CSE508 - INFORMATION RETRIEVAL

Report - Assignment 1

ANTARA DAS - MT22014
INDRAAYUDH TALUKDAR - MT22031
ARUN SEN - MT22018

---

1. Data Preparation

1.1 Relevant Text Extraction :

A new directory called temp_dir is created and using the ZipFile library all the documents of CSE508_Winter2023_Dataset.zip are extracted into temp_dir directory. BeautifulSoup library is used to extract the contents between <TITLE>...</TITLE> and <TEXT>...</TEXT> tags , concatenate those 2 string contents using blank space.

1.2 Text Preprocessing :

For this we have tried two different methods.

i) Using a customized order of the preprocessing steps :

Each of the following steps are performed on all of the files in temp_dir and the modified text is stored back in the original files. This method is more efficient in terms of the number of computational steps and efforts. After performing these steps the modified dataset is stored in CSE508_Winter2023_Dataset2_preprocessed.zip and downloaded locally.

- **Lowercasing :** The text contents of the file are converted into lowercase letters
- **Punctuations and Special Characters Removal :** All characters in the file which are not a word or a number or blank-space are replaced by a blank-space using a single substitution operation utilizing regular expression

- ➢ **Stopwords Removal :** All the stop-words defined by python's NLTK library are fetched into a list. Then using regular expression those stopwords are recognized from the file content and then simply replaced them by a blank-space character
- ➢ **Tokenization :** Using word_tokenize method from NLTK library the file content is converted into a collection of tokens. Those tokens are then stored back into the file as comma separated values.
- ➢ **Blank-space-tokens Removal :** Out of all the tokens generated in the previous step, all blank-space tokens are removed and the filtered out tokens are again stored back into the file as comma separated values.

ii) Using the given order of the preprocessing steps :

Each of the following steps are performed on all of the files in temp_dir and the modified text is stored back in the original files. After performing these steps the modified dataset is stored in CSE508_Winter2023_Dataset3_preprocessed.zip and downloaded locally.

- ➢ **Lowercasing :** The text contents of the file are converted into lowercase letters
- ➢ **Tokenization :** Using word_tokenize method from NLTK library the file content is converted into a collection of tokens. Those tokens are then stored back into the file as comma separated values.
- ➢ **Stopwords Removal :** All the stop-words defined by python's NLTK library are fetched into a list. Then these stop-words are removed from the list of tokens generated in the previous step by creating a filtered list using list comprehension
- ➢ **Punctuations and Special Characters Removal :** Out of all the tokens generated in the previous step, which are not a word or a number or blank-space are recognized using regular expressions and replaced by a blank-space using three step substitution operation, to make sure that the commas which are used as separators of the tokens don't get removed in the process.
- ➢ **Blank-space-tokens Removal :** Out of all the tokens generated in the previous step, all blank-space tokens are removed and the filtered out tokens are again stored back into the file as comma separated values.

