A Motivating Application

Data Structures

State Transition Graph of 8-Puzzle

Goal: Understand data structures by solving the puzzle problem

- Graphs
  - 3. Priority queues
  - 2. Queues
  - 1. Stacks

- Sequence Structures
  - 2. Hash tables
  - 1. Binary search trees

- Search Structures
  - 3. Trees
  - 2. Lists
  - 1. Arrays

- Elementary Structures
Graph: a very general data structure

In some graphs, edges may have additional information.

A path in a graph is a sequence of edges in which each edge follows the next.

A simple path is a path in which every node is the source and destination of at most one edge.

The length of a path is the number of edges.

A cycle is a path that begins and ends at the same node.

An Eulerian cycle is a cycle that uses every edge exactly once.

A Hamiltonian cycle is a cycle that visits every node exactly once.

In a geometric graph, nodes are points and edges are line segments.

In a network graph, nodes are points and edges are communication links.

In some graphs, node labels may be needed with N.S.W.

In some graphs, edges may have additional information.

Example: A graph with nodes A, B, C, D, E, F, and edges (A, B), (B, C), (C, D), (D, E), (E, F), and (F, A).

Graph 1: A single path from A to F.

Graph 2: A cycle that includes all nodes.

Graph 3: Two nodes connected by an edge.

Graph 4: A special node and edges between nodes.
No one knows if there is an efficient algorithm for this problem.

Fancy to come up with some algorithm for this problem?

Finding shortest paths:

For puzzle problem, answer is no for some nodes!

Many interesting problems can be phrased as path problems in graphs.

Reachability problem:

(1) Is there a path from a given node to the node representing sorted position?

Minimal path problems:

Find the shortest path from node A to node F.

Find the shortest path from every node to F.

Length of a path: number of edges in path

For puzzle problem, this corresponds to path with fewest moves.

This is more appropriate for path problems in graphs representing maps.

Sometimes edges have distances.

For puzzle problem, the corresponding to path with fewest moves.

Path problems

Cycle: Path that starts and ends at same node.

Travelling salesman’s problem:

Find the smallest length cycle that passes through all nodes.

No one knows if there is an efficient algorithm for this problem.

Easy to come up with slow algorithms for this problem.
Searching for something in a graph rather than in an array.

Think depth search is similar to breadth search except that we are

all states reachable from scrambled state.

Stop if you either generate a solved state or you have generated

"..."

Create states adjacent to those states.

Create states adjacent to scrambled state.

Start in the scrambled state.

These:

from scrambled state for puzzle problem

Code: Write a program to determine if solved state is reachable

Pseudocode for Graph Search algorithm:

Key Ideas: Keep two sets of nodes.

1. todo: set of nodes whose successors need to be examined

2. done: set of nodes whose successors have been examined

2. done: set of nodes whose successors have been examined

I. todo: set of nodes whose successors need to be examined

foreach node a in todo set:

if a is the goal node, detect a victory!

foreach node a adjacent to a do // traverse its edges (b ← a)

foreach node b move a to b one step forward // check if we have never explored a before

if b is not in todo set or done set:

\{ b is already in the done set \}

initialize todo set with scrambled configuration

\{ done set is not empty \}

initialize todo set with scrambled configuration

2. done: set of nodes whose successors have been examined

foreach node a in todo set:

if a is the goal node, detect a victory!

foreach node b move a to b one step forward // check if we have never explored a before

Add node a to done set and get it out again

if b is already in the done set:

Modifications before adding a to todo set, check it is already
Let us try to execute a few steps of the algorithm for this problem.

Algorithm for Solving Puzzle:

1. Initialize a set with scrambled configuration.
2. Add a to todo set!
3. If a is the goal node (decrease victory).
4. If a is not in todo set ()
5. For each node m adjacent to a do // here is edge (m <- a) (a) (a) in done set (cont.)
6. Add a node to todo set!
7. Remove a node from todo set!
8. While (todo set is not empty)
9. Initialize a set with scrambled configuration.
10. Explanation:
11. This is not necessary, so let us try code.
12. Modification handling set-loops more efficiently.
13. Explanation:
14. If a < - (a) is an edge, we would add a to todo set when
For search structures, best search is important

To answer the following questions:

(1) SEARCH STRUCTURE: In what order should we get nodes from the input set? When does this structure can we design to give us

(2) SEARCH STRUCTURE: How do we organize the input set so

Answer: Binary search trees, hash tables

that we can search it efficiently?

Answer: Search, graph, priority queues

the input set grows and shrinks

the nodes in the order? How can we accommodate the fact that

Two key interfaces:

For search structure, best search is important

Search interface: all search structures implement this interface

Search structure: all sequence structures implement this interface

Two key interfaces:

{ size() {
  boolean isEmpty()
  int[] get(int i) // extract from sequence structure
  void put(int i, e) // track into sequence structure
}

{ size()
  boolean search (e)
  for each e in search structure
  void insert (e) // track into search structure
}

With Generic code
```java
27
toggle(p1) {
    return !
}
}
System.out.println("Hello, World!");
)
if (!move(solved))
    if (done.search())
        move(solved) = true;
    } // move to the move
    ch = move characted();
    int n = p.occupied();
    for (int i = 0; i < moves.length(); i++)
        if (n + i < moves.length())
            if (move(i, n))
                continue;
            if (done.search())
                continue;
}
```
How do we implement a good search structure?

Good search structure

load data structure grows and shrinks. How do we implement a

Ex. this later.
cannot use it to perform a graph search in a general graph. We will.
The puzzle state transition graph is hand written into the code. We

Subject is wonderful.

sequence structure then implement the interface defined before.
The code we have written will work for any search structure and

1r