

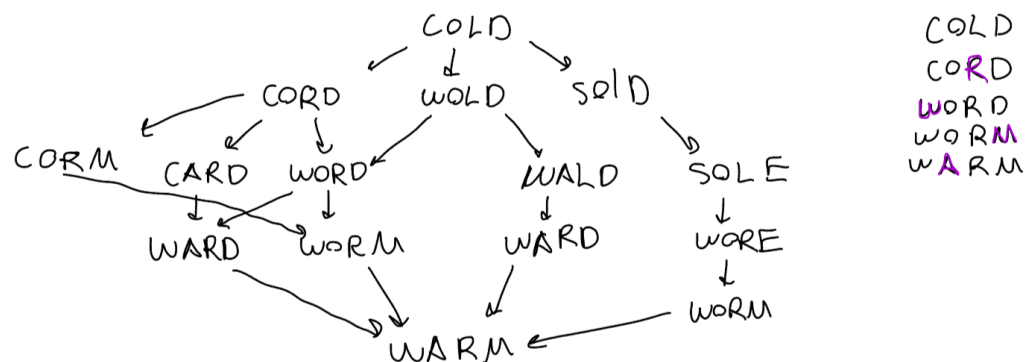
1: Graph Transversal

Define the Following:

- DFS:
- BFS:

1. Word Ladder is a game, where you are given a starting word and a target word, and you try to transform the starting word into the target word one letter at a time. Each transition word must be a real word in the dictionary. For example, a word ladder from "cold" to "warm" could be "cold-cord-word-worm-warm". Imagine there is a large graph of all words of length 4, with an edge connecting each word that differs by one letter. If given the location of your starting word, design an algorithm that finds the target word in the fewest steps possible.

COLD → WARM



2. Give an algorithm to detect whether a given undirected graph contains a cycle. If the graph contains a cycle, then your algorithm should output one. (It should not output all cycles in the graph, just one of them.) The running time of your algorithm should be $O(m + n)$ for a graph with n nodes and m edges.
3. We have a connected graph $G = (V, E)$, and a specific vertex $u \in V$. Suppose we compute a depth-first search tree rooted at u , and obtain a tree T that includes all nodes of G . Suppose we then compute a breadth-first search tree rooted at u , and obtain the same tree T . Prove that $G = T$. (In other words, if T is both a depth-first search tree and a breadth-first search tree rooted at u , then G cannot contain any edges that do not belong to T .)

2: Solutions

Define the Following:

- DFS: Depth First Search. You start with an arbitrary node, and keep following its children down as far as possible. Once you cannot go further, you backtrack until you find an unexplored path, and then explore it to its full depth. $O(V+E)$

- BFS: Breadth First Search. You choose an arbitrary node as a root, and then explore one edge distance, or "level" at a time. You explore the graph one depth level at a time. $O(V+E)$
1. *Word Ladder*: We would solve this using BFS. First pitch DFS to the students, and have them argue why or why not to use it. DFS may reach the target word more directly, but it could also go off on a tangent that steers very far from the target word. Using BFS, we can find the shortest path from "Cold" to "Warm"
 2. *Detect Cycles* You can use either DFS or BFS for this solution. I will construct a cycle-finding algorithm using BFS: First present this answer, and see if students can spot the mistake:

```
def find_cycle(G=(V, E)):
    Choose an arbitrary node S as the root, and start BFS of G
    At every level of BFS, check and see if there are edges between
        nodes of the same level
    If there is an edge (u,v) between two nodes of the same level, then
        there is a cycle. Use a path-finding algorithm between u and v
        excluding the edge (u,v)) to find the path of the cycle.
    Return the cycle.
    Return None.
```

The issue with this algorithm is that it only detects odd-length cycles. Add this line of code to also include even-length cycles:

```
If 2 or more nodes from the previous level converge into one node at the
    current level, there is a cycle. Use a path-finding algorithm between the
    nodes from the previous level to find the path of the cycle.
    Return the cycle.
```

Here is a solution using DFS:

```
def find_cycle(G=(V, E)):
    Run DFS on the entire graph G. Let T = the DFS tree.
    If G = T, then return None.
    Else, find an arbitrary edge (u, v) that is not in T, and find the path from
    u to v in T. The path from u to v in T, plus the edge (u, v) creates a cycle.
    Return the cycle.
```

3. *BFS = DFS tree implies graph is tree*. We will give a solution based on the theorems proved in class when BFS and DFS were first introduced.

Suppose G is not a tree. Then there must exist an edge $(x, y) \in G$ but $(x, y) \notin T$, where T is both a BFS and DFS tree. WLOG, let's say that x is reached first in the DFS. So x must be an ancestor of y in T . But, $(x, y) \notin T$, so we know that x cannot directly be the parent of y . This implies that x must be

further than one layer away from y . However, this contradicts the fact that T is a BFS tree, because in a BFS tree, edges never jump layers. Thus, we know that G is a tree. But if G is a tree, and T a subtree of G that reaches all of the nodes, then $G = T$, completing the proof.