discover

February 9, 2024

[1]:
```
import pandas as pa
data = pa.read_csv("./Embedded Software Engineer Quiz Resource - Sheet1.csv")
print(data)
```

```
    input output
0   15840    cGp
1   16465    cmW
2   17941    cX3
3   17942    cXB
4   18898    ctR
..    …     …
70  76082    nIv
71  79467    VXS
72  79564    VYw
73  79790    VeK
74  79791    Vel

[75 rows x 2 columns]
```

[2]:
```
import numpy as np
npdata = data.to_numpy()
print(npdata)
```

```
[[15840 'cGp']
 [16465 'cmW']
 [17941 'cX3']
 [17942 'cXB']
 [18898 'ctR']
 [19172 'ckN']
 [20512 'PVg']
 [20626 'PD4']
 [20758 'PBR']
 [25736 'MYM']
 [25893 'MSL']
 [26039 'MUk']
 [26134 'MIE']
 [26345 'Mgr']
 [26346 'MgK']
 [26676 'MKd']
```

```
[28161 'GVT']
[31622 'vNx']
[31873 'vwP']
[32028 'vV9']
[35260 'zAj']
[36368 'zO9']
[36428 'zu1']
[38716 'AcO']
[38805 'APF']
[40111 'AHl']
[40893 'ALW']
[40956 'AQ9']
[40957 'AQs']
[40958 'AQq']
[40959 'AQL']
[40960 'AQQ']
[42541 'Zcz']
[47987 'yuk']
[49137 'yh2']
[49169 'yIi']
[49443 'yau']
[49444 'yap']
[49639 'ytL']
[50047 'T7T']
[50048 'T7N']
[50127 'Tx8']
[50128 'TxE']
[50129 'Tx2']
[51183 'Tn2']
[51184 'Tn1']
[51204 'Tnf']
[52071 'T2a']
[52977 'Thu']
[54650 'NTO']
[55908 'N2e']
[56100 'N9g']
[57924 'ocd']
[60538 'oeH']
[60539 'oe0']
[60540 'oeO']
[61474 'okE']
[63141 'dH6']
[64436 'dUw']
[64437 'dUn']
[67416 'm2D']
[72677 'wtT']
[73039 'nCi']
[73040 'nCc']
```

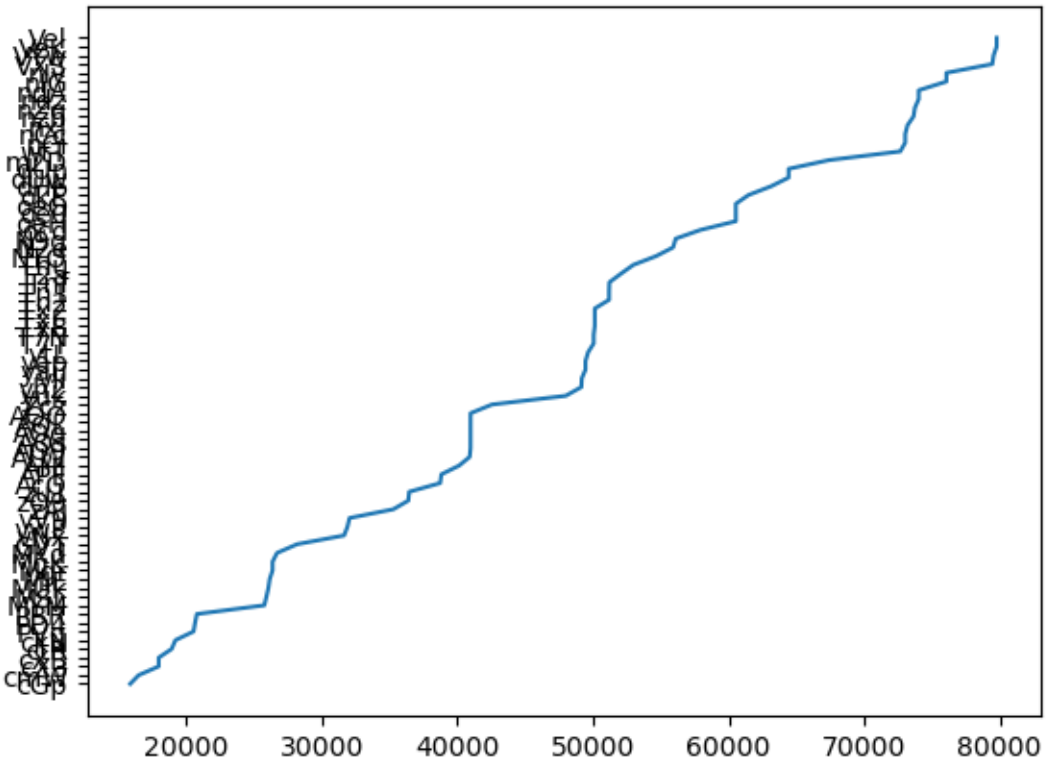
```
[73211 'nxJ']
[73654 'nzb']
[73756 'nZq']
[74037 'ndz']
[74038 'ndA']
[76081 'nIG']
[76082 'nIv']
[79467 'VXS']
[79564 'VYw']
[79790 'VeK']
[79791 'Vel']]
```

