Programming fundamentals with Python

Sorting algorithms - mergesort

Pepe García jgarciah@faculty.ie.edu

2020-11-06

Plan for today

- recursion

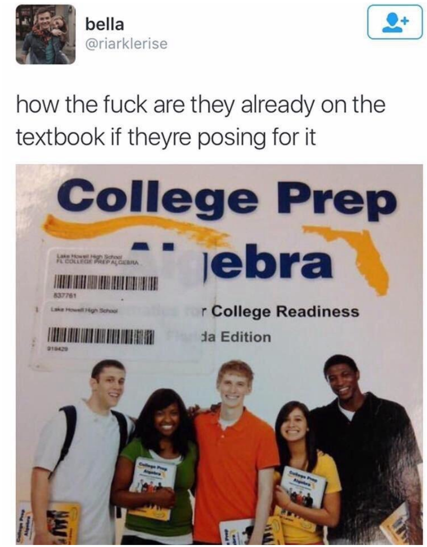
Plan for today

- recursion
- divide and conquer algorithms

Plan for today

- recursion
- divide and conquer algorithms
- mergesort

Recursion


bella
@riarklerise

how the fuck are they already on the textbook if theyre posing for it

Recursion is a technique to solve problems in terms of smaller versions of the same problem

Recursion

An example of a recursive algorithm is the calculation of a number in the Fibonacci sequence.
Remember that Fibonacci goes like this:
0, 1, 1, 2, 3, 5, 8, 13...

Recursion

An example of a recursive algorithm is the calculation of a number in the Fibonacci sequence.
Remember that Fibonacci goes like this:
0, 1, 1, 2, 3, 5, 8, 13...

Fibonacci

Let's implement the calculation of a number in the Fibonacci sequence using recursion!

Recursion

Something that we have to consider is that all problems that we solve using recursion can be solved using iteration.

Recursion

Something that we have to consider is that all problems that we solve using recursion can be solved using iteration.

Homework

Try implementing the previous function using iteration instead of recursion!

Divide and conquer

From Wikipedia:

> *Divide and rule (Latin: divide et impera), or divide and conquer, in politics and sociology is gaining and maintaining power by breaking up larger concentrations of power into pieces that individually have less power than the one implementing the strategy.*

Divide and conquer

From Wikipedia:

> *Divide and rule (Latin: divide et impera), or divide and conquer, in politics and sociology is gaining and maintaining power by breaking up larger concentrations of power into pieces that individually have less power than the one implementing the strategy.*

Divide and conquer is a technique used **in algorithmia** in which we split a problem into smaller sub-problems recursively until they're simple enough to be solved directly.

Checkpoint

How is the session going so far? - Do we have questions? - Is there anything that doesn't yet click?

Mergesort

The idea behind **mergesort** is fairly simple. What if we had a way of splitting an unsorted list into smaller ones, and then merging them together in a sorted fashion?

Mergesort

The idea behind **mergesort** is fairly simple. What if we had a way of splitting an unsorted list into smaller ones, and then merging them together in a sorted fashion?

A question we need to agree on before jumping into the implementation...

- **Is a list of only one element sorted?**

Mergesort

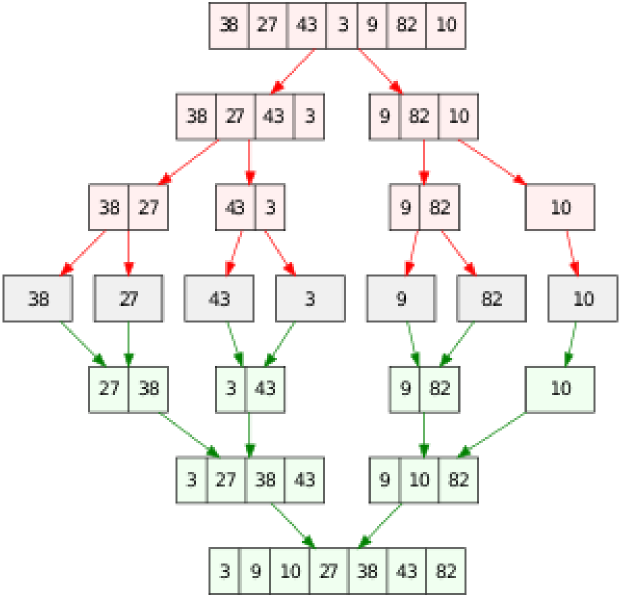
The idea behind **mergesort** is fairly simple. What if we had a way of splitting an unsorted list into smaller ones, and then merging them together in a sorted fashion?

A question we need to agree on before jumping into the implementation...

- **Is a list of only one element sorted?**

Yes, it is!

Mergesort


38 27 43 3 9 82 10
38 27 43 3
9 82 10
38 27
43 3
9 82
10
38
27
43
3
9
82
10
27 38
3 43
9 82
10
3 27 38 43
9 10 82
3 9 10 27 38 43 82

Mergesort

In our mergesort implementation we will have two different functions. **`merge`**, that will just take two **sorted** lists and merge them into a new sorted list, and **`mergesort`**, that will perform the splitting of the list and call **`merge`** on the smaller lists.

Mergesort

In our mergesort implementation we will have two different functions. **`merge`**, that will just take two **sorted** lists and merge them into a new sorted list, and **`mergesort`**, that will perform the splitting of the list and call **`merge`** on the smaller lists.

Mergesort

Let's implement mergesort!

Mergesort

What's the worst case runtime for mergesort? Why?