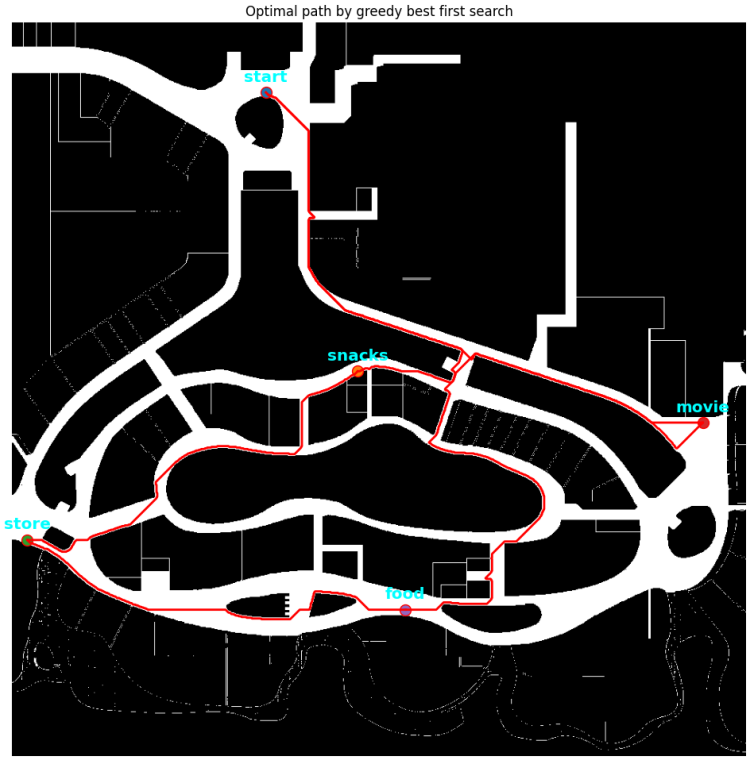

Figure 11: Optimal path by greedy best first search.