CSDS 233 Spring Session 16

SI Leader: Jakob Danninger

04/18/2023

Disclosure: This is a supplement to class, not a replacement. This should not be your only study activity for exams, it should aid you in studying. I do not have access to the actual exam so questions here will differ from those on the exam.

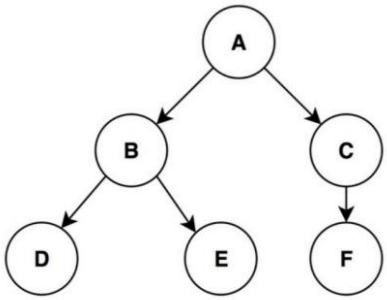
Session Objectives:

1) Understand how Depth First Search works and its uses

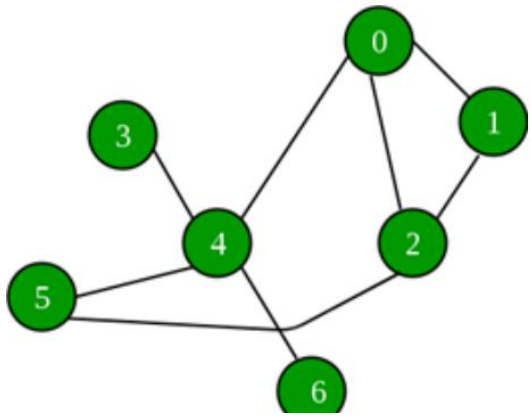
2) Understand how Breadth First Search works and its uses

Questions

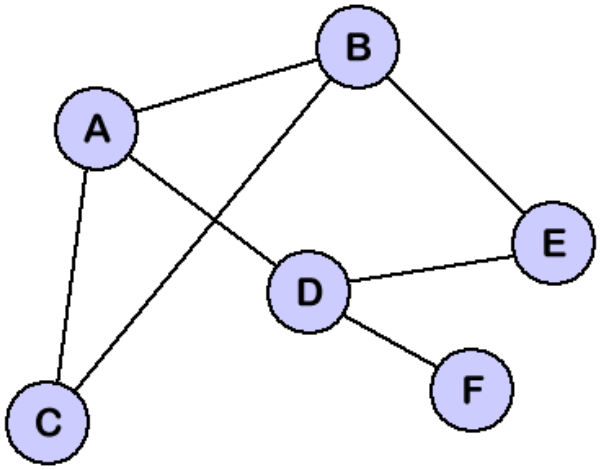
1) Fill in the blanks with neighbors, backtracking, starting, and encountered to complete the steps for the breadth and depth search:
    a. Depth First Search: Visit the_____Vertex. Then proceed as far as possible along a path before _____ and going along the next path
    b. Breadth First Search: Starting at the starting vertex, visit all of its_____and place them in a queue then take them out in order and repeat the process if the vertex from the queue is already _____ then don't add it to the queue
2) Write down the DFS and BFS of the following graph starting at A. Left children have priority over right children



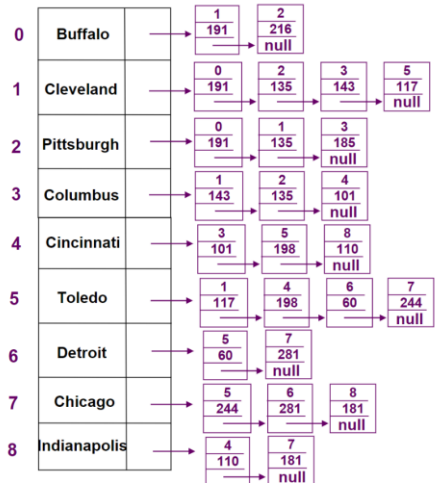
3) Identify the DFS and BFS of this graph, starting at 0 picking the lower number vertices first :

