資工碩一 113598007 楊明哲

環境設置

1. Windows
    - 環境版本
        - Java : 17.0.12
        - Python : 3.10.10
        - PySpark : 3.5.3
        - Pandas : 2.2.3
    - 執行指令 spark-class org.apache.spark.deploy.master.Master，並打開 URL
        - 指令結果

```
PS C:\Users\Zhe> spark-class org.apache.spark.deploy.master.Master
Using Spark's default log4j profile: org/apache/spark/log4j2-defaults.properties
24/12/01 18:08:37 INFO Master: Started daemon with process name: 33692@DESKTOP-F8U5CHM
24/12/01 18:08:38 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
24/12/01 18:08:38 INFO SecurityManager: Changing view acls to: Zhe
24/12/01 18:08:38 INFO SecurityManager: Changing modify acls to: Zhe
24/12/01 18:08:38 INFO SecurityManager: Changing view acls groups to:
24/12/01 18:08:38 INFO SecurityManager: Changing modify acls groups to:
24/12/01 18:08:38 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users with view perm
issions: Zhe; groups with view permissions: EMPTY; users with modify permissions: Zhe; groups with modify permissions: E
MPTY
24/12/01 18:08:38 INFO Utils: Successfully started service 'sparkMaster' on port 7077.
24/12/01 18:08:38 INFO Master: Starting Spark master at spark://192.168.18.6:7077
24/12/01 18:08:38 INFO Master: Running Spark version 3.5.3
24/12/01 18:08:38 INFO JettyUtils: Start Jetty 0.0.0.0:8080 for MasterUI
24/12/01 18:08:38 INFO Utils: Successfully started service 'MasterUI' on port 8080.
24/12/01 18:08:38 INFO MasterWebUI: Bound MasterWebUI to 0.0.0.0, and started at http://DESKTOP-F8U5CHM:8080
24/12/01 18:08:38 INFO Master: I have been elected leader! New state: ALIVE
```

2. Ubuntu
    - 環境版本
        - Memory : 8GB
        - Processors : 16
        - Java : 17.0.12
        - Python : 3.10.12
        - Spark : 3.5.3
    - 執行 spark-class org.apache.spark.deply.worker.Worker spark://192.168.18.6:7077

資料夾結構

- homework3.py
- data
    - all of datasets
- output
    - all of output csv

源代碼

```python
import os
import random
import pandas as pd
from bs4 import BeautifulSoup
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, lower, regexp_replace, udf, explode
from pyspark.sql.types import ArrayType, StringType, IntegerType
from itertools import combinations

# Create SparkSession
spark = SparkSession.builder \
    .appName("Finding Similar Documents using LSH") \
    .config('spark.executor.memory','4g')\
    .config('spark.driver.memory','4g')\
    .config('spark.driver.maxResultsSize','0')\
    .getOrCreate()

# spark = SparkSession.builder.appName("Finding Similar Documents using LSH").getOrCreate()

# Q1 : Shingles matrix
# Parse SGML files
def parse_sgml(file_path):
    with open(file_path, 'r', encoding='latin-1') as file:
        content = file.read()
        soup = BeautifulSoup(content, "html.parser")
        bodies = soup.find_all('body')
        return [body.get_text(strip=True) for body in bodies]

# Load all files in the directory
def load_all_files(directory):
    data = []
    for file_name in os.listdir(directory):
        if file_name.endswith(".sgm"):  # Ensure it is a SGML file
            file_path = os.path.join(directory, file_name)
            bodies = parse_sgml(file_path)
            for body in bodies:
                data.append({'body': body})
    return pd.DataFrame(data)

# load files
directory_path = "./data/"
df = load_all_files(directory_path)

# Create Spark DataFrame
spark_df_origin = spark.createDataFrame(df)
```

```python
spark_df = spark_df_origin.withColumn('body', lower(col('body')))
spark_df = spark_df.withColumn('body', regexp_replace(col('body'), '[^\w\s]', ''))

# k-shingles
def generate_shingles(text, k=2):
    text = text.lower().replace(" ", "")
    return [text[i:i+k] for i in range(len(text) - k + 1)] if len(text) >= k else []

# UDF
shingling_udf = udf(lambda x: generate_shingles(x, k=3), ArrayType(StringType()))

# Apply UDF
spark_df = spark_df.withColumn('shingles', shingling_udf(col('body')))
shingles_df = spark_df.withColumn('shingle', explode(col('shingles'))).select('shingle')

# Shingles Count
shingles_matrix = shingles_df.groupBy('shingle').count().orderBy(col('count').desc())
shingles_matrix.show(truncate=False)

# Convert to Pandas DataFrame
shingles_matrix_pandas = shingles_matrix.toPandas()


# Save Shingles Matrix as CSV
output_path = "./output/shingles_matrix.csv"
shingles_matrix_pandas.to_csv(output_path, index=False)
print(f"Shingles matrix saved to {output_path}")


# Q2 : Min-Hash signature matrix
# Min-Hash signature
def minhash_signature(shingle_set, num_hashes, max_shingle_id):
    signature = []
    for i in range(num_hashes):
        a, b = random.randint(1, max_shingle_id), random.randint(0, max_shingle_id)
        # hash function : h(x) = (a*x + b) % max_shingle_id
        min_hash = min([(a * hash(shingle_id) + b) % max_shingle_id for shingle_id in shingle_set])
        signature.append(min_hash)
    return signature

# UDF for Min-Hash signature
def compute_signature_udf(shingle_list, num_hashes=100, max_shingle_id=100000):
    if shingle_list is None:
        return []
    shingle_set = set(shingle_list)
```

