Problem 01 Python Code

November 29, 2020

[1]:

```python
import pandas as pd
import numpy as np
from itertools import product, chain, combinations
import itertools
from collections import defaultdict
import time
import pickle

# Systems
R1 = ['aa', 'ab', 'ai',
'bb', 'bc', 'bd',
'cc', 'cg', 'ci',
'db', 'dd',
'ec', 'ee', 'ej',
'fa', 'fh', 'fk',
'gc', 'gd', 'gi',
'hf', 'hh', 'hk',
'ic','ig', 'ii', 'ij',
'je', 'jh', 'jj',
'ka', 'kf', 'kh', 'kk']

R2 = ['11', '12', '13', '21',
      '22', '31', '33', '35',
      '41', '42', '44', '54', '55']

#Load Pickle Files for Total Sample Space
start_time = time.time()
with open('sampleSpace.pkl', 'rb') as f:
    SAMPLE_SPACE = pickle.load(f)
end_time = time.time()
print('Elapsed Time for loading `sampleSpace.pkl`: {}'.
 ↪format(end_time-start_time))

#Load Pickle Files for Homomorphic Sample Space
start_time = time.time()
with open('homomorphicSpace.pkl', 'rb') as f:
    HOMOMORPHIC_SPACE = pickle.load(f)
```

```
end_time = time.time()
print('Elapsed Time for loading `homomorphicSpace.pkl`: {}'.
 ↪format(end_time-start_time))
```

```
Elapsed Time for loading `sampleSpace.pkl`: 107.2454240322113
Elapsed Time for loading `homomorphicSpace.pkl`: 0.19091582298278809
```

[ ]:

```
#Find all Possible Mappings
start_time = time.time()
a = [list(each) for each in itertools.product('12345', repeat=11)]
space = [dict(zip([each for each in 'abcdefghijk'], i)) for i in a]
end_time = time.time()
print('Elapsed Time: {}'.format(end_time-start_time))
```

[2]:

```
#Total Possible Mappings Between S1 and S2
len(SAMPLE_SPACE)
```

[2]:

```
48828125
```

[3]:

```
def is_homomorphism(x, R1, R2):
    '''
    Returns True if R2 is a subset of initial_decode.
    '''
    initial_decode = [''.join(each) for each in [[x[each[0]], x[each[1]]] for␣
 ↪each in R1]]
    return set(initial_decode) == set(R2)


def is_strong_homomorphism(x, R1, R2):
    data = sorted(list(zip(list(x.values()), list(x.keys()))))
    d = defaultdict(list)
    for r1, r2 in data:
        d[r1].append(r2)
    h_x_inv = dict(d)

    initial_decode = [''.join(each) for each in [[x[each[0]], x[each[1]]] for␣
 ↪each in R1]]

    second_pos = []
    nested_decode = []
    dict_decode = {}

    for eachi in initial_decode:
        a = [h_x_inv[each[0][0]] for each in eachi]
        cart_prod_i = [element for element in itertools.product(a[0], a[1])]
        d = [''.join(each) for each in cart_prod_i]
        nested_decode.append(d)
        dict_decode[eachi] = d
    aaa = [i for j in [dict_decode[each] for each in R2] for i in j]
    return set(R1).issubset(aaa)
```

[4]:

```
#Find All Homomorphisms
start_time = time.time()

homomorphic_solutions = []
except_index = []
for e in SAMPLE_SPACE:
    try:
        if is_homomorphism(x=e, R1=R1, R2=R2):
            homomorphic_solutions.append(e)
        else:
            pass
    except:
        except_index.append(e)
        print('Exception: {}'.format(e))

end_time = time.time()
print('Elapsed Time: {:.3f} min'.format((end_time-start_time)/60))
print(len(homomorphic_solutions))
print(except_index)
```

```
Elapsed Time: 13.885 min
14
[]
```