[3]:
```
print(npdata[0])
print(npdata[1])
print(npdata[2])
```

```
[15840 'cGp']
[16465 'cmW']
[17941 'cX3']
```

[4]:
```
#Can we find the pattern from the graph? No, this question doesn't so easy
import matplotlib.pyplot as plt
plt.plot(*zip(*npdata))
# Nope
```

[4]:
```
[<matplotlib.lines.Line2D at 0x14a84e750>]
```



[5]:
```
#Can we find the pattern from the same last char?
# firstly, collect the item by last char.
lastCharGroup = {}
for x in npdata:
    if x[1][-1] not in lastCharGroup:
        lastCharGroup[x[1][-1]] = [x[0]]
    else:
        lastCharGroup[x[1][-1]].append(x[0])

print(lastCharGroup)
```

```
{'p': [15840, 49444], 'W': [16465, 40893], '3': [17941], 'B': [17942], 'R':
[18898, 20758], 'N': [19172, 50048], 'g': [20512, 56100], '4': [20626], 'M':
[25736], 'L': [25893, 40959, 49639], 'k': [26039, 47987], 'E': [26134, 50128,
61474], 'r': [26345], 'K': [26346, 79790], 'd': [26676, 57924], 'T': [28161,
50047, 72677], 'x': [31622], 'P': [31873], '9': [32028, 36368, 40956], 'j':
[35260], '1': [36428, 51184], 'O': [38716, 54650, 60540], 'F': [38805], 'l':
[40111, 79791], 's': [40957], 'q': [40958, 73756], 'Q': [40960], 'z': [42541,
74037], '2': [49137, 50129, 51183], 'i': [49169, 73039], 'u': [49443, 52977],
'8': [50127], 'f': [51204], 'a': [52071], 'e': [55908], 'H': [60538], '0':
[60539], '6': [63141], 'w': [64436, 79564], 'n': [64437], 'D': [67416], 'c':
[73040], 'J': [73211], 'b': [73654], 'A': [74038], 'G': [76081], 'v': [76082],
```

```
'S': [79467]}
```

[6]:
```
# build a distance list and try to find the pattern from the greatest common␣
 ↪divisor of the distance.
last_character_distance = {}
for x in lastCharGroup:
    if len(lastCharGroup[x]) >= 2:
        last_character_distance[x] = [abs(lastCharGroup[x][i+1] -␣
 ↪lastCharGroup[x][i]) for i in range(len(lastCharGroup[x]) -1)]
print(last_character_distance)

from functools import reduce
import math
gcd_of_last_character_group = list(map(lambda x: reduce(math.gcd, x),␣
 ↪list(last_character_distance.values())))
print(gcd_of_last_character_group)
gcd_of_last_character = reduce(math.gcd, gcd_of_last_character_group)
print(gcd_of_last_character)
```

```
{'p': [33604], 'W': [24428], 'R': [1860], 'N': [30876], 'g': [35588], 'L':
[15066, 8680], 'k': [21948], 'E': [23994, 11346], 'K': [53444], 'd': [31248],
'T': [21886, 22630], '9': [4340, 4588], '1': [14756], 'O': [15934, 5890], 'l':
[39680], 'q': [32798], 'z': [31496], '2': [992, 1054], 'i': [23870], 'u':
[3534], 'w': [15128]}
[33604, 24428, 1860, 30876, 35588, 62, 21948, 186, 53444, 31248, 62, 124, 14756,
62, 39680, 32798, 31496, 62, 23870, 3534, 15128]
62
```

