Import library

In [115]:
```
import os
import pandas as pd
import numpy as np
import math
import matplotlib.pyplot as plt
```

Change working directory

In [81]:
```
%cd "F:\Khang\EVRY\Transversal\Training"

%pwd
```

```
F:\Khang\EVRY\Transversal\Training
```

Out[81]:
```
'F:\\Khang\\EVRY\\Transversal\\Training'
```

Def some useful functions that will be used

In [3]:
```
def distance(c1, c2):
  x_dist = (float(c1[8]) - float(c2[8]))**2
  y_dist = (float(c1[9]) - float(c2[9]))**2
  z_dist = (float(c1[10]) - float(c2[10]))**2
  return (x_dist + y_dist + z_dist)**0.5

def flatten(t):
    return [item for sublist in t for item in sublist]
```

Part 1: Training

1.1 Read all files in directory and write proccessed files

In [82]:
```
for filename in os.listdir():
    with open(filename,'r') as infile:
        l = ''
        for ligne in infile:
            if((ligne[0:6].replace(" ", "") == "ATOM") and (ligne[13:15].repla
ce(" ","") == "C3")):
                    l += ligne
        with open("Proccessed_"+ filename, "w") as outfile:
            outfile.write(l)
```

1.2. Calculate distances of structures

In [83]:
```
dict_distance = {}
for filename in os.listdir():
    with open(filename,'r') as infile:
        if("Proccessed" in filename):
            atom = []; serial = []; atom_name = []; alt_loc = []; res_name =
[]; chain_id = []; res_num = [];
            code_res = []; x = []; y = []; z = []; occ = []; temp_fact = []; e
le_symb = []; char_atom = []
            for ligne in infile:
                atom.append(ligne[0:6].replace(" ",""))
                serial.append(ligne[6:11].replace(" ",""))
                atom_name.append(ligne[12:16].replace(" ",""))
                alt_loc.append(ligne[16:17].replace(" ",""))
                res_name.append(ligne[17:20].replace(" ",""))
                chain_id.append(ligne[21:22].replace(" ",""))
                res_num.append(ligne[22:26].replace(" ",""))
                code_res.append(ligne[26:27].replace(" ",""))
                x.append(float(ligne[30:38]))
                y.append(float(ligne[38:46]))
                z.append(float(ligne[46:54]))
                occ.append(float(ligne[54:60]))
                temp_fact.append(float(ligne[60:66]))
                ele_symb.append(ligne[70:78].replace(" ",""))
                char_atom.append(ligne[78:80].replace(" ",""))

            df = pd.DataFrame(list(zip(atom, serial, atom_name, alt_loc,
                            res_name, chain_id, res_num,
                            code_res, x, y, z,
                            occ, temp_fact, ele_symb,
                            char_atom)),
               columns =['atom', 'serial', 'atom_name', 'alt_loc',
                            'res_name', 'chain_id', 'res_num',
                            'code_res', 'x', 'y', 'z',
                            'occ', 'temp_fact', 'ele_symb',
                            'char_atom'])


            for k in df.chain_id.unique():
              sub_df = df[df.chain_id == k]
              # Only consider intrachain basepairs
              for i in range(1,sub_df.shape[0]):
                for j in range(i):
                     # Only consider residues separated by at least 3 position
s
                    if(abs(int(sub_df.iloc[i][6]) - int(sub_df.iloc[j][6])) >=
3):
                      a = str(sub_df.iloc[i][4]).strip() + " - " + str(sub_df.
iloc[j][4]).strip()
                      b = a[::-1]
                      x = distance(sub_df.iloc[i], sub_df.iloc[j])
                      if(x <= 20):
                        if(not((a in dict_distance) or (b in dict_distance))):
                            if(a in ['A - A', 'A - U', 'A - C', 'A - G', 'U -
 U', 'U - C',
                                                'U - G', 'C - C', 'C - G', 'G - G']):
                                dict_distance[a] = [x]
```

```
                    elif(b in ['A - A', 'A - U', 'A - C', 'A - G', 'U
- U', 'U - C',
                               'U - G', 'C - C', 'C - G', 'G - G']):
                        dict_distance[b] = [x]
                elif(a in dict_distance):
                  dict_distance[a].append(x)
                elif(b in dict_distance):
                  dict_distance[b].append(x)
```

