REMINDERS

• Final exam is on December 19th at 7:30 p.m.!!!

• Assignment 7 due Wednesday, Nov. 13th, at 10:00 AM
RUNTIME REVIEW

• Look at the “worst case” scenario (i.e. longest runtime)
• Assume function works (i.e. will not return an error when you run it)
• Based on the assumptions learned in class (and in the modules)
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 (assume the length of L is n):
    - L[1:], max(L), L + L, sum(L), L.remove(0)...
    - list(map(lambda x: x+1, L))
    - L.pop(2)...

RUNTIME REVIEW

- $O(n^2)$ – Quadratic
  - time proportional to square of size of the input
  - Be careful of abstract functions:
    - list(map(lambda k: list(range(k)), list(range(n)))))

- $O(2^n)$ – Exponential
  - As size of input increases by 1, the run time doubles
  - example: Module 5, Slide 15: fib
Recurrence Relations

- $T(n) = O(1) + T(n-1) \rightarrow O(n)$
- $T(n) = O(n) + T(n-1) \rightarrow O(n^2)$
- $T(n) = O(1) + T(\frac{n}{2}) \rightarrow O(\log n)$
- $T(n) = O(n) + 2T(\frac{n}{2}) \rightarrow O(n \log n)$

No need to memorize the table for exam!
USEFUL SUMMATIONS

• $\sum_{i=1}^{n} 1 = O(n)$
• $\sum_{i=1}^{n} i = O(n^2)$
  — This sum is useful in A7Q4
• $\sum_{i=1}^{n} n = O(n^2)$
• $\sum_{i=1}^{n} \sum_{j=1}^{n} 1 = O(n^2)$
# Let n = 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 fn recursively on a list of length n-1
• Add 1 to the recursive call of fn

• T(n) = O(n) + T(n-1) => O(n^2)
# 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
- \( \sum_{i=1}^{n} 1 = O(n) \)
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
- O(1)
Write a function, \texttt{largest\_diff(loi)}, that consumes \texttt{loi} (listof Int) that finds the largest difference between 2 elements in a list. Do it in $O(n)$

Examples:

- $\texttt{largest\_diff([2, 9, 3, 6, 4])} \Rightarrow 7$
- $\texttt{largest\_diff([1, 2, 3, 4])} \Rightarrow 3$
- $\texttt{largest\_diff([-20, 2, 70, 15])} \Rightarrow 90$