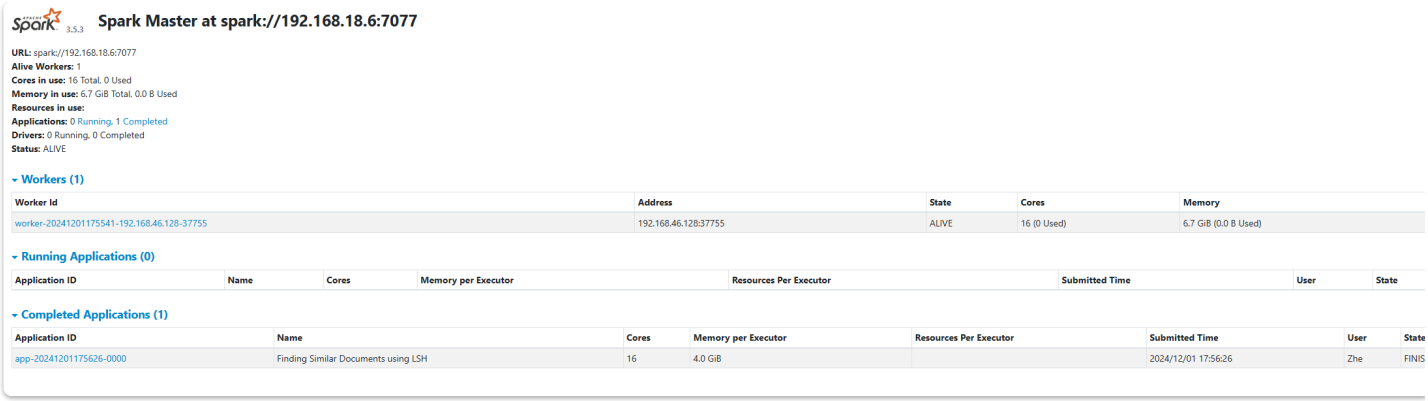
```python
    return minhash_signature(shingle_set, num_hashes, max_shingle_id)

def compute_minhash_signature_matrix(spark_df, num_hashes=100, max_shingle_id=100000):
    udf_signature = udf(lambda x: compute_signature_udf(x, num_hashes, max_shingle_id), ArrayType(IntegerType()))
    spark_df = spark_df.withColumn('signature', udf_signature(col('shingles')))
    #  Select only the signature column
    signature_df = spark_df.select(col('signature'))
    return signature_df



# Compute Min-Hash signature matrix
num_hashes = 100  # Min-Hash signatures length
max_shingle_id = 10000  # Maximum shingle ID
spark_df_with_signature = compute_minhash_signature_matrix(spark_df, num_hashes, max_shingle_id)

# Convert to Pandas DataFrame
signature_matrix_pandas = spark_df_with_signature.toPandas()

# Save Min-Hash signature matrix as CSV
signature_output_path = "./output/minhash_signature_matrix.csv"
signature_matrix_pandas.to_csv(signature_output_path, index=False)
print(f"Min-Hash signature matrix saved to {signature_output_path}")

# Q3 : Locality Sensitive Hashing (LSH)
# LSH
def lsh(signature_matrix, num_bands, rows_per_band):
    candidate_pairs = set()

    # for each band
    for band_idx in range(num_bands):
        # Set the start and end row for the current band
        start_row = band_idx * rows_per_band
        end_row = start_row + rows_per_band

        # band buckets
        buckets = {}

        for doc_id, row in signature_matrix.iterrows():
            band_signature = tuple(row['signature'][start_row:end_row])

            if band_signature not in buckets:
                buckets[band_signature] = []
            buckets[band_signature].append(doc_id)
```

```python
        print(f"Band {band_idx}: {len(buckets)} buckets")
        # find candidate pairs
        for bucket_docs in buckets.values():
            if len(bucket_docs) > 1:
                candidate_pairs.update(combinations(bucket_docs, 2))

    return candidate_pairs

# LSH parameters
num_bands = 25  # Number of bands
assert len(signature_matrix_pandas.iloc[0]['signature']) % num_bands == 0, "num_hashes 必須是 num_bands * rows_per_band 的倍數"
rows_per_band = len(signature_matrix_pandas.iloc[0]['signature']) // num_bands

# Find candidate pairs
candidate_pairs = lsh(signature_matrix_pandas, num_bands, rows_per_band)

# Print candidate pairs
print(f"Candidate pairs count: {len(candidate_pairs)}")
print(f"Candidate pairs: {candidate_pairs}")

# Save candidate pairs as CSV
candidate_pairs_output_path = "./output/candidate_pairs.csv"
candidate_pairs_df = pd.DataFrame(list(candidate_pairs), columns=['Doc1', 'Doc2'])
candidate_pairs_df.to_csv(candidate_pairs_output_path, index=False)
print(f"Candidate pairs saved to {candidate_pairs_output_path}")
```

