DIVIDE AND CONQUER, RECURSION

Dezhi “Andy” Fang ,GT TIP

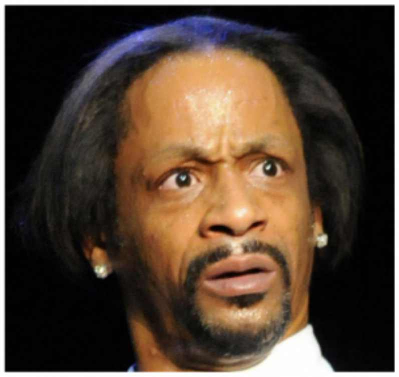
STORY TIME!

- 70+ Graded Papers
- 5 minutes
- How to sort?
- Abusing your friends.
- Divide-Conquer-Combine
- Fast forward to now...

I have no idea what quicksort is.

- Nervous Interviewee

```python
def quicksort(list):
    if len(list) <= 1:
        return list
    pivot = list[0]
    lesser = [item for item in list if item < pivot]
    pivots = [item for item in list if item == pivot]
    greater = [item for item in list if item > pivot]
    lesser = quicksort(lesser)
    greater = quicksort(greater)
    return lesser + pivots + greater
```



WAIT WHAT?

```python
def quickSort(alist):
   quickSortHelper(alist,0,len(alist)-1)

def quickSortHelper(alist,first,last):
   if first<last:

       splitpoint = partition(alist,first,last)

       quickSortHelper(alist,first,splitpoint-1)
       quickSortHelper(alist,splitpoint+1,last)


def partition(alist,first,last):
   pivotvalue = alist[first]

   leftmark = first+1
   rightmark = last

   done = False
   while not done:

       while leftmark <= rightmark and alist[leftmark] <= pivotvalue:
           leftmark = leftmark + 1

       while alist[rightmark] >= pivotvalue and rightmark >= leftmark:
           rightmark = rightmark -1

       if rightmark < leftmark:
           done = True
       else:
           temp = alist[leftmark]
           alist[leftmark] = alist[rightmark]
           alist[rightmark] = temp

   temp = alist[first]
   alist[first] = alist[rightmark]
   alist[rightmark] = temp


   return rightmark
```

Walking through

```python
def quicksort(list):
    if len(list) <= 1:
        return list
    pivot = list[0]
    lesser = [item for item in list if item < pivot]
    pivots = [item for item in list if item == pivot]
    greater = [item for item in list if item > pivot]
    lesser = quicksort(lesser)
```

[3, 2, 5, 1, 4] => [1, 2, 3, 4, 5]

RECURSIVE DATA STRUCTURES

- E.g., Linked List, Trees
-

1. Given a binary tree, find its maximum depth.
2. You have two every large binary trees: `T1`, with millions of nodes, and `T2`, with hundreds of nodes. Create an algorithm to decide if `T2` is a subtree of `T1`.
3. Calculate the `a^n % b` where `a`, `b` and `n` are all 32bit integers.