Комп'ютерний практикум

In [ ]:

```python
import hashlib as hl                    # sha1
import secrets as rnd                   # randbelow, choice
import os, sys                          # output and logging
import numpy as np, scipy.stats as sp   # stats calculations
import matplotlib.pyplot as plt         # plotting
```

In [ ]:

```python
# Some hardcoded stuff specifically for sha1
text = ' \t\n\r\v\fabcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789'
punct = r"""!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~"""
ALPH = text + punct

PIA_CMP_PREF_C = 40 - 4    # 4 chars to compare in the end
BDA_CMP_PREF_C = 40 - 8    # 8 chars to compare in the end

PIA_CMP_PREF_B = 20 - 2    # 2 bytes to compare in the end
BDA_CMP_PREF_B = 20 - 4    # 4 bytes to compare in the end

N = 125                    # Iteration count

STD_OS = sys.stdout

gamma = 0.95               # Confidence value
# ------------------------------------------

BASE_MESSAGE_PARTS = ["Bondar", "Petro", "Olexandrovych", ""]
```

In [ ]:

```python
def gen_bm_sample(char_counts=[0, 0, 0, 0]):
    result_msg_parts = []

    for i in range(4):
        if char_counts[i] != 0:
            gen_part = str()
            for _ in range(char_counts[i]):
                gen_part += rnd.choice(ALPH[6:])
            result_msg_parts.append(gen_part)
        result_msg_parts.append(BASE_MESSAGE_PARTS[i])

    return "".join(result_msg_parts)

def random_modification(base: str):
    res = list(base)

    for i in range(len(res)):
        if rnd.randbelow(2) == 1:
            res[i] = rnd.choice(ALPH)

    return "".join(res)

def rand_count_vec():
    return ([rnd.randbelow(6),
             rnd.randbelow(6),
             rnd.randbelow(6),
             rnd.randbelow(6)])
```

In [ ]:

```
def preimage_attack_1(base_message: str, log=False, log_output='preimage_attack_1.txt'):
    print(f"Base message is: {base_message}\n")
    # Formating parameters and output
    pw = len(base_message) + 6

    out = sys.stdout
    if log:
        out = open(log_output, 'w')
    #

    base_sha = hl.sha1(base_message.encode())

    out.write(f"{base_message:{pw}s} : {base_sha.hexdigest()[:PIA_CMP_PREF_C]}\t{base_sha.hexdigest()[PIA_CMP_PREF_C:]}\n")
    if log: print(f"{base_message:{pw}s} : {base_sha.hexdigest()[:PIA_CMP_PREF_C]}\t{base_sha.hexdigest()[PIA_CMP_PREF_C:]}")

    app = 0
    while True:
        app += 1

        msg = base_message + str(app)
        msg_sha = hl.sha1(msg.encode())

        out.write(f"{msg:{pw}s} : {msg_sha.hexdigest()[:PIA_CMP_PREF_C]}\t{msg_sha.hexdigest()[PIA_CMP_PREF_C:]}")

        if msg_sha.digest()[PIA_CMP_PREF_B:] == base_sha.digest()[PIA_CMP_PREF_B:]:
            out.write(f"\tSecond preimage on {app}!")
            if log:
                print(f"{msg:{pw}s} : {msg_sha.hexdigest()[:PIA_CMP_PREF_C]}\t{msg_sha.hexdigest()[PIA_CMP_PREF_C:]}", end='')
                print(f"\tSecond preimage on {app}!")

            break
        else:
            out.write('\n')

    if out != sys.stdout:
        out.close()

    return app
```

In [ ]:

```
def preimage_attack_2(base_message: str, log=False, log_output='preimage_attack_2.txt'):
    print(f"Base message is: {base_message}\n")
    # Formating parameters
    pw = len(base_message) + 8

    out = sys.stdout
    if log:
        out = open(log_output, 'w')
    #

    base_sha = hl.sha1(base_message.encode())

    out.write(f"{repr(base_message):{pw}s} : {base_sha.hexdigest()[:PIA_CMP_PREF_C]}\t{base_sha.hexdigest()[PIA_CMP_PREF_C:]}\n")
    if log: print(f"{repr(base_message):{pw}s} : {base_sha.hexdigest()[:PIA_CMP_PREF_C]}\t{base_sha.hexdigest()[PIA_CMP_PREF_C:]}")

    itr = 0
    while True:
        itr += 1
        msg = random_modification(base_message)
        msg_sha = hl.sha1(msg.encode())

        out.write(f"{repr(msg):{pw}s} : {msg_sha.hexdigest()[:PIA_CMP_PREF_C]}\t{msg_sha.hexdigest()[PIA_CMP_PREF_C:]}")

        if msg_sha.digest()[PIA_CMP_PREF_B:] == base_sha.digest()[PIA_CMP_PREF_B:] and msg != base_message:
            out.write(f"\tSecond preimage in {itr} iterations!\n")
            if log:
                print(f"{repr(msg):{pw}s} : {msg_sha.hexdigest()[:PIA_CMP_PREF_C]}\t{msg_sha.hexdigest()[PIA_CMP_PREF_C:]}", end='')
                print(f"\tSecond preimage in {itr} iterations!")

            break
        else:
            out.write('\n')

    if out != sys.stdout:
        out.close()

    return itr
```