Sample File contents after performing all preprocessing steps :

```
********** File_1 **********
effect,mach,number,boundary,layer,transition,presence,pressure,rise,surface,roughness,ogive-cylinder,body,cold,wall,conditions,effect,mach,number,va
riation,1.8,7.4,boundary-layer,transition,investigated,slender,fin-stabilized,ogive-cylinder,body,free,flight,constant,length,reynolds,number,13.8,m
illion,wall,free-stream,temperature,ratio,constant,value,1.0,mach,number,4.5,value,test,showed,increasing,mach,number,favorable,effect,increasing,ex
tent,laminar,boundary,layer,given,surface,roughness,transition,data,plotted,function,factor,indicative,heat,transfer,showed,heat,transfer,possibly,r
esponsible,good,deal,increase,transition,reynolds,number,mach,number,transition,found,occur,farther,forward,sheltered,side,body,windward,side,angle
s,attack,low,0.4,mach,numbers,pressure,rise,along,sheltered-side,streamlines,examined,found,pressure-rise,coefficient,transition,point,showed,variat
ion,mach,number,data,sources,different,test,conditions,reduced,values,pressure-rise,coefficient,also,found,correlate,well,present,investigation,exce
ption,data,low,subsonic,mach,numbers,present,results,also,show,mach,number,surface,roughness,pressure,rise,length,reynolds,number,affected,boundary-
layer,transition,region,theoretical,infinite,laminar,stability,small,two-dimensional,disturbances,calculated,flat,plate,zero,pressure,gradient

********** File_2 **********
flow,compressible,fluid,past,sphere,flow,compressible,fluid,past,sphere,fixed,uniform,stream,calculated,third,order,approximation,means,janzen-rayle
igh,method,velocity,pressure,distributions,surface,sphere,computed,terms,involving,fourth,power,mach,number,neglected,rayleigh's,calculation,shown,c
onsiderable,importance,local,velocity,sound,approached,sphere,critical,mach,number,,,value,mach,number,maximum,velocity,fluid,past,sphere,equal,loca
l,velocity,sound,calculated,second,third,approximations,found,respectively,

********** File_3 **********
aerodynamic,investigation,parabolic,body,revolution,mach,number,1,92,effects,annular,supersonic,jet,exhausting,base,aerodynamic,investigation,parabo
lic,body,revolution,conducted,mach,number,1.92,without,annular,supersonic,jet,exhausting,base,measurements,jet,inoperative,made,lift,drag,pitching,m
oment,radial,longitudinal,pressure,distributions,base,pressures,jet,operation,measurements,made,pressures,rear,body,primary,variables,angle,attack,r
atio,jet,velocity,freestream,velocity,ratio,jet,pressure,stream,pressure,results,jet,inoperative,showed,radial,pressures,body,varied,appreciably,dis
tribution,generally,employed,approximate,theories,linearized,solutions,lift,pitching,moment,center,pressure,gave,relatively,poor,predictions,experim
ental,results,analysis,several,theoretical,methods,calculating,pressure,distribution,wave,drag,showed,methods,gave,results,considerable,disagreemen
t,experimental,values,maximum,effects,jet,obtained,lower,ratio,jet,velocity,stream,velocity,highest,ratio,jet,pressure,stream,pressure,effects,amoun
ted,slight,decrease,foredrag,reduction,lift,shift,center,pressure,destabilizing,direction

********** File_4 **********
free-flight,measurements,static,dynamic,air-flow,properties,nozzles,calculated,charted,equilibrium,flow,two,types,frozen,flows,one,type,frozen,flow,
air,assumed,equilibrium,nozzle,reservoir,arbitrary,points,chemical,reactions,molecular,vibrations,became,frozen,type,assumed,molecular,vibrations,eq
uilibrium,throughout,nozzle,chemical,reactions,became,frozen,arbitrary,points,calculations,made,range,stagnation,pressures,10,000,poinds,per,square,
inch,absolute,stagnation,enthalpies,24,500,btu,per,pound,flow,properties,charted,temperature,pressure,density,velocity,dynamic,pressure,mach,number,
reynolds,number,molecular,weight,fraction,mass,flow,equilibrium,flow,properties,normal,shock,waves,also,included

********** File_5 **********
interaction,shock,waves,boundary,layers,note,effects,interaction,performance,supersonic,intakes,interaction,shock,waves,boundary,layers,important,ef
fects,many,problems,high-speed,flow,paper,written,guide,literature,subject,critical,review,present,state,knowledge,concerning,underlying,physical,pr
ocesses,practical,applications,clear,reader,although,substantial,progress,made,knowledge,still,far,complete,work,fundamental,nature,specific,applica
tions,needed,problem,understood,sufficiently,well,design,purposes,part,paper,describes,experiments,comparatively,simple,types,flow,designed,provide,
fundamental,information,assist,development,theory,experiments,show,interaction,depends,mainly,mach,reynolds,numbers,strength,shock,wave,particular,i
nteraction,shock,wave,laminar,boundary,layer,shown,produce,much,larger,effects,boundary,layer,turbulent,cases,effects,interaction,large,enough,serio
us,practical,consequences,found,boundary,layer,separates,surface,difference,interaction,laminar,turbulent,layers,arises,mainly,laminar,layer,separat
es,much,readily,adverse,pressure,gradient,details,interaction,downstream,separation,point,thus,depend,critically,behaviour,separated,layer,condition
s,reattaches,surface,many,features,found,fundamental,experiments,appear,also,practical,applications,considered,parts,ii,iii,paper,although,emphasis,
hero,performance,aerfoils,wings,moving,high,subsonic,speeds,importance,interaction,examples,supersonic,trailing,edges,supersonic,intakes,also,discus
sed,briefly,differences,interaction,laminar,turbulent,boundary,layers,often,source,serious,discrepancy,model,experiments,full-scale,conditions,small
-scale,models,therefore,frequently,essential,make,boundary,layer,turbulent,artificial,means,difficulties,involved,certain,promising,methods,briefly,
discussed,shown,experiments,models,transition,fixed,used,explain,number,aerodynamic,effects,encountered,transonic,flight,connected,occurrence,shock-
induced,separation,turbulent,boundary,layers,two-dimensional,aerofoils,straight,sweptback,wings,turbulent,separation,occurs,shocks,certain,strength,
applies,model,full-scale,conditions,full-scale,conditions,.,differences,magnitude,would,expected,pressure,recovery,along,separated,layer,shock,trail
ing,edge,affected,reynolds,number,little,information,present,available,point,repercussions,turbulent,separation,steady-motion,characteristics,aerofo
ils,wings,traced,associated,reduction,pressure,recovery,roar,surface,pressure,trailing,edge,controls,inter-relation,two,surfaces,/so,long,flow,trail
ing,edge,remains,subsonic/,particular,relative,movements,shock,waves,extents,local,regions,supersonic,flow,certain,unsteady-flow,characteristics,buf
feting,control,surface,separation,evidence,presented,influence,section,shape,occurrence,effects,separation,,,many,respects,information,relevant,turb
ulent,boundary,layers,scarce,notes,work,required,given,part,iv,paper
```

