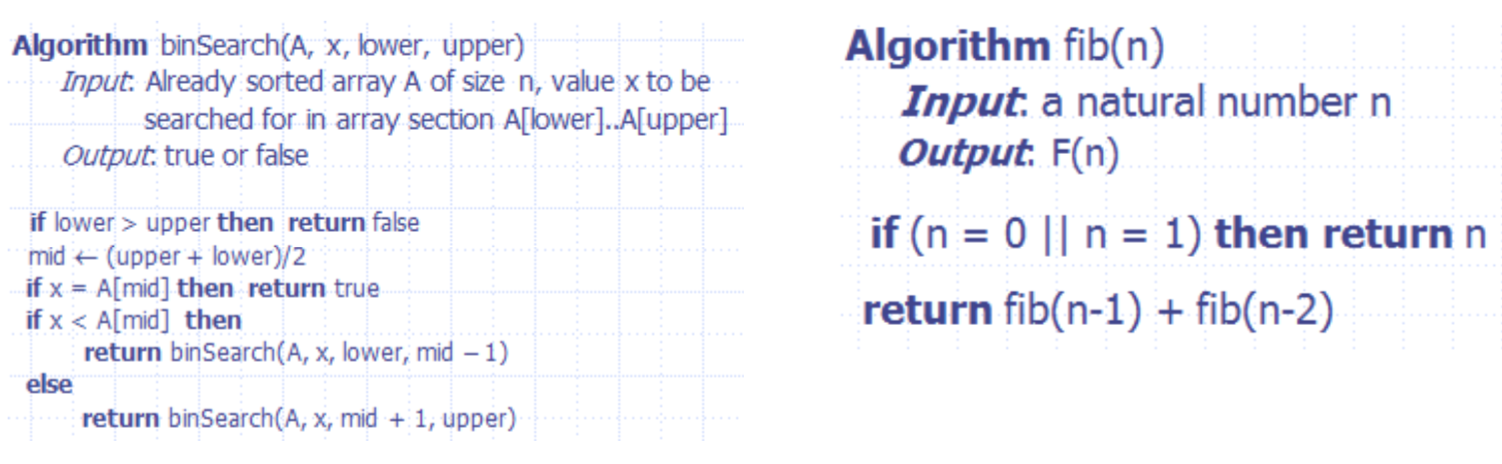
**LAB 10**

**Jimmy Palma**

**610756**

1. **Below, the BinarySearch and Recursive Fibonacci algorithms are shown. In each case, what are the subproblems? Why do we say that the subproblems of BinarySearch do not overlap and the subproblems of Recursive Fibonacci overlap? Explain.**



Subproblems are when a big problem is divided into smaller ones.

Binary Search does not overlap because every subproblem works separately in the if else. Fibonacci overlap the subproblem because proccess the same over and over

1. **Consider the following instance of the Edit Distance problem: EditDistance(“maple”, “kale”). Taking the iterative dynamic programming approach to solve this problem, fill out the values in the table.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **D** | **""** | **"k"** | **"ka"** | **"kal"** | **"kale"** |
| **""** | 0 | 1 | 2 | 3 | 4 |
| **"m"** | 1 | 1 | 2 | 3 | 4 |
| **"ma"** | 2 | 2 | 1 | 2 | 3 |
| **"map"** | 3 | 3 | 2 | 2 | 3 |
| **"mapl"** | 4 | 4 | 3 | 2 | 3 |
| **"maple"** | 5 | 5 | 4 | 3 | 2 |

1. **(Interview Question) Devise a dynamic programming solution for the following problem: Given two strings, find the length of longest subsequence that they share in common.**

**Different between substring and subsequence:**  
 **Substring: the characters in a substring of S must occur contiguously in S. Subsequence: the characters can be interspersed with gaps.**  
 **For example: Given two Strings - “regular” and “ruler”, your algorithm should output 4.**

Algorithm LLCS(Sᵢ, Tᵤ)

Input String Si, Tᵤ

Output Length of the LCS of Si and Tᵤ

if i = 0 || u = 0 then

return 0

else if S[i] = T[j] then

return LLCS(Si-1, Tᵤ -1) + 1

else  
 return max { LLCS(Si-1, Tᵤ), LLCS(Si, Tᵤ -1)

}

1. **(Optional Interview Question) Devise a dynamic programming solution for the following problem:**

**Given a positive integer n, find the least number of perfect square numbers which sum to n. (Perfect square numbers are 1, 4, 9, 16, 25, 36, 49, ...)**  
 **For example, given n = 12, return 3; (12 = 4 + 4 + 4)**  
 **Given n = 13, return 2; (13 = 4 + 9)**

**Given n = 67 return 3; (67 = 49 + 9 + 9)**