REVIEW: GRAPH TRAVERSAL INTERVIEW PRACTICE

Problem Solving with Computers-II

```
#include <iostream>
using namespace std;

int main(){
    cout<<"Hola Facebook\n";
    return 0;

}
```

How is PA 2 going?

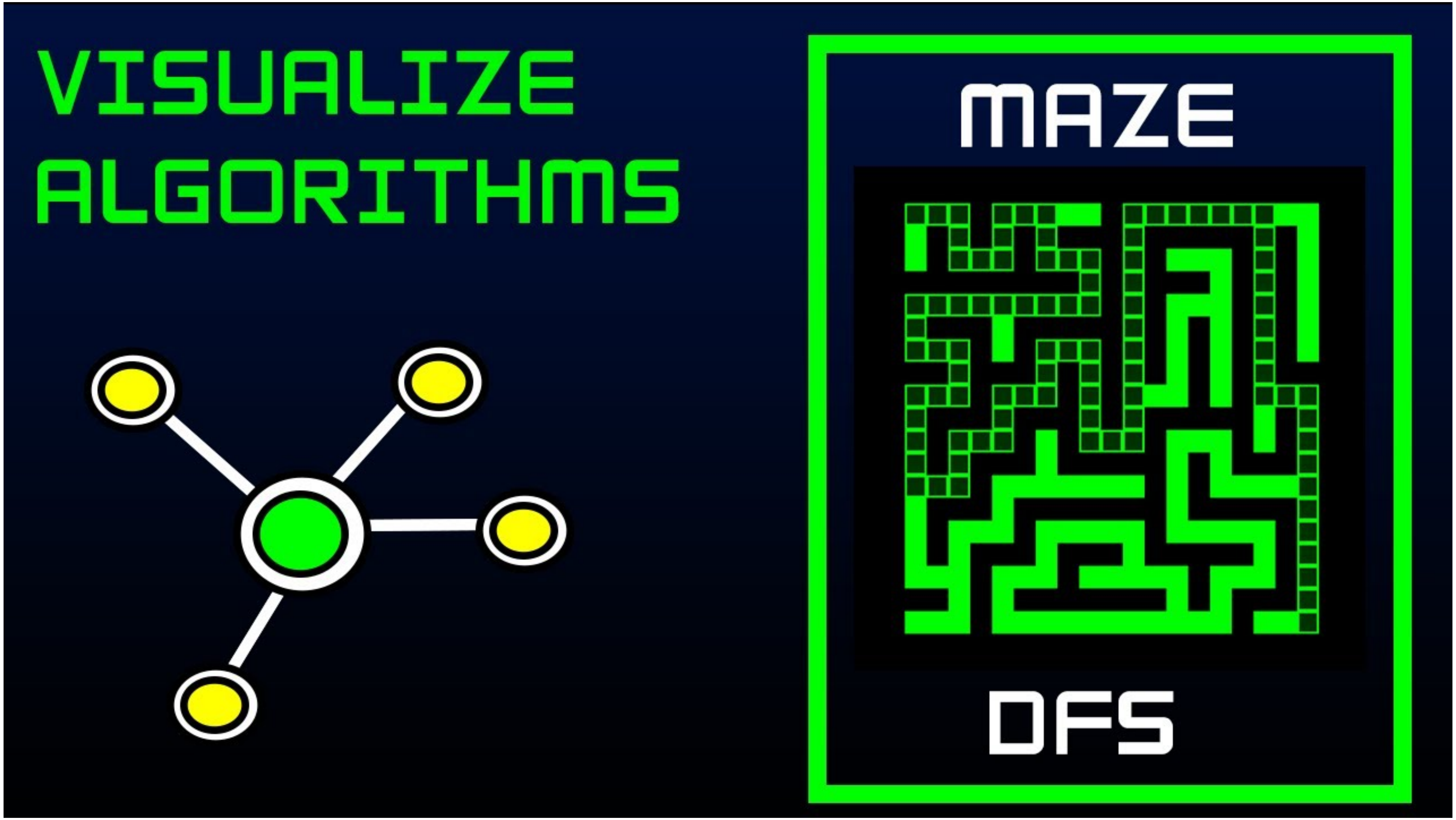
A. Done! **B.** On track to finish and having fun. **C.** On track to finish but struggling (a bit).

D. Falling behind and struggling a lot. **E.** Haven't read the assignment.

I can deal with pressure, and deadlines.

DO YOU HAVE AN IDEA FOR YOUR STORY YET?
NO, I'M WAITING FOR INSPIRATION.
YOU CAN'T JUST TURN ON CREATIVITY LIKE A FAUCET. YOU HAVE TO BE IN THE RIGHT MOOD.
WHAT MOOD IS THAT?
LAST-MINUTE PANIC.