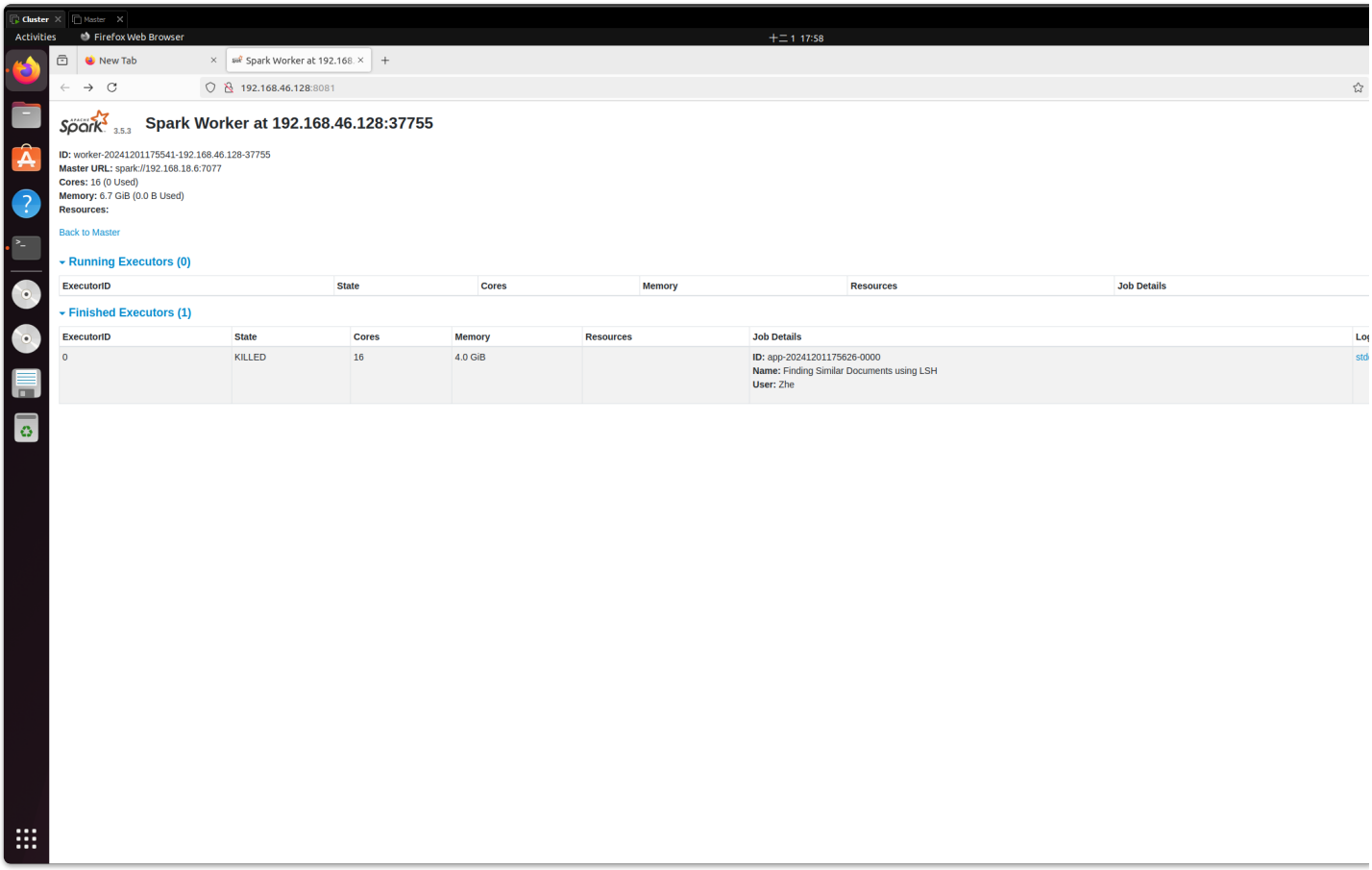
輸出結果

- 執行 spark-submit --master spark://192.168.18.6:7077 --conf spark.driver.host=192.168.18.6 homework3.py

