```
start state, delta transitions [], alphabet [], accepting state, states[] are global


main (){
    find shortest super string
    find min in stored array and return it
}
Find shortest super string (current state, string w, current index, retval)
{
    if (current state == accepting state && current index == w.length)
    {
        store retval in some possible result array
    }
    else
    {
        for every character c in the alphabet for the delta transitions for current
state
        {
            if(C == w[current index])
                current index ++;
                retval += c
            else
                retval += c

            findshortestsuperstring(delta(current state, c), w, current index,
retval)
        }
    }
}
```

HW 7

Due on **Wednesday, March 13, 2019 at 10am**

CS/ECE 374: Algorithms & Models of Computation, Spring 2019 Version: **1.21**

Unless stated otherwise, for all questions in this homework involving dynamic programming, you need to provide a solution with explicit memoization.

Worked with Jeremy Varghese.

19 (100 PTS.) Superstring theory.

You are given a DFA $M = (Q, \Sigma, \delta, s, A)$ with n states, where $|\Sigma| = O(1)$.

For two strings $x, y \in \Sigma^*$, the string x is a ***superstring*** of y, if one can delete characters from x and get y.

Let w be a given input string with m characters.

19.A. (25 PTS.) Let $q, q' \in Q$ be two states of M. Prove, that the shortest string w'', such that $\delta^*(q, w'') = q'$ is of length at most $n - 1$.

19.B. (25 PTS.) Prove, that if there is a superstring x of w, such that x is accepted by M, then the shortest such superstring is of length at most $(n + 1)(m + 1)$, where $n = |Q|$ and $m = |w|$.

19.C. (50 PTS.) Describe an algorithm, as efficient as possible, that computes the shortest superstring z of w, such that z is accepted by M. (One can solve this problem using dynamic programming, but this is definitely not the only way.)

19

Solution:

1. We will prove by induction that the shortest string w" such that w" can be an input from state q to reach q' has length at most n-1.

   Inductive Variable: We will induct on n, where n is the number of states in the DFA

   Base case: Our base case is that n = 1, therefore q = q'. With this, we assume the number of transitions within our DFA to get from q to q' for any string w" is n - 1 = 0.

   Inductive Hypothesis: Let a DFA M' be defined by Q', Σ, δ', s', A' have k states where k is an integer such that 0 < k < n. Assume that to get from q to q' with input string x, it is true such that the number of transitions required is less than k-1.

   Inductive Step: Using the DFA in the inductive hypothesis, we assume the DFA we now iterate on is made up of a subset of states Q - q, We will iterate on the first character in w" and use a single transition away from q in the path to q'. Therefore the number of states remaining needed to get to q' are now (k-1)-1. The necessary states to then get to q' are now k-2. We continue with the ith character of w" where iterator i will go through all of w". The number of transitions needed to get to q' is k-2-i until k = 0. In the beginning, we began with k states which is already less than n so by definition, k <= n-1. All other iterations require less than k states therefore we prove that the input states for w" to get from q to q' are at most n-1.

2. We will do a direct proof to prove if a superstring x of w such that such that x is accepted by M, then the shortest such superstring is of length at most (n + 1)(m + 1), where n = |Q| and m = |w|.

   There exists two cases:

   Case 1: M accepts w which implies w is equal to x. if w = x, the number of states required to represent all characters of x is n. We also know m = |w| from the given statement. In this case, by iterating over the entirety of x in the DFA M, we will iterate over n states. Because x = m, n = |w| = m therefore the operation is bound simply by m. This is less than our upper bound of (n+1)(m+1).

   Case 2: w is a subsequence of x where w != x. Therefore M is not required to accept w. we also know |w| < |x| by definition of subsequence and the above statement. for w qualify as a subsequence of x that exists, DFA M must iterate over at least |w| states which is equivalent to m states. To then get to an accepting state, we must encompass all of x, therefore we must transition between at most n-1 states proven from the first part of this question if q is deemed the state that we left off of after iterating over all of w and q' is deemed the accepting state. This way we are bounded by O(n-1)(m) because of the requirement that w is only a subsequence of x therefore any number of characters can exists in x between subsequent characters of w within x. This is less than the requirement (n + 1)(m + 1) stated in the problem.

3. See attached page for pseudo code. The approach of the algorithm depends on the given the DFA's components of a start state, delta transitions in an array, the alphabet character array, accepting state, and all states as arguements. In our first main function, we will call our recursive function first to find all shortest superstring possibilities and store them in a array. To do so we must iterate through all DFA states taking O(n) time. In each state, we recursively call a case for each character in the alphabet. Because we know |Sigma| = O(1), this takes O(1) time. Through each recursive call, we only store a super string as a potential answer if every character in w is satisfied making it a valid super string. Ensuring this has a bound of O(m). Binding these two together because any character may appear in between characters of w, this makes the total run time of the algorithm O(nm).
