Randy Van 88081986

Park Him Timothy Tse 14284355

**CS 260P Project 2**

1. **Algorithm**

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

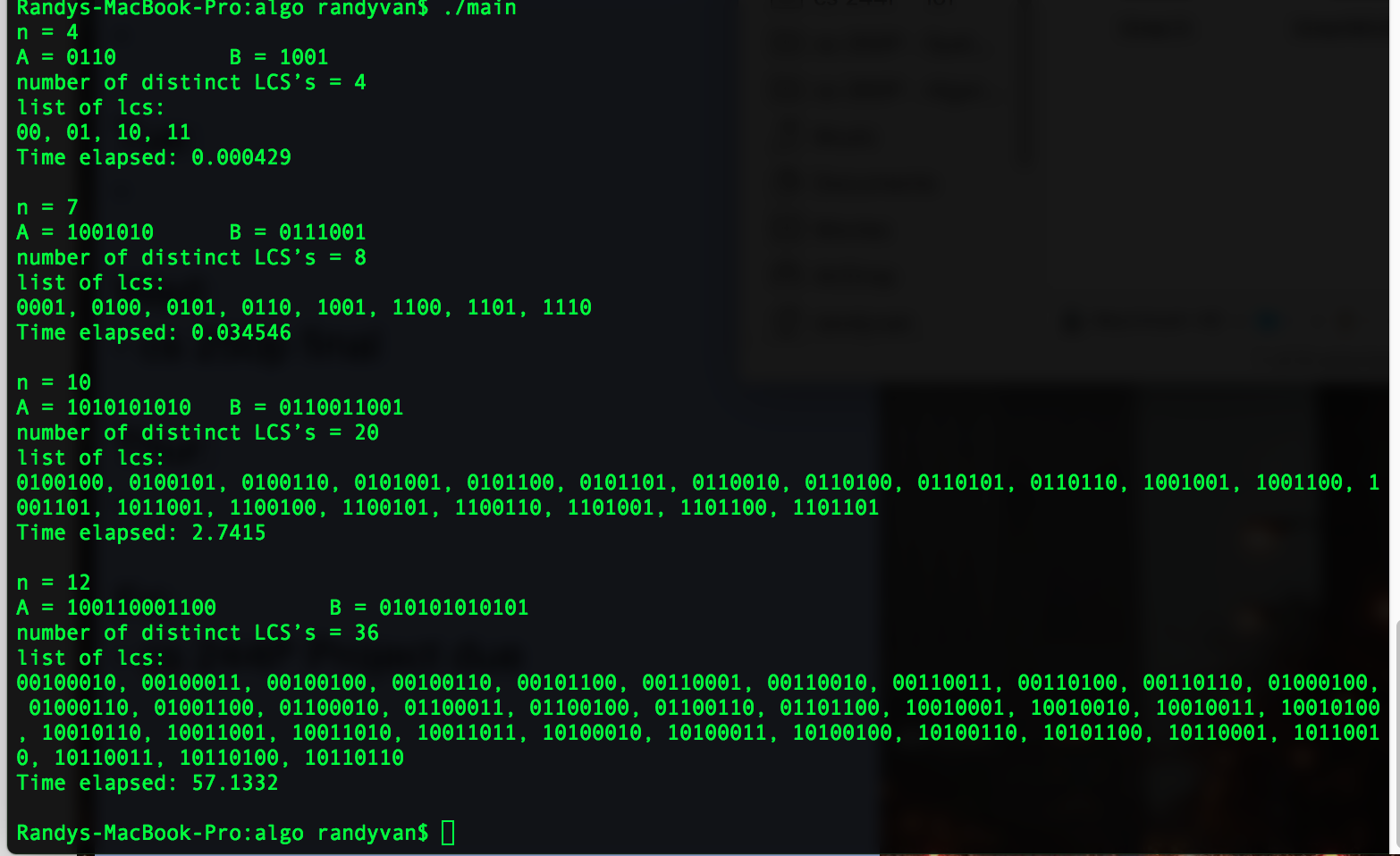
1. Generating binary strings of length n
2. Determine strings A and B by comparing largest number of unique LCS for each pairs of string
3. Finding the list of unique LCS for that pair of strings