[7]:
```
# This is good! 26(English letters number) x 2(ordinary and capitals) + 10␣
 ↪(0-9) = 62 which make sense, like base62
# Since that, we can calc the char_position from the last char
character_position = {}
for x in lastCharGroup:
    character_position[x] = lastCharGroup[x][0] % gcd_of_last_character
print(character_position)
{k: v for k, v in sorted(character_position.items(), key=lambda item: item[1])}
```

```
{'p': 30, 'W': 35, '3': 23, 'B': 24, 'R': 50, 'N': 14, 'g': 52, '4': 42, 'M': 6,
'L': 39, 'k': 61, 'E': 32, 'r': 57, 'K': 58, 'd': 16, 'T': 13, 'x': 2, 'P': 5,
'9': 36, 'j': 44, '1': 34, 'O': 28, 'F': 55, 'l': 59, 's': 37, 'q': 38, 'Q': 40,
'z': 9, '2': 33, 'i': 3, 'u': 29, '8': 31, 'f': 54, 'a': 53, 'e': 46, 'H': 26,
'0': 27, '6': 25, 'w': 18, 'n': 19, 'D': 22, 'c': 4, 'J': 51, 'b': 60, 'A': 10,
'G': 7, 'v': 8, 'S': 45}
```

[7]:
```
{'x': 2,
 'i': 3,
 'c': 4,
 'P': 5,
```

```
 'M': 6,
 'G': 7,
 'v': 8,
 'z': 9,
 'A': 10,
 'T': 13,
 'N': 14,
 'd': 16,
 'w': 18,
 'n': 19,
 'D': 22,
 '3': 23,
 'B': 24,
 '6': 25,
 'H': 26,
 'O': 27,
 'O': 28,
 'u': 29,
 'p': 30,
 '8': 31,
 'E': 32,
 '2': 33,
 '1': 34,
 'W': 35,
 '9': 36,
 's': 37,
 'q': 38,
 'L': 39,
 'Q': 40,
 '4': 42,
 'j': 44,
 'S': 45,
 'e': 46,
 'R': 50,
 'J': 51,
 'g': 52,
 'a': 53,
 'f': 54,
 'F': 55,
 'r': 57,
 'K': 58,
 'l': 59,
 'b': 60,
 'k': 61}
```

