Artificial Intelligence

Program – Search in Pacman

EECS 26' You-Ting Liu 111060013

Q4: A* Search

This algorithm is designed based on BFS algorithm framework. The slight differences between them are the data structure of **process** and **discovered_states**. **process** here is not a pure first-in-first-out queue but a lowest-first priority queue sorted by the **heuristic value** of each element. All the **heuristic values** are counted by the **heuristic function f** as below.

$$f(state) = g(state) + h(state)$$
$$g(state) = g(parent\ state) + the\ cost\ of\ state, \qquad g(start\ state) = 0$$
$$h(state) = ManhattanDistance(state, goal\ state)$$

It is trivial that the values counted by the **heuristic function** are always lower than the actual lowest cost from the start state to the goal state since Manhattan distance counts the least number of steps without any walls, increasing or decreasing 1 for each step. In other words, Manhattan distance is admissible and consistent, which means that it is a valid heuristic function. The **heuristic values** are saved in the dictionary **discovered_states** which are indexed by **state**. Below is the structure of **discovered_states**.

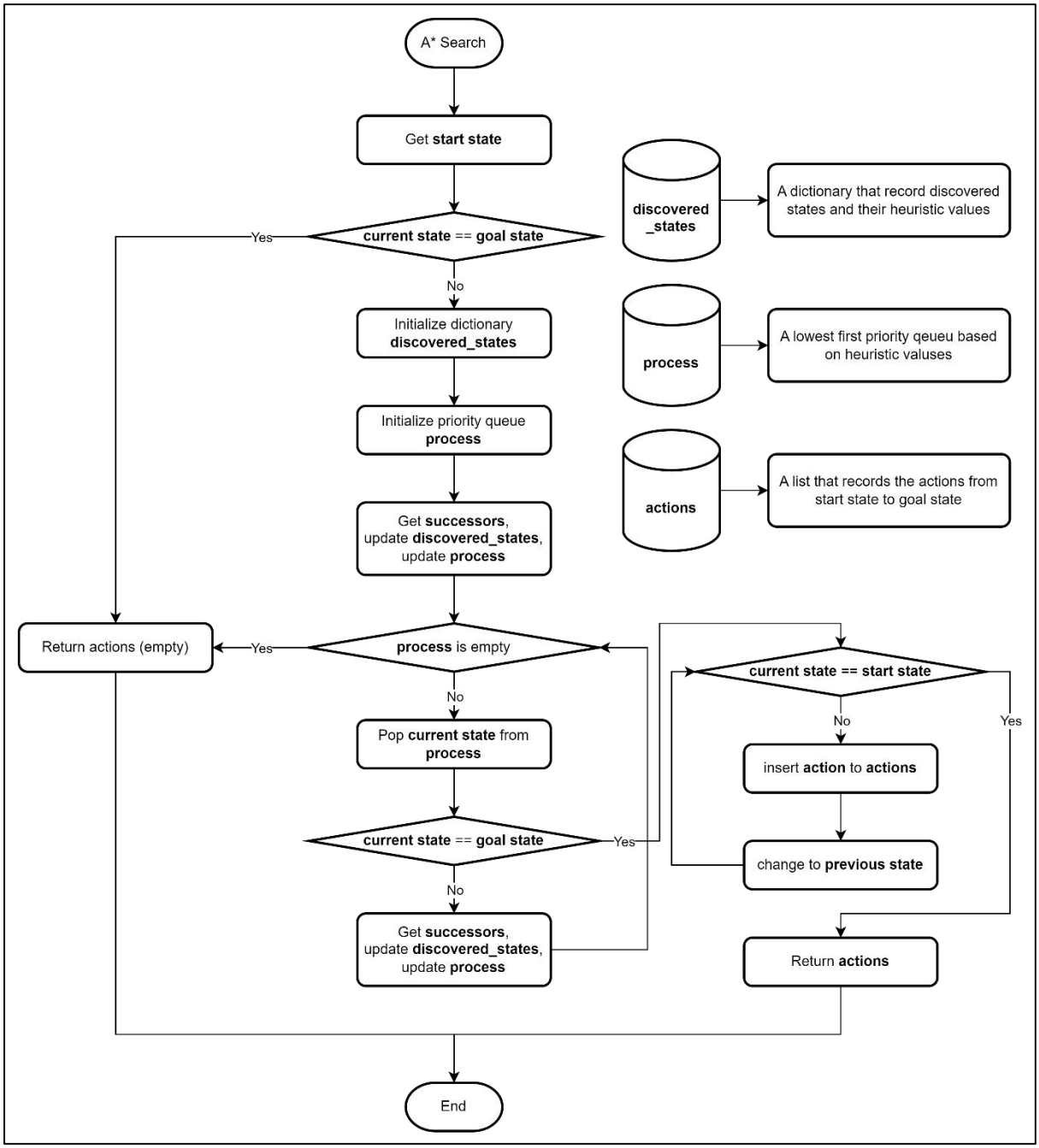
$$discovered\_states = \{state1: (parent\ state, action, heuristice\ value), state2: \dots\}$$

