1) PREPROCESSING

```
import os
import nltk
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
import string

# Ensure stopwords and punkt are downloaded
nltk.download('stopwords')
nltk.download('punkt')

# Directory of 1000 documents
directory_path = '/content/drive/MyDrive/1000_documents'

def preprocess_text(text):
    stop_words = set(stopwords.words('english'))
    stemmer = PorterStemmer()
    tokens = nltk.word_tokenize(text)
    tokens = [token.lower() for token in tokens if token.isalpha()]
    filtered_tokens = [stemmer.stem(token) for token in tokens
if token not in stop_words]
    return filtered_tokens

def preprocess_documents(directory_path):
    processed_docs = {}
    for filename in os.listdir(directory_path):
        file_path = os.path.join(directory_path, filename)
        with open(file_path, 'r', encoding='utf-8') as file:
            text = file.read()
            processed_docs[filename] = preprocess_text(text)
    return processed_docs

preprocessed_data = preprocess_documents(directory_path)
print(preprocessed_data)
```

2) SORT BASED INDEXING

```
import os

directory_path =
'/content/drive/MyDrive/1000_documents'

def sort_based_indexing(directory_path):
    inverted_index = {}
    for filename in os.listdir(directory_path):
        file_path = os.path.join(directory_path, filename)
        with open(file_path, 'r', encoding='utf-8') as file:
            content = file.read().split()
            for term in set(content):
                if term not in inverted_index:
                    inverted_index[term] = []
                inverted_index[term].append(filename)
    sorted_index = dict(sorted(inverted_index.items()))
    return sorted_index

sorted_index = sort_based_indexing(directory_path)
print(sorted_index)
```

3) BSBI

```
import os
from collections import defaultdict

directory_path = '/content/drive/MyDrive/1000_documents'

def bsbi_algorithm(directory_path):
    block_size = 100
    block_count = 0
    intermediate_indices = []

    for block_start in range(0, len(os.listdir(directory_path)),
block_size):
        block_count += 1
        block_files =
os.listdir(directory_path)[block_start:block_start + block_size]
        block_index = defaultdict(list)

        for filename in block_files:
            file_path = os.path.join(directory_path, filename)
            with open(file_path, 'r', encoding='utf-8') as file:
                content = file.read().split()
                for word in set(content):
                    block_index[word].append(filename)


intermediate_indices.append(dict(sorted(block_index.items())))
    return intermediate_indices

bsbi_indices = bsbi_algorithm(directory_path)
for block in bsbi_indices:
    print(block)
```

4)SPIMI

```
import os
from collections import defaultdict

directory_path = '/content/drive/MyDrive/ktextfiles'

def spimi_algorithm(directory_path, block_size=100):
    block_count = 0
    intermediate_indices = []
    filenames = sorted(os.listdir(directory_path))
    for block_start in range(0, len(filenames), block_size):
        block_count += 1
        block_files     =     filenames[block_start:block_start     +
block_size]
        block_index = defaultdict(list)

        for filename in block_files:
            file_path = os.path.join(directory_path, filename)
            with open(file_path, 'r', encoding='utf-8') as file:
                content = file.read().split()
                for word in set(content):
                    block_index[word].append(filename)

        intermediate_indices.append(block_index)

    global_index = merge_indexes(intermediate_indices)

    return global_index

def merge_indexes(intermediate_indices):
    final_index = defaultdict(list)

    for block_index in intermediate_indices:
        for term, doc_list in block_index.items():
            final_index[term].extend(doc_list)

    for term in final_index:
        final_index[term] = sorted(set(final_index[term]))

    return final_index

spimi_index = spimi_algorithm(directory_path)

for term, doc_list in sorted(spimi_index.items()):
    print(f"{term} -> {doc_list}")
```

5) Dynamic Logarithmic

```
import os
import math

directory_path                                                    =
'/content/drive/MyDrive/1000_documents'

def dynamic_logarithmic_indexing(directory_path):
    inverted_index = {}
    doc_count = len(os.listdir(directory_path))

    for filename in os.listdir(directory_path):
        file_path = os.path.join(directory_path, filename)
        with open(file_path, 'r', encoding='utf-8') as file:
            content = file.read().split()
            unique_terms = set(content)
            for term in unique_terms:
                if term not in inverted_index:
                    inverted_index[term] = []

idf=math.log(doc_count/(1+len(inverted_index[term])))
inverted_index[term].append((filename, idf))

    return inverted_index

dynamic_index=dynamic_logarithmic_indexing(directo
ry_path)
print(dynamic_index)
```