Ejercicio 1:

Las top 10 fechas donde hay más tweets. Mencionar el usuario (username) que más publicaciones tiene por cada uno de esos días.

Supuesto1: Para cumplir con lo solicitado, se trabaja el USERNAME como id unico representante de una cuenta, pero, como regla de negocio esto es incorrecto, ya que un USERNAME en twitter puede cambiar para una misma cuenta.

Ejecucion1 optimizada para Tiempo

```
%load_ext memory_profiler

import q1_time as q1t
from memory_profiler import profile
import cProfile
import pstats

@profile
def try_stat():
    profiler=cProfile.Profile()
    profiler.enable()
    li_q1t = q1t.q1_time('./farmers-protest-tweets-2021-2-4.json')
    profiler.disable()
    stats = pstats.Stats(profiler)
    stats.print_stats(0)
    print(li_q1t)

if __name__ == '__main__':
    try_stat()
```

```
The memory_profiler extension is already loaded. To reload it, use:
  %reload_ext memory_profiler
ERROR: Could not find file C:\Users\Usuario\AppData\Local\Temp\ipykernel_8176\681458841.py
Filename: c:\Users\Usuario\Downloads\challenge_DE\src\q1_time.py

Line #    Mem usage    Increment  Occurrences   Line Contents
=============================================================
    16    154.0 MiB    154.0 MiB           1   @profile
    17                                         def q1_time(file_path: str) -> List[Tuple[datetime.date, str]]:
    18    154.0 MiB      0.0 MiB           1       resp = []
    19    154.0 MiB      0.0 MiB           1       pd.set_option('mode.chained_assignment', None)  # Levantar una excepción
    20   1675.6 MiB   1521.6 MiB           1       pddf=pd.read_json(file_path, lines=True)
    21   1633.5 MiB    -42.1 MiB           1       pddf=pddf[['date','url','user']]
    22   1635.3 MiB      1.7 MiB           1       pddf['date_fecha'] = pddf['date'].dt.date
    23
    24   1636.3 MiB      1.0 MiB           1       df_topdates = pddf.groupby('date_fecha').agg({'url': ['count']})
    25   1636.3 MiB      0.0 MiB           1       df_topdates.columns = ['url_count']
    26   1636.3 MiB      0.0 MiB           1       df_topdates = df_topdates.reset_index()
    27   1636.3 MiB      0.0 MiB           1       df_topdates=df_topdates.sort_values(by='url_count', ascending=False)
    28   1636.3 MiB      0.0 MiB           1       df_topdates = df_topdates.head(10)
    29
    30
    31   1639.9 MiB      0.0 MiB          11       for indice, fila in df_topdates.iterrows():
    32   1639.8 MiB      2.2 MiB          10           df_topuser_xdate = pddf[pddf['date_fecha'] == fila['date_fecha']]
    33   1639.8 MiB      0.3 MiB      198744           df_topuser_xdate['identificador'] = df_topuser_xdate['user'].apply(lambda x:
    34   1639.8 MiB      0.7 MiB          10           df_topuser_xdate = df_topuser_xdate.groupby('identificador').agg({'url': ['co
    35   1639.8 MiB      0.0 MiB          10           df_topuser_xdate=df_topuser_xdate.reset_index()
    36   1639.8 MiB      0.0 MiB          10           df_topuser_xdate.columns = ['identificador','url_count']
    37   1639.9 MiB      0.4 MiB          10           df_topuser_xdate=df_topuser_xdate.sort_values(by='url_count', ascending=False
    38   1639.9 MiB      0.0 MiB          10           df_topuser_xdate = df_topuser_xdate.head(1)
    39   1639.9 MiB      0.0 MiB          10           resp.append((fila['date_fecha'],df_topuser_xdate['identificador'].values[0])
    40
    41   1639.9 MiB      0.0 MiB           1       return resp


         672858 function calls (671326 primitive calls) in 10.129 seconds

[(datetime.date(2021, 2, 12), 'RanbirS00614606'), (datetime.date(2021, 2, 13), 'MaanDee08215437'), (datetime.date(2021, 2, 17), 'Raa
```

Resultado

[
(datetime.date(2021, 2, 12), 'RanbirS00614606'),
(datetime.date(2021, 2, 13), 'MaanDee08215437'),
(datetime.date(2021, 2, 17), 'RaaJVinderkaur'),
(datetime.date(2021, 2, 16), 'jot__b'),

