Homework 4

Evan Waldmann
COP 3503H

August 30, 2018

Problem 1. Give an efficient dynamic programming algorithm to find the longest palindrome that is a sub-sequence of a given input string.

(a) Characterize longest palindrome.
The longest palindrome will be the longest common sub-sequence that appears the same when read in reverse. The length of a sub-sequence that is a palindrome can be characterized by,

$$\text{table}[i,j] = \begin{cases} 1 & i = j \\ 2 & i = j-1 \ \& \ \text{str}[i] = \text{str}[j] \\ \text{table}[i+1, j-1] + 2 & i < j-1 \ \& \ \text{str}[i] = \text{str}[j] \\ MAX(\text{table}[i, j-1], \text{table}[i+1, j]) & i < j-1 \ \& \ \text{str}[i] \neq \text{str}[j] \end{cases}$$

(b) Define a recursive solution.

```
1  function rec-Sol(str[1..n],i,j)
2  if i == j then
3  |  return 1
4  if str[i]== str[j] then
5  |  if i == j − 1 then
6  |  |  return 2
7  |  else
8  |  |  return rec-Sol(str, i + 1,j − 1) +2
9  else
10 |  return MAX(rec-Sol(str, i + 1,j) , rec-Sol(str, i,j − 1))
11 end
```

(c) Compute the length of a longest palindrome sub-sequence.

```
1  function Longest-Palindrome(str[1..n])
2  Let table[1..n,1..n] be a two dimensional array
3  for r ← 0 to n do
4  |  table[r][r] ← 1
5  end
6  for i ← 2 to n do
7  |  for r ← 0 to n − i + 1 do
8  |  |  c ← r + i − 1
9  |  |  if str[r]== str[c] & i == 2 then
10 |  |  |  table[r][c] ← 2
11 |  |  else if str[r] == str[c] then
12 |  |  |  table[r][c] ← table[r + 1][c − 1] +2
13 |  |  else
14 |  |  |  table[r][c] ← MAX(table[r][c − 1] , table[r + 1][c])
15 |  |  end
16 |  end
17 end
18 return table[0][n − 1]
```

(d) Construct a longest palindrome sub-sequence. (assume that table[0][$n-1$] is even)

```
1  function Generate-LPS(str[1..n], table[1..n,1..n])
2  Let result be an empty string
3  end ← table[0][n − 1]
4  r ← 0
5  c ← n − 1
6  while end≥ 0 & r ≤ c do
7  |  if str[r] == str[c] then
8  |  |  result ∪ str[r]
9  |  |  end−−
10 |  |  r + +
11 |  |  c − −
12 |  else
13 |  |  if table[r + 1][c] > table[r][c − 1] then
14 |  |  |  r + +
15 |  |  else
16 |  |  |  c − −
17 |  |  end
18 |  end
19 end
20 r ← 0
21 m ← table[0][n − 1]/2
22 c ← n − 1
23 while c ≥ m do
24 |  result[r] ← result[c]
25 |  r + +
26 |  c − −
27 end
28 return result
```

(e) The running time of the algorithm is $\Theta(n^2)$.

Problem 2. Java implementation of Longest Common Palindrome Sub-sequence with examples.

```
// Evan Waldmann
// COP 3503H
// 10/15/17

public class LCSpalindrome {

   public static void printouttable(int[][] arr)
   {
      for (int i=0; i<arr.length; i++)
      {
         for (int j=0; j<arr[0].length; j++)
         {
            System.out.print(arr[i][j] + " ");
         }
         System.out.println();
      }
      System.out.println("END");
   }

   public static String generateLengthTable(char[] str)
   {
      int n = str.length;
      int r, c, i;
      int[][] table = new int[n][n];

      for (r = 0; r < n; r++){
         table[r][r] = 1; //single chars are palindromes of length 1
      }

      for (i=2; i<=n; i++)
      {
         for (r=0; r<n-i+1; r++)
         {
            c = r+i-1;
            if (str[r] == str[c] && i == 2){
               table[r][c] = 2;
            }
            else if (str[r] == str[c]) {
               table[r][c] = table[r+1][c-1] + 2;
            }
            else
            {
               if (table[r][c-1] > table[r+1][c])
                  table[r][c] = table[r][c-1];
               else
                  table[r][c] = table[r+1][c];
            }
         }
      }

      // printouttable(table); //test to see if table is working
      return printStringFromTable(str, table);

   }

   private static String printStringFromTable(char[] a, int[][] table)
   {
      int len = a.length;
      int end = table[0][len-1]; // the longest palindrome's length
      char result[] = new char[end+1];
```

```
   result[end] = '\0';//null terminator
   end--;

   int r = 0;
   int c = len - 1;
   while(end >= 0 && r <= c)
   {
      if(a[r] == a[c])
      {
         result[end] = a[r];
         end--;
         r++;
         c--;
      }
      else
      {
         if(table[r+1][c] > table[r][c-1])
         {
            r++;
         }
         else
         {
            c--;
         }
      }
   }

   //even or odd length changes bounds of loop
   if(table[0][len-1]%2 == 0)
   {
      r=0;
      int mid = table[0][len-1]/2;
      c = result.length - 2;
      while(c >= mid)
      {
         result[r++] = result[c--];
      }
   }
   else
   {
      r = 0;
      int mid = table[0][len-1]/2;
      c = result.length - 2;
      while(c > mid)
      {
         result[r++] = result[c--];
      }
   }

   return (String.valueOf(result)).substring(0, table[0][len-1]);
   //null terminator was printing for some reason so I cut it off
}


public static void main(String args[])
{

   String str = "character";
   System.out.println("Longest commmon palindrome of "+str+" is "+
       generateLengthTable(str.toCharArray()));

   str = "BBABCBCAB";
```

```
        System.out.println("Longest commmon palindrome of "+str+" is "+
            generateLengthTable(str.toCharArray()));

        str = "aibohphobia";
        System.out.println("Longest commmon palindrome of "+str+" is "+
            generateLengthTable(str.toCharArray()));

        str = "racecar";
        System.out.println("Longest commmon palindrome of "+str+" is "+
            generateLengthTable(str.toCharArray()));

        str = "one";
        System.out.println("Longest commmon palindrome of "+str+" is "+
            generateLengthTable(str.toCharArray()));

        str = "palindrome";
        System.out.println("Longest commmon palindrome of "+str+" is "+
            generateLengthTable(str.toCharArray()));

        str = "input";
        System.out.println("Longest commmon palindrome of "+str+" is "+
            generateLengthTable(str.toCharArray()));

        str = "example";
        System.out.println("Longest commmon palindrome of "+str+" is "+
            generateLengthTable(str.toCharArray()));

        str = "sequence";
        System.out.println("Longest commmon palindrome of "+str+" is "+
            generateLengthTable(str.toCharArray()));
    }
}
```

Output from the code above:

```
Longest commmon palindrome of character is carac
Longest commmon palindrome of BBABCBCAB is BABCBAB
Longest commmon palindrome of aibohphobia is aibohphobia
Longest commmon palindrome of racecar is racecar
Longest commmon palindrome of one is o
Longest commmon palindrome of palindrome is p
Longest commmon palindrome of input is i
Longest commmon palindrome of example is exe
Longest commmon palindrome of sequence is eqe
```