[8]:
```
# There are two possibilities:
```

```
# 1. The second char comes from a different dataset than the last char. Which␣
 ↪more complex.
# 2. The second char comes from the same dataset, we can simply use the {A1} /␣
 ↪62 % 62 to predict it.
# Verify the second possibility
# The idea is to try to find two items, the first char and the last char both␣
 ↪of them already in the character_position
# If we cannot find it, we can try to build one, since we know the dataset of␣
 ↪last one.

# collect by the first char
same_first_group = {}
for x in npdata:
    if x[1][0] not in same_first_group:
        same_first_group[x[1][0]] = [(x[1] , x[0])]
    else:
        same_first_group[x[1][0]].append((x[1] , x[0]))

print(same_first_group)
```

```
{'c': [('cGp', 15840), ('cmW', 16465), ('cX3', 17941), ('cXB', 17942), ('ctR',
18898), ('ckN', 19172)], 'P': [('PVg', 20512), ('PD4', 20626), ('PBR', 20758)],
'M': [('MYM', 25736), ('MSL', 25893), ('MUk', 26039), ('MIE', 26134), ('Mgr',
26345), ('MgK', 26346), ('MKd', 26676)], 'G': [('GVT', 28161)], 'v': [('vNx',
31622), ('vwP', 31873), ('vV9', 32028)], 'z': [('zAj', 35260), ('zO9', 36368),
('zu1', 36428)], 'A': [('AcO', 38716), ('APF', 38805), ('AHl', 40111), ('ALW',
40893), ('AQ9', 40956), ('AQs', 40957), ('AQq', 40958), ('AQL', 40959), ('AQQ',
40960)], 'Z': [('Zcz', 42541)], 'y': [('yuk', 47987), ('yh2', 49137), ('yIi',
49169), ('yau', 49443), ('yap', 49444), ('ytL', 49639)], 'T': [('T7T', 50047),
('T7N', 50048), ('Tx8', 50127), ('TxE', 50128), ('Tx2', 50129), ('Tn2', 51183),
('Tn1', 51184), ('Tnf', 51204), ('T2a', 52071), ('Thu', 52977)], 'N': [('NTO',
54650), ('N2e', 55908), ('N9g', 56100)], 'o': [('ocd', 57924), ('oeH', 60538),
('oe0', 60539), ('oeO', 60540), ('okE', 61474)], 'd': [('dH6', 63141), ('dUw',
64436), ('dUn', 64437)], 'm': [('m2D', 67416)], 'w': [('wtT', 72677)], 'n':
[('nCi', 73039), ('nCc', 73040), ('nxJ', 73211), ('nzb', 73654), ('nZq', 73756),
('ndz', 74037), ('ndA', 74038), ('nIG', 76081), ('nIv', 76082)], 'V': [('VXS',
79467), ('VYw', 79564), ('VeK', 79790), ('Vel', 79791)]}
```

[9]:
```
# Try to find the first char and the last char both of them already in the␣
 ↪character_position
same_first_group_12_in_dict = {}
for x in same_first_group:
    t = list(filter(lambda x: x[0][1] in character_position and x[0][2] in␣
 ↪character_position ,same_first_group[x]))
    if len(t) > 0:
        same_first_group_12_in_dict[x] = t
```

```
print(same_first_group_12_in_dict)
```

```
{'c': [('cGp', 15840), ('ckN', 19172)], 'P': [('PD4', 20626), ('PBR', 20758)],
'M': [('MSL', 25893), ('Mgr', 26345), ('MgK', 26346), ('MKd', 26676)], 'v':
[('vNx', 31622), ('vwP', 31873)], 'z': [('zAj', 35260), ('zO9', 36368), ('zu1',
36428)], 'A': [('AcO', 38716), ('APF', 38805), ('AHl', 40111), ('ALW', 40893),
('AQ9', 40956), ('AQs', 40957), ('AQq', 40958), ('AQL', 40959), ('AQQ', 40960)],
'Z': [('Zcz', 42541)], 'y': [('yuk', 47987), ('yau', 49443), ('yap', 49444)],
'T': [('Tx8', 50127), ('TxE', 50128), ('Tx2', 50129), ('Tn2', 51183), ('Tn1',
51184), ('Tnf', 51204), ('T2a', 52071)], 'N': [('NTO', 54650), ('N2e', 55908),
('N9g', 56100)], 'o': [('ocd', 57924), ('oeH', 60538), ('oe0', 60539), ('oeO',
60540), ('okE', 61474)], 'd': [('dH6', 63141)], 'm': [('m2D', 67416)], 'n':
[('nxJ', 73211), ('nzb', 73654), ('ndz', 74037), ('ndA', 74038)], 'V': [('VeK',
79790), ('Vel', 79791)]}
```

[10]:
```
# We can pick the 'cGp' and 'ckN'. More than this, since we knew the distance␣
 ↪between 'p' and 'N', We can build the 'ckp'.
ckp = 19172 + (character_position['p'] - character_position['N'])
print(ckp)
cGp = same_first_group['c'][0][1] #get the A of 'cGp'
print(cGp)
```

```
19188
15840
```

[11]:
```
# What I expect is that:
# 1. If the second char dataset is the same as the last char dataset
# 2. If there is no offset between them
# 3. The distance of 'ckp' and 'cGp' == distance_'k'_'G' * 62
print(ckp - cGp)
print((character_position['k'] - character_position['G']) *␣
 ↪gcd_of_last_character)
```

```
3348
3348
```

