Week 2

Review last week's assignments

Q: 2017 j1
Q: 2017 j2
Q: 2016 j1
Q: 2016 j2

About your homework:

Data Structure

Data structure is _____, so that it _____. It is a collection of _____, _____, and _____.
In python, we already learned _____. These are all data structures.

Algorithms

Algorithms: an effective method expressed as a finite list of well-defined instructions for calculating a function. It tells you how to solve a class of problems.

Properties:

1. finiteness: _____
2. definiteness: _____
3. input
4. output

Common algorithms:

1. reading data from external source
2. computations and comparisons, testing logical conditions
3. selection:_____
4. iterations:_____
5. output:_____

Testing correctness:

1. mathematical proof
2. choose different data sets

Cost of the algorithms:

1. Time
2. Space
3. Both

What is cost? Algorithm analysis.

```
def sumOfN(n):
   theSum = 0
   for i in range(1,n+1):
       theSum = theSum + i

   return theSum

print(sumOfN(10))
```

PseudoCode

The first thing we need for a program is to decide the name for the program.
We learn pseudo code so that we can understand what algorithm means.

- Let's say if we want to write a program that can calculate the interest rate. So we can call the name of the program `CalculateInterest`
- The naming style is called CamelCase :)

So this is our program

```
PROGRAM <ProgramName>:
       <Do stuff>
END.
```

Computer starts from the beginning to the end. We call this **Sequence**. This is the basic design of the algorithm.

```
Statement1;
Statement2;
Statement3;
Statement4;
Statement5;
Statement6;
```

Selection Condition

```
IF <CONDITION>
       THEN <Statement>;
       ELSE <Statement>;
ENDIF;
```

When you make coffee

```
IF (sugar is required)
       THEN add sugar;
       ELSE don't add sugar;
ENDIF
```

Iteration

Concept:

```
WHILE (Kettle is not full)
       DO keep filling kettle;
ENDWHILE;
```

The syntax is

```
WHILE (<CONDITION>)
       DO <Statement>;
ENDWHILE;
```

Example:

If we want to express the following algorithm:

- read in a number and print it out

```
PROGRAM PrintNumber:
       Read A
       Print A
END.
```

If we want to change the algorithm:

- Read in a number and double the number and print it out

```
PROGRAM PrintDoubleNumber:
       Read A
       B = A * 2
       Print B
END.
```

- If we want to check if it is odd or even

```
PROGRAM IsOddOrEven:
       Read A
       IF (A/2 gives a remainder)
              THEN Print "it's Odd";
              ELSE Print "it's Even";
       ENDIF
END.
```

We want to print out numbers from 1 to 5

```
PROGRAM Print1to5:
	A = 1;
	WHILE (A != 6)
	DO Print A;
		A = A + 1;
	ENDWHILE;
END.
```

Next Topic is **Searching and Sorting**, we look into search first then sort

Searching

Searching

In python, we have an easy way to ask whether an item is in the list

```
>>> 15 in [3,5,2,4,1]
False
>>> 3 in [3,5,2,4,1]
True
```

We're more interested in how the searching algorithm is implemented.
We can do a linear search or sequential search to find the value.

1. **Linear Search**
   Problem: Given a list N values, determine whether a given value X occurs in the list
   For example, consider the problem whether value 55 occurs in

1	2	3	4	5	6	7	8
17	31	9	73	55	12	19	7

   Linear search algorithm:

```
def linearSearch(lst, value):
	"""
	the obvious solution is that we start from the top, and move to the next element
	stop when we find the value
	"""
```

