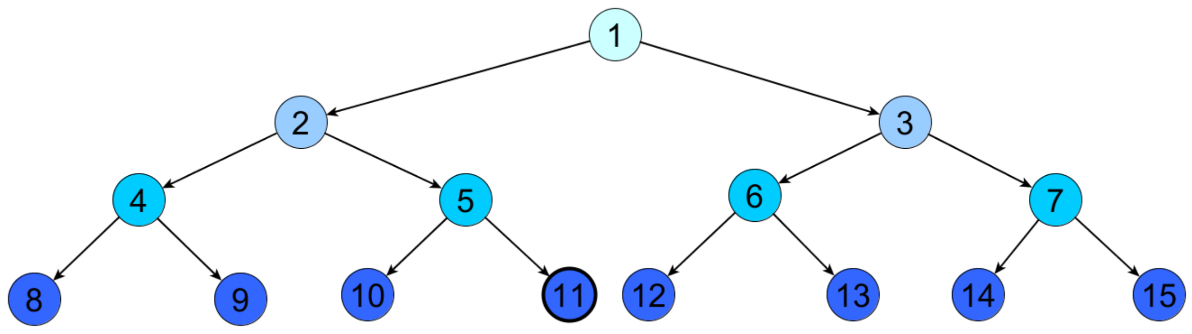
Question 1 (Question 3.8 of AIMA 2ed)

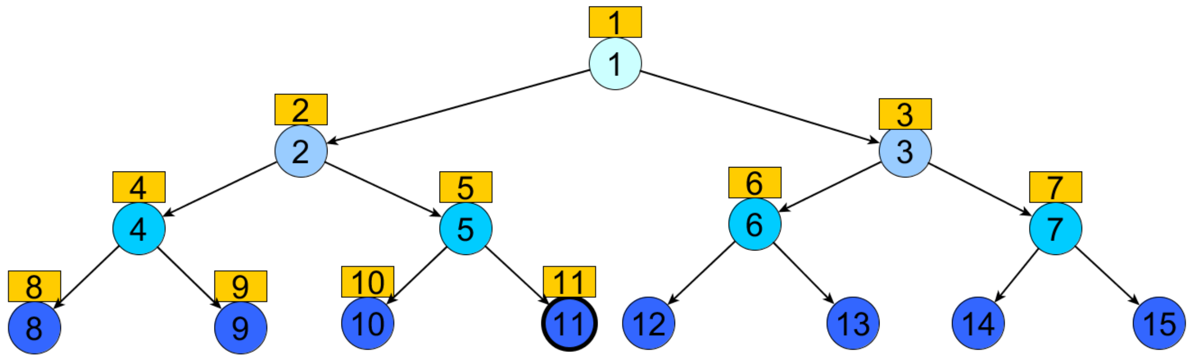
Consider a state space where the start space is 1 and the successor function for state n returns two states: 2n and 2n+1.

(a) Draw a portion of the state space up to state 15.

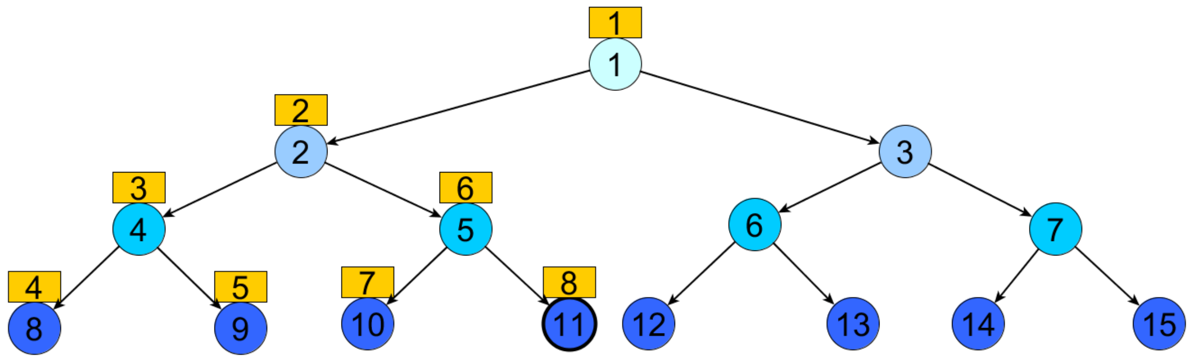


(b) Suppose the goal state is 11. List the order in which the nodes would be visited for:

