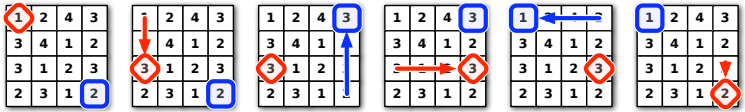
15.6
Exercise: Modefling problems using graphs

Modeling Problems as Search

The following puzzle was invented by the infamous Mongolian puzzle-warrior Vidrach Itky Leda in the year 1473. The puzzle consists of an $n \times n$ grid of squares, where each square is labeled with a positive integer, and two tokens, one red and the other blue. The tokens always lie on distinct squares of the grid. The tokens start in the top left and bottom right corners of the grid; the goal of the puzzle is to swap the tokens.

In a single turn, you may move either token up, right, down, or left *by a distance determined by the **other** token*. For example, if the red token is on a square labeled 3, then you may move the blue token 3 steps up, 3 steps left, 3 steps right, or 3 steps down. However, you may not move a token off the grid or to the same square as the other token.



A five-move solution for a 4×4 Vidrach Itky Leda puzzle.

Describe and analyze an efficient algorithm that either returns the minimum number of moves required to solve a given Vidrach Itky Leda puzzle, or correctly reports that the puzzle has no solution. For example, given the puzzle above, your algorithm would return the number 5.

Undirected vs Directed Connectivity

Consider following problem.

- Given undirected graph $G = (V, E)$.
- Two subsets of nodes $R \subset V$ (red nodes) and $B \subset V$ (blue nodes). R and B non-empty.
- Describe linear-time algorithm to decide whether every red node can reach every blue node.

How does the problem differ in directed graphs?

Undirected vs Directed Connectivity

Consider following problem.

- Given undirected graph $G = (V, E)$.
- Two subsets of nodes $R \subset V$ (red nodes) and $B \subset V$ (blue nodes). R and B non-empty.
- Describe linear-time algorithm to decide whether every red node can reach every blue node.

How does the problem differ in directed graphs?

Undirected vs Directed Connectivity

Consider following problem.

- Given directed graph $G = (V, E)$.
- Two subsets of nodes $R \subset V$ (red nodes) and $B \subset V$ (blue nodes).
- Describe linear-time algorithm to decide whether every red node can be reached by some blue node.

THE END

...

(for now)