**Assignment 1**

**IR – Winter 2019 (CSL 436)**

**Maximum marks 5**

**Due Date: M.Tech and Ph.D. students 19th September 2019**

**B. Tech Students: 20th September 2019**

**Late submissions will have a penalty (Early submissions will be possible with mutual convenience)**

Implement in-memory array-based version of inverted indices presented in the notes, including all 3 versions of the next method: binary search, sequential search and galloping search. Using your implementation of inverted indices, implement the phrase searching algorithm given in the notes.

To test your phrase searching implementation, you will need a corpus of 256 MB or larger that fits comfortably in main memory of your computer. The English Wikipedia could be a good corpus to use. A subset of that is also fine. Check out how big it is.

Select at least 10,000 phrases of various lengths from the corpus and verify that your implementation successfully locates these phrases. Your selection should include short phrases of length 2-3, longer ones of length 4-10 and very long ones too. Include phrases containing frequent and infrequent terms both. At least half of the phrases should contain at least one very common term, such as an article or a preposition.

Following guidelines from notes, compare the efficiency of phrase searching, using your three versions of the next method. Compute average response times. Plot response times against phrase length for all three versions. For clarity you may need to plot the results for linear scan separately from those of the other two methods.

Select another set of phrases all with length 2. Include phrases containing combinations of frequent and infrequent terms. Plot the response times against L (the length of the longest posting list) for all three versions.

Implement the phrase searching algorithm described below:

NextPhrase2 (t1, t2, t3……. tn, position)

u = next (t1, position)

v = u

for i = 2 to n do

v = next (ti,v)

if v= infinity then

return [ infinity, infinity]

if v – u = n – 1 then

return [ u, v]

else

return next Phrase2(t1,t2,…..tn, v-n)

Repeat the performance measurements using this implementation. Report the results.