In [122]:
```
dict_distance['A - G'][:20]
```

Out[122]:
```
[15.620086811538531,
 12.312247073544293,
 14.733483023372306,
 16.929337287679044,
 11.975991691713887,
 15.634781386383375,
 14.506780242355642,
 16.52999147005225,
 19.207756141725664,
 15.589266980842941,
 19.835075043971976,
 15.137478786112302,
 17.460282529214698,
 18.91435708661545,
 15.52628841674661,
 15.65891327646973,
 18.325649401862954,
 14.875037243650853,
 19.25940676656475,
 19.360167612910793]
```

1.3. Calculate the reference frequency (P_ref), observed frequency (P_obs) and the score

In [84]:

```
P_ref = {}
P_obs = {}
score = {}
for j in range(20):
    P_ref[str(j) + "-" + str(j + 1)] = len([m for m in flatten(dict_distance.v
alues()) if ((m > j) & (m <= j+1))]) / len(flatten(dict_distance.values()))


for i in dict_distance.keys():
    if(not(i in score.keys())):
        score[i] = []
    for j in range(20):
        s = 0
        a = str(i) + "_" + str(j) + "-" + str(j + 1)
        P_obs[a] = len([m for m in dict_distance[i] if((m > j) & (m <= j+1))])
/len(flatten([dict_distance[i]]))
        if((P_ref[str(j) + "-" + str(j + 1)] != 0) & (P_obs[a] != 0)):
            t = -math.log(P_obs[a]/P_ref[str(j) + "-" + str(j + 1)])
            print(i, j, t)
            score[i].append([j,t])
```

A - G 5 -0.7944666027982701
A - G 6 -0.2348508148628473
A - G 7 -0.9279979954227926
A - G 8 -0.43156110910890155
A - G 9 -0.9390478316093774
A - G 10 0.109245346869025
A - G 11 -0.25547010206558285
A - G 12 -0.2084175577946917
A - G 13 0.29172316587128266
A - G 14 0.267006138920383
A - G 15 -0.014667305177114292
A - G 16 -0.0844718516657134
A - G 17 -0.07832990401362581
A - G 18 0.24871588093050898
A - G 19 0.04902462946737108
A - U 4 -2.6671137311654256
A - U 5 -1.280819370045535
A - U 6 -0.02805640155016707
A - U 7 -0.7212035821101124
A - U 8 0.468380484763724
A - U 9 -0.8063613904504192
A - U 10 -0.6002509716924498
A - U 11 0.3286185423885654
A - U 12 -0.1734734014086707
A - U 13 -0.36452863817138004
A - U 14 -0.3371296639832653
A - U 15 -0.0016231444820113377
A - U 16 -0.030440192905443253
A - U 17 -0.09931942157165767
A - U 18 0.3983518804032405
A - U 19 0.39239457778580206
U - C 6 -0.5522493596431051
U - C 8 -0.7489596538891593
U - C 9 1.3085029810719015
U - C 10 1.8712883437686032
U - C 11 -0.06204302307985008
U - C 12 -0.42737602976169703
U - C 13 0.0921076567474082
U - C 14 0.08313898676464783
U - C 15 0.019695079178931107
U - C 16 0.12198472761035321
U - C 17 -0.0018241630867953786
U - C 18 -0.02867732923604974
U - C 19 -0.1071057677167646
C - C 6 0.10787595261189395
C - C 7 -0.5852712279480513
C - C 9 0.5823339322070097
C - C 10 0.45197211434376616
C - C 11 -0.04627472721536443
C - C 12 -0.08883434163416047
C - C 13 0.6721902613288707
C - C 14 -0.30655782547903093
C - C 15 -0.09788417715407825
C - C 16 -0.2428145330115984
C - C 17 -0.03484603139174185
C - C 18 0.11654903386532259
C - C 19 0.3298759932240246

