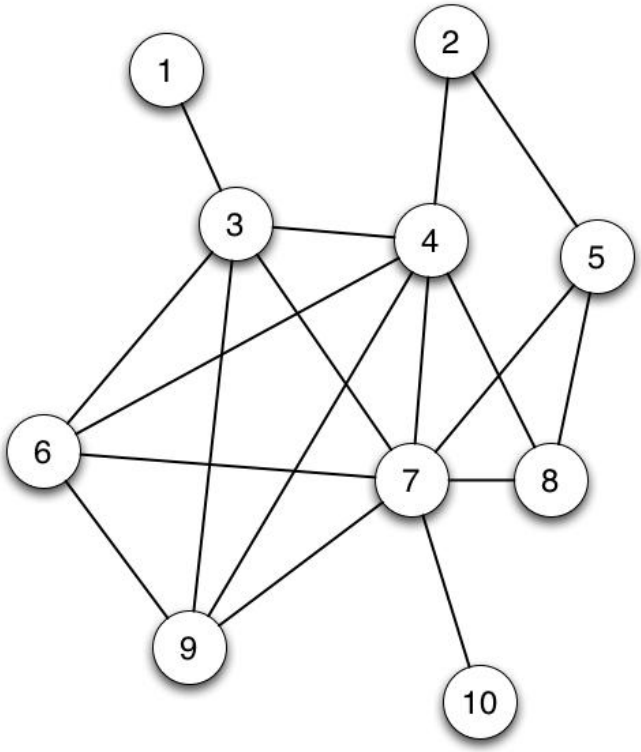
22C:16 (CS:1210) Quiz 9

You have 20 minutes to complete this quiz. This quiz depends on the `searchWordNetwork` function that was part of the program `playLaddersGame2.py`. Code from this function appears on the back of this page.



Consider the network of "words" shown above. Suppose that we call the function `searchWordNetwork` on this word network with source "`2`" and target "`10`".

1. Show the contents of the `reached` dictionary and the `processed` dictionary at the beginning of each iteration of the while-loop in `searchWordNetwork`. Make sure you show the key-value pairs in each dictionary and not just a list of keys. Assume that each time we pull an element out of `reached` using `popitem()`, *we get the element that is numerically largest.*

2. Following up on Problem 1, show the contents of the `processed` dictionary, when it is returned from `searchWordNetwork`.

```
def searchWordNetwork(source, target, D):

    # Initialization: processed and reached are two dictionaries that will help in the
    # exploration.
    # reached: contains all words that have been reached but not processed.
    # processed: contains all words that have been reached and processed, i.e., their neighbors
    # have also been explored.
    processed = {source:0}
    reached = {}
    for e in D[source]:
        reached[e] = source # the value in the dictionary of a key k is the "parent" of k

    # Repeat until reached set becomes empty or target is reached
    while reached:
        # Check if target is in reached; this would imply there is path from source to target
        if target in reached:
            processed.update({target:reached[target]})
            return processed

        # Pick an item in reached and process it
        item = reached.popitem() # returns an arbitrary key-value pair as a tuple
        newWord = item[0]
        parent = item[1]

        # Find all neighbors of this item and add new neighbors to reached
        processed[newWord] = parent
        for neighbor in D[newWord]:
            if neighbor not in reached and neighbor not in processed:
                reached[neighbor] =  newWord

    return {}
```