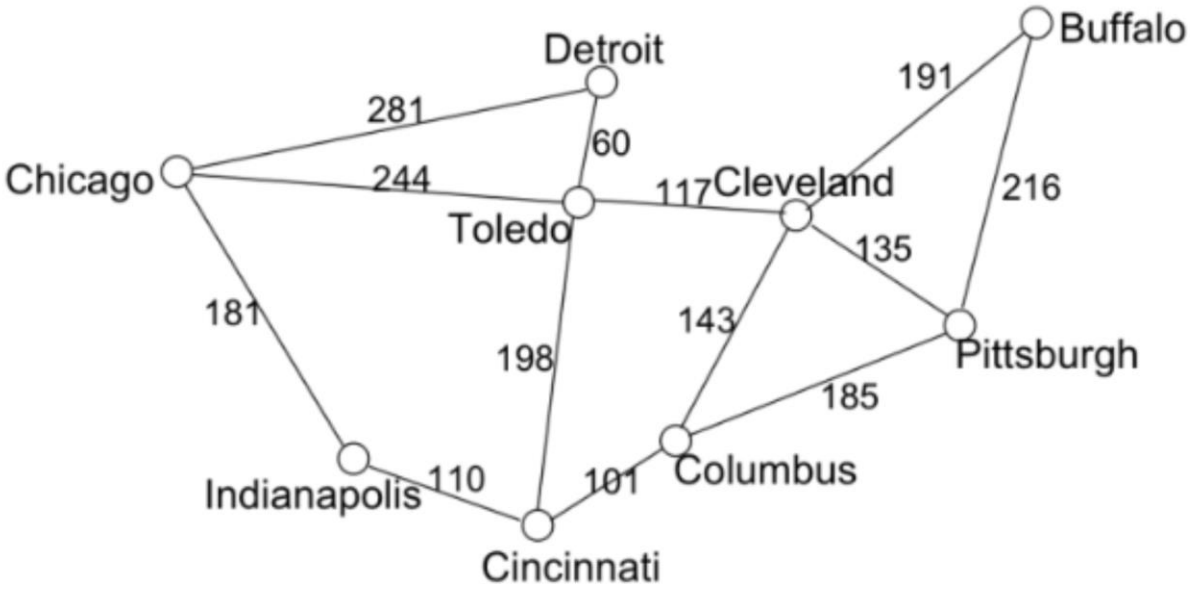


4) Draw the spanning tree created form using DFS and BFS for the graph below. . . use alphabetic order and start at A



5) Do DFS and BFS from Detroit





Check for Understanding

6) Fill in Blanks for DFS

```
public void dfTrav(Vertex v, Vertex parent) {
        System.out.println(v.id);
        v.encounter = [          ];
        v.parent = [          ];
        Edge e = v.edges;
        while (e != null) {
          Vertex w = e.end;
          if (w.encounter != [          ])
                dfTrav(w, v);
          e = e.next; }
}
```

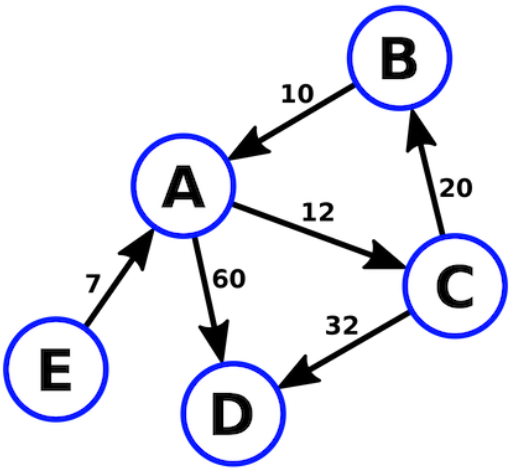
7) Fill in blanks for BFS

```
public void bfTrav(Vertex origin) {
        origin.encountered = [          ];
        origin.parent = null;
        Queue q = new LLQueue();
        q.insert(origin);
        while (!q.isEmpty()) {
                Vertex v = q.remove();
                System.out.println(v.id);    // Visit v. /
        // Add v's unencountered neighbors to the queue
                Edge e = v.edges;
                while (e != null) {
                        Vertex w = e.end;
                        if (w.encountered != [          ]) {
                                w.encountered = true;
                                w.parent = v;
                                q.[          ]; }
                        e = e.next; }
        }
}
```

8) What is the Big O of graph traversal if it is sparse and dense

Test Like Question

For the graph below



a) Draw the matrix version of the graph (the 2d array)

b) Run DFS and show the spanning tree created, use alphabetic order starting at E

Bonus Question (real interview question I got) → Answer at a high level, no need to give code just a conceptual algorithm to solve the problem

You have an array of every 5 letter word, a starting 5 letter word, and an ending 5 letter word

Your task is to devise an algorithm to make a chain from the start word to the end word where every word in that chain is only one letter apart from the previous

Here is an example: start: leave end years

Leave → lease → leare → learn →yearn → years