A - C 5 -1.6096287339211788
A - C 6 -1.2731564972999658
A - C 7 -1.0500129459857561
A - C 8 -0.553576059671865
A - C 9 0.11759221416930517
A - C 10 -0.49827741947563936
A - C 11 -0.22333437280128815
A - C 12 0.04426094108375539
A - C 13 0.20744854329116616
A - C 14 0.11421952969066598
A - C 15 0.13957112088808044
A - C 16 0.3338976237788582
A - C 17 -0.5166821827887464
A - C 18 -0.17557994150538828
A - C 19 0.02765320468409504
C - G 8 0.3045678940247289
C - G 9 0.975736167865899
C - G 10 -0.25323793866545424
C - G 11 -0.1835007426186455
C - G 12 -0.00780679101742359
C - G 13 -0.7001914858370343
C - G 14 -0.23765163539463874
C - G 15 0.14265520886198135
C - G 16 0.30735618501662815
C - G 17 -0.02756594099788587
C - G 18 0.04402096566611219
C - G 19 0.03682003647376523
G - G 6 -0.2247666957962211
G - G 8 0.27167019051766983
G - G 9 -0.1557738243092697
G - G 10 -0.19912426518288362
G - G 11 -0.21639844612570452
G - G 12 -0.028434401932668073
G - G 13 0.10113658947575754
G - G 14 0.029254094082427903
G - G 15 -0.038484737786169704
G - G 16 -0.18459087273270164
G - G 17 0.13760626925714883
G - G 18 0.2587999999971349
G - G 19 0.09154402428715089
U - G 6 0.46503039787844225
U - G 8 0.9614672841923333
U - G 9 -0.4468059836463325
U - G 10 -0.44363640888505357
U - G 11 0.26208955388175187
U - G 12 0.005955839164896932
U - G 13 0.16030685957180943
U - G 14 0.1559571354453436
U - G 15 -0.05508005142147234
U - G 16 0.03520259169622618
U - G 17 0.026042597731634132
U - G 18 -0.07540433120513675
U - G 19 -0.04533745522265373
A - A 11 0.6705702447744037
A - A 12 0.8511541816698177
A - A 13 0.6217800806049718
A - A 14 0.6334306978249473

```
A - A 15 -0.10958831277517118
A - A 16 -0.1831847323557117
A - A 17 0.7170902604092967
A - A 18 -0.25705497217385775
A - A 19 -0.4072320443799124
U - U 7 -0.7184600961643615
U - U 8 -0.9151703904104157
U - U 9 1.1422922445506452
U - U 11 1.0245092088942618
U - U 12 0.9819495944754658
U - U 13 1.3121912813460426
U - U 14 0.03025693555039469
U - U 15 -0.10651032272862608
U - U 16 -0.19603202177890725
U - U 17 0.6343115729168857
U - U 18 -0.41394163181999083
U - U 19 -0.18174931071253037
```