2. Boolean Queries

For building the Unigram inverted index from scratch firstly, all of the unique tokens present in the preprocessed dataset are fetched and those words are considered as unigrams. Then the unique Document-IDs in which each of these unigrams present are found iteratively and the unigrams and its corresponding sorted list of inverted indices is stored in a python dictionary, where each unique token acts as a key.

For boolean operators are implemented here : OR, AND, AND NOT, OR NOT
The number of comparisons required for performing each of these operations are computed. All these operations are implemented in linear time complexity.

- **T1 AND T2 :** Documents in which both of token T1 and T2 are present. From a set theory perspective it is (T1 ∩ T2). This task is implemented as finding the common document-IDs from two lists of inverted indices for T1 and T2.
- **T1 AND NOT T2 :** Documents in which token T1 is present but not token T2. From a set theory perspective it is (T1 ∩ T2′). This task is implemented as removing the document-IDs present in the list of inverted indices for T2 from the lists of inverted indices for T1.
- **T1 OR T2 :** Documents in which either token T1 or token T2 is present. From a set theory perspective it is (T1 ∪ T2). This task is implemented as merging two sorted lists of inverted indices for T1 and T2.
- **T1 OR NOT T2 :** Documents in which either token T1 is present or token T2 is not present.From a set theory perspective it is (T1 ∪ T2′), which can be also represented as (Universe − T2). This task is implemented as removing all the document-IDs that are present in the list of inverted indices of T2 from the set of all possible document-IDs, by iterating over the range of document-IDs starting and ending with the first and last document-ID present in posting list of T2 respectively.

All the user input sequences of any query are preprocessed using the preprocessing steps described before. For the complex queries consisting more than one boolean operator, operators are considered one by one starting from the extreme left moving towards the right. The first operator is applied on the first two tokens and from the second operator onwards the left operand is basically the result of previous operation and the right operand is the next available token present in the preprocessed input sequence. Comparisons required for performing each operator will be added cumulatively to get the total count of comparisons performed for computing the entire complex query.

