Chapter 5, Dynamic Programming:

- Introduction and sample problems (previous week Wednesday)
- Principles of DP (previous week Friday)
- DP algorithms, solutions to sample problems (last week Monday)
- Introduce optimal BSTs / review for test 2 (last week Wednesday)
- **Test 2, not covering DP** (last week Friday)
- Test 2 retrospective (**Today**)
- Finish up optimal BSTs (Wednesday)
- [Begin hash tables (Friday)]

Today:

- How I scored tests
- Background of the chutes and ladders problem
- Comments on other problems
- Looking ahead
before asking about the following generation: Consider the family and interviews going deep down branches versus broad across generations:

[Diagram of family tree]

Who is the first ancestor? Patricia
Who is the next child? Karin
Who is the next child? Esther
Who is the next child? Annika
Who is the next child? Elizabeth
Who is the next child? Megan
Who is the next child? Kristen
Who is the next child? Raylee
Who is the next child? Emma

Who is the first ancestor? Patricia
Who is the next child? Karin
Who is the next child? Joan
Who is the next child? Kristen
Who is the next child? Raylee
Who is the next child? Emma

The point is not that there is a right way or wrong way to do it, but that these two strategies can lead to essentially
public int[] distances() { return distances; }

public int[] traverse(Graph g, int start, PerformOnVertex op) {
    int[] parents = new int[g.numVertices()];
    int[] distances = new int[g.numVertices()];
    Queue<Integer> worklist = new ListQueue<Integer>();
    worklist.enqueue(start);
    distances[start] = 0;
    parents[start] = start;
    while (!worklist.isEmpty()) {
        int front = worklist.remove();
        op.perform(front);
        for (int u : g.adjacents(front)) {
            if (parents[u] == -1) {
                worklist.enqueue(u);
                distances[u] = distances[front] + 1;
                parents[u] = front;
            }
        }
    }
}
Reading and practice: Sections 4.(1-3)

Due: Friday, September 30, 2022 at 11:59 pm

Read Sections 4.(1-3), pg 265-302. This is a big chunk---spread it out! I'm putting this, together with the exercises, as "due" on Friday, but you should work on it in parallel to what we're doing in class, with the Friday due date being for any pieces you still need to finish up.

Do Exercise 4.(26-29) [pg 360-361]. This is a programming assignment, kind of like a mini-project. You will find starter code in the repository under practice/traversal, and a solution under practice/traversal-so1n.

Posted Wed Aug 17, 2022 at 2:00 pm

Comments

There are no comments

Write a comment

Post
<table>
<thead>
<tr>
<th>Exercise</th>
<th>True/False</th>
<th>Points</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>2. I did Exercises 4.(26-29)</td>
<td>True/False</td>
<td>1 Point</td>
<td>100.0%</td>
</tr>
<tr>
<td>4. Indicate which algorithm uses which ADT</td>
<td>Matching</td>
<td>1 Point</td>
<td>100.0%</td>
</tr>
</tbody>
</table>
6. Consider the following code assuming $g$ is a graph with the usual interface.

```java
Queue<Integer> q = new SomeQueue<Integer>();
Set<Integer> discovered = new SomeSet<Integer>();
q.enqueue(0);
discovered.add(0);
while (!q.isEmpty()) {
    int u = q.remove();
    // ....do something with u...
    for (int v : g.getAdjacents(u))
        if (!discovered.contains(v)) {
            q.enqueue(v);
            discovered.add(v);
        }
}
```

(2 points each)

a. Does the following perform breadth-first traversal or depth-first traversal of $g$ from vertex 0?

b. What would you change in the above code to transform it into the other (breadth-first traversal vs depth-first traversal)?

c. Which algorithm (breadth-first or depth-first) would you use to find the shortest path between vertices in terms of number of edges (not in terms of edge weights)?
3. **[Graphs.]** Recall that *breadth first traversal* is an algorithm pattern used to iterate over all the vertices in a graph that are reachable from a given starting vertex in an order based on the structure of the graph. Specifically, breadth first traversal first visits the source vertex, then all vertices one edge away from the source, then all the vertices two edges away from the source, etc.

For an example, consider the following graph. The circled numbers indicate the order in which they are visited in breadth first traversal starting from 0.

![Graph diagram]

Finish the class `bfi.BreadthFirstIterator`, which iterates over the vertices in the graph in breadth-first order from a given source. Specifically, when a given a graph and a starting vertex `source` are passed to this class’s constructor, the resulting iterator behaves such that successive calls to `next()` return the vertices reachable from `source` in breadth-first order. For example, when given the graph above and vertex 0 as `source`, then calls to `next()` return 0, 1, 3, 4, 2, respectively, after which calls to `hasNext()` return false.

Graphs are represented by the same `Graph` interface that was used in class and the projects, with vertices identified by whole numbers. The class used is `AdjListGraph`, but that shouldn’t matter.

4. **[Graphs.]** Suppose you are given a directed graph and a source vertex `s`. What
package q2graph;

import java.util.Iterator;

/**
 * CheatsAndLadders
 * * Placeholder for method to find the fewest moves needed
 * to finish a chutes and ladders board
 */

public class CheatsAndLadders {
    /**
     * Find the fewest number of moves needed to get to the end of the
     * given board.
     * * @param board A Chutes and Ladders board
     * @return The fewest number of moves needed to get to the finish
     */
    public static int fewestMoves(ChutesAndLaddersBoard board) {
        // What could you use this for?
        int[] d = new int[board.size()];
        throw new UnsupportedOperationException();
    }
}
Coming up:

Catch up on projects…

Due Mon, Nov 14 (end of day)
Do Project 6.1.b as a practice problem
Take quiz (on Section 6.4)

Due Wed, Nov 16 (end of day)
Read Section 6.5
(No quiz on Section 6.5)

Do Optimal BST project (suggested by Monday, Nov 21)

Due Fri, Nov 18 (end of day)
Read Sections 7.(1 & 2)
Take quiz