```
import sys
import bz2
import json
from pathlib import Path

input_directory = Path(sys.argv[1])

hashtags_nodes = {}
hashtags_edges = {}

users_nodes = {}
users_edges = {}

for input_filename in input_directory.glob("**/*.bz2"):

    with bz2.open(input_filename, "rb") as f:

        for line in f:

            tweet = json.loads(line)

            if "delete" in tweet:
                continue

            if "extended_tweet" in tweet:
                tweet_hashtags = tweet["extended_tweet"]["entities"]["hashtags"]
                tweet_mentions = tweet["extended_tweet"]["entities"]["user_mentions"]
            else:
                tweet_hashtags = tweet["entities"]["hashtags"]
                tweet_mentions = tweet["entities"]["user_mentions"]

            # Calcolo le occorrenze degli hashtag
            for hashtag in tweet_hashtags:
                hashtag_text = hashtag["text"]
                if hashtag_text not in hashtags_nodes:
                    hashtags_nodes[hashtag_text] = 1
                else:
                    hashtags_nodes[hashtag_text] += 1

            # Calcolo le occorrenze delle menzioni
            for mention in tweet_mentions:
                screen_name = mention["screen_name"]
                if screen_name not in users_nodes:
```

```
        users_nodes[screen_name] = 1
    else:
        users_nodes[screen_name] += 1

# Calcolo le occorrenze dell'autore del tweet
screen_name = tweet["user"]["screen_name"]
if screen_name not in users_nodes:
    users_nodes[screen_name] = 1
else:
    users_nodes[screen_name] += 1

# Calcolo le occorrenze dell'autore del tweet retwittato
if "retweeted_status" in tweet:
    screen_name = tweet["retweeted_status"]["user"]["screen_name"]
    if screen_name not in users_nodes:
        users_nodes[screen_name] = 1
    else:
        users_nodes[screen_name] += 1

# Calcolo le co-occorrenze degli hashtag
for h1 in tweet_hashtags: # Ciclo sugli hashtag: A, B, C

    for h2 in tweet_hashtags: # Ciclo sugli hashtag: A, B, C

        if h2["text"] != h1["text"]: # Considero solo gli hashtag diversi

            # Ordino la coppia di hashtag in ordine alfabetico
            if h2["text"] > h1["text"]:
                hashtags_edge_key = h1["text"] + "," + h2["text"] # Esempio: "A,B"
            else:
                hashtags_edge_key = h2["text"] + "," + h1["text"] # Esempio: "A,B"

            if hashtags_edge_key not in hashtags_edges:
                hashtags_edges[hashtags_edge_key] = 1
            else:
                hashtags_edges[hashtags_edge_key] += 1

    # "Sono un tweet #A #B #C"
    # A,B -> 1
    # A,C -> 1
    # B,C -> 1

# Calcolo le co-occorrenze delle menzioni
```

```
        for u1 in tweet_mentions: # Ciclo sulle menzioni: A, B, C

            # Ordino la coppia autore / menzione in ordine alfabetico
            if tweet["user"]["screen_name"] > u1["screen_name"]:
                mentions_edge_key = u1["screen_name"] + "," + tweet["user"]["screen_name"] #
Esempio: "A,B"
            else:
                mentions_edge_key = tweet["user"]["screen_name"] + "," + u1["screen_name"] #
Esempio: "A,B"

            if mentions_edge_key not in users_edges:
                users_edges[mentions_edge_key] = 2
            else:
                users_edges[mentions_edge_key] += 2

            for u2 in tweet_mentions: # Ciclo sulle menzioni: A, B, C

                if u2["screen_name"] != u1["screen_name"]: # Considero solo le menzioni diverse

                    # Ordino la coppia di menzioni in ordine alfabetico
                    if u2["screen_name"] > u1["screen_name"]:
                        mentions_edge_key = u1["screen_name"] + "," + u2["screen_name"] #
Esempio: "A,B"
                    else:
                        mentions_edge_key = u2["screen_name"] + "," + u1["screen_name"] #
Esempio: "A,B"

                    if mentions_edge_key not in users_edges:
                        users_edges[mentions_edge_key] = 1
                    else:
                        users_edges[mentions_edge_key] += 1

            # "Sono un tweet @A @B @C"
            # A,B -> 1
            # A,C -> 1
            # B,C -> 1

with open("hashtags_nodes.csv", "w") as f:
    f.write("Id,Label,Weight\n")
    for key in hashtags_nodes:
        if hashtags_nodes[key] > 1:
            try:
                f.write(key + "," + key + "," + str(hashtags_nodes[key]) + "\n")
```

```
            except:
                continue

with open("hashtags_edges.csv", "w") as f:
    f.write("Source,Target,Weight\n")
    for key in hashtags_edges:
        if hashtags_edges[key]//2 > 1:
            try:
                f.write(key + "," + str(hashtags_edges[key]//2) + "\n")
            except:
                continue

with open("users_nodes.csv", "w") as f:
    f.write("Id,Label,Weight\n")
    for key in users_nodes:
        if users_nodes[key] > 1:
            try:
                f.write(key + "," + key + "," + str(users_nodes[key]) + "\n")
            except:
                continue

with open("users_edges.csv", "w") as f:
    f.write("Source,Target,Weight\n")
    for key in users_edges:
        if users_edges[key]//2 > 1:
            try:
                f.write(key + "," + str(users_edges[key]//2) + "\n")
            except:
                continue
```