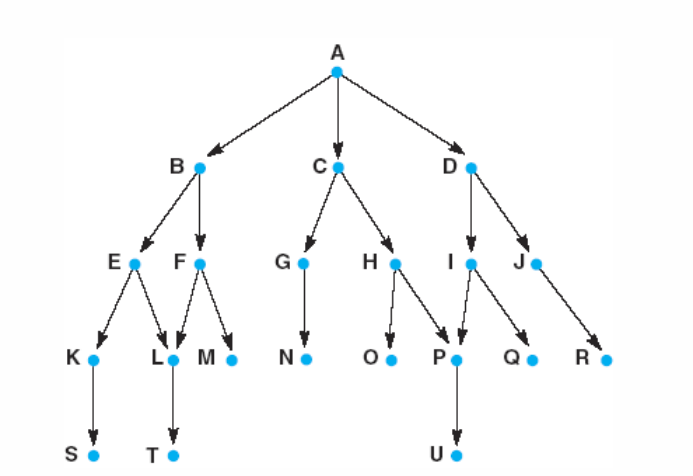
Project 2. Uninformed search

The types of search considered here are called *uninformed* because they do not use any *a priori* knowledge about the search domain and the search criteria.

Uninformed search algorithms are implemented on state space graphs such as the following one.



A state space graph

Here A, B, C, D, E, …, are the states with A as the Start state and U as the Goal state where the search conditions are fulfilled. The most common uninformed search algorithms are the breadth-first and depth-first searches.

The above state space graph is traversed in a breadth-first search of the goal as follows:

The BFS trace

1. **open = [A]; closed = [ ]**
2. **open = [B,C,D]; closed = [A]**
3. **open = [C,D,E,F]; closed = [B,A]**
4. **open = [D,E,F,G,H]; closed = [C,B,A]**
5. **open = [E,F,G,H,I,J]; closed = [D,C,B,A]**
6. **open = [F,G,H,I,J,K,L]; closed = [E,D,C,B,A]**
7. **open = [G,H,I,J,K,L,M]** (as L is already on open); **closed = [F,E,D,C,B,A]**
8. **open = [H,I,J,K,L,M,N]; closed = [G,F,E,D,C,B,A]**
9. and so on until either U is found or **open** = [ ]

This trace of BFS is performed using two stacks, **open** and **close**.

Task 1.

Consider the depth-first search for the same graph.

Develop the trace of the DFS similar to the above BFS trace.

Task 2.

Develop the pseudocodes for both BFS and DFS using the stacks as above.

Task 3.

This task is about using the BFS and DFS for solving the 8-puzzle.

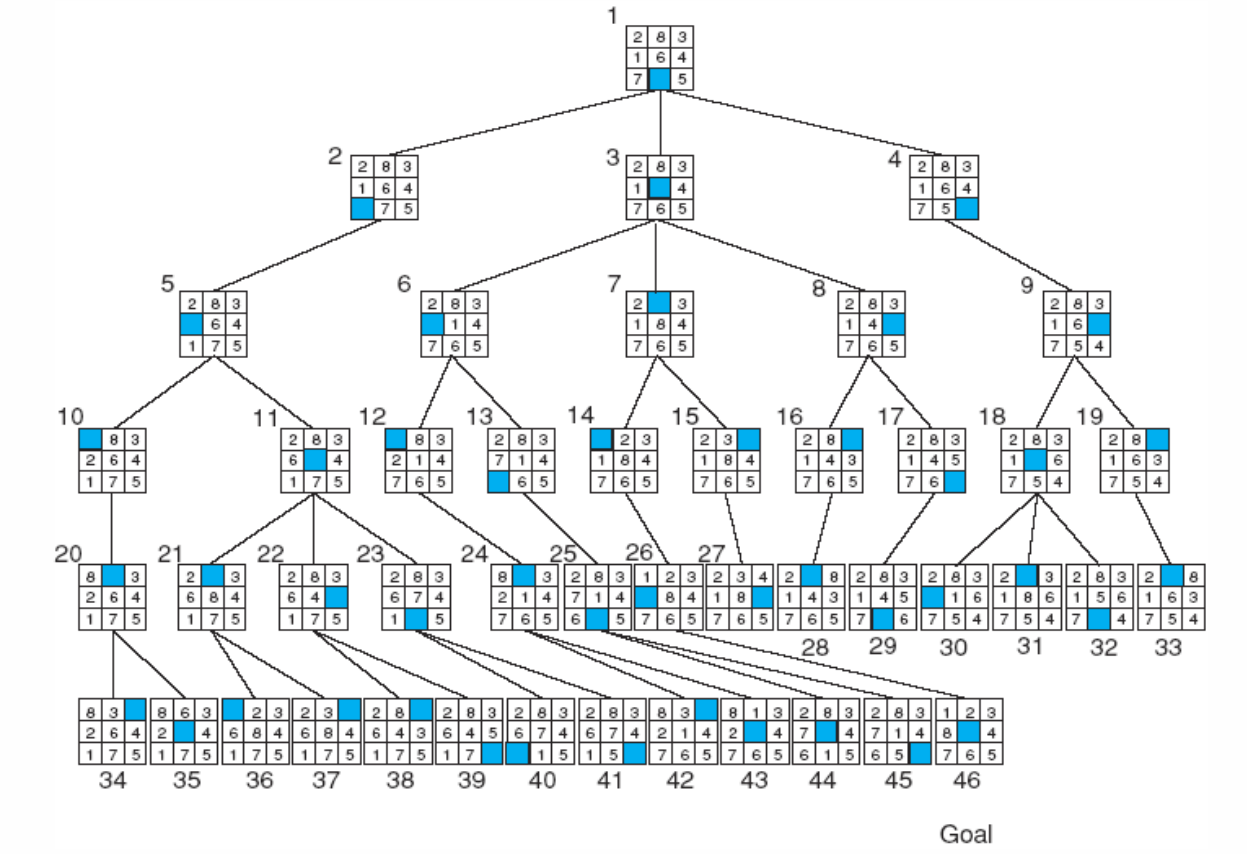
8-puzzle

The 8-puzzle belongs to the family of sliding block puzzles used as test problems for search algorithms in AI. It uses a 3 by 3 board of numbered (from 1 to 8) tiles and one empty space (shown below as a blue square). A tile adjacent to the empty square can be "swapped" with this empty space.

Starting with a given unordered displacement the steps should end up with the ordered tiles:

START → GOAL

The solution is illustrated with the following search tree expansion:



Using your favorite programming language develop two 8-puzzle programs implementing the BFS and DFS methods in accordance with the pseudocodes from Task 2.

You should decide on

(a) representation of needed data structures to make the program more efficient;

(b) method of the tree expansion for an arbitrary start position of the tiles;
(c) evaluation of a position – is it the goal?

Run the both programs for some start positions. Compare the results. Which of the two methods works better? What criteria should be used for comparison?

Submission

This Project can be performed by one or two students working together if you decide so.

Put your work into a single file including answers to all tasks above.

Include your codes, and the run results.

Submit your work to the Beachboard by Friday September 16, 11:59 pm.