CS 61A Discussion 5

February 25, 2016

Agenda

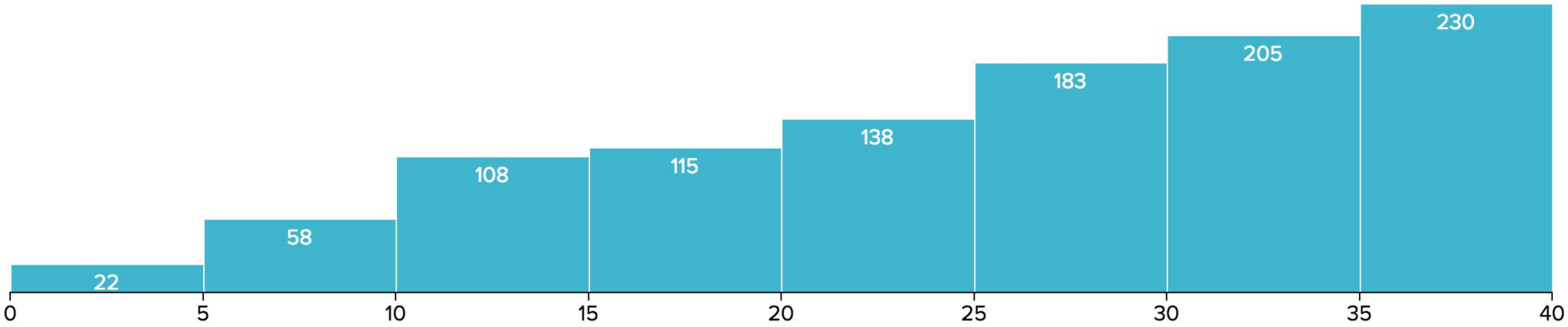
- Quiz 05
- Midterm 1 Reflection
- Mutation
- Dictionaries

Quiz 05: Solution

```
def tree_path(t, num, directions):
    """Given a tree T that is filled with numbers, mutates DIRECTIONS
    so that it contains a path to NUM. DIRECTIONS is given as a
    list of child indices to follow.
    """
    while directions:
        directions.pop()
    def helper(t):
        if label(t) == num:
            return True
        elif is_leaf(t):
            return False
        for i in range(len(children(t))):
            directions.append(i)
            if helper(children(t)[i]):
                return True
            else:
                directions.pop()
    helper(t)
```

MT1 was hard

...but fun, right?
(It was empirically pretty hard. This is the distribution from last semester's MT1.)



MINIMUM	MEDIAN	MAXIMUM	MEAN	STD DEV
0.5	27.5	40.0	25.66	9.89

MT1 Comments:

+ 20/40 (or even 10/40) is not the same as failing the course. We'll account for brutal test scores through the rest of the assignments.

+ Tests (and general content) is cumulative. You should go over MT1 and make sure you understand the concepts that we tested you on.

+ No matter how you did, don't let your guard down. Keep working hard and ask for help when you need it. Remember: tenacity!

+ I'm happy to meet with anyone who wants to talk one-on-one about their midterm.

Trees

There is a lot of terminology surrounding trees. (Sorry.) Be familiar with all of them.

+ **Node** – think of it as a point in the tree. (It's still a tree itself.)
+ **Parent node** – one half of the parent/child relationship
+ **Child node** – the other half of the parent/child relationship
+ **Root** – the topmost node
+ **Leaf** – one of the nodes on the bottom
+ **Subtree** – a smaller tree within a larger tree
+ **Depth** – distance from the root. (Root has depth 0, its children have depth 1…)
+ **Height** – max distance from the root

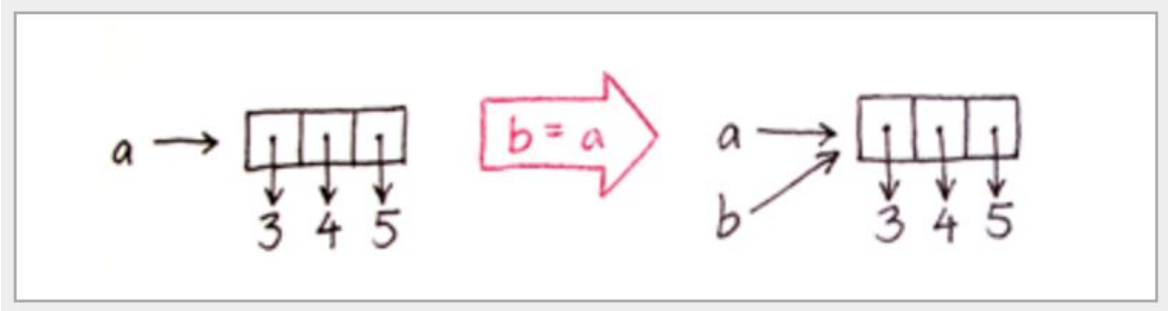
The Tree ADT

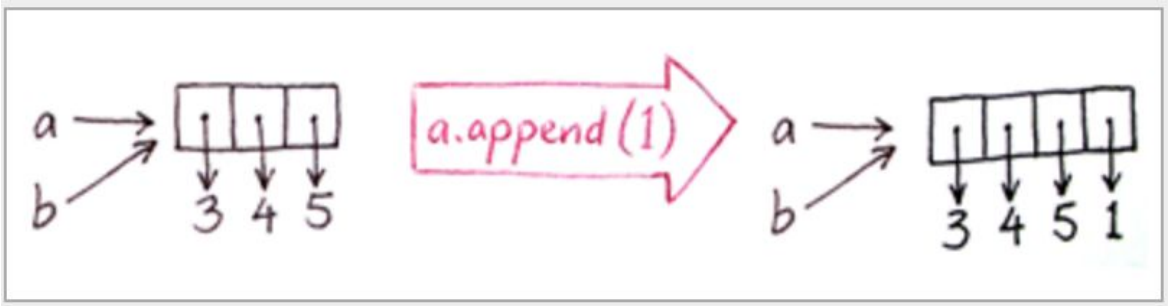
We represent trees as an **abstract data type**.

+ `tree(label, children=[])` – creates and returns a tree
+ `label(tree)` – returns the label at the topmost node of the tree
+ `children(tree)` – returns the children of the topmost node of the tree
+ `is_leaf(tree)` – True if the tree is a leaf. False if it isn't

Mutation

Mutation means that you're **changing the actual object in memory**.





Mutation, *cont.*

+ Mutation functions often return `None`.
+ Also, functions that mutate a list usually don't need to create a *new* list.
+ The + operator creates a new list. It does not mutate.
+ The += operator mutates, for some reason.

Mutable Lists (i.e. the lists you've already been using)

Lists are mutable. You mutate them with **list methods**, which are basically functions that act on a list:

+ `append(elt)` – adds something to the end of the list
+ `insert(i, elt)` – inserts something at index `i`
+ `remove(elt)` – removes something from a list
+ `pop(i)` – removes AND RETURNS the element at index `i`

Dictionaries

Dictionaries are basically a bunch of key/value pairs.

+ Only one value per key!
+ `dictionary[key] = val` – adds the (`key`, `val`) pair to the dictionary
+ `del dictionary[key]` – deletes a key/value pair from the dictionary

+ To iterate over keys, we use

```
for key in dictionary:
    print(key) # this is just an example
    print(dictionary[key]) # prints values
```