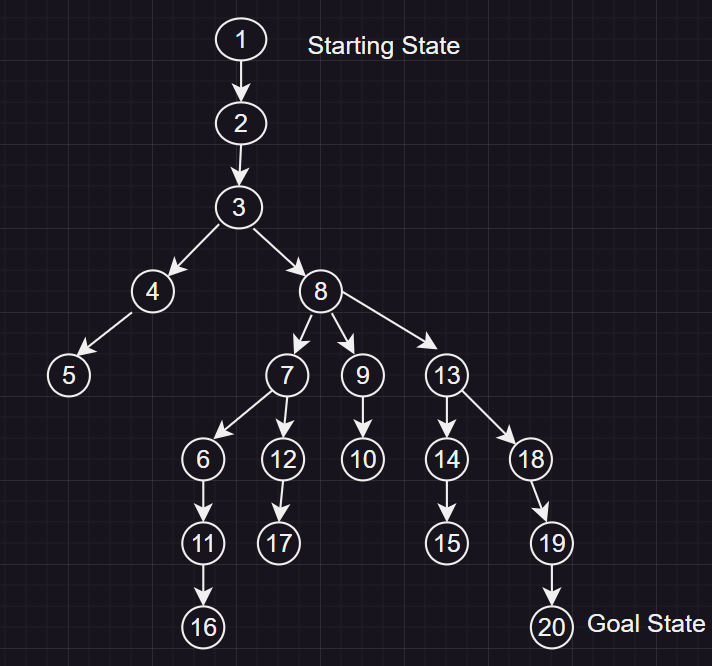
Assignment1 – Ben Smerd 22072922

* **State-space structure**



* **Depth-first search**

1. Open=[1];closed=[]
2. Open=[2];closed=[1]
3. Open=[3];closed=[2,1]
4. Open=[4,8];closed=[3,2,1]
5. Open=[5,8];closed=[4,3,2,1]
6. Open=[8];closed=[5,4,3,2,1]
7. Open=[7,9,13];closed=[8,5,4,3,2,1]
8. Open=[6,12,9,13];closed=[7,8,5,4,3,2,1]
9. Open=[11,12,9,13];closed=[6,7,8,5,4,3,2,1]
10. Open=[16,12,9,13];closed=[11,6,7,8,5,4,3,2,1]
11. Open=[12,9,13];closed=[16,11,6,7,8,5,4,3,2,1]
12. Open=[17,9,13];closed=[12,16,11,6,7,8,5,4,3,2,1]
13. Open=[9,13[;closed=[17,12,16,11,6,7,8,5,4,3,2,1]
14. Open=[10,13];closed=[9,17,12,16,11,6,7,8,5,4,3,2,1]
15. Open=[13];closed=[10,9,17,12,16,11,6,7,8,5,4,3,2,1]
16. Open=[14,18];closed=[13,10,9,17,12,16,11,6,7,8,5,4,3,2,1]
17. Open=[15,18];closed=[14,13,10,9,17,12,16,11,6,7,8,5,4,3,2,1]
18. Open=[18];closed=[15,14,13,10,9,17,12,16,11,6,7,8,5,4,3,2,1]
19. Open=[19];closed=[18,15,14,13,10,9,17,12,16,11,6,7,8,5,4,3,2,1]
20. Open=[20];closed=[19, 18,15,14,13,10,9,17,12,16,11,6,7,8,5,4,3,2,1]
21. Goal reached [20]
    * Path(with backtracking): 1->2->3->4->5->4->3->8->7->6->11->16->11->6->7->12->17->12->7->8->9->10->9->8->13->14->15->14->13->18->19->20
    * Discussion: Depth-first search will go down to the bottom of each branch of a child node of a parent, before moving on to a sibling branch of the same parent node.

* **Breadth-first search**

1. Open=[1];closed=[]
2. Open=[2];closed=[1]
3. Open=[3];closed=[2,1]
4. Open=[4,8];closed=[3,2,1]
5. Open=[8,5];closed=[4,3,2,1]
6. Open=[5,7,9,13];closed=[8,4,3,2,1]
7. Open=[7,9,13];closed=[5,8,4,3,2,1]
8. Open=[9,13,6,12];closed=[7,5,8,4,3,2,1]
9. Open=[13,6,12,10];closed=[9,7,5,8,4,3,2,1]
10. Open=[6,12,10,14,18];closed=[13,9,7,5,8,4,3,2,1]
11. Open=[12,10,14,18,11];closed=[6,13,9,7,5,8,4,3,2,1]
12. Open=[10,14,18,11,17];closed=[12,6,13,9,7,5,8,4,3,2,1]
13. Open=[14,18,11,17];closed=[10,12,6,13,9,7,5,8,4,3,2,1]
14. Open=[18,11,17,15];closed=[14,10,12,6,13,9,7,5,8,4,3,2,1]
15. Open=[11,17,15,19];closed=[18,14,10,12,6,13,9,7,5,8,4,3,2,1]
16. Open=[17,15,19,16];closed=[11,18,14,10,12,6,13,9,7,5,8,4,3,2,1]
17. Open=[15,19,16,20];closed=[17,11,18,14,10,12,6,13,9,7,5,8,4,3,2,1]
18. Open=[19,16,20];closed=[15,17,11,18,14,10,12,6,13,9,7,5,8,4,3,2,1]
19. Open=[16,20];closed=[19,15,17,11,18,14,10,12,6,13,9,7,5,8,4,3,2,1]
20. Open=[20];closed=[16,19,15,17,11,18,14,10,12,6,13,9,7,5,8,4,3,2,1]
21. Goal reached[20]
    * Path: 1->2->3->4->->3->8->5->7->9->13->6->12->10->14->18->11->17->15->19->16->20
    * Discussion: Breadth first uses a queue when processing nodes (first in, first out order). Once the goal node has been dequeued from the open list then the goal has been reached. Breadth-first works by processing each node within a level independent of how each node on that level is related to one another, and it will only go to the next level once every node on the level has been processed. This can make sure that the search never goes deeper within the tree than where the goal state is.

* **A\* search**
  + TODO need to define the g(n) and h(n) as tiles moved (both have to be the same) and then make that the cost function so g(n) would be tiles moved to a certain position I think and h(n) is tiles left to the goal state
  + TODO maybe c an use straight line distance for the puzzle? Maybe this is the straight line from the state to goal avoiding the walls so it would just be the optimal path if I was drawing it like a maze, that is a straight path???