Searching and Sorting Algorithms - Assignments

1. CSV Data ¶

The csv file `student_database.csv` in `data` folder contains list of students' data. Following are some samples.

```
StudID,LastName,FirstName
605729443,Lilah,Arnold
112147292,Alberto,Turner
404336848,Santiago,Krause
699024055,Edward,Duarte
```

Create Student Class

Implement a class `Student`.

- It has 3 instance variables, `stud_id`, `last_name` and `first_name`.
- Implement `__init__()` method to initialize its instance variables.
- Implement `__str__()` method to return a string in the format of `Student(605729443, 'Lilah', 'Arnold')`.

In [20]:

```python
class Student:

    def __init__(self, stud_id, last_name, first_name):
        self.stud_id = stud_id
        self.last_name = last_name
        self.first_name = first_name

    def __str__(self):
        return "{}({}, '{}', '{}')".format(self.__class__.__name__,
                                           self.stud_id, self.last_name, self.first_name)

```

Load Data from File

Write a function `load_students()`, which has a parameter `csv_path` pointing to the csv file.

- For each line of data it reads, it converts it to a `Student` instance.
- It returns list of students converts from the data in the csv file.

In [21]:

```
import csv

def load_students(csv_path):
    result = []
    with open(csv_path) as f:
        reader = csv.reader(f)
        header = next(reader)
        for row in reader:
            s = Student(*row)
            result.append(s)
    return result

```

In [22]:

```
students = load_students('data/student_database.csv')
print(len(students))
print(students[0])
```

```
200
Student(605729443, 'Lilah', 'Arnold')
```

2. Linear Search

Implement a function `find_student_linear()` , which search for a student in a csv file.

- The function takes 2 parameters: a list `students` which is list of Student instances, and a string `first_name` .
- It returns a student instance whose first name matches `first_name` parameter.
- It returns `None` if no matching student is found.

In [23]:

```
def find_student_linear(students, first_name):
    for i in range(len(students)):
        if students[i].first_name == first_name:
            return students[i]
    return None
```

In [25]:

```
s = find_student_linear(students, 'Ferrell')
print(s)
```

```
Student(136113514, 'Misael', 'Ferrell')
```

3. Bubble Sort

Implement a sorting function `sort_students_bubble()` which sorts a list of students using bubble sort algorithm.

- It takes in 2 parameters, `students` which is the list of unsorted students, and `field` which is the name of the field to be sorted.
- It does NOT affect the original `students` list, i.e. it returns a sorted list of students.

In [30]:

```
def sort_students_bubble(students, field):
    arr = list(students)
    for j in range(len(arr)-1, 0, -1):
        for i in range(0,j):
            if getattr(arr[i], field) > getattr(arr[i+1],field):
                arr[i], arr[i+1] = arr[i+1], arr[i]
    return arr

```

In [33]:

```
result = sort_students_bubble(students, 'first_name')
[i.first_name for i in result[0:10]]
```

Out[33]:

```
['Aguilar',
 'Alexander',
 'Ali',
 'Anderson',
 'Anderson',
 'Andrews',
 'Archer',
 'Arnold',
 'Arnold',
 'Atkinson']
```

4. Binary Search

Implement a binary search function `find_student_binary()` which takes in a list of `students` and a parameter `first_name` .

- It returns first student instance whose first_name matches `first_name` .

In [34]:

```
def find_student_binary(arr, first_name):
    left=0
    right=len(arr)-1

    while left<=right:
        mid=(left+right)//2
        if arr[mid].first_name==first_name:
            # bingo
            return arr[mid]
        else:
            if first_name>arr[mid].first_name:
                # move left pointer
                left = mid+1
            else:
                # move right pointer
                right = mid-1

    return None

```

In [36]:

```
sorted_students = result
s = find_student_binary(result, 'Giles')
print(s)
```

```
Student(635829987, 'Kaden', 'Giles')
```