[12]:
```
# It is eq!
# There is a possibility that the first char come from another dataset.
# But we can complete the char position list, and verify the result. If it␣
 ↪doesn't match, then we can try to figure it out.

# Get miss char from the first position
for x in npdata:
    if x[1][0] not in character_position:
        character_position[x[1][0]] = int(x[0] / (gcd_of_last_character ** 2))
print(character_position)
{k: v for k, v in sorted(character_position.items(), key=lambda item: item[1])}
```

```
{'p': 30, 'W': 35, '3': 23, 'B': 24, 'R': 50, 'N': 14, 'g': 52, '4': 42, 'M': 6,
```

```
'L': 39, 'k': 61, 'E': 32, 'r': 57, 'K': 58, 'd': 16, 'T': 13, 'x': 2, 'P': 5,
'9': 36, 'j': 44, '1': 34, 'O': 28, 'F': 55, 'l': 59, 's': 37, 'q': 38, 'Q': 40,
'z': 9, '2': 33, 'i': 3, 'u': 29, '8': 31, 'f': 54, 'a': 53, 'e': 46, 'H': 26,
'0': 27, '6': 25, 'w': 18, 'n': 19, 'D': 22, 'c': 4, 'J': 51, 'b': 60, 'A': 10,
'G': 7, 'v': 8, 'S': 45, 'Z': 11, 'y': 12, 'o': 15, 'm': 17, 'V': 20}
```

[12]:
```
{'x': 2,
 'i': 3,
 'c': 4,
 'P': 5,
 'M': 6,
 'G': 7,
 'v': 8,
 'z': 9,
 'A': 10,
 'Z': 11,
 'y': 12,
 'T': 13,
 'N': 14,
 'o': 15,
 'd': 16,
 'm': 17,
 'w': 18,
 'n': 19,
 'V': 20,
 'D': 22,
 '3': 23,
 'B': 24,
 '6': 25,
 'H': 26,
 '0': 27,
 'O': 28,
 'u': 29,
 'p': 30,
 '8': 31,
 'E': 32,
 '2': 33,
 '1': 34,
 'W': 35,
 '9': 36,
 's': 37,
 'q': 38,
 'L': 39,
 'Q': 40,
 '4': 42,
 'j': 44,
 'S': 45,
```

```
 'e': 46,
 'R': 50,
 'J': 51,
 'g': 52,
 'a': 53,
 'f': 54,
 'F': 55,
 'r': 57,
 'K': 58,
 'l': 59,
 'b': 60,
 'k': 61}
```

[13]:
```
# Get miss char from the second position
for x in npdata:
    if x[1][1] not in character_position:
        character_position[x[1][1]] = int(x[0] / (gcd_of_last_character ** 1))␣
 ↪% gcd_of_last_character

print(character_position)
[v for v in sorted(character_position.items(), key=lambda item: item[1])]
```

```
{'p': 30, 'W': 35, '3': 23, 'B': 24, 'R': 50, 'N': 14, 'g': 52, '4': 42, 'M': 6,
'L': 39, 'k': 61, 'E': 32, 'r': 57, 'K': 58, 'd': 16, 'T': 13, 'x': 2, 'P': 5,
'9': 36, 'j': 44, '1': 34, 'O': 28, 'F': 55, 'l': 59, 's': 37, 'q': 38, 'Q': 40,
'z': 9, '2': 33, 'i': 3, 'u': 29, '8': 31, 'f': 54, 'a': 53, 'e': 46, 'H': 26,
'0': 27, '6': 25, 'w': 18, 'n': 19, 'D': 22, 'c': 4, 'J': 51, 'b': 60, 'A': 10,
'G': 7, 'v': 8, 'S': 45, 'Z': 11, 'y': 12, 'o': 15, 'm': 17, 'V': 20, 'X': 41,
't': 56, 'Y': 43, 'U': 47, 'I': 49, 'h': 48, '7': 1, 'C': 0}
```