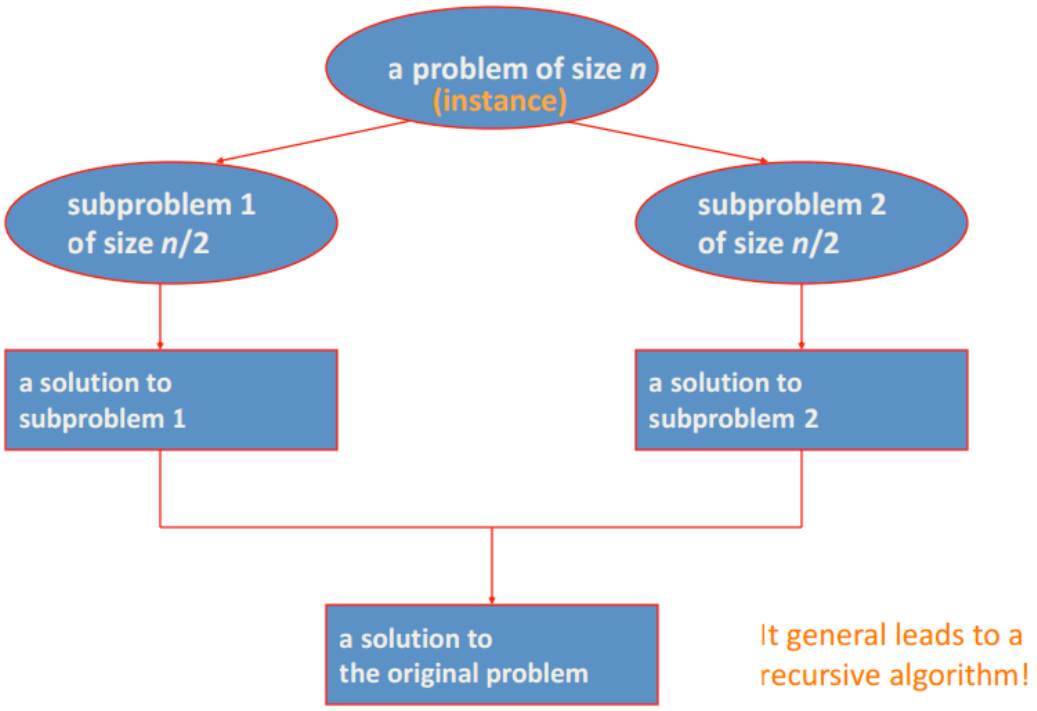
If the list is an ordered list, the linear search algorithm would be the same.

```
def orderedLinearSearch(lst, value):
        """
        for ordered list, the solution is the same, starting from the top, and moving to
        the next element
        stop when we find the value
        """
```

2. Search with log2(n) time

Divide-and-Conquer

1. Divide _____
2. Solve _____
3. obtain solution to original instance by _____



Basic idea:

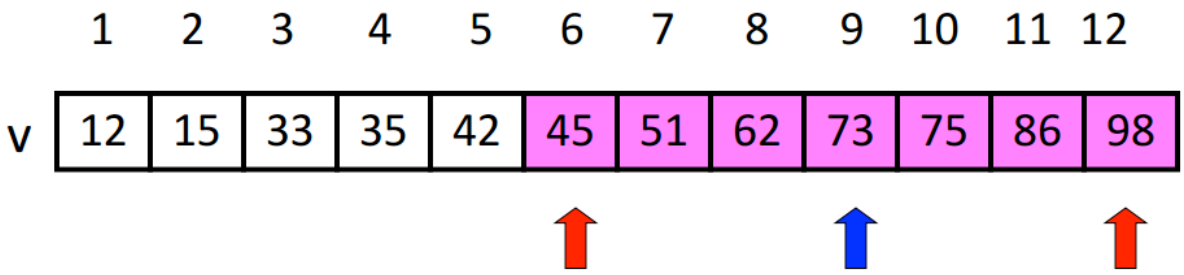
-

An item in a sorted array of length n can be located with just $\log_2 n$ comparisons

n	log2(n)
100	7
1000	10
10000	13

Big saving!!

We look for x = 70



x > 45 → x < 73 → x > 51 → nope

So the new search method that uses divide-and-conquer strategy is called **binary search**

```
def binarySearch(lst, value):
        """
        this is the binary search, but the condition is that we need to sort the list first
        and then we can implement the search algorithm
        """
        ## lst.sort()
```

Sorting

Sorting is the process of placing elements from a collection in some order.