Spark Master Web



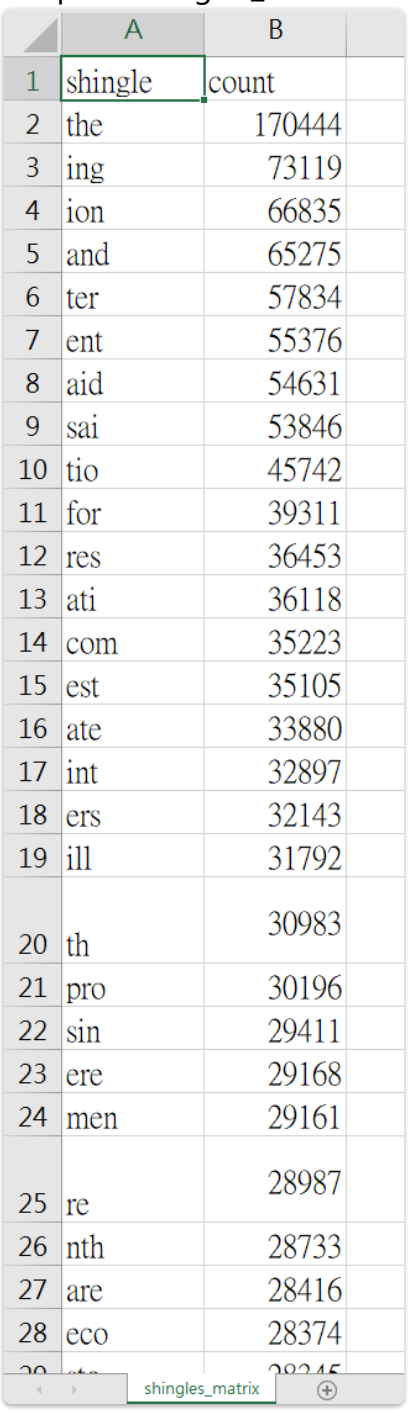
Spark Cluster Web



第一題

- set representation

- Output：shingles_matrix.csv



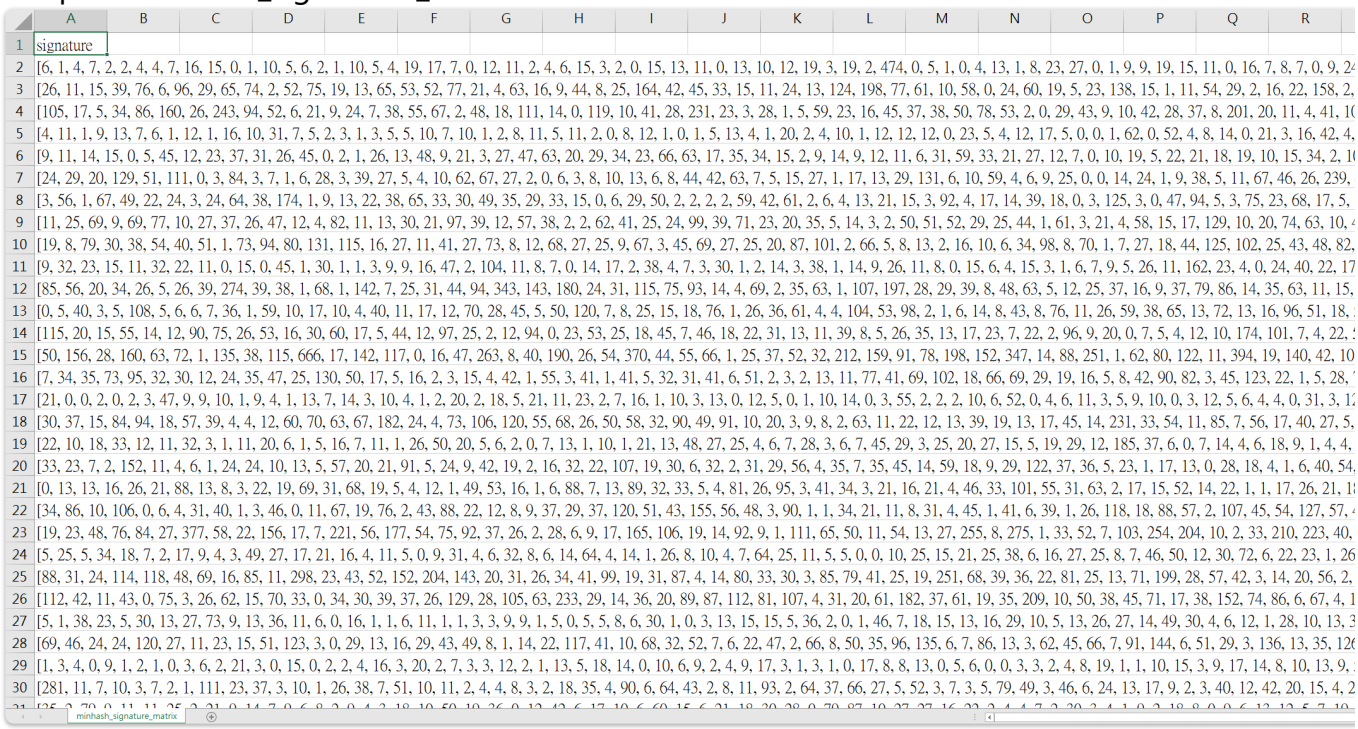
	A	B
1	shingle	count
2	the	170444
3	ing	73119
4	ion	66835
5	and	65275
6	ter	57834
7	ent	55376
8	aid	54631
9	sai	53846
10	tio	45742
11	for	39311
12	res	36453
13	ati	36118
14	com	35223
15	est	35105
16	ate	33880
17	int	32897
18	ers	32143
19	ill	31792
20	th	30983
21	pro	30196
22	sin	29411
23	ere	29168
24	men	29161
25	re	28987
26	nth	28733
27	are	28416
28	eco	28374