Sample Test Case of Boolean query :

```
Query 1: body AND shapes OR well
Number of documents retrieved for query 1: 170
Names of the documents retrieved for query 1: cranfield0001.txt, cranfield0008.txt, cranfield0012.txt, cranfield0014.txt, cranfield0020.txt, cranfie
ld0022.txt, cranfield0044.txt, cranfield0059.txt, cranfield0063.txt, cranfield0077.txt, cranfield0079.txt, cranfield0089.txt, cranfield0094.txt, cra
nfield0096.txt, cranfield0115.txt, cranfield0122.txt, cranfield0126.txt, cranfield0146.txt, cranfield0152.txt, cranfield0160.txt, cranfield0165.txt,
cranfield0166.txt, cranfield0168.txt, cranfield0173.txt, cranfield0177.txt, cranfield0186.txt, cranfield0187.txt, cranfield0196.txt, cranfield0202.t
xt, cranfield0213.txt, cranfield0232.txt, cranfield0233.txt, cranfield0257.txt, cranfield0274.txt, cranfield0287.txt, cranfield0289.txt, cranfield03
06.txt, cranfield0307.txt, cranfield0330.txt, cranfield0343.txt, cranfield0356.txt, cranfield0363.txt, cranfield0369.txt, cranfield0373.txt, cranfie
ld0376.txt, cranfield0400.txt, cranfield0417.txt, cranfield0433.txt, cranfield0443.txt, cranfield0447.txt, cranfield0453.txt, cranfield0456.txt, cra
nfield0463.txt, cranfield0469.txt, cranfield0486.txt, cranfield0500.txt, cranfield0521.txt, cranfield0528.txt, cranfield0556.txt, cranfield0572.txt,
cranfield0599.txt, cranfield0602.txt, cranfield0606.txt, cranfield0620.txt, cranfield0622.txt, cranfield0626.txt, cranfield0630.txt, cranfield0636.t
xt, cranfield0637.txt, cranfield0640.txt, cranfield0648.txt, cranfield0686.txt, cranfield0688.txt, cranfield0697.txt, cranfield0703.txt, cranfield07
04.txt, cranfield0710.txt, cranfield0711.txt, cranfield0732.txt, cranfield0747.txt, cranfield0758.txt, cranfield0762.txt, cranfield0782.txt, cranfie
ld0794.txt, cranfield0796.txt, cranfield0798.txt, cranfield0801.txt, cranfield0808.txt, cranfield0809.txt, cranfield0814.txt, cranfield0815.txt, cra
nfield0816.txt, cranfield0825.txt, cranfield0830.txt, cranfield0847.txt, cranfield0850.txt, cranfield0874.txt, cranfield0887.txt, cranfield0890.txt,
cranfield0893.txt, cranfield0902.txt, cranfield0903.txt, cranfield0908.txt, cranfield0912.txt, cranfield0926.txt, cranfield0928.txt, cranfield0934.t
xt, cranfield0947.txt, cranfield0959.txt, cranfield0960.txt, cranfield0962.txt, cranfield0975.txt, cranfield0980.txt, cranfield1002.txt, cranfield10
23.txt, cranfield1024.txt, cranfield1026.txt, cranfield1027.txt, cranfield1047.txt, cranfield1052.txt, cranfield1053.txt, cranfield1065.txt, cranfie
ld1074.txt, cranfield1079.txt, cranfield1097.txt, cranfield1104.txt, cranfield1106.txt, cranfield1108.txt, cranfield1110.txt, cranfield1118.txt, cra
nfield1126.txt, cranfield1139.txt, cranfield1150.txt, cranfield1179.txt, cranfield1185.txt, cranfield1187.txt, cranfield1191.txt, cranfield1195.txt,
cranfield1197.txt, cranfield1198.txt, cranfield1201.txt, cranfield1218.txt, cranfield1225.txt, cranfield1229.txt, cranfield1239.txt, cranfield1240.t
xt, cranfield1241.txt, cranfield1247.txt, cranfield1268.txt, cranfield1269.txt, cranfield1271.txt, cranfield1273.txt, cranfield1274.txt, cranfield12
93.txt, cranfield1299.txt, cranfield1301.txt, cranfield1303.txt, cranfield1309.txt, cranfield1310.txt, cranfield1319.txt, cranfield1322.txt, cranfie
ld1332.txt, cranfield1367.txt, cranfield1370.txt, cranfield1371.txt, cranfield1373.txt, cranfield1375.txt, cranfield1377.txt, cranfield1380.txt, cra
nfield1381.txt
Number of comparisons required for query 1: 396


Query 2: profiles AND NOT match
Number of documents retrieved for query 2: 64
Names of the documents retrieved for query 2: cranfield0049.txt, cranfield0054.txt, cranfield0061.txt, cranfield0071.txt, cranfield0074.txt, cranfie