(datetime.date(2021, 2, 14), 'rebelpacifist'),
(datetime.date(2021, 2, 18), 'neetuanjle_nitu'),
(datetime.date(2021, 2, 15), 'jot__b'),
(datetime.date(2021, 2, 20), 'MangalJ23056160'),
(datetime.date(2021, 2, 23), 'Surrypuria'),
(datetime.date(2021, 2, 19), 'Preetm91')
]

(t+) se aprecia que el consumo de memoria en cada instruccion es de ~1640MiB para casi todas las instrucciones del primer y segundo bloque, el tiempo es de aproximados 10 seconds

Ejecucion1 optimizada para Memoria

```
%load_ext memory_profiler

import q1_memory as q1m
from memory_profiler import profile
import cProfile
import pstats

@profile
def try_stat():
    profiler=cProfile.Profile()
    profiler.enable()
    li_q1m = q1m.q1_memory('./farmers-protest-tweets-2021-2-4.json')
    profiler.disable()
    stats = pstats.Stats(profiler)
    stats.print_stats(0)
    print(li_q1m)

if __name__ == '__main__':
    try_stat()
```

```
The memory_profiler extension is already loaded. To reload it, use:
  %reload_ext memory_profiler
ERROR: Could not find file C:\Users\Usuario\AppData\Local\Temp\ipykernel_8176\4294011426.py
Filename: c:\Users\Usuario\Downloads\challenge_DE\src\q1_memory.py

Line #    Mem usage    Increment  Occurrences   Line Contents
=============================================================
    17    157.4 MiB    157.4 MiB           1   @profile
    18                                         def q1_memory(file_path: str) -> List[Tuple[datetime.date, str]]:
    19    157.4 MiB      0.0 MiB           1       resp = []
    20    157.4 MiB      0.0 MiB           1       pd.set_option('mode.chained_assignment', None)  # Levantar una excepción
    21    157.4 MiB      0.0 MiB           1       with open(file_path, 'r') as f:
    22    570.8 MiB -15426.6 MiB      117410           data = [[json.loads(line)['url'], pd.to_datetime(json.loads(line)['date']), 
    23    438.5 MiB   -132.3 MiB           1       columnas = ['url', 'date', 'identificador']
    24    440.4 MiB      1.9 MiB           1       pddf = pd.DataFrame(data, columns=columnas)
    25    444.1 MiB      3.7 MiB           1       pddf['date_fecha'] = pddf['date'].dt.date
    26
    27    444.2 MiB      0.1 MiB           1       df_topdates = pddf.groupby('date_fecha').agg({'url': ['count']})
    28    444.2 MiB      0.0 MiB           1       df_topdates.columns = ['url_count']
    29    444.2 MiB      0.0 MiB           1       df_topdates = df_topdates.reset_index()
    30    444.2 MiB      0.0 MiB           1       df_topdates=df_topdates.sort_values(by='url_count', ascending=False)
    31    444.2 MiB      0.0 MiB           1       df_topdates = df_topdates.head(10)
    32
    33    444.2 MiB      0.0 MiB          11       for indice, fila in df_topdates.iterrows():
    34    444.2 MiB    -27.3 MiB          10           df_topuser_xdate = pddf[pddf['date_fecha'] == fila['date_fecha']]
    35                                                 #df_topuser_xdate['identificador'] = df_topuser_xdate['user'].apply(lambda x
    36    441.2 MiB    -30.3 MiB          10           df_topuser_xdate = df_topuser_xdate.groupby('identificador').agg({'url': ['cc
    37    441.2 MiB      0.0 MiB          10           df_topuser_xdate=df_topuser_xdate.reset_index()
    38    441.2 MiB      0.0 MiB          10           df_topuser_xdate.columns = ['identificador','url_count']
    39    441.2 MiB      0.0 MiB          10           df_topuser_xdate=df_topuser_xdate.sort_values(by='url_count', ascending=False
    40    441.2 MiB      0.0 MiB          10           df_topuser_xdate = df_topuser_xdate.head(1)
    41    441.2 MiB      0.0 MiB          10           resp.append((fila['date_fecha'],df_topuser_xdate['identificador'].values[0])
    42
    43    441.2 MiB     -3.0 MiB           1       return resp


         144211904 function calls (144093254 primitive calls) in 403.207 seconds

[(datetime.date(2021, 2, 12), 'RanbirS00614606'), (datetime.date(2021, 2, 13), 'MaanDee08215437'), (datetime.date(2021, 2, 17), 'Raa
```

