**Run time analysis for Cricket Batting Order:**

**Available Functions:**

1. \_\_init\_\_(self)

* Initialization function

2. read\_file(self, inputfile)

* Reads input file and parses the data to form preference array

3. def assign(self)

* calculate the total number of unique batting charts such that every player gets exactly one batting position from their list of positions and no two players are given the same batting position in one batting chart.

**Time Complexity :**

**1**. def \_\_init\_\_(self, inputfile):

* This method has 1 "for" loop.
* Time Complexity = O(n)

**2.** read\_file(self, inputfile):

* This method has 3 "for" loops. 1 main "for" loop and 2 inner "for" loops.
* Main "for" loop reads each line from the input file with time complexity O(n)
* 2nd "for" loop which is dependent on main for loop splits every line string using "/" with time complexity O(n)
* 3nd "for" loop which is dependent on main for loop sets the prefered position with time complexity O(n)
* Even though this method has 2 inner loops both are independent of each other.
* Since inner "for" loops are dependent on the main "for" loop we know that this method has below time complexity.
* Time Complexity = O(n^2)

3. def assign(self):

* This method has 2 "while" loop and 1 "for" loop.
* Main "while" loop has a time complexity of O(n)
* Inner "while" loop which is dependent on main "while" loop has a time complexity of O(log n)
* Inner "for" loop which is independent of main "while" loop has a time complexity on O(n)
* Even though this method has 2 inner loops both are independent of each other.
* Thus we know that this method has below time complexity.
* TIme Complexity = O(nlogn)

**Over all time complexity of "Cricket Batting Order" algorithm is O(n^2)**