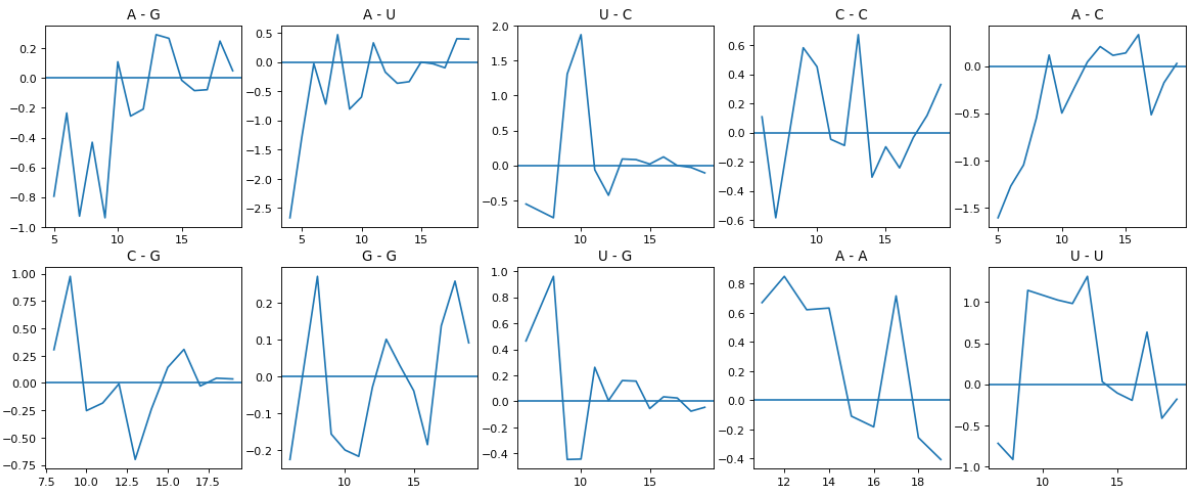
1.4. Plot the interaction profiles

In [85]:
```
from matplotlib.pyplot import figure

figure(figsize=(18, 15), dpi=80)

m = 1
for j in score.keys():
  plt.subplot(4,5,m)
  x = [score[j][i][0] for i in range(len(score[j]))]
  y = [score[j][i][1] for i in range(len(score[j]))]
  plt.title(j)
  plt.axhline(y=0)
  plt.plot(x,y)
  m += 1

plt.show()
```



Part 2: Compute the total score of each structure

2.1. Create linear interpolation function for dict from training datasets

the formula is: yj = (ya - yb)*(j - b)/(a - b) + yb

In [ ]:
```
def linear_interpol(key, dict_score, each_dict):
    list_score = []
    for i in each_dict[key]:
        for j in range(len(dict_score[key])):
            if(dict_score[key][j][0] == math.floor(i)-1):
                list_score.append((dict_score[key][j+1][1] - dict_score[key][j
][1])*(i - math.floor(i)) +  dict_score[key][j][1])

    return list_score
```

2.2. Apply the formula to all structures

In [96]:

```
score_each_struct = {}
for filename in os.listdir():
    with open(filename,'r') as infile:
        if("Proccessed" in filename):
            each_dict_distance = {}
            atom = []; serial = []; atom_name = []; alt_loc = []; res_name =
[]; chain_id = []; res_num = [];
            code_res = []; x = []; y = []; z = []; occ = []; temp_fact = []; e
le_symb = []; char_atom = []
            for ligne in infile:
                atom.append(ligne[0:6].replace(" ",""))
                serial.append(ligne[6:11].replace(" ",""))
                atom_name.append(ligne[12:16].replace(" ",""))
                alt_loc.append(ligne[16:17].replace(" ",""))
                res_name.append(ligne[17:20].replace(" ",""))
                chain_id.append(ligne[21:22].replace(" ",""))
                res_num.append(ligne[22:26].replace(" ",""))
                code_res.append(ligne[26:27].replace(" ",""))
                x.append(float(ligne[30:38]))
                y.append(float(ligne[38:46]))
                z.append(float(ligne[46:54]))
                occ.append(float(ligne[54:60]))
                temp_fact.append(float(ligne[60:66]))
                ele_symb.append(ligne[70:78].replace(" ",""))
                char_atom.append(ligne[78:80].replace(" ",""))

            df = pd.DataFrame(list(zip(atom, serial, atom_name, alt_loc,
                            res_name, chain_id, res_num,
                            code_res, x, y, z,
                            occ, temp_fact, ele_symb,
                            char_atom)),
               columns =['atom', 'serial', 'atom_name', 'alt_loc',
                            'res_name', 'chain_id', 'res_num',
                            'code_res', 'x', 'y', 'z',
                            'occ', 'temp_fact', 'ele_symb',
                            'char_atom'])


            for k in df.chain_id.unique():
                sub_df = df[df.chain_id == k]
              #print(sub_df, sub_df.shape[0])
                for i in range(1,sub_df.shape[0]):
                    for j in range(i):
                        if(abs(int(sub_df.iloc[i][6]) - int(sub_df.iloc[j][6
])) >= 3):
                            a = str(sub_df.iloc[i][4]).strip() + " - " + str(sub
_df.iloc[j][4]).strip()
                            b = a[::-1]
                            x = distance(sub_df.iloc[i], sub_df.iloc[j])
                            if(x <= 20):
                              if(not((a in each_dict_distance) or (b in each_dic
t_distance))):
                                  if(a in ['A - A', 'A - U', 'A - C', 'A - G',
'U - U', 'U - C',
                                           'U - G', 'C - C', 'C - G', 'G - G'
]):
```

```
                                    each_dict_distance[a] = [x]
                                elif(b in ['A - A', 'A - U', 'A - C', 'A - G',
'U - U', 'U - C',
                                                'U - G', 'C - C', 'C - G', 'G - G'
]):
                                    each_dict_distance[b] = [x]
                            elif(a in each_dict_distance):
                              each_dict_distance[a].append(x)
                            elif(b in dict_distance):
                              each_dict_distance[b].append(x)

            # Create a temporary list that save scores of each structure
            m = []
            for u in each_dict_distance.keys():
                t = linear_interpol(u,score, each_dict_distance)
                m.append(sum(t)/len(t))

            # Save the sum of scores of each structure to a dict
            score_each_struct[filename[11:-4]] = sum(m)
```

