HW1 for CS4013/5013: Artificial Intelligence

Fall 2024

Release Date: Aug 26, 2024.

Due Date: Sep 15, 2024.

Programming Tasks

We will implement BFS, DFS and GBFS in Python based on the graph in Figure 1. Assume the starting point is arbitrary but the ending point is always G. The list next to the graph is the estimated cost of the cheapest path from a node to the goal.

Tip: You can store the graph of n nodes using an n-by-n adjacency matrix G, whose element $G(i, j)$ stores the weight between node i and node j. (If the two nodes do not connect, you may set $G(i, j)$ to a unique value such as -1.) Figure 2 shows part of the adjacency matrix for the graph in Figure 1. You can hard code it into the program, and store it using any data structure.

Rule: You need to implement all algorithms by yourself, which means not calling an existing search function from Python libraries. But feel free to call any other function e.g., queue.

Task 1. Implement breadth-first search and output the search sequence (which is the sequence of nodes being visited by your search algorithm). The sequence should begin with any starting node and terminate at any ending node. If multiple nodes are eligible to visit, prioritize them based on alphabetical order e.g., if you can visit 'D', 'E' or 'H', visit D first, then E, then H.

Task 2. Implement depth-first search and output the search sequence.

Task 3. Implement greedy-best-first search and output the search sequence.

A Python template 'hw1.py' will be provided with more instructions.

Written Tasks

We will practice A* search in theory based on the graph in Figure 3. The estimated cost of the cheapest path from a node to Store A or Store B is listed in Figure 4.

Task 4. Apply A* to find an optimal path from Store A to B and show the search sequence.

Task 5. Apply A* to find an optimal path from J to Store A and show the search sequence.

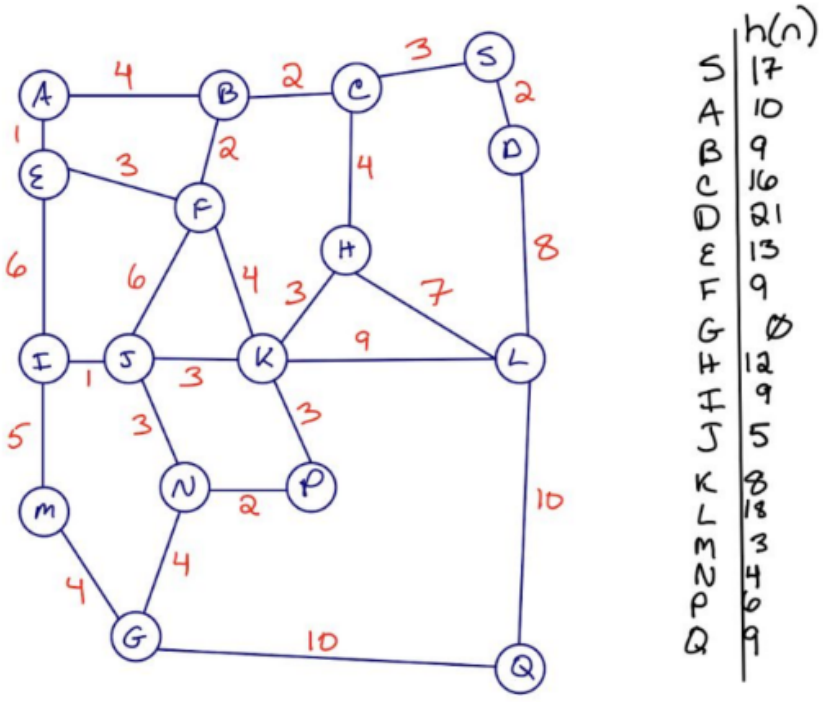

Figure 1: Graph for BFS/DFS/GBFS

	A	B	C	D	E	F	G	H	…
A	0	4	-1	-1	1	-1	-1	-1	…
B	4	0	2	-1	-1	2	-1	-1	…
C	-1	2	0	-1	-1	-1	-1	4	…
D	-1	-1	-1	0	-1	-1	-1	-1	…
E	1	-1	-1	-1	0	3	-1	-1	…
F	-1	2	-1	-1	3	0	-1	-1	…
G	-1	-1	-1	-1	-1	-1	0	-1	…
H	-1	-1	4	-1	-1	-1	-1	0	…
…	…	…	…	…	…	…	…	…	…

Figure 2: Part of the Adjacency Matrix for the Graph in Figure 1

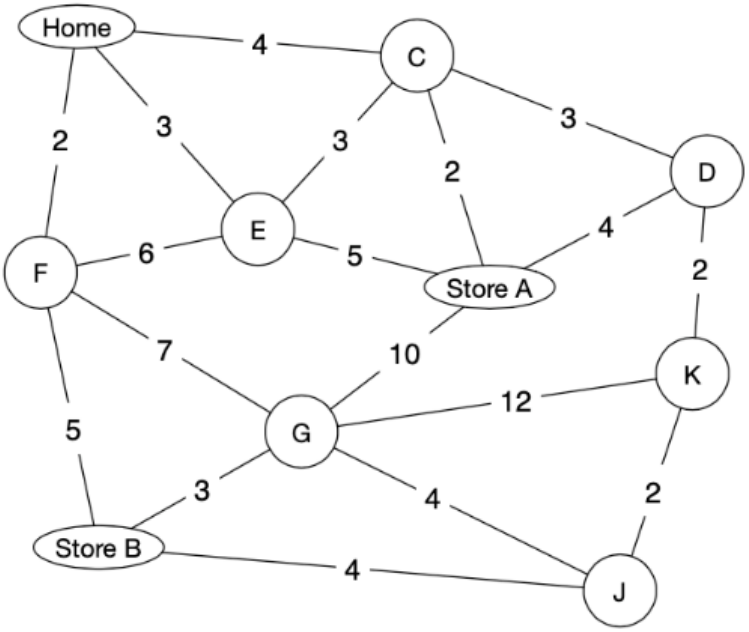

Figure 3: Graph for A* Search

Node	h(n): Store A	h(n): Store B
Home	5	7
Store A	0	11
Store B	11	0
C	2	8
D	4	7
E	4	10
F	10	4
G	9	3
J	7	4
K	6	6

Figure 4: Estimated Cost for the Graph in Figure 3