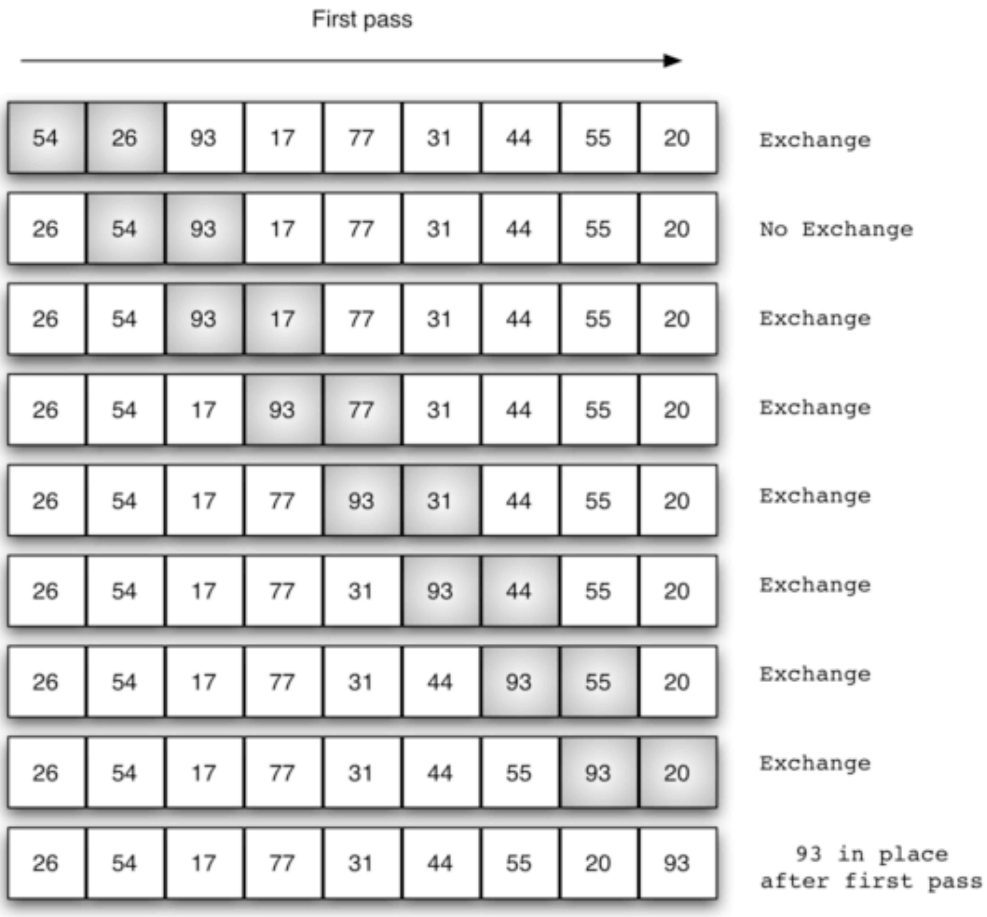
1. it is necessary to _____
2. when values are not in the correct position with respect to one another, it is necessary to exchange them
3. We need to look at the _____

implement sorting on a list. Do not use any python method.

```
def mySorting(alist):
    “””
    implement your algorithm here
    “””
    pass
```

1. Bubble Sort

Bubble sort makes multiple passes through a list. It compares the adjacent items and exchanges them if they are out of the order. Each pass places the larger one in the upper place. It looks like ‘bubble up’.



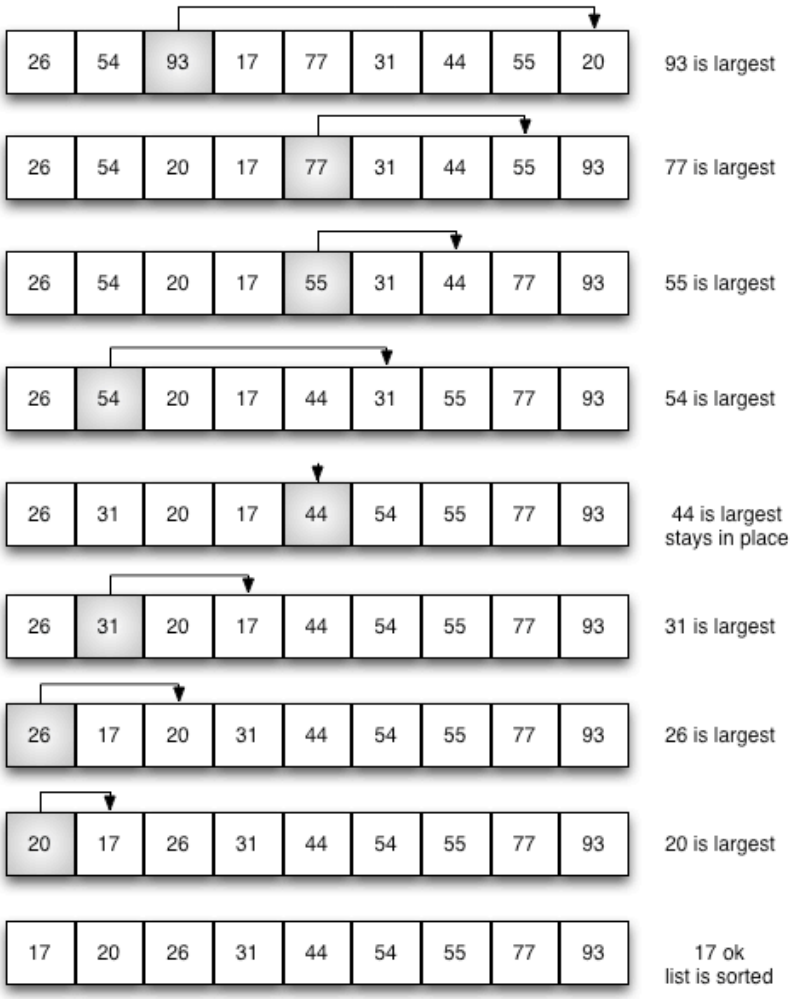
```
temp = alist[i]
alist[i] = alist[j]
alist[j] = temp

def bubbleSort(alist):
        """
        compare the adjacent items and bubble up
        """
```