In [97]:
```
score_each_struct
```

Out[97]:

```
{'1F27': -0.30513458681979644,
 '1J6S': -0.6305405901622058,
 '1J9H': 1.0089668045409685,
 '1L2X': -0.18613034264484074,
 '1MDG': -0.622228088598846,
 '1P79': -0.8537229867112206,
 '1PJG': 0.2978628513219765,
 '1QCU': 0.030082213713090188,
 '1WPU': -0.04892147101308732,
 '259D': -0.15617832440807458,
 '2A43': -0.18361203709583038,
 '2ASB': 0.12796674381421935,
 '2G3S': 0.4298987463119339,
 '2G91': 0.3352537276819977,
 '2GRB': 0.02624148843882808,
 '2Q1O': -0.14880878947429932,
 '2R1S': 0.41280652591767764,
 '2V7R': 0.4134506361040248,
 '2VUQ': 0.11370151256154601,
 '2XS2': 0.19340074602555601,
 '2XS7': 0.157820246450459,
 '2Y8Y': -0.3439335239029159,
 '315D': 0.04849927561016909,
 '354D': 0.8262459108425376,
 '397D': -0.5269308667229889,
 '3C3Z': -0.13783987344858142,
 '3CZW': 0.6249702377865788,
 '3G9Y': -0.41571853563855793,
 '3GLP': -0.3824028763360475,
 '3GVN': -0.0833612928359212,
 '3HGA': 0.014662655135750338,
 '3JXQ': 0.2528971231294277,
 '3ND3': 0.08243970214903731,
 '3NJ6': -0.03879002308022644,
 '3OK4': -0.045100613123365565,
 '3P4C': 0.09354497250926869,
 '3PF4': -0.10489454129167033,
 '3R1D': -0.18554350892150562,
 '3R1E': -0.13213467925514544,
 '3SJ2': -0.04891455856570931,
 '435D': -0.025286594611969276,
 '464D': -0.6191786342985609,
 '485D': 0.03404551810106504,
 '4E6B': 0.09091172220584227,
 '4FEN': 0.46464067886696414,
 '4JAH': 0.13905985892648048,
 '4JRD': 0.1478101420327395,
 '4K31': 0.8904912220796083,
 '4LGT': -0.663951728726942,
 '4MDX': 0.25886930415131193,
 '4NLF': -0.45188157667361356,
 '4O41': -0.2818755367011884,
 '4OQ9': -0.021912831849996824,
 '4PCO': 0.3454668042290297,
 '4QM6': -0.6095963755341962,
 '4RBQ': -0.3242947641990406,
 '4RBY': -0.0662233864353023,
```

```
'4RBZ': 0.18784682207177583,
'4RC0': -0.10815062789668617,
'4RJ1': -0.19177220773486786,
'4RKV': -0.13990312503323146,
'4S2X': 0,
'4U34': -0.4674683201189218,
'4U3L': 0.633138538295246,
'4U6K': -0.14108332669877488,
'5AY2': -0.2087561419457264,
'5C5W': 0.07168474814400376,
'5D8T': -0.012490399911963379,
'5EBI': 0.15647215829083422,
'5EME': -0.41581271034877965,
'5EMF': -0.46836720495460804,
'5EV3': -0.09990493023396366,
'5EW4': -0.0010491752492193633,
'5HBY': -0.1574307177324693,
'5HNJ': -0.4655787083467477,
'5JAJ': -0.013857563356778457,
'5K8H': 0,
'5KLA': -0.2622970325848909,
'5L00': -0.29629243932255256,
'5LQT': 0.10009090256286421,
'5NXT': -0.15993991312189432,
'5TDJ': -0.04284933693333917,
'5U0Q': -0.09708411217513671,
'5UED': -0.5211284224996489,
'5V0J': -0.5534905987659203,
'5V1K': 0.16614369712676405,
'6D2Z': 0,
'6D30': 0,
'6DO8': -0.21773125967770943,
'6DOY': -0.507172858298418,
'6DP8': -0.01205414289621276,
'6E0O': 0,
'6FPQ': -0.3268467017960289,
'6GD3': -0.7717254110074596,
'6HC5': 0.06825466442409298,
'6KUG': -0.12110966506363124,
'6M7K': 0,
'6N6I': 0,
'6N6J': 0,
'6N6K': 0,
'6Q1H': 0,
'6QIT': 0.1442679285365735,
'6RT4': 0,
'6RT6': 0,
'6VRD': -0.16261352259006334,
'6XLW': -0.06866469449506679,
'6XRQ': 0.05327279193008638,
'6XUS': -0.5122151284255793,
'6ZQ9': 0.15058824976098625,
'7EOG': 0.31742546914787606,
'7K5L': 0,
'7MJW': 0.6041406110594219,
'7MKY': 0.3025591149303621,
'7NJC': 0}
```

