1. B+ Trees are balanced trees such that:
   - Same height for paths from root to leaf
   - Given a search-key K, nearly same access time for different K values

B+ Tree is constructed by parameter $n$
   - Each Node (except root) has $\lceil n/2 \rceil$ to $n$ pointers
   - Each Node (except root) has $\lceil n/2 \rceil - 1$ to $n-1$ search-key values

Suppose $n = 1,024$. What would be the maximum height of a B+ tree with 8 billion keys?

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

   ![Diagram of the graph]

   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 paths 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.

3. 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.

   $$
   \begin{array}{cccccc}
   & v_0 & v_1 & v_2 & v_3 & v_4 \\
   v_0 & & & & & \\
v_1 & & & & & \\
v_2 & & & & & \\
v_3 & & & & & \\
v_4 & & & & & \\
   \end{array}
   $$

   (to vertex)
4. 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:

5. In the word ladder puzzle you transform 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?

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

```
fool  fool  foul  pool  cool
foil  fall  fail  pale
pall  pope  sale  sage
```

Lecture 26 Page 2
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 would be helpful for finding the shortest solving the word ladder puzzle?

7. What data structure would be helpful in each type of search? Why?
a) Breadth First Search (BFS):

b) Depth First Search (DFS):
8. Consider the following directed graph (diagraph) $G = \{ V, E \}$ with adjacency matrix $W$:

\[
\begin{pmatrix}
0 & 1 & 3 & 4 \\
2 & 0 & 3 & 3 \\
1 & \infty & 0 & 4 & \infty \\
3 & \infty & 2 & 0 & 3 \\
4 & 3 & \infty & \infty & 0
\end{pmatrix}
\]

In Dijkstra’s Algorithm two arrays $[1..(n-1)]$ are used:

- $\text{length}[i] = \text{length of current shortest path from } v_0 \text{ to } v_i \text{ using only vertices in } Y \text{ as intermediates}$
- $\text{touch}[i] = \text{index of last vertex in } Y \text{ on current shortest path from } v_0 \text{ to } v_i$

Initially, the length and touch arrays are shown below. Complete the trace of the algorithm.

\[
\begin{align*}
\text{length:} & \quad \begin{array}{cccc}
1 & 2 & 3 & 4 \\
1 & \infty & 1 & 5 \\
1 & 2 & 3 & 4 \\
\end{array} \\
\text{touch:} & \quad \begin{array}{cccc}
1 & 2 & 3 & 4 \\
0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
\end{array}
\end{align*}
\]

Algorithm:

1) Find smallest nonneg. value in length (next closest vertex to $v_0$ that’s not in $Y$).
2) Update lengths now that this vertex is in $Y$.
3) Update touch accordingly if you find a shorter path.
4) Mark this vertex in $Y$ (make its length value -1).