Randy Van 88081986
Park Him Timothy Tse 14284355

CS 260P Project 2

I. Algorithm

The code takes the following steps to find the two binary strings that have the largest Longest Common Subsequence (LCS).

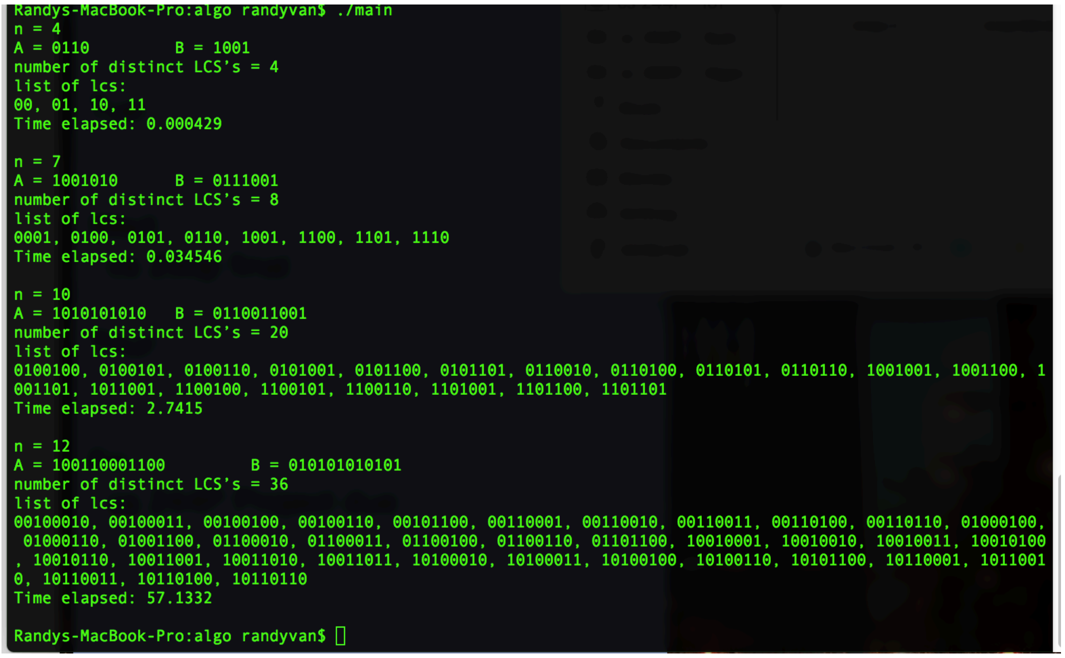
1. Generating binary strings of length n
2. Determine strings A and B by comparing largest number of unique LCS for each pairs of string
3. Finding the list of unique LCS for that pair of strings

This code uses *memoization* to find the number of distinct LCS, but uses *recursion* to find the actual unique LCS strings.

II. Analysis

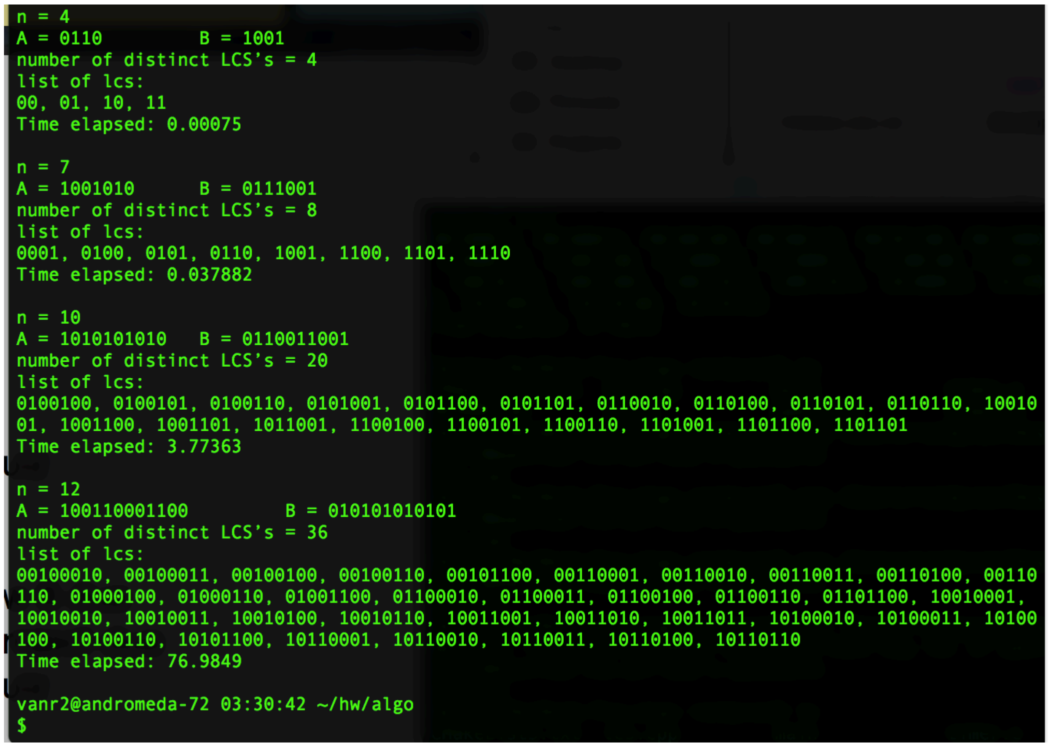
Component	Description	Time Complexity
Binary_string_permutation	Generate all strings	$O(2^n)$
Determined_string	Determine strings A and B	$O(n^2)$
Num_distinct_lcs	Find number of distinct lcs	$O(n^2)$
Lcs_len_vector	[n+1][n+1] vector of lcs length	$O(n^2)$
lcs	Find length of lcs	$O(n^2)$
Make_distinct_list	Make list of distinct lcs	$O(n^2)$
Overall :		$O(2^n)$