Rank structures by their score

In [126]:

```
s = 1
for i in sorted(set(score_each_struct.values()), reverse = True):
    for j in score_each_struct.keys():
        if(score_each_struct[j] == i):
            x = j + " has rank " + str(s) + " with score: " + str(i)
            print(x)
    s += 1
```

1J9H has rank 1 with score: 1.0089668045409685
4K31 has rank 2 with score: 0.8904912220796083
354D has rank 3 with score: 0.8262459108425376
4U3L has rank 4 with score: 0.633138538295246
3CZW has rank 5 with score: 0.6249702377865788
7MJW has rank 6 with score: 0.6041406110594219
4FEN has rank 7 with score: 0.46464067886696414
2G3S has rank 8 with score: 0.4298987463119339
2V7R has rank 9 with score: 0.4134506361040248
2R1S has rank 10 with score: 0.41280652591767764
4PCO has rank 11 with score: 0.3454668042290297
2G91 has rank 12 with score: 0.3352537276819977
7EOG has rank 13 with score: 0.31742546914787606
7MKY has rank 14 with score: 0.3025591149303621
1PJG has rank 15 with score: 0.2978628513219765
4MDX has rank 16 with score: 0.25886930415131193
3JXQ has rank 17 with score: 0.2528971231294277
2XS2 has rank 18 with score: 0.19340074602555601
4RBZ has rank 19 with score: 0.18784682207177583
5V1K has rank 20 with score: 0.16614369712676405
2XS7 has rank 21 with score: 0.157820246450459
5EBI has rank 22 with score: 0.15647215829083422
6ZQ9 has rank 23 with score: 0.15058824976098625
4JRD has rank 24 with score: 0.1478101420327395
6QIT has rank 25 with score: 0.1442679285365735
4JAH has rank 26 with score: 0.13905985892648048
2ASB has rank 27 with score: 0.12796674381421935
2VUQ has rank 28 with score: 0.11370151256154601
5LQT has rank 29 with score: 0.10009090256286421
3P4C has rank 30 with score: 0.09354497250926869
4E6B has rank 31 with score: 0.09091172220584227
3ND3 has rank 32 with score: 0.08243970214903731
5C5W has rank 33 with score: 0.07168474814400376
6HC5 has rank 34 with score: 0.06825466442409298
6XRQ has rank 35 with score: 0.05327279193008638
315D has rank 36 with score: 0.04849927561016909
485D has rank 37 with score: 0.03404551810106504
1QCU has rank 38 with score: 0.030082213713090188
2GRB has rank 39 with score: 0.02624148843882808
3HGA has rank 40 with score: 0.014662655135750338
4S2X has rank 41 with score: 0
5K8H has rank 41 with score: 0
6D2Z has rank 41 with score: 0
6D30 has rank 41 with score: 0
6E0O has rank 41 with score: 0
6M7K has rank 41 with score: 0
6N6I has rank 41 with score: 0
6N6J has rank 41 with score: 0
6N6K has rank 41 with score: 0
6Q1H has rank 41 with score: 0
6RT4 has rank 41 with score: 0
6RT6 has rank 41 with score: 0
7K5L has rank 41 with score: 0
7NJC has rank 41 with score: 0
5EW4 has rank 42 with score: -0.0010491752492193633
6DP8 has rank 43 with score: -0.01205414289621276
5D8T has rank 44 with score: -0.012490399911963379