Quiz:
What is the order of the list after 3 rounds of bubble sort?
>>> [19, 1, 9, 7, 3, 10, 13, 15, 8, 12]

2. Selection Sort

We look for the largest value for each pass, and we place the largest value at the end. So we need n-1 passes to complete sorting n items. For each pass, we need to check every rest items.

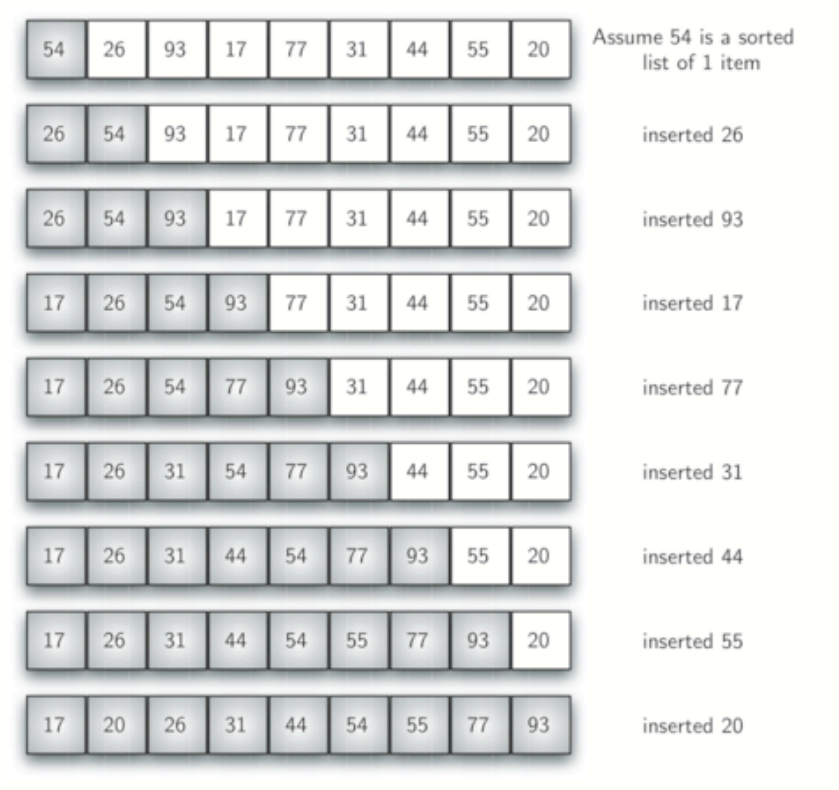


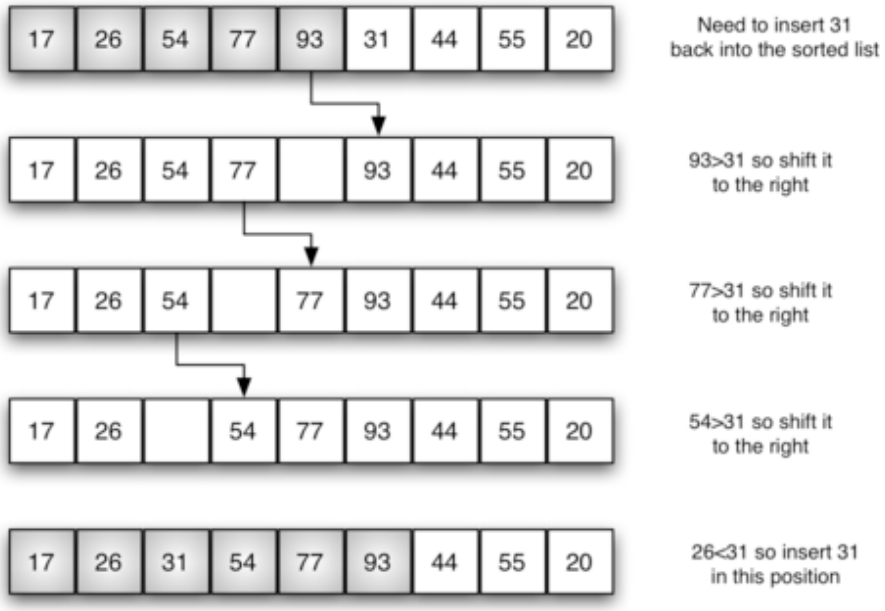
```
def selectionSort(alist):
        """
        put the largest value at the end and iterate
        """
```

Quiz:
What's the order of the list after 3 rounds of selection sort?
>>> [11, 7, 12, 14, 19, 1, 6, 18, 8, 20]

3. Insertion Sort

We change the algorithm a little different. We start from the left and 'insert' the new item back to the sublist that we have checked. The checked sublist is already sorted.





The above procedure shows how 31 is inserted in to the sublist.