Resultado

[
(datetime.date(2021, 2, 12), 'RanbirS00614606'),

(datetime.date(2021, 2, 13), 'MaanDee08215437'),
(datetime.date(2021, 2, 17), 'RaaJVinderkaur'),
(datetime.date(2021, 2, 16), 'jot__b'),
(datetime.date(2021, 2, 14), 'rebelpacifist'),
(datetime.date(2021, 2, 18), 'neetuanjle_nitu'),
(datetime.date(2021, 2, 15), 'jot__b'),
(datetime.date(2021, 2, 20), 'MangalJ23056160'),
(datetime.date(2021, 2, 23), 'Surrypuria'),
(datetime.date(2021, 2, 19), 'Preetm91')
]

(m+) se aprecia que el consumo de memoria en cada instruccion del primer y segundo bloque es de aproximados ~440MiB. y en consecuencia, el tiempo subio considerablemente a aproximados ~403 seconds

Ejercicio 2:

Los top 10 emojis más usados con su respectivo conteo.

Supuesto1: No tengo una definicion clara de que es un emoji, ya que tecnicamente un emoji es un caracter unicode representado por un \u*, por lo que bajo esa definicion se trabajo la solucion.

Ejecucion2 optimizada para Tiempo

```
%load_ext memory_profiler

import q2_time as q2t
from memory_profiler import profile
import cProfile
import pstats

@profile
def try_stat():
    profiler=cProfile.Profile()
    profiler.enable()
    li_q2t = q2t.q2_time('./farmers-protest-tweets-2021-2-4.json')
    profiler.disable()
    stats = pstats.Stats(profiler)
    stats.print_stats(0)
    print(li_q2t)

if __name__ == '__main__':
    try_stat()
```

```
The memory_profiler extension is already loaded. To reload it, use:
  %reload_ext memory_profiler
ERROR: Could not find file C:\Users\Usuario\AppData\Local\Temp\ipykernel_8176\1025412768.py
Filename: c:\Users\Usuario\Downloads\challenge_DE\src\q2_time.py

Line #    Mem usage    Increment  Occurrences   Line Contents
=============================================================
    17    430.2 MiB    430.2 MiB           1   @profile
    18                                         def q2_time(file_path: str) -> List[Tuple[datetime.date, str]]:
    19    430.2 MiB      0.0 MiB           1       pd.set_option('mode.chained_assignment', None)  #evita warning
    20                                             #cargo el jsonl como dataframe pandas
    21   1665.7 MiB   1235.5 MiB           1       pddf=pd.read_json(file_path, lines=True)
    22                                             #trabajo solo con una columna, la que hace referencia al body del tweet
    23   1500.9 MiB   -164.8 MiB           1       pddf=pddf[['content']]
    24                                             # selecciono solo las filas que tengan al menos un emoji
    25   1502.7 MiB      1.8 MiB           1       pddf = pddf[pddf['content'].str.contains(r'[\U0001F300-\U0001F5FF\U0001F600-\U000
    26                                             # elimino todo contenido que no sea un emoji
    27   1502.8 MiB      0.1 MiB       33901       pddf['content'] = pddf['content'].apply(lambda x: ' '.join(re.findall(r'[\U0001F
    28                                             # separo las celdas que tengan mas de un emoji en filas
    29    246.7 MiB  -1256.2 MiB           1       pddf = pddf.assign(content=pddf['content'].str.split(' ')).explode('content').res
    30                                             # selecciono las filas que no sean vacias
    31    246.7 MiB      0.0 MiB           1       pddf = pddf[pddf['content'] != ' ']
    32
    33                                             # count de emojis
    34    246.7 MiB      0.1 MiB           1       df_topemoji = pddf.groupby('content').agg({'content': ['count']})
    35    246.7 MiB      0.0 MiB           1       df_topemoji = df_topemoji.reset_index()
    36                                             # rename de columnas
    37    246.7 MiB      0.0 MiB           1       df_topemoji.columns = ['content','emoji_count']
    38                                             # orderno desc
    39    246.7 MiB      0.0 MiB           1       df_topemoji=df_topemoji.sort_values(by='emoji_count', ascending=False)
    40                                             # selecciono las top10
    41    246.7 MiB      0.0 MiB           1       df_topemoji = df_topemoji.head(10)
    42                                             # dataframe a lista de tuplas
    43    246.7 MiB      0.0 MiB           1       list_topemoji = list(df_topemoji.to_records(index=False))
```

```
        44    246.7 MiB      0.0 MiB           1       return list_topemoji


         905398 function calls (904779 primitive calls) in 9.277 seconds

    [('🙏', 7286), ('😂', 3072), ('', 3061), ('🚜', 2972), ('✊', 2411), ('🌾', 2363), ('ɪ', 2096), ('ɴ', 2094), ('🏼', 2080), ('❤️',
```

