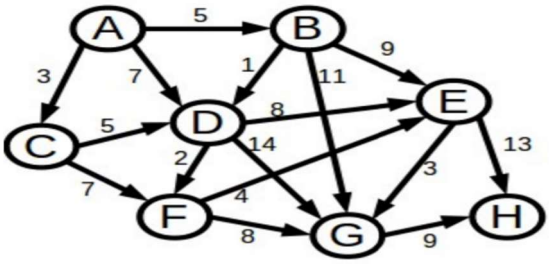
Yahya Alhinai

Oct 1, 2019

CSCI4511W

Problem 1:



Heuristic (for goal node H):

A: 30	E: 8
B: 29	F: 13
C: 25	G: 9
D: 15	H: 0

1: C(3+25 = 28) || **D(7+15 = 22)** || B(5+29 = 34)

DONE: A

2: C(3+25 = 28) || B(5+29 = 34) || **F(9+13 = 22)** || G(21+9 = 30) || E(15+8 = 23)

DONE: A, D

3: C(3+25 = 28) || B(5+29 = 34) || G(21+9 = 30) || **E(15+8 = 23)** || G(17+9 = 26)

DONE: A, D, F

4: C(3+25 = 28) || B(5+29 = 34) || G(21+9 = 30) || **G(17+9 = 26)** || G(18+9 = 27) || H(28+0 = 28)

DONE: D, F, E, G

5: C(3+25 = 28) || B(5+29 = 34) || G(21+9 = 30) || G(18+9 = 27) || H(28+0 = 28) || H(26+0 = 26)

DONE: D, F, E, G, **H**

The result of A* search:

A → D → F → G → H

Problem 2:

Are the heuristics above admissible? h(node) ≤ optimal path

A: h(A) ≤ cost(A to H)

A: 30 ≤ 26 (does not holds)

The heuristics is not admissible

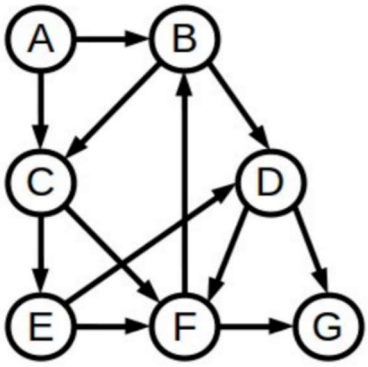
Are they consistent? h(node) ≤ cost(node to child) + h(child)

A: h(A) ≤ cost(A to B) + h(B)

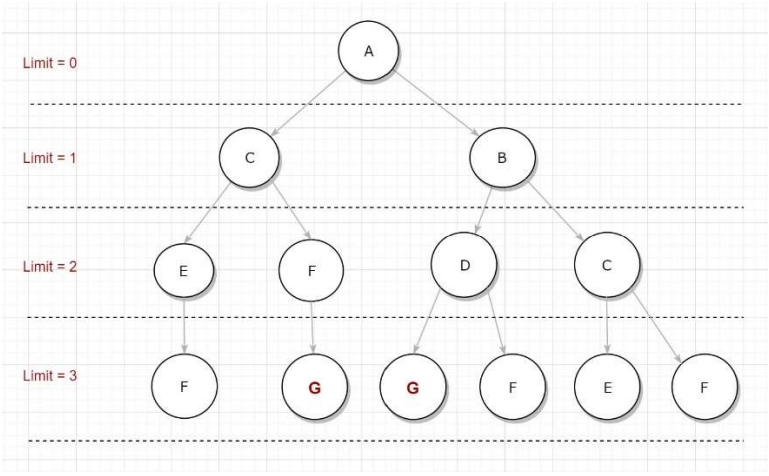
A: 30 ≤ (5 + 20) (does not holds)

The heuristics is not consistent

Problem 3:



1. **Convert the shown graph below into a tree:**



2. **Iterative Deepening Depth First Search (DFS):**
   - **Limit = 0**
     Search A
     A is not the goal

   - **Limit = 1**
     Search A → C → B
     C and B are not the goal

   - **Limit = 2**
     Search A → C → E → F → B → D → C
     E, F, and D are not the goal

   - **Limit = 3**
     Search A → C → E → F → F → G
     G is the Goal

Problem 4:

Situation 1:

- **Example state**: the location of the eight numbers in 3x3 grid and a blank square. An example would be (8, 6, 4, 2, blank, 7, 5, 3, 1).
- **Actions:** The actions are the movements of the blank space to either Left, Right, Up, or Down. The movement is depending on where the blank is.
- **Complete-state:** Because the game states with all elements in board (eight numbers + a blank) which represents state. Move elements around would transit to another state.

Situation 2:

- **Example state** Any arrangement of the n-items that account for the degree of fragility of all items as well as the items that can and cannot be put together in a bag. An example would be a bag that started with milk followed by half dozens of eggs followed by a small bag of chips.
- **Actions:** The action is adding one item at a time to a bag.
- **incremental:** Because it starts with an empty state and with every action you add an item to the state.

Situation 3:

- **Example state** Putting together any pieces of the couch together would be a state. An example is assembling the 4th leg of the couch to its main body.
- **Actions:** Putting together any of the wooden planks together.
- **incremental:** Because it starts with nothing being assembled and every time there are two or more pieces being assembled would be a state.

Problem 5:

Situation 1:

- **Graph**: Graph representation would work the best in this situation because the complete-state nature of this this game and because there are several different moves that can lead to the same state which, unlike tree representation, exclude repetitive states.

Situation 2:

- **Tree**: This is true because of the incremental nature of the problem. It starts with an N options and then followed by N-1 options and so forth until there are no items left.

Situation 3:

- **Tree:** This problem also requires a tree representation because of it's incremental states where you have to start with a wood plank then you add another wood plank to it until the couch get assembled as one piece.

Problem 6:

Situation 1:

- **Iterative deepening depth first search:** The produced solution would be optimal because this algorithm is searching into states as shallow as possible which guarantee reaching solution with the least moves possible. Iterative deepening depth first search was used over breadth first search because the fact it's a lot more efficient in terms of the usage of memory.

Situation 2:

- **Uniform cost search:** This search would be ideal to solve this problem because to solve such a problem we need to gather N items together as well as account for which items can be gathered together and with items cannot.

Situation 3:

- **Bi-directional search:** The fact that we know the initial state and the final state would make Bi-directional search is the ideal in this situation. Search the best possibilities of assembling the couch and at the same time the best possibilities of dissembling the couch and meet in the middle.

Problem 7:

The run-time and memory of the three different searches solving 8-puzzle game that has the values (1, 4, 2, 3, 5, 7, 8, 6, 0):

	Time	Memory
bi-directional	**1.548s**	**99, 336 kB**
breadth first	**Long time (not solved)**	**30231584 kB (not solved)**
A*	**7.081s**	**102, 884 kB**

- **Bi-directional search** would be the best search to solve 8-puzzles in both aspects: run-time and the usage of memory. The table above shoes an example of solving 8-puzzle game being solve using the three searches.
- **Bi-directional** searches a path from the initial state to the goal state. At the same time, it searches a path from the goal state to the initial state. As a results they meet in the middle, cutting out a great deal of memory usage because it visits less states as well as run-time to go through states making the search area look like two ellipses started from the initial and the goal state and meets in the middle.