III. Results



```
Randys-MacBook-Pro:algo randyvan$ ./main
n = 4
A = 0110         B = 1001
number of distinct LCS's = 4
list of lcs:
00, 01, 10, 11
Time elapsed: 0.000429

n = 7
A = 1001010      B = 0111001
number of distinct LCS's = 8
list of lcs:
0001, 0100, 0101, 0110, 1001, 1100, 1101, 1110
Time elapsed: 0.034546

n = 10
A = 1010101010   B = 0110011001
number of distinct LCS's = 20
list of lcs:
0100100, 0100101, 0100110, 0101001, 0101100, 0101101, 0110010, 0110100, 0110101, 0110110, 1001001, 1001100, 1
001101, 1011001, 1100100, 1100101, 1100110, 1101001, 1101100, 1101101
Time elapsed: 2.7415

n = 12
A = 100110001100         B = 010101010101
number of distinct LCS's = 36
list of lcs:
00100010, 00100011, 00100100, 00100110, 00101100, 00110001, 00110010, 00110011, 00110100, 00110110, 01000100,
 01000110, 01001100, 01100010, 01100011, 01100100, 01100110, 01101100, 10010001, 10010010, 10010011, 10010100
, 10010110, 10011001, 10011010, 10011011, 10100010, 10100011, 10100100, 10100110, 10101100, 10110001, 1011001
0, 10110011, 10110100, 10110110
Time elapsed: 57.1332

Randys-MacBook-Pro:algo randyvan$ 
```

Figure 1: *Results from running on local machine*



```
n = 4
A = 0110         B = 1001
number of distinct LCS's = 4
list of lcs:
00, 01, 10, 11
Time elapsed: 0.00075

n = 7
A = 1001010      B = 0111001
number of distinct LCS's = 8
list of lcs:
0001, 0100, 0101, 0110, 1001, 1100, 1101, 1110
Time elapsed: 0.037882

n = 10
A = 1010101010   B = 0110011001
number of distinct LCS's = 20
list of lcs:
0100100, 0100101, 0100110, 0101001, 0101100, 0101101, 0110010, 0110100, 0110101, 0110110, 10010
01, 1001100, 1001101, 1011001, 1100100, 1100101, 1100110, 1101001, 1101100, 1101101
Time elapsed: 3.77363

n = 12
A = 100110001100         B = 010101010101
number of distinct LCS's = 36
list of lcs:
00100010, 00100011, 00100100, 00100110, 00101100, 00110001, 00110010, 00110011, 00110100, 00110
110, 01000100, 01000110, 01001100, 01100010, 01100011, 01100100, 01100110, 01101100, 10010001,
10010010, 10010011, 10010100, 10010110, 10011001, 10011010, 10011011, 10100010, 10100011, 10100
100, 10100110, 10101100, 10110001, 10110010, 10110011, 10110100, 10110110
Time elapsed: 76.9849

vanr2@andromeda-72 03:30:42 ~/hw/algo
$
```

Figure 2: *Results from running on Openlab*

IV. Discussion

Though the code produces correct results, the runtime is much longer than anticipated. With a Macbook Pro 2016, the results were found barely within the 60 second time limit. With the Openlab computers, runtime could vary from 80 seconds to 70 seconds.

The code could be further improved and optimized by streamlining results and reducing the number of calls to get one piece of information at a time. In the rush to meet deadlines, shortcuts were taken to hastily get data. By compacting results and reusing more function calls, the number of redundant LCS generations would be reduced.

The algorithm also be improved by filtering the two strings A and B instead of comparing each different pair. By following trends such as having different first and last digits, only a quarter of the total strings would be compared. Finding more trends and filters would significantly reduce runtime and singlehandedly put the code below the maximum 60 second runtime restriction.

V. Resources

1. https://comeoncodeon.wordpress.com/2009/11/13/number-of-distinct-lcs/
2. http://www.geeksforgeeks.org/printing-longest-common-subsequence/
3. http://www.geeksforgeeks.org/print-longest-common-sub-sequences-lexicographical-order/
4. http://www.thelearningpoint.net/computer-science/algorithms-dynamic-programming---longest-common-subsequence
5. https://www.codechef.com/wiki/tutorial-dynamic-programming
6. https://www.codechef.com/wiki/tutorial-lcs-problem-revisited
7. https://arxiv.org/pdf/cs/0301034.pdf