Resultado

[
('🙏', 7286),
('😂', 3072),
('', 3061),
('🚜', 2972),
('✊', 2411),
('🌾', 2363),
('ɪ', 2096),
('ɴ', 2094),
('🏼', 2080),
('❤', 1779)
]

(t+) se aprecia que el consumo de memoria en al menos 4 instrucciones es de ~1500MiB luego bajando a ~246MiB, el tiempo es de aproximados 9.2 seconds

Ejecucion2 optimizada para Memoria

```
%load_ext memory_profiler

import q2_memory as q2m
from memory_profiler import profile
import cProfile
import pstats

@profile
def try_stat():
    profiler=cProfile.Profile()
    profiler.enable()
    li_q2m = q2m.q2_memory('./farmers-protest-tweets-2021-2-4.json')
    profiler.disable()
    stats = pstats.Stats(profiler)
    stats.print_stats(0)
    print(li_q2m)

if __name__ == '__main__':
    try_stat()
```

```
    The memory_profiler extension is already loaded. To reload it, use:
      %reload_ext memory_profiler
    ERROR: Could not find file C:\Users\Usuario\AppData\Local\Temp\ipykernel_8176\3497146961.py
    Filename: c:\Users\Usuario\Downloads\challenge_DE\src\q2_memory.py

    Line #    Mem usage    Increment  Occurrences   Line Contents
    =============================================================
        18    236.2 MiB    236.2 MiB           1   @profile
        19                                         def q2_memory(file_path: str) -> List[Tuple[datetime.date, str]]:
        20    236.2 MiB      0.0 MiB           1       with open(file_path, 'r') as f:
        21    640.6 MiB -1166168.4 MiB      117410           data = [[json.loads(line)['content']]  for line in f.readlines()]
        22    555.0 MiB    -85.6 MiB           1       columnas = ['content']
        23    554.4 MiB     -0.6 MiB           1       pddf = pd.DataFrame(data, columns=columnas)
        24                                             # elimino todo contenido que no sea un emoji
        25    556.2 MiB      1.8 MiB      234815       pddf['content'] = pddf['content'].apply(lambda x: ' '.join(re.findall(r'[\U0001F
        26                                             # separo las celdas que tengan mas de un emoji en filas
        27    558.3 MiB      2.1 MiB           1       pddf = pddf.assign(content=pddf['content'].str.split(' ')).explode('content').res
        28                                             # selecciono las filas que no sean vacias
        29    558.3 MiB      0.0 MiB           1       pddf = pddf[pddf['content'] != ' ']
        30
        31                                             # count de emojis
        32    558.3 MiB      0.0 MiB           1       df_topemoji = pddf.groupby('content').agg({'content': ['count']})
        33    558.3 MiB      0.0 MiB           1       df_topemoji = df_topemoji.reset_index()
        34                                             # rename de columnas
        35    558.3 MiB      0.0 MiB           1       df_topemoji.columns = ['content','emoji_count']
        36                                             # orderno desc
        37    558.3 MiB      0.0 MiB           1       df_topemoji=df_topemoji.sort_values(by='emoji_count', ascending=False)
        38                                             # selecciono las top10
    39    558.3 MiB      0.0 MiB           1       df_topemoji = df_topemoji.head(10)
    40                                             # dataframe a lista de tuplas
    41    558.3 MiB      0.0 MiB           1       list_topemoji = list(df_topemoji.to_records(index=False))
    42    558.3 MiB      0.0 MiB           1       return list_topemoji


         2232748 function calls (2232547 primitive calls) in 11.611 seconds

[('', 100457), ('🙏', 7286), ('😂', 3072), ('', 3061), ('🚜', 2972), ('✊', 2411), ('🌾', 2363), ('ɪ', 2096), ('ɴ', 2094), ('🏽',
```

Resultado

[
('🙏', 7286),
('😂', 3072),
('', 3061),
('🚜', 2972),
('✊', 2411),
('🌾', 2363),
('ɪ', 2096),
('ɴ', 2094),
('🏽', 2080),
('❤', 1779)
]

(m+) se aprecia que el consumo de memoria en cada instruccion es tiene como maximo 640MiB en una sola instruccion, luego bajando a 558MiB para la mayoria. el tiempo subio a aproximados ~11.6 seconds

Ejercicio 3:

El top 10 histórico de usuarios (username) más influyentes en función del conteo de las menciones (@) que registra cada uno de ellos.

