ICS Problem Set 4

Exercise 1 - All subsets (1 point)

Write a function called `all_subsets(lst)` which returns all the subsets of `lst`

Examples:

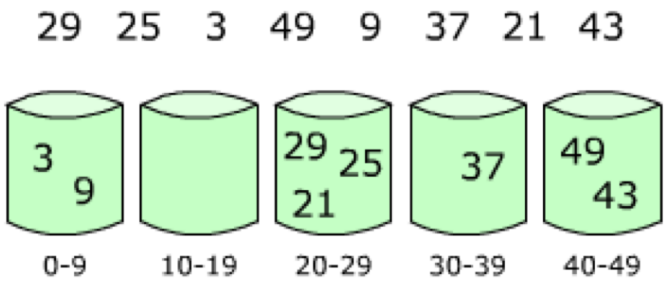
Given		Return
`[1]`	→	`[[], [1]]`
`[1, 2]`	→	`[[], [1], [2], [1, 2]]`

Bucket Sort

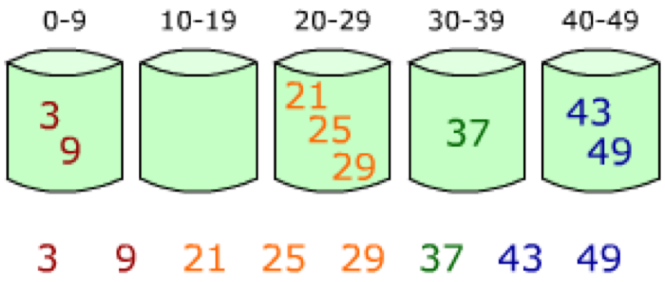
We have learned bubble sort and merge sort. Another very useful sorting algorithm in practice is *bucket sort*. The intuition is simple:

- You throw items into a set of buckets, each is responsible for a *fixed* and *constant* range
- Sort items within a bucket using another other sorting algorithms
- Go through the buckets in order, you are done!

Step 1: distribute items into the buckets



Step 2: sort items within the buckets



Implement bucket sort, we will sort a list of random integers range in 0~99, and will have every bucket responsible for exactly the same range. So the first bucket takes numbers from 0 to 9, and second 10 to 19, and so on.

Exercise 2 – Using the procedural programming paradigm (2 points)

The following hints are ways to implement the above intuition:

- Use the starting code provided from **bucket_sort_student.py**
- Use a dictionary to implement buckets, each bucket holds a list.
- Distribute items into the list in each bucket. In our case, if x is a number in the list, x//10 will find its bucket. If the bucket doesn't exist, start a list with just x, otherwise append to the list.
- For each bucket, sort its list; you can use built-in list sort, i.e. `mylist.sort()` will sort `mylist` in ascending order
- Then go through the dictionary in order of keys, using `sorted(mydict.keys())`.

Exercise 3 – Using the OOP programming paradigm (2 points)

Use the starting code provide from **bucket_sort_oop_student.py**

Define a `Bucketsort` class: it should have 5 attributes,

- `data` to store the input list
- `bucketWidth`
- `sorted` which is a Boolean variable to indicate whether `data` is already sorted or not
- `buckets` which is a dictionary to store all buckets.

The `Bucketsort` class should contain 3 methods, apart from the `__init__`.

- `_ change_bucketWidth(self, newWidth):` this function update `bucketWidth`
- `_distributeElementsIntoBuckets(self)`: thisfunction distributes the data into the buckets.
- `sort(self, inputList):` this function sorts the elements in each bucket, merges the buckets, and update `data`.

Note: we can also hide the method in a class by starting the method's name with a "_" (i.e., weakly hidden) or "__"(strong hidden). A hidden method means this method is designed to be used only by the other methods that defined in the class, not for being called outside of that class.