Week 06 Presentation

Sunday, October 24, 2021
6:12 PM

Agenda:

1. Go through Exercise04 solution - compare answers!
2. Hints for Assignment 2 (DUE: OCT 31ST!)
3. BRIEFLY go through tutorial video content (VS CODE)
4. Answer and discuss practice problems for this week.

Recursive Lists:

Iterating through a list recursively!

I.e.

```
rec_func :: Num a => [a] -> [a]
rec_func [] = []
rec_func (x:xs) = x+1 : rec_func xs
```

BubbleSort Algorithm:

Iterate through every element in a list
If element is bigger than the one if front of it, swap the two elements

REPEAT UNTIL LIST IS SORTED FROM SMALLEST TO GREATEST (will go through list multiple times)
It will check adjacent elements

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

1. [1,2,4,5,2,3]
2. [1,2,4,2,5,3]
3. [1,2,4,2,3,5]

Goes through list again

4. [1,2,2,4,3,5]
5. [1,2,2,3,4,5]

Worst case scenario:
i.e. [5,4,3,2,1]

1. [4,5,3,2,1]
2. [4,3,5,2,1]
3. [4,3,2,5,1]
4. [4,3,2,1,5]

1. [3,4,2,1,5]
2. [3,2,4,1,5]
3. [3,2,1,4,5]

1. [2,3,1,4,5]
2. [2,1,3,4,5]

1. [1,2,3,4,5]

1. [1,2,3,4,5]

This alogirthm is very inefficient, since it has to iterate through the list multiple times.
Most intuitive sorting algorithm -> most understandable.

Insertion Sort

Set up a sorted and unsorted list i.e. in the beginning, the "sorted" list could be the empty list.
Takes an element of the unsorted list, and insert it into the sorted list.

Since the list is already sorted, you move through the list, and insert the element of the unsorted list, to the sorted list

i.e.

Unsorted list = [6,5,3,1,8]
Sorted list = []

1. Unsorted list [5,3,1,8] Sorted list = [6]
2. Unsorted list [3,1,8] Sorted list = [5,6]
3. Unsorted list [1,8] Sorted list = [3,5,6]
4. Unsorted list [8] Sorted list = [1,3,5,6]
5. Unsorted list [] Sorted list = [1,3,5,6,8]

Essentially, it takes the first element of out unsorted list, and inserts it into the correct place of the sorted list.

QuickSort

Chooses a pivot point of a list (i.e. the first element of the list)
Splits this list into two lists
Smaller -> all the elements of the original list that is smaller than or equal to the pivot point
Greater -> all the elements of the original list that is greater than the pivot point

Then does this process on the smaller and greater lists respectively.

I.e.

List = [7,3,8,1,2,1]
Pivot = 7
Smaller = [3,1,2,1] ++ [7] ++ Greater = [8]
*smaller gets broken down

Unsorted list = [3,1,2,1]
Pivot =3
Smaller1 = [1,2,1]
Greater1 = []

Smaller = [1,2,1] ++ 3 ++ []
*smaller gets broken down again
Unsorted list = 1,2,1
Pivot = 1
Smaller2 = [1]
Greater2 = [2]
Smaller = [1] ++ [1] ++ [2] = [1,1,2]

Combine the lists!

[1,1,2,3,7,8]