Supuesto1:

Ejecucion3 optimizada para Tiempo

```
%load_ext memory_profiler

import q3_time as q3t
from memory_profiler import profile
import cProfile
import pstats

@profile
def try_stat():
    profiler=cProfile.Profile()
    profiler.enable()
    li_q3t = q3t.q3_time('./farmers-protest-tweets-2021-2-4.json')
    profiler.disable()
    stats = pstats.Stats(profiler)
    stats.print_stats(0)
    print(li_q3t)

if __name__ == '__main__':
    try_stat()
```

```
The memory_profiler extension is already loaded. To reload it, use:
  %reload_ext memory_profiler
ERROR: Could not find file C:\Users\Usuario\AppData\Local\Temp\ipykernel_8176\565335560.py
c:\Users\Usuario\Downloads\challenge_DE\src\q3_time.py:25: UserWarning: This pattern is interpreted as a regular expression, and ha
  pddf = pddf[pddf['content'].str.contains(r'(?:[@]([a-zA-Z0-9_]+|$))', regex=True)]
Filename: c:\Users\Usuario\Downloads\challenge_DE\src\q3_time.py

Line #    Mem usage    Increment  Occurrences   Line Contents
=============================================================
    17    163.7 MiB    163.7 MiB           1   @profile
    18                                         def q3_time(file_path: str) -> List[Tuple[datetime.date, str]]:
    19    163.7 MiB      0.0 MiB           1       pd.set_option('mode.chained_assignment', None)  #evita warning
    20                                             #cargo el jsonl como dataframe pandas
    21   1958.3 MiB   1794.5 MiB           1       pddf=pd.read_json(file_path, lines=True)
    22                                             #trabajo solo con una columna, la que hace referencia al body del tweet
    23   1901.1 MiB    -57.2 MiB           1       pddf=pddf[['content']]
    24                                             # selecciono solo las filas que tengan al menos un emoji
    25   1899.4 MiB     -1.7 MiB           1       pddf = pddf[pddf['content'].str.contains(r'(?:[@]([a-zA-Z0-9_]+|$))', regex=True
    26                                            # elimino todo contenido que no sea un emoji
    27   1899.4 MiB      0.0 MiB       76349       pddf['content'] = pddf['content'].apply(lambda x: ' '.join(re.findall(r'(?:[@]([a
    28                                            # separo las celdas que tengan mas de un emoji en filas
    29    659.9 MiB  -1239.5 MiB           1       pddf = pddf.assign(content=pddf['content'].str.split(' ')).explode('content').res
    30                                            # selecciono las filas que no sean vacias
    31    659.9 MiB      0.0 MiB           1       pddf = pddf[pddf['content'] != ' ']
    32
    33                                            # count de emojis
    34    660.0 MiB      0.0 MiB           1       df_topmention = pddf.groupby('content').agg({'content': ['count']})
    35    660.0 MiB      0.0 MiB           1       df_topmention = df_topmention.reset_index()
    36                                            # rename de columnas
    37    660.0 MiB      0.0 MiB           1       df_topmention.columns = ['content','mention_count']
    38                                            # orderno desc
    39    660.0 MiB      0.0 MiB           1       df_topmention=df_topmention.sort_values(by='mention_count', ascending=False)
    40                                            # selecciono las top10
    41    660.0 MiB      0.0 MiB           1       df_topmention = df_topmention.head(10)
    42                                            # dataframe a lista de tuplas
    43    660.0 MiB      0.0 MiB           1       list_topmention = list(df_topmention.to_records(index=False))
    44    660.0 MiB      0.0 MiB           1       return list_topmention


         1075448 function calls (1074848 primitive calls) in 9.591 seconds

[('narendramodi', 2261), ('Kisanektamorcha', 1836), ('RakeshTikaitBKU', 1641), ('PMOIndia', 1422), ('RahulGandhi', 1125), ('GretaThu
```

