TUTORIAL 8

O(\_\_)

EFFICIENCY, SEARCHING AND SORTING ALGORITHMS
REMINDERS

• Assignment 7 due Wed, July 5th, at 10am
RUNTIME REVIEW

• Look at the “worst case” scenario (i.e. longest runtime)
• Only for code that gets executed when you run it
• Assume function works (i.e. will not produce an error when you run it)
RUNTIME REVIEW

• O(1) – Constant
  – does not depend on the size of the input
  – For numbers:
    • Numeric operations: +, *, /, -, %, //
    • max, min
  – For list L:
    • L[0], len(L)...
    • L.append(4)...

• O(n) – Linear
  – depends on the size of the input
  – For list L:
    • L[1:], max(L), L + L, sum(L), L.remove(0)...
    • list(map(lambda x: x+1, L))
RUNTIME REVIEW

• $O(n^2)$ – Quadratic
  – time proportional to square of input
  – For list $L$, $n = \text{len}(L)$:
    • $\text{list(map(lambda k: list(range(n)), list(range(n)))))}$

• $O(2^n)$ – Exponential
  – As size of input increases, run time doubles

Module 5, Slide 15: fib
# Let \( n = \text{len}(L) \)
def fn(L):
    if L == []:
        return 0
    else:
        return 1 + fn(L[1:])

Count steps for:
- Compare \( L \) with \( [] \)
- Calculate \( L[1:] \)
- Call \( \text{fn} \) recursively on a list of length \( n-1 \)
- Add 1 to the recursive call of \( \text{fn} \)
- \( T(n) = O(n) + T(n-1) \)
# Let n = len(L)
def fn(L):
    ans = []
    for x in L:
        if x[0]=='A':
            ans.append(x)
    return ans

Count steps for:
• Assign [] to ans
• Loop:
  – Number of Iterations
  – Asymptotic run time of the body of loop:
    • Check if x[0] == 'A'
    • ans.append(x)
• Return ans
RUNTIME EXAMPLE 3

# Let n = len(L)
def fn(L):
    L1 = L[0::2]
    if L==[]:
        return []
    else:
        return fn(L1)

Count steps for:

- L1 = L[0::2]
- Compare L with []
- Call fn recursively on a list of length n//2
- T(n) = O(n) + T(n/2)
# Q5

def fn(n):
    if n % 2 == 0:
        return "outcome1"
    elif n % 3 == 0:
        return "outcome2"
    elif n % 5 == 0:
        return "outcome3"
    else:
        return "outcome4"

Count steps for:

- Calculate n%2
- Compare it with 0
- Calculate n%3
- Compare it with 0
- Calculate n%5
- Compare it with 0
- Return the answer
QUESTION 2 - QUICKSORT

Consider a different way of sorting a list \( L \) of distinct integers:

- Let \( n \) be the first element of the list
- Let \( \text{lst1} \) be all the elements in the list smaller than \( n \)
- Let \( \text{lst2} \) be all the elements in the list larger than \( n \)
- Recursively quicksort \( \text{lst1} \) and \( \text{lst2} \)
- \( \text{lst1} + [n] + \text{lst2} \)

Write a function \texttt{quicksort} which consumes a list of distinct integers, \( \text{lst} \), and sorts it using the quicksort algorithm.
EXAMPLE

quicksort([2,3,1,4,0])

- quicksort([1,0]) + [2] + quicksort([3,4])
- (quicksort([0]) + [1]) + [2] + ([3] + quicksort([4]))
- ([0] + [1]) + [2] + ([3] + [4])
- [0,1] + [2] + [3,4]
- [0,1,2,3,4]
RUNTIME OF QUICKSORT

• **Worst case runtime:**
  - $T(n) = O(n) + T(n-1) \Rightarrow O(n^2)$
  - The list is already sorted

• **Best case runtime:**
  - $T(n) = O(n) + 2*T(n/2) \Rightarrow O(n \log n)$
  - The first element is always the middle number of the list

• There are techniques that you can use to avoid $O(n^2)$ runtime for Quicksort