Maze Solver



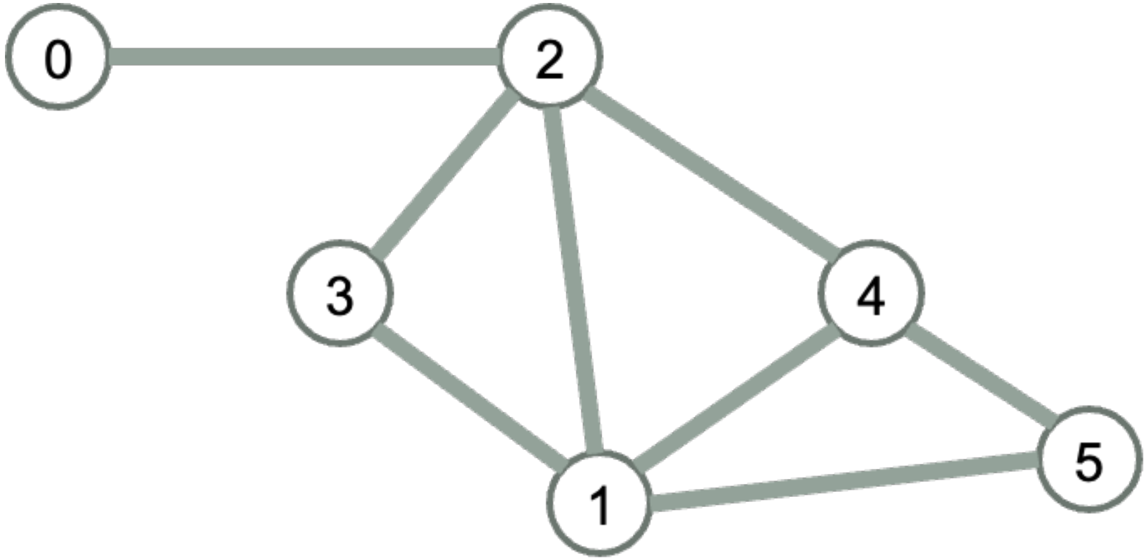
We’ll work to complete the handout (from previous lecture)

```
class graph{
  public:
     graph(int n = 0) { // n is the number of vertices
        adjList = vector<list<int>>(n);
     }
     Other public functions
     // (New!) Implement depth-first search
    // (New!)  Implement a variation of BFS that computes the shortest path
                  from a source vertex to all vertices reachable from it
   private:
     vector<list<int>> adjList;
};
```

Link to hand out: https://bit.ly/CS24-Graph-SearchHandout

Review: Graph Search

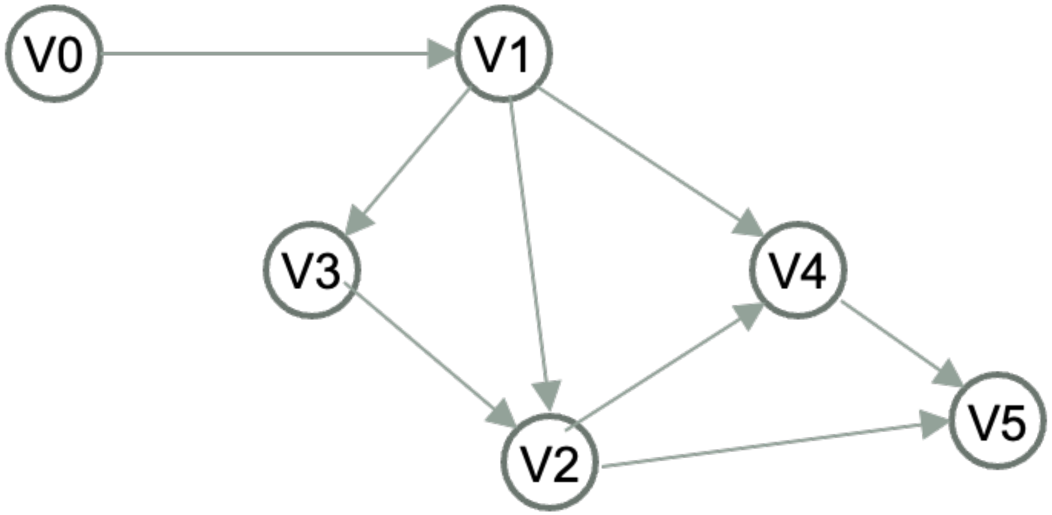
Depth First Search: Search as far down a single path as possible, backtrack as needed



Visualize DFS and BSF algorithms at https://visualgo.net/en/dfsbfs
Then, code each algo and analyze run time and space complexity

Depth First Search

Search as far down a single path as possible, backtrack as needed



Assuming DFS chooses the lower number node to explore first, in what order does DFS visit the nodes in this graph?