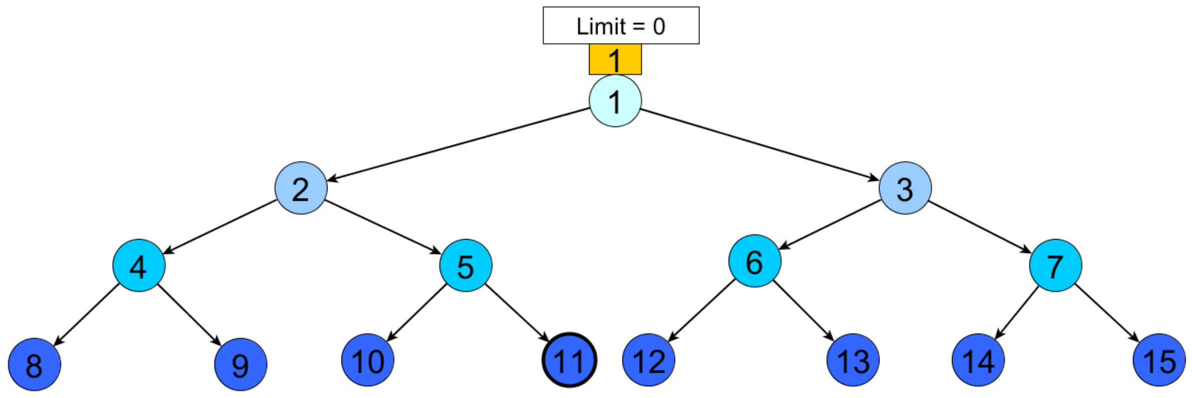
1. Breadth-first

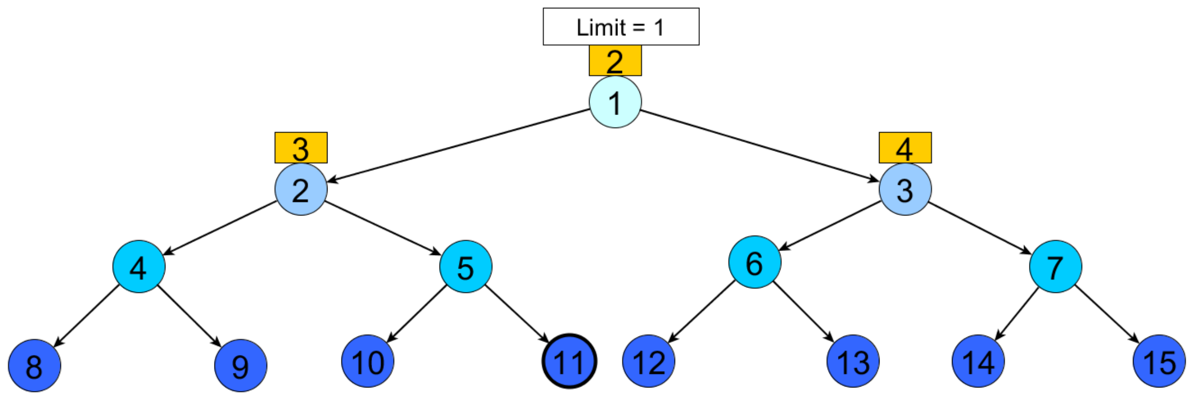


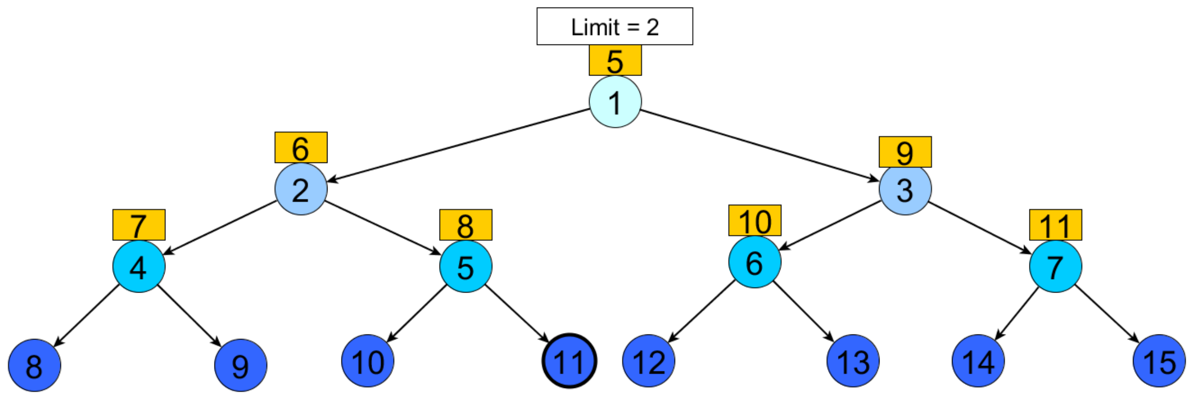
2. Depth-limited (to 3)

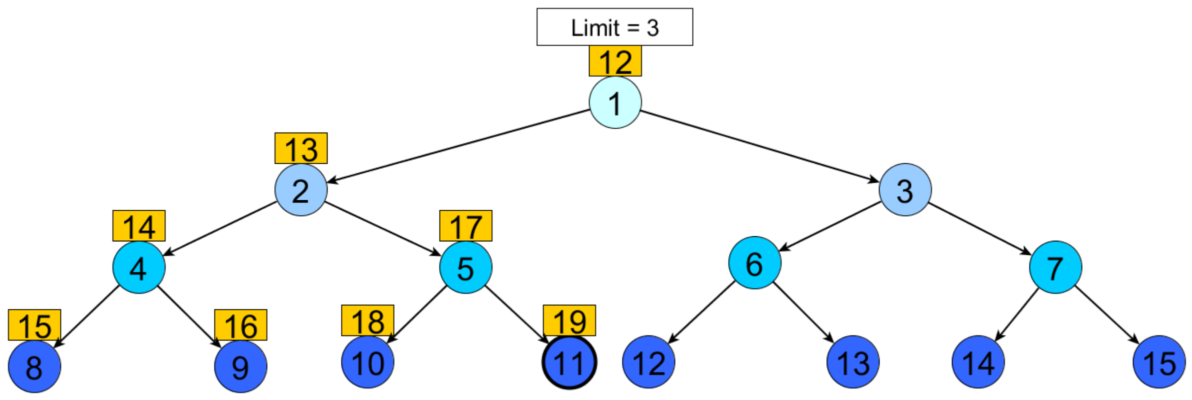


3. Iterative-deepening