ld0076.txt, cranfield0082.txt, cranfield0088.txt, cranfield0094.txt, cranfield0101.txt, cranfield0165.txt, cranfield0168.txt, cranfield0171.txt, cra
nfield0177.txt, cranfield0178.txt, cranfield0189.txt, cranfield0206.txt, cranfield0211.txt, cranfield0218.txt, cranfield0219.txt, cranfield0233.txt,
cranfield0240.txt, cranfield0261.txt, cranfield0327.txt, cranfield0328.txt, cranfield0344.txt, cranfield0351.txt, cranfield0364.txt, cranfield0365.t
xt, cranfield0366.txt, cranfield0370.txt, cranfield0387.txt, cranfield0397.txt, cranfield0417.txt, cranfield0435.txt, cranfield0440.txt, cranfield04
74.txt, cranfield0491.txt, cranfield0495.txt, cranfield0534.txt, cranfield0538.txt, cranfield0547.txt, cranfield0557.txt, cranfield0569.txt, cranfie
ld0689.txt, cranfield0774.txt, cranfield0797.txt, cranfield0943.txt, cranfield0961.txt, cranfield0963.txt, cranfield0967.txt, cranfield0978.txt, cra
nfield0985.txt, cranfield1061.txt, cranfield1109.txt, cranfield1216.txt, cranfield1235.txt, cranfield1241.txt, cranfield1242.txt, cranfield1300.txt,
cranfield1326.txt, cranfield1375.txt, cranfield1383.txt, cranfield1386.txt
Number of comparisons required for query 2: 59
```

```
uii.input_output()          # call the input_output method using pkl object

Enter number of queries to execute : 2
Enter Input sequence : Details of Design, and Review.
Enter Operations separated by comma : AND,OR NOT
Enter Input sequence : Body in Shapes, is well !!!!
Enter Operations separated by comma : AND,AND NOT


Query 1: details AND design OR NOT review
Number of documents retrieved for query 1: 1377
Names of the documents retrieved for query 1: cranfield0001.txt, cranfield0002.txt, cranfield0003.txt, cranfield0004.txt, cranfield0005.txt, cranfield(
Number of comparisons required for query 1: 1443


Query 2: body AND shapes AND NOT well
Number of documents retrieved for query 2: 23
Names of the documents retrieved for query 2: cranfield0044.txt, cranfield0079.txt, cranfield0094.txt, cranfield0160.txt, cranfield0186.txt, cranfield(
Number of comparisons required for query 2: 396
```

```
Query 1: profiles OR match
Number of documents retrieved for query 1: 68
Names of the documents retrieved for query 1: cranfield0049.txt, cranfield0054.txt, cranfield0061.txt, cranfield0071.txt, cranfield0074.txt, cranfi
eld0076.txt, cranfield0082.txt, cranfield0088.txt, cranfield0094.txt, cranfield0101.txt, cranfield0138.txt, cranfield0165.txt, cranfield0168.txt, c
ranfield0171.txt, cranfield0177.txt, cranfield0178.txt, cranfield0189.txt, cranfield0206.txt, cranfield0211.txt, cranfield0218.txt, cranfield0219.t
xt, cranfield0233.txt, cranfield0240.txt, cranfield0261.txt, cranfield0327.txt, cranfield0328.txt, cranfield0344.txt, cranfield0351.txt, cranfield0
364.txt, cranfield0365.txt, cranfield0366.txt, cranfield0370.txt, cranfield0387.txt, cranfield0397.txt, cranfield0417.txt, cranfield0435.txt, cranf
ield0440.txt, cranfield0474.txt, cranfield0491.txt, cranfield0495.txt, cranfield0534.txt, cranfield0538.txt, cranfield0547.txt, cranfield0557.txt,
cranfield0569.txt, cranfield0689.txt, cranfield0774.txt, cranfield0797.txt, cranfield0943.txt, cranfield0961.txt, cranfield0963.txt, cranfield0967.
txt, cranfield0978.txt, cranfield0985.txt, cranfield1061.txt, cranfield1109.txt, cranfield1198.txt, cranfield1216.txt, cranfield1235.txt, cranfield
1241.txt, cranfield1242.txt, cranfield1251.txt, cranfield1300.txt, cranfield1307.txt, cranfield1326.txt, cranfield1375.txt, cranfield1383.txt, cran
field1386.txt
Number of comparisons required for query 1: 59
```