[5]:

```
strong_homomorphic_solutions = []
except_index = []
for e in HOMOMORPHIC_SPACE:
    try:
        if is_strong_homomorphism(x=e, R1=R1, R2=R2):
            strong_homomorphic_solutions.append(e)
        else:
            pass
    except:
        except_index.append(e)
        print('Exception: {}'.format(e))

end_time = time.time()
print('Elapsed Time: {:.3f} min'.format((end_time-start_time)/60))
print(len(strong_homomorphic_solutions))
print(except_index)
```

```
Elapsed Time: 14.084 min
14
```

```
[]
```

```
[6]: #Example of 1 of 14 Strong Homomorphic Solutions
     example_map = strong_homomorphic_solutions[0]
     example_map
```

```
[6]: {'a': '4',
      'b': '2',
      'c': '1',
      'd': '1',
      'e': '1',
      'f': '5',
      'g': '1',
      'h': '5',
      'i': '1',
      'j': '3',
      'k': '5'}
```

```
[18]: #Forward Decode
      initial_decode = [''.join(each) for each in [[example_map[each[0]],␣
       ↪example_map[each[1]]] for each in R1]]
      forward_decode = dict(zip(R1, initial_decode))
      forward_decode
```

```
[18]: {'aa': '44',
       'ab': '42',
       'ai': '41',
       'bb': '22',
       'bc': '21',
       'bd': '21',
       'cc': '11',
       'cg': '11',
       'ci': '11',
       'db': '12',
       'dd': '11',
       'ec': '11',
       'ee': '11',
       'ej': '13',
       'fa': '54',
       'fh': '55',
       'fk': '55',
       'gc': '11',
       'gd': '11',
       'gi': '11',
       'hf': '55',
       'hh': '55',
       'hk': '55',
       'ic': '11',
       'ig': '11',
       'ii': '11',
       'ij': '13',
       'je': '31',
       'jh': '35',
       'jj': '33',
       'ka': '54',
       'kf': '55',
       'kh': '55',
       'kk': '55'}
```

[17]:

```
#Reverse Decode
second_pos = []
nested_decode = []
dict_decode = {}

for eachi in initial_decode:
    a = [h_x_inv[each[0][0]] for each in eachi]
    cart_prod_i = [element for element in itertools.product(a[0], a[1])]
    d = [''.join(each) for each in cart_prod_i]
    nested_decode.append(d)
    dict_decode[eachi] = d
reverse_decode = dict_decode
reverse_decode
```

[17]:

```
{'44': ['aa'],
 '42': ['ab'],
 '41': ['ac', 'ad', 'ae', 'ag', 'ai'],
 '22': ['bb'],
 '21': ['bc', 'bd', 'be', 'bg', 'bi'],
 '11': ['cc',
  'cd',
  'ce',
  'cg',
  'ci',
  'dc',
  'dd',
  'de',
  'dg',
  'di',
  'ec',
  'ed',
  'ee',
  'eg',
  'ei',
  'gc',
  'gd',
  'ge',
  'gg',
  'gi',
  'ic',
  'id',
  'ie',
  'ig',
  'ii'],
 '12': ['cb', 'db', 'eb', 'gb', 'ib'],
 '13': ['cj', 'dj', 'ej', 'gj', 'ij'],
 '54': ['fa', 'ha', 'ka'],
 '55': ['ff', 'fh', 'fk', 'hf', 'hh', 'hk', 'kf', 'kh', 'kk'],
 '31': ['jc', 'jd', 'je', 'jg', 'ji'],
 '35': ['jf', 'jh', 'jk'],
 '33': ['jj']}
```

[ ]:

```python
#Initial Code to Find all Homomorphisms
start_time = time.time()

homomorphic_solutions = []
except_index = []
for e in SAMPLE_SPACE:
    try:
        if is_homomorphism(x=e, R1=R1, R2=R2):
            homomorphic_solutions.append(e)
        else:
            pass
    except:
        except_index.append(e)
        print('Exception: {}'.format(e))

end_time = time.time()
print('Elapsed Time: {:.3f} min'.format((end_time-start_time)/60))
print(len(homomorphic_solutions))
print(except_index)
```

[ ]:

```python
#Save var `homomorphic_solutions` as pkl file.
with open('homomorphicSpace.pkl', 'wb') as f:
    pickle.dump(homomorphic_solutions, f)
```