shingles_matrix

第二題

- minhash signatures

- Output：minhash_signatures_matrix.csv



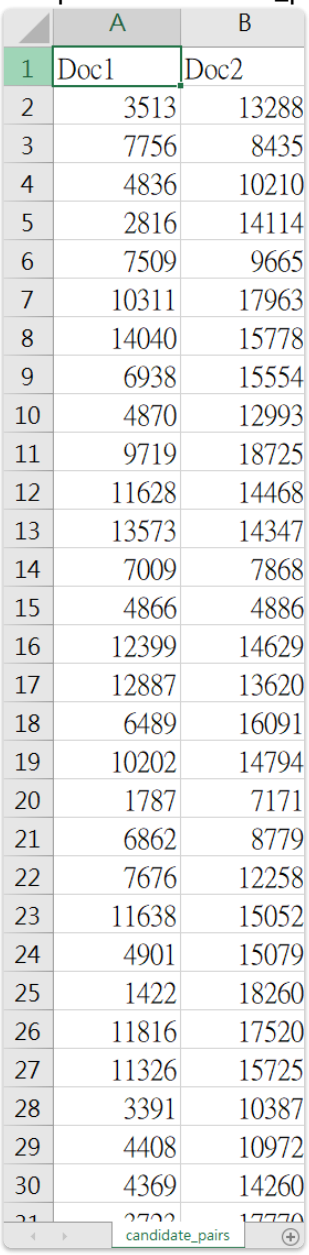
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	signature																	
2	[6, 1, 4, 7, 2, 2, 4, 4, 7, 16, 15, 0, 1, 10, 5, 6, 2, 1, 10, 5, 4, 19, 17, 7, 0, 12, 11, 2, 4, 6, 15, 3, 2, 0, 15, 13, 11, 0, 13, 10, 12, 19, 3, 19, 2, 474, 0, 5, 1, 0, 4, 13, 1, 8, 23, 27, 0, 1, 9, 9, 19, 15, 11, 0, 16, 7, 8, 7, 0, 9, 2																	
3	[26, 11, 15, 39, 76, 6, 96, 29, 65, 74, 2, 52, 75, 19, 13, 65, 53, 52, 77, 21, 4, 63, 16, 9, 44, 8, 25, 164, 42, 45, 33, 15, 11, 24, 13, 124, 198, 77, 61, 10, 58, 0, 24, 60, 19, 5, 23, 138, 15, 1, 11, 54, 29, 2, 16, 22, 158, 2,																	
4	[105, 17, 5, 34, 86, 160, 26, 243, 94, 52, 6, 21, 9, 24, 7, 38, 55, 67, 2, 48, 18, 111, 14, 0, 119, 10, 41, 28, 231, 23, 3, 28, 1, 5, 59, 23, 16, 45, 37, 38, 50, 78, 53, 2, 0, 29, 43, 9, 10, 42, 28, 37, 8, 201, 20, 11, 4, 41, 1(																	
5	[4, 11, 1, 9, 13, 7, 6, 1, 12, 1, 16, 10, 31, 7, 5, 2, 3, 1, 3, 5, 5, 10, 7, 10, 1, 2, 8, 11, 5, 11, 2, 0, 8, 12, 1, 0, 1, 5, 13, 4, 1, 20, 2, 4, 10, 1, 12, 12, 12, 0, 23, 5, 4, 12, 17, 5, 0, 0, 1, 62, 0, 52, 4, 8, 14, 0, 21, 3, 16, 42, 4,																	
6	[9, 11, 14, 15, 0, 5, 45, 12, 23, 37, 31, 26, 45, 0, 2, 1, 26, 13, 48, 9, 21, 3, 27, 47, 63, 20, 29, 34, 23, 66, 63, 17, 35, 34, 15, 2, 9, 14, 9, 12, 11, 6, 31, 59, 33, 21, 27, 12, 7, 0, 10, 19, 5, 22, 21, 18, 19, 10, 15, 34, 2, 1(																	
7	[24, 29, 20, 129, 51, 111, 0, 3, 84, 3, 7, 1, 6, 28, 3, 39, 27, 5, 4, 10, 62, 67, 27, 2, 0, 6, 3, 8, 10, 13, 6, 8, 44, 42, 63, 7, 5, 15, 27, 1, 17, 13, 29, 131, 6, 10, 59, 4, 6, 9, 25, 0, 0, 14, 24, 1, 9, 38, 5, 11, 67, 46, 26, 239,																	
8	[3, 56, 1, 67, 49, 22, 24, 3, 24, 64, 38, 174, 1, 9, 13, 22, 38, 65, 33, 30, 49, 35, 29, 33, 15, 0, 6, 29, 50, 2, 2, 2, 2, 59, 42, 61, 2, 6, 4, 13, 21, 15, 3, 92, 4, 17, 14, 39, 18, 0, 3, 125, 3, 0, 47, 94, 5, 3, 75, 23, 68, 17, 5,																	