Атака пошуку другого прообразу 1

```
In [ ]: def birthday_attack_1(base_message: str, log=False, log_output='birthday_attack_1.txt'):
            print(f"Base message is: {base_message}\n")
            # Formating parameters
            pw = len(base_message) + 6

            out = sys.stdout
            if log:
                out = open(log_output, 'w')
            #

            base_sha = hl.sha1(base_message.encode())

            out.write(f"{base_message:{pw}s} : {base_sha.hexdigest()[:BDA_CMP_PREF_C]}\t{base_sha.hexdigest()[BDA_CMP_PREF_C:]}\n")
            if log: print(f"{base_message:{pw}s} : {base_sha.hexdigest()[:BDA_CMP_PREF_C]}\t{base_sha.hexdigest()[BDA_CMP_PREF_C:]}")

            prev_dict = {base_sha.digest()[BDA_CMP_PREF_B:]: base_message}
            app = 0
            while True:
                app += 1

                msg = base_message + str(app)
                msg_sha = hl.sha1(msg.encode())

                if msg_sha.digest()[BDA_CMP_PREF_B:] in prev_dict.keys():
                    col = prev_dict[msg_sha.digest()[BDA_CMP_PREF_B:]]
                    col_sha = hl.sha1(col.encode())

                    out.write(f"\nCollision found in {app} iterations!\n")
                    out.write(f"Msg: {msg:{pw}s} : {msg_sha.hexdigest()[:BDA_CMP_PREF_C]}\t{msg_sha.hexdigest()[BDA_CMP_PREF_C:]}\n")
                    out.write(f"Col: {col:{pw}s} : {col_sha.hexdigest()[:BDA_CMP_PREF_C]}\t{col_sha.hexdigest()[BDA_CMP_PREF_C:]}\n")
                    if log:
                        print(f"\nCollision found in {app} iterations!")
                        print(f"Msg: {msg:{pw}s} : {msg_sha.hexdigest()[:BDA_CMP_PREF_C]}\t{msg_sha.hexdigest()[BDA_CMP_PREF_C:]}")
                        print(f"Col: {col:{pw}s} : {col_sha.hexdigest()[:BDA_CMP_PREF_C]}\t{col_sha.hexdigest()[BDA_CMP_PREF_C:]}")

                    break
                else:
                    out.write(f"{msg:{pw}s} : {msg_sha.hexdigest()[:BDA_CMP_PREF_C]}\t{msg_sha.hexdigest()[BDA_CMP_PREF_C:]}\n")
                    prev_dict[msg_sha.digest()[BDA_CMP_PREF_B:]] = msg

            if out != sys.stdout:
                out.close()

            return app
```

In [ ]:

```
def birthday_attack_2(base_message: str, log=False, log_output='birthday_attack_2.txt'):
    print(f"Base message is: {base_message}\n")
    # Formating parameters
    pw = len(base_message) + 8

    out = sys.stdout
    if log:
        out = open(log_output, 'w')
    #

    base_sha = hl.sha1(base_message.encode())

    out.write(f"{repr(base_message):{pw}s} : {base_sha.hexdigest()[:BDA_CMP_PREF_C]}\t{base_sha.hexdigest()[BDA_CMP_PREF_C:]}\n")
    if log: print(f"{repr(base_message):{pw}s} : {base_sha.hexdigest()[:BDA_CMP_PREF_C]}\t{base_sha.hexdigest()[BDA_CMP_PREF_C:]}")

    prev_dict = {base_sha.digest()[BDA_CMP_PREF_B:]: base_message}
    itr = 0
    while True:
        itr += 1
        msg = random_modification(base_message)
        msg_sha = hl.sha1(msg.encode())

        if msg_sha.digest()[BDA_CMP_PREF_B:] in prev_dict.keys() and msg != base_message:
            col = prev_dict[msg_sha.digest()[BDA_CMP_PREF_B:]]
            col_sha = hl.sha1(col.encode())

            out.write(f"\nCollision found in {itr} iterations!\n")
            out.write(f"Msg: {repr(msg):{pw}s} : {msg_sha.hexdigest()[:BDA_CMP_PREF_C]}\t{msg_sha.hexdigest()[BDA_CMP_PREF_C:]}\n")
            out.write(f"Col: {repr(col):{pw}s} : {col_sha.hexdigest()[:BDA_CMP_PREF_C]}\t{col_sha.hexdigest()[BDA_CMP_PREF_C:]}\n")
            if log:
                print(f"\nCollision found in {itr} iterations!")
                print(f"Msg: {repr(msg):{pw}s} : {msg_sha.hexdigest()[:BDA_CMP_PREF_C]}\t{msg_sha.hexdigest()[BDA_CMP_PREF_C:]}")
                print(f"Col: {repr(col):{pw}s} : {col_sha.hexdigest()[:BDA_CMP_PREF_C]}\t{col_sha.hexdigest()[BDA_CMP_PREF_C:]}")

            break
        else:
            out.write(f"{repr(msg):{pw}s} : {msg_sha.hexdigest()[:BDA_CMP_PREF_C]}\t{msg_sha.hexdigest()[BDA_CMP_PREF_C:]}\n")
            prev_dict[msg_sha.digest()[BDA_CMP_PREF_B:]] = msg

    if out != sys.stdout:
        out.close()

    return itr
```

