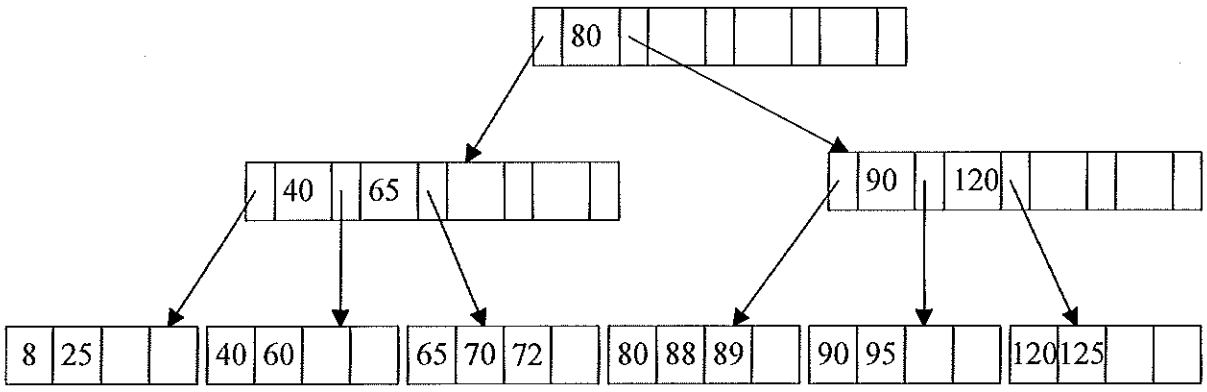
Situation		B+ Tree Insertion Algorithm
Data Page Full?	Parent Index Page Full?	
No	No	Place record in sorted position in the appropriate data page.
Yes	No	1. Split data page with records < middle key going in left data page and records ≥ middle key going in right data page. 2. Place middle key in index page in sorted order with the pointer immediately to its left pointing to the left data page and the pointer immediately to its right pointing to the right data page.
Yes	Yes	1. Split data page with records < middle key going in left data page and records ≥ middle key going in right data page. 2. Adding middle key to parent index page causes it to split with keys < middle key going into the left index page, keys > middle key going in right index page, **and** the middle key inserted into the next higher level index page. If the next higher index page is full continue to splitting index pages up the B+ tree as necessary.

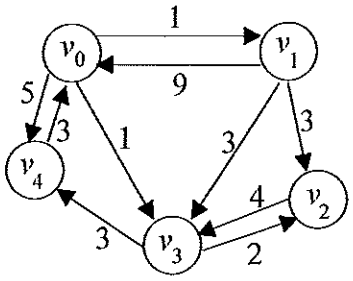
The deletion algorithm for a B+ tree is summarized by the below table.

Situation		B+ Tree Deletion Algorithm
Data Page Below Fill Factor?	Parent Index Page Below Fill Factor?	
No	No	Delete record from the data page. Shifting records with larger keys to left to fill in the hole. If the deleted key appears in the index page, use the next key to replace it.
Yes	No	1. Combine data page and its sibling. Change the index page to reflect the change.
Yes	Yes	1. Combine data page and its sibling. 2. Adjusting the index page to reflect the change causes it to drop below the fill factor, so combine the index page with its sibling. 3. Continue combining the next higher level index pages until you reach an index page with the correct fill factor or you reach the root index page.

0. Consider an B+ tree example with b = 5 and 50% fill factor. Delete 89, 65, and 88. What is the resulting B+ tree?



1. Consider the following directed graph (diagraph) $G = \{ V, E \}$:



a) What is the set of vertices? $V =$

b) An edge can be represented by a tuple (from vertex, to vertex [, weight]). What is the set of edges?

$E =$

c) A path is a sequence of vertices that are connected by edges. In the graph G above, list two different pathes from v_0 to v_3.

d) A cycle in a directed graph is a path that starts and ends at the same vertex. Find a cycle in the above graph.