9	[11, 25, 69, 9, 69, 77, 10, 27, 37, 26, 47, 12, 4, 82, 11, 13, 30, 21, 97, 39, 12, 57, 38, 2, 2, 62, 41, 25, 24, 99, 39, 71, 23, 20, 35, 5, 14, 3, 2, 50, 51, 52, 29, 25, 44, 1, 61, 3, 21, 4, 58, 15, 17, 129, 10, 20, 74, 63, 10, 4																	
10	[19, 8, 79, 30, 38, 54, 40, 51, 1, 73, 94, 80, 131, 115, 16, 27, 11, 41, 27, 73, 8, 12, 68, 27, 25, 9, 67, 3, 45, 69, 27, 25, 20, 87, 101, 2, 66, 5, 8, 13, 2, 16, 10, 6, 34, 98, 8, 70, 1, 7, 27, 18, 44, 125, 102, 25, 43, 48, 82,																	
11	[9, 32, 23, 15, 11, 32, 22, 11, 0, 15, 0, 45, 1, 30, 1, 1, 3, 9, 9, 16, 47, 2, 104, 11, 8, 7, 0, 14, 17, 2, 38, 4, 7, 3, 30, 1, 2, 14, 3, 38, 1, 14, 9, 26, 11, 8, 0, 15, 6, 4, 15, 3, 1, 6, 7, 9, 5, 26, 11, 162, 23, 4, 0, 24, 40, 22, 17																	
12	[85, 56, 20, 34, 26, 5, 26, 39, 274, 39, 38, 1, 68, 1, 142, 7, 25, 31, 44, 94, 343, 143, 180, 24, 31, 115, 75, 93, 14, 4, 69, 2, 35, 63, 1, 107, 197, 28, 29, 39, 8, 48, 63, 5, 12, 25, 37, 16, 9, 37, 79, 86, 14, 35, 63, 11, 15,																	
13	[0, 5, 40, 3, 5, 108, 5, 6, 6, 7, 36, 1, 59, 10, 17, 10, 4, 40, 11, 17, 12, 70, 28, 45, 5, 50, 120, 7, 8, 25, 15, 18, 76, 1, 26, 36, 61, 4, 4, 104, 53, 98, 2, 1, 6, 14, 8, 43, 8, 76, 11, 26, 59, 38, 65, 13, 72, 13, 16, 96, 51, 18,																	
14	[115, 20, 15, 55, 14, 12, 90, 75, 26, 53, 16, 30, 60, 17, 5, 44, 12, 97, 25, 2, 12, 94, 0, 23, 53, 25, 18, 45, 7, 46, 18, 22, 31, 13, 11, 39, 8, 5, 26, 35, 13, 17, 23, 7, 22, 2, 96, 9, 20, 0, 7, 5, 4, 12, 10, 174, 101, 7, 4, 22, 5																	
15	[50, 156, 28, 160, 63, 72, 1, 135, 38, 115, 666, 17, 142, 117, 0, 16, 47, 263, 8, 40, 190, 26, 54, 370, 44, 55, 66, 1, 25, 37, 52, 32, 212, 159, 91, 78, 198, 152, 347, 14, 88, 251, 1, 62, 80, 122, 11, 394, 19, 140, 42, 10																	
16	[7, 34, 35, 73, 95, 32, 30, 12, 24, 35, 47, 25, 130, 50, 17, 5, 16, 2, 3, 15, 4, 42, 1, 55, 3, 41, 1, 41, 5, 32, 31, 41, 6, 51, 2, 3, 2, 13, 11, 77, 41, 69, 102, 18, 66, 69, 29, 19, 16, 5, 8, 42, 90, 82, 3, 45, 123, 22, 1, 5, 28, 7																	
17	[21, 0, 0, 2, 0, 2, 3, 47, 9, 9, 10, 1, 9, 4, 1, 13, 7, 14, 3, 10, 4, 1, 2, 20, 2, 18, 5, 21, 11, 23, 2, 7, 16, 1, 10, 3, 13, 0, 12, 5, 0, 1, 10, 14, 0, 3, 55, 2, 2, 2, 10, 6, 52, 0, 4, 6, 11, 3, 5, 9, 10, 0, 3, 12, 5, 6, 4, 4, 0, 31, 3, 12																	
