Intro to BFS

BFS (breadth-first-search) is a way to search (or traverse) a tree data structure. This tree does *not* have to be binary. Each node can have any number of children. (Other than that it's just like a binary tree). Examples include: the files in a directory structure on your computer (nodes are folders, children are files or folders inside that folder), a family tree (children are literal children, although if we include marriages, this isn't quite a proper tree anymore); and the friend tree (first level is all your close friends, then second level is all their close friends, etc. To be a tree you only list each person once)

BFS can be implemented with a queue. Starting at the root, we add all the children of the root to a queue. Then we start popping elements off the (front of) the queue and pushing that node's children to the (end of) the queue. Repeat until the queue is empty. Here's an algorithm

```
function BFS(root):
  Queue q = new Queue()
  q.push(root)
  while (q is not empty) do:
    current = q.pop()
    for each child in current.children do:
        q.push(child)
```

Example

Make a depth 3 permutation tree for {A,B,C,D}. The root of a permutation tree is an empty string. Each node is a string containing a subset of {A,B,C,D} in some order. The children of a node s containing a string of length n are all strings on $n+1$ created by appending a character *not* already in s onto the end of s.

Now trace BFS starting at the root of the tree you just made. Show the contents of the stack after each push and pop.

Coding

Code up this BFS algorithm in Java.