1 Birthday Attack

To run the script do:

```
cd problem-2
python3 bithdayattack.py
```

A cryptographic hash function is an exceptional class of hash function that has certain properties that make it appropriate for use in cryptography. It is a numerical algorithm that maps information of self-assertive size to a piece line of a fixed size (a hash function) which is intended to likewise be a one-way output function, that is, a function which is infeasible to revert.

To calculate the cryptographic hash value in Python, "hashlib" Module is used. The hashlib gives the following cryptographic hash functions to discover the hash output of a text as follows:

- SHA3-224 – 28 bit Digest-Size
- SHA3-256 – 32 bit Digest-Size
- SHA3-384 – 48 bit Digest-Size
- SHA3-512 – 64 bit Digest-Size

I have used SHA3-256 for the birthday attack problem.

Birthday Attack Algorithm:

```
def birthday_attack(d):
  log_dict = {}
  attempts = 0
  while True:
    hash_num = random.random()
    digest = hashlib.sha3_256(str(hash_num)).hexdigest()
    dbits = hex_to_bin(digest)[:d]
    if dbits in log_dict:
      return attempts, hash_num, log_dict[dbits], digest
    log_dict[dbits] = hash_num
    attempts += 1
```

I have computed the tuple (s1, s2, h, m, n) for d = 1 to d = 24. The maximum number of attempts required over all d values is 105 attempts and the maximum memory used over all values of d is 20024 KB.

Conclusion: As the value of d increases i.e the number of bits to be same in the hash output of two strings, the memory used increases and the number of attempts also increase. The values of d asked from us is very small, however in real life it might be large and thus require more memory and attempts to find the collision. In such cases the hash function can be optimized to speedup the process of finding hash output. The following strategy can be used as described here in point 3
https://iopscience.iop.org/article/10.1088/1742-6596/1486/3/032004/pdf