18	[30, 37, 15, 84, 94, 18, 57, 39, 4, 4, 12, 60, 70, 63, 67, 182, 24, 4, 73, 106, 120, 55, 68, 26, 50, 58, 32, 90, 49, 91, 10, 20, 3, 9, 8, 2, 63, 11, 22, 12, 13, 39, 19, 13, 17, 45, 14, 231, 33, 54, 11, 85, 7, 56, 17, 40, 27, 5,																	
19	[22, 10, 18, 33, 12, 11, 32, 3, 1, 11, 20, 6, 1, 5, 16, 7, 11, 1, 26, 50, 20, 5, 6, 2, 0, 7, 13, 1, 10, 1, 21, 13, 48, 27, 25, 4, 6, 7, 28, 3, 6, 7, 45, 29, 3, 25, 20, 27, 15, 5, 19, 29, 12, 185, 37, 6, 0, 7, 14, 4, 6, 18, 9, 1, 4, 4,																	
20	[33, 23, 7, 2, 152, 11, 4, 6, 1, 24, 24, 10, 13, 5, 57, 20, 21, 91, 5, 24, 9, 42, 19, 2, 16, 32, 22, 107, 19, 30, 6, 32, 2, 31, 29, 56, 4, 35, 7, 35, 45, 14, 59, 18, 9, 29, 122, 37, 36, 5, 23, 1, 17, 13, 0, 28, 18, 4, 1, 6, 40, 54,																	
21	[0, 13, 13, 16, 26, 21, 88, 13, 8, 3, 22, 19, 69, 31, 68, 19, 5, 4, 12, 1, 49, 53, 16, 1, 6, 88, 7, 13, 89, 32, 33, 5, 4, 81, 26, 95, 3, 41, 34, 3, 21, 16, 21, 4, 46, 33, 101, 55, 31, 63, 2, 17, 15, 52, 14, 22, 1, 1, 17, 26, 21, 1																	
22	[34, 86, 10, 106, 0, 6, 4, 31, 40, 1, 3, 46, 0, 11, 67, 19, 76, 2, 43, 88, 22, 12, 8, 9, 37, 29, 37, 120, 51, 43, 155, 56, 48, 3, 90, 1, 1, 34, 21, 11, 8, 31, 4, 45, 1, 41, 6, 39, 1, 26, 118, 18, 88, 57, 2, 107, 45, 54, 127, 57, 4																	
23	[19, 23, 48, 76, 84, 27, 377, 58, 22, 156, 17, 7, 221, 56, 177, 54, 75, 92, 37, 26, 2, 28, 6, 9, 17, 165, 106, 19, 14, 92, 9, 1, 111, 65, 50, 11, 54, 13, 27, 255, 8, 275, 1, 33, 52, 7, 103, 254, 204, 10, 2, 33, 210, 223, 40,																	
24	[5, 25, 5, 34, 18, 7, 2, 17, 9, 4, 3, 49, 27, 17, 21, 16, 4, 11, 5, 0, 9, 31, 4, 6, 32, 8, 6, 14, 64, 4, 14, 1, 26, 8, 10, 4, 7, 64, 25, 11, 5, 5, 0, 0, 10, 25, 15, 21, 25, 38, 6, 16, 27, 25, 8, 7, 46, 50, 12, 30, 72, 6, 22, 23, 1, 26																	
25	[88, 31, 24, 114, 118, 48, 69, 16, 85, 11, 298, 23, 43, 52, 152, 204, 143, 20, 31, 26, 34, 41, 99, 19, 31, 87, 4, 14, 80, 33, 30, 3, 85, 79, 41, 25, 19, 251, 68, 39, 36, 22, 81, 25, 13, 71, 199, 28, 57, 42, 3, 14, 20, 56, 2,																	
