Activity 1: Recursive array_max

like you did for the section 0 mini assignment, draw out the call stack for each recursion of array_max([5, 1, 9, 2], 4). When you reach the base case and the function returns, write the return value. Continue to write the return value as you pop calls off the stack. you can work with a neighbor. put "N/A" for the non-base-case return values. the first one is done for you!


compare your result with a different neighbor and resolve any differences.

Activity 2: Plug 'n Chug

the recurrence relation T(n) given below describes the number of operations needed for a problem of size n in terms of the number of ops needed for a problem of size n - 1:

T(n) = c₁ + T(n - 1)
T(1) = c₀

Continue filling out the 'plug n chug' solutions for T(5) and T(6), using the recurrence relation.

T(2) = c₁ + T(2-1) = c₁ + T(1) = c₁ + c₀
T(3) = c₁ + T(3-1) = c₁ + T(2) = c₁ + c₁ + c₀ = 2c₁ + c₀
T(4) = c₁ + T(4-1) = c₁ + T(3) = c₁ + 2c₁ + c₀
T(5) =
T(6) =

Based on the results for T(1) - T(6), what do you think T(n) is in terms of constants and n?

T(n) =

What is big-O of T(n)?
Activity 3: Induction Proof

Follow along with the induction proof example shown in class (also shown here on the right-hand side of the page in italics) and fill in your own proof on the left-hand side. Prove the following for all positive integers $n$:

$$\sum_{x=1}^{n} x = \frac{n(n+1)}{2}$$

Your Proof:

Sample Proof:

The solution for the recurrence relation $T(1) = c0, T(n) = c1 + T(n-1)$ is $T(n) = (n-1)c1 + c0$

Base case (show that $P(n)$ is true for $n = 1$):

$T(1) = (1 - 1)c1 + c0 = c0$

Inductive Assumption (write out $P(k)$):

Assume the proposition is true for $k$: $T(k) = (k - 1)c1 + c0$

Now, write out $P(k + 1)$, what you want to prove:

$T(k + 1) = (k)c1 + c0$

Inductive Step: Show that $P(k + 1)$ is true given $P(k)$

Hint: Start with writing out the left side of $P(k + 1)$ and filling in the right side based on the definition of a sum. Using simplification and your inductive assumption, make the right side look like $P(k + 1)$.

$T(k + 1) = c1 + T(k)$ (by recurrence relation)

$T(k + 1) = c1 + (k - 1)c1 + c0$ (by induct. assump.)

$T(k + 1) = (k)c1 + c0$

This is $P(k + 1)$.

Conclusion (how does your work show that the claim is true?)

We've proven $P(n)$ for the base case $n = 1$ and shown that for some $k$, $P(k)$ implies $P(k + 1)$, therefore $T(n) = (n - 1)c1 + c0$ for positive integers $n$. 

Activity 1: Pseudocode for a Capped-capacity Stack

Write pseudocode for the functions $\text{isEmpty()}$, $\text{push(obj)}$, and $\text{pop()}$ for a capped-capacity stack. Assume your stack has the following constructor and $\text{size()}$ functions.

Stack(): $\text{O( } )$

function push(obj): $\text{O( } )$
  data = array of size 20
  # TODO
  count = 0
  # TODO

function size(): $\text{O(1) }$
  return count

function isEmpty(): $\text{O( } )$
  # TODO
  # TODO

function $\text{pop()}$: $\text{O( } )$
  # TODO
  # TODO

Write the big-O runtime on each operation above.

What should happen if the user tries to push to a stack that is at full capacity? What about when someone tries to pop from an empty stack?

Activity 2: Expanding Stack - Analysis of Incremental Strategy

Based on the calculations in lecture of the number of operations per push for 5, 10, and 15 pushes, using an incremental expansion strategy where $c = 5$, what would be the average number of operations per push for 20 pushes?