Resultado

[
('narendramodi', 2261),
('Kisanektamorcha', 1836),
('RakeshTikaitBKU', 1641),
('PMOIndia', 1422),
('RahulGandhi', 1125),
('GretaThunberg', 1046),
('RaviSinghKA', 1015),
('rihanna', 972),
('UNHumanRights', 962),
('meenaharris', 925)
]

(t+) se aprecia que el consumo de memoria en al menos 4 instrucciones es de ~1900MiB luego bajando a ~660MiB, el tiempo es de aproximados 9.5 seconds

Ejecucion3 optimizada para Memoria

```
%load_ext memory_profiler

import q3_memory as q3m
from memory_profiler import profile
import cProfile
import pstats

@profile
def try_stat():
    profiler=cProfile.Profile()
    profiler.enable()
    li_q3m = q3m.q3_memory('./farmers-protest-tweets-2021-2-4.json')
    profiler.disable()
    stats = pstats.Stats(profiler)
    stats.print_stats(0)
    print(li_q3m)

if __name__ == '__main__':
    try_stat()
```

```
The memory_profiler extension is already loaded. To reload it, use:
  %reload_ext memory_profiler
ERROR: Could not find file C:\Users\Usuario\AppData\Local\Temp\ipykernel_8176\3965048329.py
Filename: c:\Users\Usuario\Downloads\challenge_DE\src\q3_memory.py

Line #    Mem usage    Increment  Occurrences   Line Contents
=============================================================
    18    649.5 MiB    649.5 MiB           1   @profile
    19                                         def q3_memory(file_path: str) -> List[Tuple[datetime.date, str]]:
    20    649.5 MiB      0.0 MiB           1       with open(file_path, 'r') as f:
```

```
    21    703.9 MiB -765573.5 MiB      117410           data = [[json.loads(line)['content']]  for line in f.readlines()]
    22    641.8 MiB    -62.1 MiB           1       columnas = ['content']
    23    642.7 MiB      0.9 MiB           1       pddf = pd.DataFrame(data, columns=columnas)
    24                                             #trabajo solo con una columna, la que hace referencia al body del tweet
    25    643.6 MiB      0.9 MiB           1       pddf=pddf[['content']]
    26                                             # elimino todo contenido que no sea un emoji
    27    644.5 MiB      0.9 MiB      234815       pddf['content'] = pddf['content'].apply(lambda x: ' '.join(re.findall(r'(?:[@]([a
    28                                             # separo las celdas que tengan mas de un emoji en filas
    29    645.9 MiB      1.4 MiB           1       pddf = pddf.assign(content=pddf['content'].str.split(' ')).explode('content').res
    30                                             # selecciono las filas que no sean vacias
    31    645.9 MiB      0.0 MiB           1       pddf = pddf[pddf['content'] != ' ']
    32
    33                                             # count de emojis
    34    645.9 MiB      0.0 MiB           1       df_topmention = pddf.groupby('content').agg({'content': ['count']})
    35    645.9 MiB      0.0 MiB           1       df_topmention = df_topmention.reset_index()
    36                                             # rename de columnas
    37    645.9 MiB      0.0 MiB           1       df_topmention.columns = ['content','mention_count']
    38                                             # orderno desc
    39    645.9 MiB      0.0 MiB           1       df_topmention=df_topmention.sort_values(by='mention_count', ascending=False)
    40                                             # selecciono las top10
    41    645.9 MiB      0.0 MiB           1       df_topmention = df_topmention.head(10)
    42                                             # dataframe a lista de tuplas
    43    645.9 MiB      0.0 MiB           1       list_topmention = list(df_topmention.to_records(index=False))
    44    645.9 MiB      0.0 MiB           1       return list_topmention


         2233520 function calls (2233304 primitive calls) in 11.398 seconds

[('', 79253), ('narendramodi', 2261), ('Kisanektamorcha', 1836), ('RakeshTikaitBKU', 1641), ('PMOIndia', 1422), ('RahulGandhi', 112
```

Resultado

[
('narendramodi', 2261),
('Kisanektamorcha', 1836),
('RakeshTikaitBKU', 1641),
('PMOIndia', 1422),
('RahulGandhi', 1125),
('GretaThunberg', 1046),
('RaviSinghKA', 1015),
('rihanna', 972),
('UNHumanRights', 962),
('meenaharris', 925)
]

(m+) se aprecia que el consumo de memoria tiene como maximo 650MiB en una sola instruccion, luego bajando a 150MiB para la mayoria. el tiempo subio a aproximados ~11.3 seconds