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

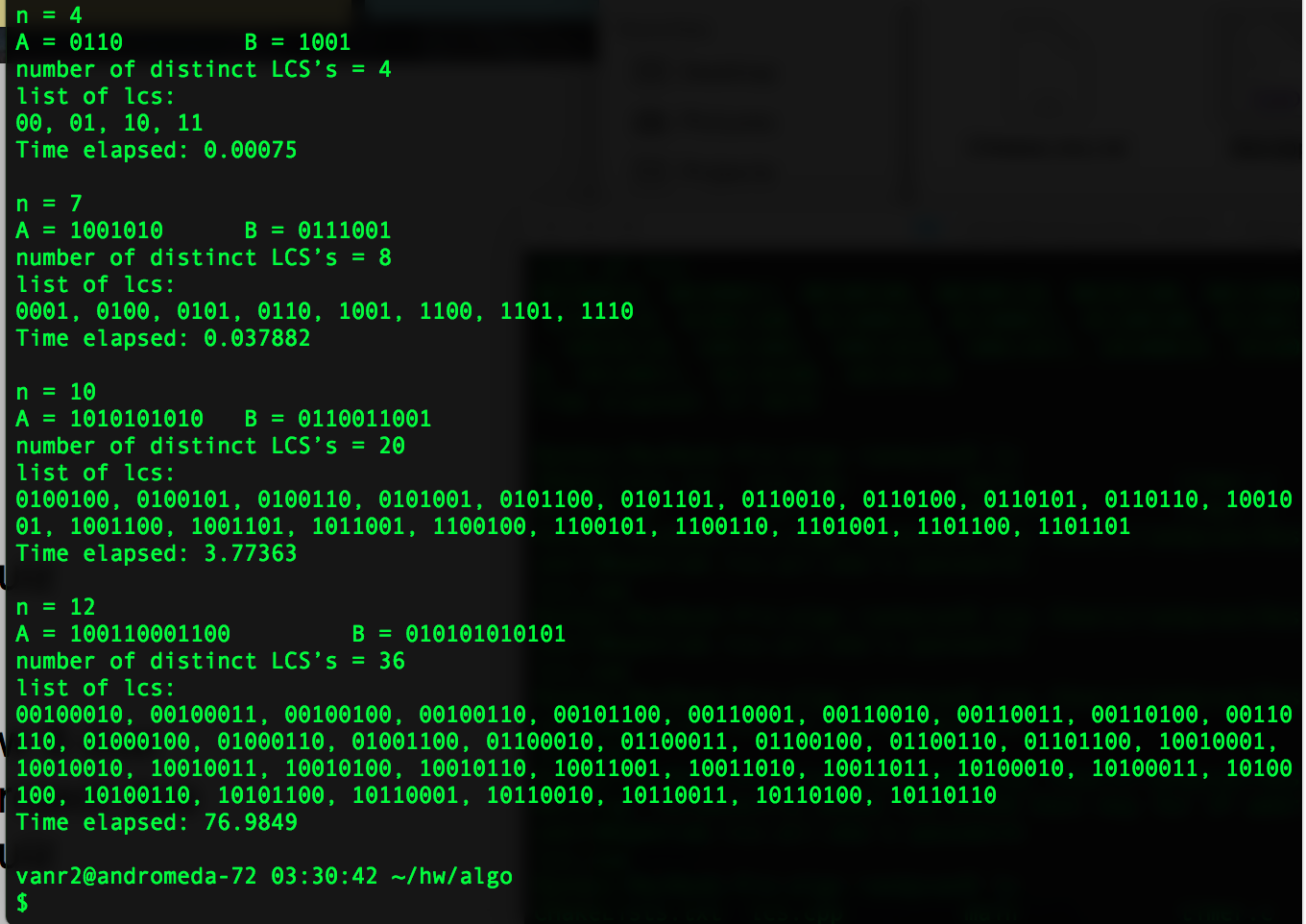
1. **Analysis**

|  |  |  |
| --- | --- | --- |
| Component | Description | Time Complexity |
| Binary\_string\_permutation | Generate all strings | O(2­n) |
| Determined\_string | Determine strings A and B | O(n2) |
| Num\_distinct\_lcs | Find number of distinct lcs | O(n2) |
| Lcs\_len\_vector | [n+1][n+1] vector of lcs length | O(n2) |
| lcs | Find length of lcs | O(n2) |
| Make\_distinct\_list | Make list of distinct lcs | O(n2) |
|  |  |  |
| Overall : | | O(2n) |

1. **Results**



**Figure 1**: *Results from running on local machine*



**Figure 2**: *Results from running on Openlab*

1. **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.

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