26	[112, 42, 11, 43, 0, 75, 3, 26, 62, 15, 70, 33, 0, 34, 30, 39, 37, 26, 129, 28, 105, 63, 233, 29, 14, 36, 20, 89, 87, 112, 81, 107, 4, 31, 20, 61, 182, 37, 61, 19, 35, 209, 10, 50, 38, 45, 71, 17, 38, 152, 74, 86, 6, 67, 4, 1																	
27	[5, 1, 38, 23, 5, 30, 13, 27, 73, 9, 13, 36, 11, 6, 0, 16, 1, 1, 6, 11, 1, 1, 3, 3, 9, 9, 1, 5, 0, 5, 5, 8, 6, 30, 1, 0, 3, 13, 15, 15, 5, 36, 2, 0, 1, 46, 7, 18, 15, 13, 16, 29, 10, 5, 13, 26, 27, 14, 49, 30, 4, 6, 12, 1, 28, 10, 13, 3																	
28	[69, 46, 24, 24, 120, 27, 11, 23, 15, 51, 123, 3, 0, 29, 13, 16, 29, 43, 49, 8, 1, 14, 22, 117, 41, 10, 68, 32, 52, 7, 6, 22, 47, 2, 66, 8, 50, 35, 96, 135, 6, 7, 86, 13, 3, 62, 45, 66, 7, 91, 144, 6, 51, 29, 3, 136, 13, 35, 126																	
29	[1, 3, 4, 0, 9, 1, 2, 1, 0, 3, 6, 2, 21, 3, 0, 15, 0, 2, 2, 4, 16, 3, 20, 2, 7, 3, 3, 12, 2, 1, 13, 5, 18, 14, 0, 10, 6, 9, 2, 4, 9, 17, 3, 1, 3, 1, 0, 17, 8, 8, 13, 0, 5, 6, 0, 0, 3, 3, 2, 4, 8, 19, 1, 1, 10, 15, 3, 9, 17, 14, 8, 10, 13, 9,																	
30	[281, 11, 7, 10, 3, 7, 2, 1, 111, 23, 37, 3, 10, 1, 26, 38, 7, 51, 10, 11, 2, 4, 4, 8, 3, 2, 18, 35, 4, 90, 6, 64, 43, 2, 8, 11, 93, 2, 64, 37, 66, 27, 5, 52, 3, 7, 3, 5, 79, 49, 3, 46, 6, 24, 13, 17, 9, 2, 3, 40, 12, 42, 20, 15, 4, 2																	
31	[25, 2, 70, 0, 11, 11, 25, 2, 21, 0, 14, 7, 0, 6, 8, 2, 0, 4, 2, 18, 10, 50, 10, 26, 0, 12, 42, 6, 17, 10, 6, 60, 15, 6, 21, 18, 30, 28, 0, 70, 87, 10, 27, 27, 16, 22, 2, 4, 4, 7, 2, 30, 2, 4, 1, 0, 2, 18, 8, 0, 0, 6, 13, 12, 5, 7, 10																	

minhash_signature_matrix

第三題

- candidate pairs

- Output：candidate_paris.csv



	A	B
1	Doc1	Doc2
2	3513	13288
3	7756	8435
4	4836	10210
5	2816	14114
6	7509	9665
7	10311	17963
8	14040	15778
9	6938	15554
10	4870	12993
11	9719	18725
12	11628	14468
13	13573	14347
14	7009	7868
15	4866	4886
16	12399	14629
17	12887	13620
18	6489	16091
19	10202	14794
20	1787	7171
21	6862	8779
22	7676	12258
23	11638	15052
24	4901	15079
25	1422	18260
26	11816	17520
27	11326	15725
28	3391	10387
29	4408	10972
30	4369	14260

小組分工

學號、姓名	貢獻比例	工作內容
楊明哲	100%	程式編寫、除錯以及文書處理