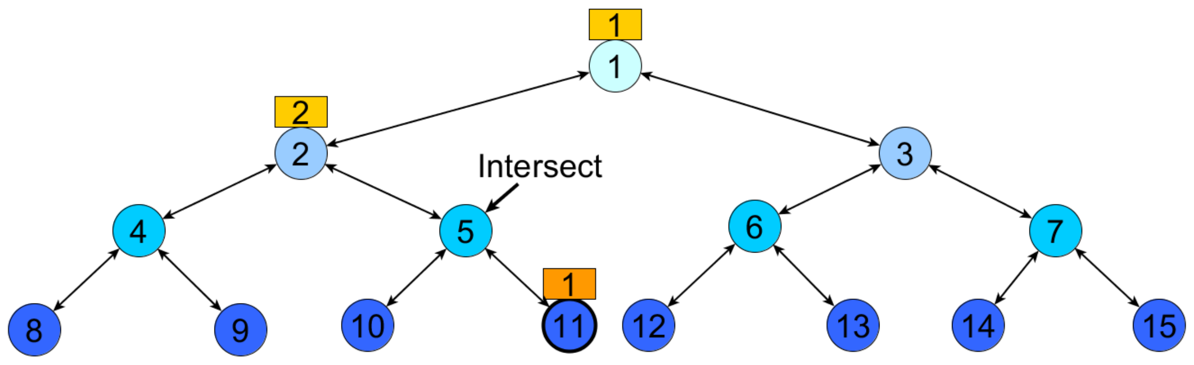








4. Bi-directional search starts the search from the start state and end state, and stops as soon as the two search trees meet. Would bi-directional search work for this problem? If so describe how it would work.