For every push, **process** sorts the elements by checking the **heuristic values** stored in **discovered_states**. The structure of the nodes in **process** is defined as below.

$$node = (parrent\ state, (current\ state, action, cost))$$

Before every push, **discovered_states** is updated by the new heuristic value. If the state is undiscovered, insert the state with the new heuristic value into **discovered_states** directly. Otherwise, update **discovered_states[state]** if the new heuristic value is less than the value recorded in **discovered_states**.

In summary, this algorithm follows the BFS framework but incorporates a priority queue to explore the lowest-cost path first, guided by the heuristic function. By maintaining **discovered_states** and updating heuristic values dynamically, it ensures an efficient search toward the goal while considering optimality.



Picture 1: The flow chart of the A* search algorithm

Table 1: The number of expanded nodes in Q4

# expanded node	tinyMaze	mediumMaze	bigMaze
BFS	15	269	620
A* Search	8	232	513

Q6: Corners Problem: Heuristic

Based on Q4, I modified the structure of **state**, **goal state**, and the **heuristic function**. The new structure of **state** is defined as below.

$$state = (current\ position, (positions\ of\ unvisited\ corners))$$

To fit the problem requirement, any two states with the same current position but with different unvisited corners are different, separating different layers with different goals. The **goal state** is defined as below, which means that the goal position is reached and there is only one goal.

$$goal\ state = (len(state[1]) ==\ 1\ and\ state[0]\ in\ state[1])$$

In **getSuccessors()** function, the new tuple of **positions of unvisited corners** is remain the same if the current position is not an element of the original tuple of **positions of unvisited corners**. Otherwise, this tuple should be updated as below and combined into the success state.

$$i\ =\ state[1].index(state[0])$$
$$new\ goal\ tuple =\ state[1][:i] +\ state[1][i+1:]$$
$$nextState\ =\ (next\ position, new\ goal\ tuple)$$

The **heuristic function f** is defined here. (Denote Manhattan distance as MD)

$$f(state) = g(state) + h(state)$$
$$g(state) =\ g(parent\ state) + the\ cost\ of\ state, \qquad g(start\ state) = 0$$
$$h(state) =\ mean\ (MD(state[0], state[1][0]), MD(state[0], state[1][0]), \dots)$$

I used the mean value of Manhattan distance of different goals, which must be lower than the actual cost from the current state to the goal state, expecting to guide the path toward to the positions of unvisited corners. Manhattan distance is admissible. For the case with multiple goals, the real cost from the current position to the farthest goal with some intermediate goals is greater than the mean Manhattan distance from the current position to all goals. Therefore, the **heuristic function** I designed is admissible. Since every step can only increase or decrease 0 **heuristic value** for all the corners are not visited, 1 / 3 for three corners are unvisited, 1 for no more than two corners are unvisited, which is not greater than the actual cost of each step, the **heuristic function** is consistent.

In conclusion, by redefining the **state structure** and refining the **heuristic function**, this approach effectively handles multiple goals while ensuring admissibility and consistency. Through experimentation, the max Manhattan distance heuristic proved to be the most efficient, making it the optimal choice for evaluating heuristic values in this problem.

Table 2: The number of expanded nodes in Q6

# expanded node	tinyCorners	mediumCorners	bigCorners
BFS	269	1988	7974
A* Search	212	937	4157