A. V0, V1, V2, V3, V4, V5
B. V0, V1, V3, V4, V2, V5
C. V0, V1, V3, V2, V4, V5
D. V0, V1, V2, V4, V5, V3

BFS Traverse

```
Input: Graph G = (V, E), source vertex s, Let n = |V|, m = |E|
Start at source s;
Mark all the vertices as “not visited”
Mark s as visited
push s into a queue
while the queue is not empty:
   - pop the vertex u from the front of the queue
     - for each of u’s neighbor (v)
          - If v has not yet been visited (v):
               • Mark v as visited
               • Push v in the queue
```

What is the time complexity of BFS?

BFS Traverse: Time Complexity (express in terms of n, m)

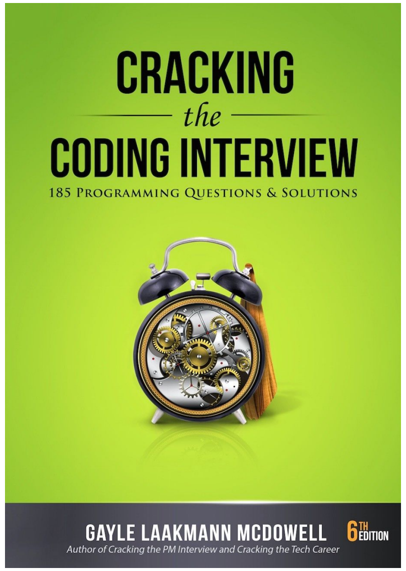
```
Input: Graph G = (V, E), source vertex s, Let n = |V|, m = |E|
Start at source s;
Mark all the vertices as “not visited”
Mark s as visited
push s into a queue
while the queue is not empty:
   - pop the vertex u from the front of the queue
     - for each of u’s neighbor (v)
          - If v has not yet been visited (v):
               • Mark v as visited
               • Push v in the queue
```

What is the time complexity of BFS?

A. O(n)
B. O(m)
C. O(n + m)
D. O(n^2)
E. None of the above

- How many times does the while loop run?
- How many times do we check if a vertex has been visited?

Tips for Technical Interviews

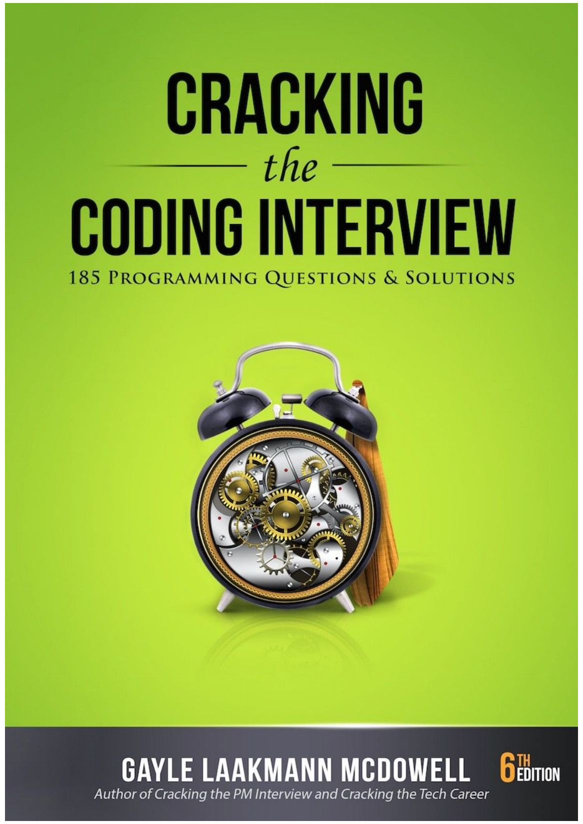

CRACKING
the
CODING INTERVIEW
185 Programming Questions & Solutions
GAYLE LAAKMANN MCDOWELL
6TH EDITION
Author of Cracking the PM Interview and Cracking the Tech Career

1. Listen carefully
2. Draw an example
3. State the brute force or a partially correct solution
   - then work to get at a better solution
4. Optimize:
   - Make time-space tradeoffs to optimize runtime
   - Precompute information: Reorganize the data e.g. by sorting
5. Solidify your understanding of your algo before diving into writing code.
6. Start coding!

Interview practice!

Write a ADT called minStack that provides the following methods

- push() // inserts an element to the “top” of the minStack
- pop() // removes the last element that was pushed on the stack
- top () // returns the last element that was pushed on the stack
- min() // returns the minimum value of the elements stored so far


CRACKING
the
CODING INTERVIEW
185 PROGRAMMING QUESTIONS & SOLUTIONS
GAYLE LAAKMANN MCDOWELL
6TH EDITION
Author of Cracking the PM Interview and Cracking the Tech Career

Practice the interview tips:

- Draw/solve a small example! (2 min)
  - Think of the most straightforward approach (1 min)
  - Evaluate its performance (1 min)
  - Think of another approach and evaluate it (5 min)
    - Can you trade off space/memory for better runtime?
- Pick the most promising approach and start coding! (10 min)