Discussion 7

Trees and Linked Lists

Antonio Kam

anto [at] berkeley [dot] edu

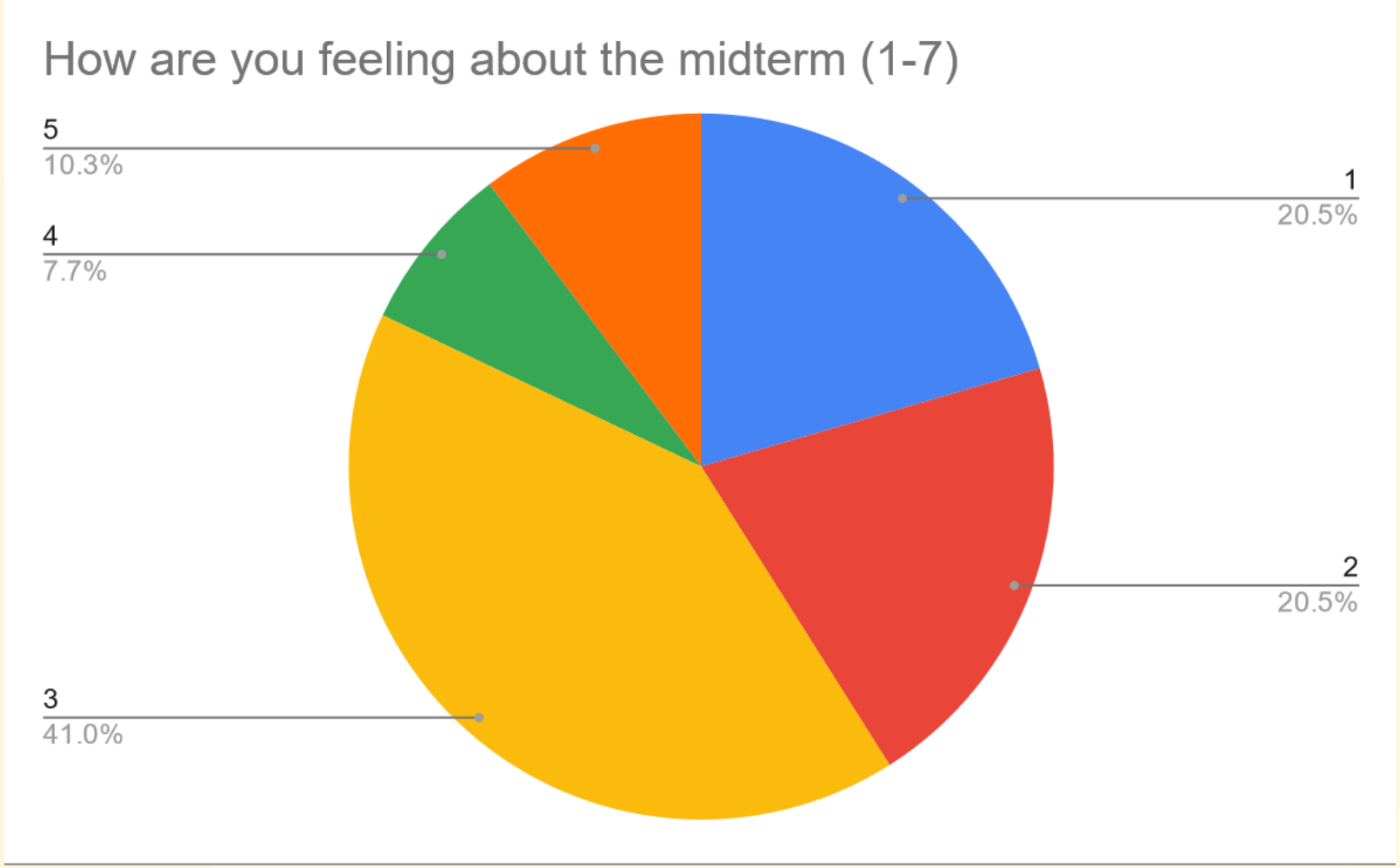
All slides can be found on

`teaching.rouxl.es`

Announcements

- Cats due today!
- Ants gets released tomorrow!
  - This is my favourite project by far 👀 I hope you enjoy it
  - Checkpoint is Friday
- Magic: the Lambda-ing will be released soon as an extra credit project
  - It's definitely a small project - you aren't expected to do nearly as much as the normal CS 61A projects
- HW 04, Lab 07 due Thursday
- CS 61A has a final exam *full* clobber policy
  - This is not usually a thing during regular semesters, but we can all take advantage of it!