[13]:
```
[('C', 0),
 ('7', 1),
 ('x', 2),
 ('i', 3),
 ('c', 4),
 ('P', 5),
 ('M', 6),
 ('G', 7),
 ('v', 8),
 ('z', 9),
 ('A', 10),
 ('Z', 11),
 ('y', 12),
 ('T', 13),
 ('N', 14),
 ('o', 15),
 ('d', 16),
 ('m', 17),
```

```
('w', 18),
('n', 19),
('V', 20),
('D', 22),
('3', 23),
('B', 24),
('6', 25),
('H', 26),
('0', 27),
('O', 28),
('u', 29),
('p', 30),
('8', 31),
('E', 32),
('2', 33),
('1', 34),
('W', 35),
('9', 36),
('s', 37),
('q', 38),
('L', 39),
('Q', 40),
('X', 41),
('4', 42),
('Y', 43),
('j', 44),
('S', 45),
('e', 46),
('U', 47),
('h', 48),
('I', 49),
('R', 50),
('J', 51),
('g', 52),
('a', 53),
('f', 54),
('F', 55),
('t', 56),
('r', 57),
('K', 58),
('l', 59),
('b', 60),
('k', 61)]
```

[14]:
```
len({k: v for k, v in sorted(character_position.items(), key=lambda item:␣
↪item[1])})
```

[14]: 61

[15]:
```
# We miss one of them. That's fine, we can rebuild it.
import string
s = string.digits + string.ascii_lowercase + string.ascii_uppercase
miss_char = ''
for x in s:
    if x not in character_position:
        print(x)
        miss_char = x
        print(miss_char)
        break
# Build a array for find the miss position
char_position = [v for v in sorted(character_position.items(), key=lambda item:␣
 ↪item[1])]
for x in range(61):
    if char_position[x][1] != x:
        character_position[miss_char] = x
        break
print(len({k: v for k, v in sorted(character_position.items(), key=lambda item:␣
 ↪item[1])}))
char_position = [v for v in sorted(character_position.items(), key=lambda item:␣
 ↪item[1])]
print(char_position)
```

```
5
5
62
[('C', 0), ('7', 1), ('x', 2), ('i', 3), ('c', 4), ('P', 5), ('M', 6), ('G', 7),
('v', 8), ('z', 9), ('A', 10), ('Z', 11), ('y', 12), ('T', 13), ('N', 14), ('o',
15), ('d', 16), ('m', 17), ('w', 18), ('n', 19), ('V', 20), ('5', 21), ('D',
22), ('3', 23), ('B', 24), ('6', 25), ('H', 26), ('0', 27), ('O', 28), ('u',
29), ('p', 30), ('8', 31), ('E', 32), ('2', 33), ('1', 34), ('W', 35), ('9',
36), ('s', 37), ('q', 38), ('L', 39), ('Q', 40), ('X', 41), ('4', 42), ('Y',
43), ('j', 44), ('S', 45), ('e', 46), ('U', 47), ('h', 48), ('I', 49), ('R',
50), ('J', 51), ('g', 52), ('a', 53), ('f', 54), ('F', 55), ('t', 56), ('r',
57), ('K', 58), ('l', 59), ('b', 60), ('k', 61)]
```

[16]:
```
# Build a simple function to verify it.
def myBase62(a):
    first = int(a / (62 ** 2)) % 62
    second = int(a / 62) % 62
    last = a % 62
    return char_position[first][0] + char_position[second][0] +␣
 ↪char_position[last][0]

print(myBase62(15840))
```

```
cGp
```

[17]:
```
# Test all the data
for x in npdata:
    res = myBase62(x[0])
    if res != x[1]:
        print(res + " " + str(x[0]) + " " + x[1])

print("finish test")
```

```
finish test
```

[18]:
```
# Q2.b
print("f(30001) = " + myBase62(30001))
print("f(55555) = " + myBase62(55555))
print("f(77788) = " + myBase62(77788))
```

```
f(30001) = GIF
f(55555) = NOi
f(77788) = VNQ
```

[19]:
```
# Q3.c
# The max number of A shoud be 62 ** 3 - 1
print("The max number: " + str(62 ** 3 - 1))
```

```
The max number: 238327
```