Bidirectional search runs BFS from both the starting node and the goal node simultaneously until their frontiers intersect. Bidirectional search will only work if each node can be reached from the other node within the graph – for this problem we need to assume each edge is undirected to use bidirectional search.

Bidirectional search is advantageous over BFS and Iterative Deepening Search as it has half the time complexity. It also has half the space complexity of BFS and is complete (if applicable and using BFS in both directions), unlike DFS. Using Bidirectional Search with Iterative Deepening in one direction and BFS in the other direction will have half the time complexity again and is another useful approach.

Question 2 (Question 3.2 of AIMA 3ed)

Your goal is to navigate a robot through a maze; see Figure 1. The robot enters the maze from the top left facing south. You can turn the robot to face north, east, south, or west. You can direct the robot to move forward a certain distance, although it will stop before hitting a wall.

(a) Formulate this problem. How large is the state space?

$$S = \{Orientations\} * \{Positions\}$$

$$= \{N, E, S, W\} * \{Positons\}$$

$$= \infty$$

(b) In navigating a maze, the only place we need to turn is at the intersection of two or more corridors. Reformulate this problem using this observation. How large is the state space now?

… You can direct the robot to move forward until in encounters an intersection or a wall.

$$S = \{Orientations\} * \{Positions\}$$

$$= \{N, E, S, W\} * \{Intersections\}$$

$$= 4I$$

(c) From each point in the maze, we can move in any of the four directions until we reach a turning point, and this is the only action we need to do. Reformulate the problem using these actions. Do we need to keep track of the robot's orientation now?

… You can move the robot north, south, east or west.

$S = \{Positions\}$

$\quad = \{Intersections\}$

$\quad = I$

(d) Define an appropriate path cost and heuristic function for this problem.

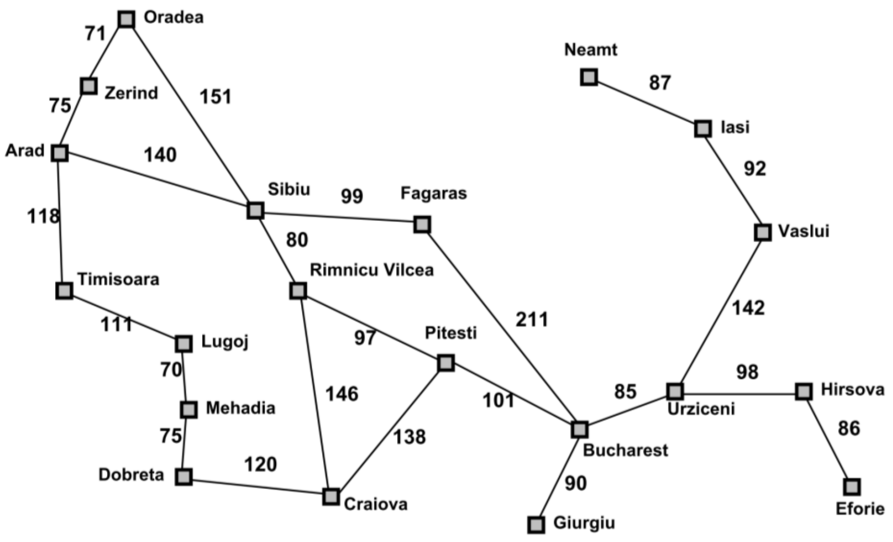
- Path cost could be distance travelled (Euclidian/Manhattan), time, number of intersections encountered (complexity), etc.
- Heuristic could be distance to goal (Euclidian/Manhattan)
- Manhattan distance is applicable as we're moving in a grid – it will have a lower computational cost than Euclidian.

(e) In our initial description of the problem we already abstracted from the real world, restricting actions and removing details. List three such simplifications we made.

- Precise turns/directions
- Cannot go through walls
- 2D representation of 3D, eg. Assuming maze is flat with on hills
- Accurate localisation
- Precise stopping
- Must go through maze, eg. Can't fly or drive around
- Etc.

<u>**Question 3 (Question 4.1 of AIMA 2ed)**</u>

Using Figure 2, show the steps in expanding nodes for an A* search for the shorted path from Lugoj to Bucharest (using the straight line distance heuristic).



