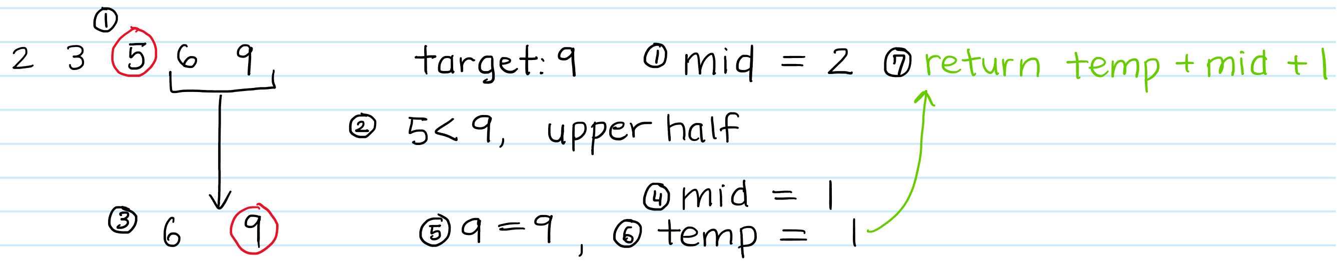
Lab 1

Sunday, January 12, 2020 10:17 PM

Recursive Binary Search:

This algorithm works by splitting the list in half, and comparing the center value with the target value. If the center value is larger than the target value, then the function calls itself again, this time with the list in range from 0 to the center value. The upper half of the previous list is discarded. It repeats this until either the target value is found, or found that it does not exist within the list, which happens when the list length is 0. If the target is found, the index is returned. If it does not exist, it returns None.

Algorithm Complexity:
O(log n) Logarithmic

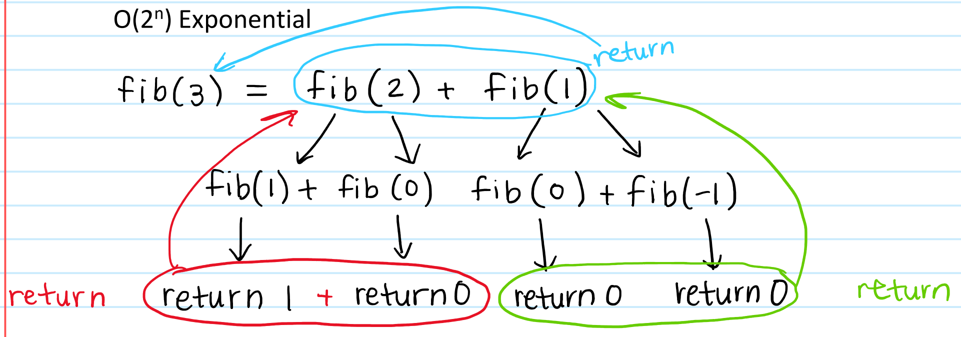


① compute middle value of list
② compare middle value with target
③ run search function on upper half
④ compute new middle value
⑤ compare middle value with target
⑥ assign temp as the new mid value (1)
⑦ end recursion, sum up temp, mid, +1

Fibonacci:

The nth term in a Fibonacci sequence is the sum of the two previous terms. The two previous terms are computed by recursive calls to the function. Each of these calls also call another two instances with reduced indices until it reaches n = 0 and n =1, which then returns 0 and 1, respectively. The time complexity of this algorithm is exponential, since for every call of the function, there are two calls, each with two calls and so on.

Algorithm Complexity:
O(2^n) Exponential



Factorial:

When using the iterative function to compute 1000!, The function does the multiplication operation 1000 times, decrementing the index until it reaches 1. For the recursive function, it calls itself with an index 1 less of the original while the index is greater than 0. Having an index of 1000, the function would recursively call itself 1000 times, but due to Python's recursion limit of 1000, it cannot run and will raise a recursion error.

Algorithm Complexity:
O(n) Linear (both)