Quiz:
What is the list after 3 passes of insertion sort?
>>> [15, 5, 4, 18, 12, 19, 14, 10, 8, 20]

Algorithm:

```
def insertionSort(alist):
        """
        loop through the list, and insert new item into the list that you have checked on the
left
        """
```

4. Merge Sort

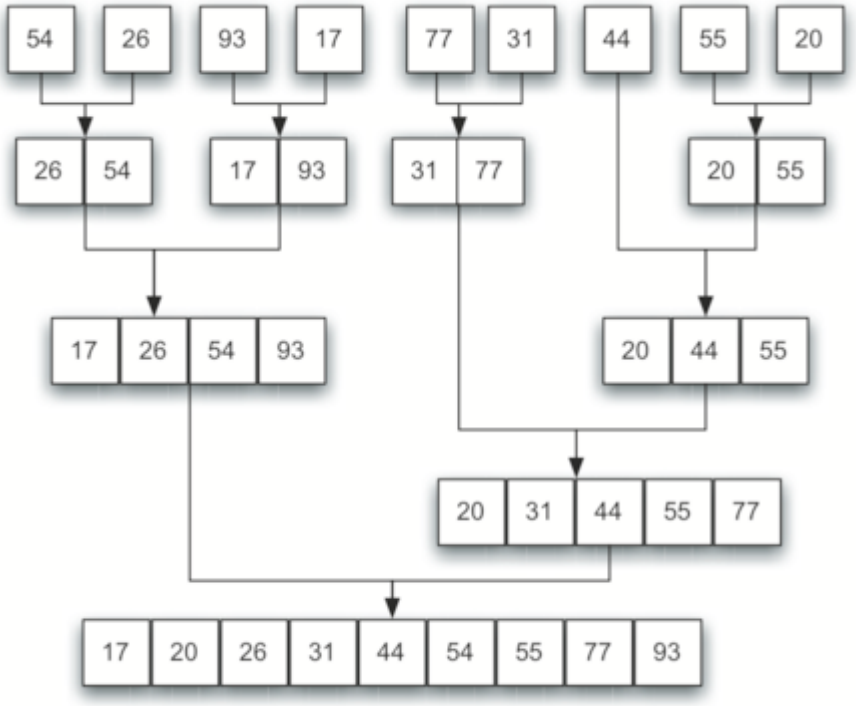
We use divide-and-conquer strategy. We recursively split the list into half.

- Base case: _____
- Else: _____

Merge process is taking 2 smaller sorted lists and combining them into a single sorted list.

First step:

Second step:



Quiz:
What are the sublists to sort after 3 recursive calls?
>>> [21, 1, 26, 45, 29, 28, 2, 9, 16, 49, 39, 27, 43, 34, 46, 40]

```
def mergeSort(alist):
        """
        the algorithm should achieve splitting a list into half and merging them together
        """
```