Straight−line distance
to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

1.

Current = Lugoj

Frontier = {[Timisoara, 111+329=440], [Mehadia, 70+241=311]}

Explored = { Lugoj}

2.

Current = Mehadia, 70

Frontier = {[Timisoara, 111+329=440], [Dobreta, 70+75+242=387]}

Explored = {Lugoj, Mehadia}

3.

Current = Dobreta, 145

Frontier = {[Timisoara, 111+329=440], [Craiova, 145+120+160=425]}

Explored = {Lugoj, Mehadia, Dobreta}

4.

Current = Craiova, 265

Frontier = {[Timisoara, 111+329=440], [Pitesti, 265+138+98=501], [Rimnicu Vilcea, 265+146+193=604]}

Explored = {Lugoj, Mehadia, Dobreta, Craiova}

5.

Current = Timisoara, 111

Frontier = {[Pitesti, 265+138+98=501], [Arad, 111+118+366=595], [Rimnicu Vilcea, 265+146+193=604]}

Explored = {Lugoj, Mehadia, Dobreta, Craiova, Timisoara}

6.

Current = Pitesti, 403

Frontier = {[Bucharest, 403+101+0=504], [Arad, 111+118+366=595], [Rimnicu Vilcea, 265+146+193=604]}

Explored = {Lugoj, Mehadia, Dobreta, Craiova, Timisoara, Pitesti}

7.

Current = Bucharest, 504

Frontier = {[Arad, 111+118+366=595], [Rimnicu Vilcea, 265+146+193=604]}

Explored = {Lugoj, Mehadia, Dobreta, Craiova, Timisoara, Pitesti, Bucharest}

Question 4 (Question 4.2 of AIMA 2ed)

A heuristic path algorithm is a best-first search algorithm with the objective function $f(n) = (2-w)g(n)+wh(n)$. Assuming $h(n)$ is admissible, then:

(a) For what values of w is it guaranteed to be optimal?

For the search algorithm to be admissible $wh(n)$ must never overestimate the cost, thus:

$$0 \leq w \leq 1$$

Largest admissible $h(n)$ would be setting it to the exact distance remaining, in this case if $w > 1$ then $wh(n)$ will overestimate, hence $w \leq 1$

Setting $w < 0$ may cause negative path costs, hence $w \geq 1$

(b) What kind of search is performed in the special case w = 0?

$$f(n) = 2g(n)$$

Uniform-cost search (Dijkstra's)

Doubling all path costs in a graph will not change the search

(c) What kind of search is performed in the special case w = 1?

$$f(n) = g(n) + h(n)$$

A* search

(d) What kind of search is performed in the special case w = 2?

$$f(n) = 2h(n)$$

Greedy Search

<u>Question 5 (Question 4.3 of AIMA 2ed)</u>

Prove the following:

(a) Breadth first is a special case of uniform cost search.

Breadth first search is uniform cost search where the path cost is always identical (eg. 1 or c)

(b) Breadth first, depth-first and uniform-cost searches are special cases of best-first search.

Breadth first uses a heuristic where the cost is the depth

Depth-first uses a heuristic where the cost is the negative depth

Uniform-cost search uses a heuristic where the cost is the path cost

(c) Uniform-cost search is a special case of A* search.

Uniform-cost search is A* search with the heuristic set to zero.

<u>Question 6 (Question 4.5 of AIMA 2ed)</u>

A "greedy" bestfirst approach to going from Iasi to Fagaras is at best suboptimal (it goes first to the deadend Neamt) and possibly fatally flawed (never gets to Fagaras) if repeated states are not detected (where it may oscillate between Vaslui and Neamt). Firstly, you should work this out for yourselves – assume the map in Figure 2 is reasonably indicative of the missing straightline distances (i.e., closeness on the map is actually closeness in real life). Likewise, note that there is

no problem in solving the reverse trip – from Fagaras to Iasi. Then, answer the following question: are there problems for which the straightline distance heuristic fails in both directions (still in the context of best-first search)?

Yes, greedy search can fail in both directions

A->B Failure (Dark Green)

B->A Failure (Light Green)

A<->B Shortest Path (Purple)