Notes from last section



Notes from last section/feedback form

- There's going be more resources for finals - we'll have specific topical review sections along with the discussion/lab final review sections
- Lab and Discussion are fixed timeslots - I'm not allowed to extend the amount of time for each section
- More problem-solving tips
- Try to do more questions for each topic
- Teach us how to cube
  - There's a DeCal for that! (I'm teaching it 👀)

Temperature Check 🌡️

- Linked lists (have not been covered in lecture yet)
- Trees
- Tree Recursion (they're pretty related with trees)

Trees

Trees

- Trees are recursive data structures (as in, trees contain more trees)
- Important terms:
  - Root Node
  - Branch(es)
  - Leaf Node
  - Children
- Sort of looks like an upside-down tree compared to the real world
- Questions are generally solved using tree recursions

Trees (construction)

- The `Tree` constructor takes in a `label`, and an *optional* `list` of `branches`. If `branches` isn't given, it defaults to an empty list `[]`

```
class Tree: # simplified version compared to the 61A implementation
    def __init__(self, label, branches = []):
        assert type(branches) == list
        for item in branches:
            assert isinstance(item, Tree)
        self.label = label
        self.branches = branches
```

Construction: `Tree(2)`; `Tree(2, [Tree(3), Tree(4)])`

Note that the branches **must** be in a list.

Trees (access)

```
t = Tree(3, [Tree(4, [Tree(5)]), Tree(6)])
>>> t.label
3
>>> t.branches
[Tree(4, [Tree(5)]), Tree(6)]
```

Question 1 (mini-lecture)

Recall that the height of a tree is the length of the longest path from the root to a leaf.

```
def height(t):
    """Return the height of a tree.

    >>> t = Tree(3, [Tree(5, [Tree(1)]), Tree(2)])
    >>> height(t)
    2
    >>> t = Tree(3, [Tree(1), Tree(2, [Tree(5, [Tree(6)]), Tree(1)])])
    >>> height(t)
    3
    """
    "*** YOUR CODE HERE ***"
```

Worksheet!

Attendance

`links.rouxl.es/disc`

Linked Lists

Linked Lists

- You can use linked lists to create your own version of a sequence
- They are generally useful when you want to have an infinitely-sized list, or want to dynamically change the size of the list (more useful in 61B if you do end up taking it)
- In general, linked lists problems can be solved using both iteration and recursion
  - Like trees, they are recursive data structures, but unlike trees, you can use both recursion and iteration to solve them.

Linked Lists (Anatomy)

```
class Link:
    empty = ()

    def __init__(self, first, rest = Link.empty):
        assert rest is Link.empty or isinstance(rest, Link)
        self.first = first
        self.rest = rest
```

- Linked lists have a `first` (similar to `label`) attribute and a `rest` (similar-ish to `branches`) attribute.

Linked Lists (Construction)

Note to Anto: Draw box-and-pointer diagram

```
s = Link(1, Link(2, Link(3)))
>>> s.first
1
>>> s.rest
Link(2, Link(3))
>>> s.rest.first
2
s2 = Link(1, 2) # This will error because rest is not a linked list
>>> s.rest.rest.first
3
```

Question 5: WWPD

```
>>> link = Link(1, Link(2, Link(3)))
>>> link.first
>>> link.rest.first
>>> link.rest.rest.rest is Link.empty
>>> link.rest = link.rest.rest
>>> link.rest.first
>>> link = Link(1)
>>> link.rest = link
>>> link.rest.rest.rest.rest.first
>>> link = Link(2, Link(3, Link(4)))
>>> link2 = Link(1, link)
>>> link2.first
>>> link2.rest.first
```

Mental Health Resources

- CAPS:
  - If you need to talk to a professional, please call CAPS at 510-642-9494.
- After Hours Assistance
  - For any assistance after hours, details on what to do can be found at this link

Anonymous Feedback Form

`links.rouxl.es/feedback`

Thanks for coming! 🥳

Please give me feedback on what to improve!