2. Like most data structures, a graph can be represented using an array, or as a linked list of nodes.
a) The array representation is called an *adjacency matrix* which consists of a two-dimensional array (matrix) whose elements contain information about the edges and the vertices corresponding to the indices.

Complete the following adjacency matrix for the above graph.

(from vertex) \ (to vertex)	v_0	v_1	v_2	v_3	v_4
v_0					
v_1					
v_2					
v_3					
v_4					

3. The linked representation maintains a list of vertices with each vertex maintaining a list of other vertices that it connects to. Draw the adjacency list representation below:

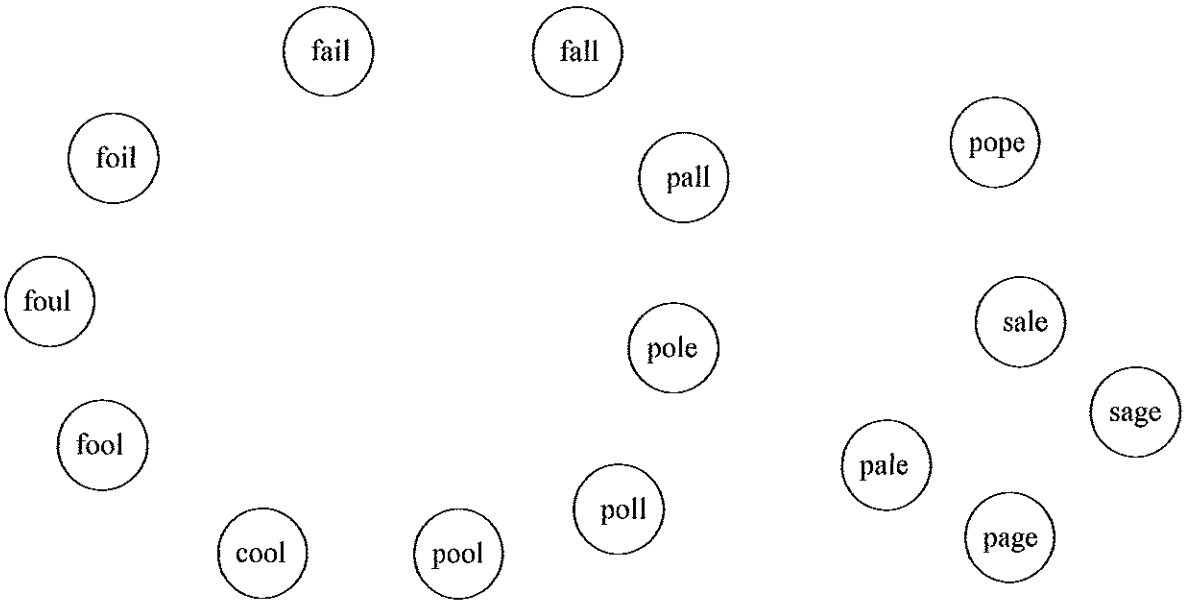
4. Graphs can be used to solve many problems by modeling the problem as a graph and using "known" graph algorithm(s). For example, consider the *word-ladder puzzle* where you tranform one word into another by changing one letter at a time, e.g., transform FOOL into SAGE by FOOL → FOIL → FAIL → FALL → PALL → PALE → SALE → SAGE.

We can use a graph algorithm to solve this problem by constructing a graph such that

- a word represents a vertex
- an edge represents?

- a word ladder transformation from one word to another represents?

5. For the words listed below, draw the graph of question 4



a) List a different transformation from FOOL to SAGE

b) If we wanted to find the shortest transformation from FOOL to SAGE, what does that represent in the graph?

c) There are two general approaches for traversing a graph from some starting vertex *s*:

- Breadth First Search (BFS) where you find all vertices a distance 1 (directly connected) from s, before finding all vertices a distance 2 from s, etc.
- Depth First Search (DFS) where you explore as deeply into the graph as possible. If you reach a "dead end," we backtrack to the deepest vertex that allows us to try a different path.

Which of these traversals would be helpful for finding the **shortest** solution to the word-ladder puzzle?