3. Phrase Queries

To find phrasal queries we use two methods of bigram inverted index and positional index as follows :

3.1 Bigram Inverted Index

For bigram inverted index, we are splitting our preprocessed tokens into word tokens and then we resort to nltk's bigram function to gather all the bigrams. We are storing all the bigrams for all the docs in a list of lists. We also gather all the unique bigrams together and create a dictionary using them as keys. The values will be lists consisting of docIDs of documents containing the bigrams respectively. All of these are encapsulated in a class called Bigram_Index.

For making the bigram inverted index, we just checked if the current bigram is in the document under question and the documentID is not yet added to the respective list of the bigram, we add the documentID to the list.

For processing the query, we preprocess the query first. After preprocessing we have tokens of the query. Next we prepare the bigrams from the word tokens we obtained. We maintain a set which will contain the documentIDs of documents of all the bigrams of the query explored previously. Now to find the documents containing the bigrams explored previously and also the current bigram, we do an intersection of the two sets. Finally, we get the intersection of all the docs. For the first bigram, union has to be done with a null set to initialize the set. Rest will follow the intersection. To get the document names, we keep aside a docID → docName mapping in a dictionary.

3.2 Positional Index

Another approach which we took for phrasal query resolution is positional indexing. Here we are maintaining a dictionary of dictionaries. For the mother dictionary, the keys are unigram terms. With a term as a key we will have a dictionary as value. In this internal dictionary, the document IDs where the term is stored will be used as the key and the corresponding value will be a list containing the positions of the term in the document. Also, we will create a list of lists which will cater to the tokens of the documents. A dictionary mapping the document ID to document name is maintained as well.

For creating the positional index, as mentioned above a dictionary of dictionaries is maintained. Now, at first we create a list of the unique terms. Next, we start parsing through the token list. The token list is parsed through indices. Parsing through indices helps us determine the position of the term or token in a document. Now for adding the positions, we dereference the indices of doc and the indices of the tokens for that doc as present in the list of lists. We take each position in a doc, and treat it as a key in the mother dictionary. Next, for that token in the doc, we add the doc as a key in its inner dictionary and we append the current position to the list (which is the value in the inner dictionary). Basically, what we are doing, for a particular document and given a position there, we use the token as key for the mother dictionary and the doc ID as key for the inner dictionary for a particular token. Then, we append the position of the token.

We assume that in our queries, we are expected to find documents which contain the whole phrase in order as it is given, barring the stopwords. We preprocess the query phrase to align with our data repository. We maintain a dictionary with the keys as the documents containing all the tokens. We parse over the tokens and then the docs associated with each token. Now to make sure the tokens are aligned consecutively, we decrement the positions of each token present in a document by their index. So if a token has position p in a document d and index i in the query, then it will become p-i. These decremented positions will be intersected with the positions of the first token. So for the docs which contain our query phrase, we will be left with the positions of the first token only. Now, we have gathered the docs ,but some docs might not contain our current token, so it has to be removed. For that, we make a copy of the dictionary with only the keys being docs containing our current token, since we already know that our previous words are contained in them. Also we are keeping a check that the intersection is not null, as null intersection indicates the tokens are not contiguous in a document.

The two forms of indexings are combined in a single function, where they are pickled and downloaded for further use.

Exact method of querying is depicted in the comments in the code file.

Output containing both the indices: (The picture below will show you the output)

```
+ Code + Text
bi=load(fb)
get_query_results(bi,pi)
fp.close()
fb.close()

enter number of queries:1
enter query number 1
stressing heated wings
Number of documents retrieved for query  1  using bigram inverted index: 1
Names of documents retrieved for query  1  using bigram inverted index: ['cranfield0013']
Number of documents retrieved for query  1  using positional index: 1
Names of documents retrieved for query  1  using positional index: ['cranfield0013']
```

```
enter number of queries:2
enter query number 1
stressing heated wings
Number of documents retrieved for query  1  using bigram inverted index: 1
Names of documents retrieved for query  1  using bigram inverted index: ['cranfield0013']
Number of documents retrieved for query  1  using positional index: 1
Names of documents retrieved for query  1  using positional index: ['cranfield0013']
enter query number 2
distribution conical bodies hypersonic
Number of documents retrieved for query  2  using bigram inverted index: 1
Names of documents retrieved for query  2  using bigram inverted index: ['cranfield0019']
Number of documents retrieved for query  2  using positional index: 1
Names of documents retrieved for query  2  using positional index: ['cranfield0019']
```

Contributions :

Indraayudh Talukdar 1(i), 3(ii), 3(iii)

Arun Sen - 2.1, 2.3.a, 2.3.c

Antara Das - 1(ii), Rest part of 2, 3(i)

[GitHub Repository Link]