5JAJ has rank 45 with score: -0.013857563356778457
4OQ9 has rank 46 with score: -0.021912831849996824
435D has rank 47 with score: -0.025286594611969276
3NJ6 has rank 48 with score: -0.03879002308022644
5TDJ has rank 49 with score: -0.04284933693333917
3OK4 has rank 50 with score: -0.045100613123365565
3SJ2 has rank 51 with score: -0.04891455856570931
1WPU has rank 52 with score: -0.04892147101308732
4RBY has rank 53 with score: -0.0662233864353023
6XLW has rank 54 with score: -0.06866469449506679
3GVN has rank 55 with score: -0.0833612928359212
5U0Q has rank 56 with score: -0.09708411217513671
5EV3 has rank 57 with score: -0.09990493023396366
3PF4 has rank 58 with score: -0.10489454129167033
4RC0 has rank 59 with score: -0.10815062789668617
6KUG has rank 60 with score: -0.12110966506363124
3R1E has rank 61 with score: -0.13213467925514544
3C3Z has rank 62 with score: -0.13783987344858142
4RKV has rank 63 with score: -0.13990312503323146
4U6K has rank 64 with score: -0.14108332669877488
2Q1O has rank 65 with score: -0.14880878947429932
259D has rank 66 with score: -0.15617832440807458
5HBY has rank 67 with score: -0.1574307177324693
5NXT has rank 68 with score: -0.15993991312189432
6VRD has rank 69 with score: -0.16261352259006334
2A43 has rank 70 with score: -0.18361203709583038
3R1D has rank 71 with score: -0.18554350892150562
1L2X has rank 72 with score: -0.18613034264484074
4RJ1 has rank 73 with score: -0.19177220773486786
5AY2 has rank 74 with score: -0.2087561419457264
6DO8 has rank 75 with score: -0.21773125967770943
5KLA has rank 76 with score: -0.2622970325848909
4O41 has rank 77 with score: -0.2818755367011884
5L00 has rank 78 with score: -0.29629243932255256
1F27 has rank 79 with score: -0.30513458681979644
4RBQ has rank 80 with score: -0.3242947641990406
6FPQ has rank 81 with score: -0.3268467017960289
2Y8Y has rank 82 with score: -0.3439335239029159
3GLP has rank 83 with score: -0.3824028763360475
3G9Y has rank 84 with score: -0.41571853563855793
5EME has rank 85 with score: -0.41581271034877965
4NLF has rank 86 with score: -0.45188157667361356
5HNJ has rank 87 with score: -0.4655787083467477
4U34 has rank 88 with score: -0.4674683201189218
5EMF has rank 89 with score: -0.46836720495460804
6DOY has rank 90 with score: -0.507172858298418
6XUS has rank 91 with score: -0.5122151284255793
5UED has rank 92 with score: -0.5211284224996489
397D has rank 93 with score: -0.5269308667229889
5V0J has rank 94 with score: -0.5534905987659203
4QM6 has rank 95 with score: -0.6095963755341962
464D has rank 96 with score: -0.6191786342985609
1MDG has rank 97 with score: -0.622228088598846
1J6S has rank 98 with score: -0.6305405901622058
4LGT has rank 99 with score: -0.663951728726942
6GD3 has rank 100 with score: -0.7717254110074596
1P79 has rank 101 with score: -0.8537229867112206