Запуск атак

Атака пошуку другого прообразу 1

In [ ]:

```
general_stats = {}
```

Перша атака

In [ ]:

```
preimage_attack_1(gen_bm_sample([5, 0, 0, 0]), True)
```

```
Base message is: +Qx11BondarPetroOlexandrovych

+Qx11BondarPetroOlexandrovych       : a36f5af81f2a094e4dd4929be35a39424414      3950
+Qx11BondarPetroOlexandrovych8681   : 42deaf1f1aa95ad85b35a3bdaed18f09db36      3950     Second preimage on 8681!
```

Out[ ]:

```
8681
```

In [ ]:

```
stats = []

with open(os.devnull, 'w') as f:
    sys.stdout = f
    for i in range(N):
        msg = gen_bm_sample(rand_count_vec())
        stats.append((msg, preimage_attack_1(msg, True, os.devnull)))
    sys.stdout = STD_OS

with open('preimage_attack_1_stats.txt', 'w') as f:
    f.write('-' + ('-'*52) + '-' + ('-'*12) + '-\n')
    f.write(f"| {'Start message':50s} | {'Iterations':10s} |\n")
    f.write('|' + ('-'*52) + '+' + ('-'*12) + '|\n')
    for msg, st in stats:
        f.write(f"| {repr(msg):50s} | {st:10} |\n")
    f.write('-' + ('-'*52) + '-' + ('-'*12) + '-\n')

data = [count for _, count in stats]
sum_itr = np.sum(data)
mean_val = np.mean(data)
variance = np.var(data)

# Computing confidence interval
q = 1 - ((1 - gamma) / 2)   # quantile
s = np.sqrt(variance)
scale_t = sp.t.ppf(q, N - 1)     # N - 1 degrees of freedom
confidence_interval = (mean_val - (scale_t * s / np.sqrt(N)), mean_val + (scale_t * s / np.sqrt(N)))

print(f"Sum of iterations: {sum_itr}")
print(f"Mean: {mean_val}")
print(f"Variance: {variance} -> Standart deviance: {s}")
print(f"Confidence interval: {confidence_interval}")

general_stats["1.1"] = [sum_itr, mean_val, variance, s, confidence_interval, stats.copy()]
```

```
Sum of iterations: 8350721
Mean: 66805.768
Variance: 4928728861.362176 -> Standart deviance: 70204.9062485107
Confidence interval: (54377.239312836595, 79234.2966871634)
```

Атака пошуку другого прообразу 2

In [ ]:

```python
preimage_attack_2(gen_bm_sample([0, 5, 0, 0]), True)
```

```
Base message is: Bondar,hEvRPetroOlexandrovych

'Bondar,hEvRPetroOlexandrovych'       : aefbf6bc2e0937fcf2f29cad0f094afe6fca    bfb8
'BJn%ai,hEvR#_troOleTa@drcvych'       : 99cec1c019babb96f89fb964ff12fd70e8d7    bfb8    Second preimage in 165207 iterations!
```

Out[ ]:

```
165207
```

In [ ]:

```python
stats = []

with open(os.devnull, 'w') as f:
    sys.stdout = f
    for i in range(N):
        msg = gen_bm_sample(rand_count_vec())
        stats.append((msg, preimage_attack_2(msg, True, os.devnull)))
    sys.stdout = STD_OS

with open('preimage_attack_2_stats.txt', 'w') as f:
    f.write('-' + ('-'*52) + '-' + ('-'*12) + '-\n')
    f.write(f"| {'Start message':50s} | {'Iterations':10s} |\n")
    f.write('|' + ('-'*52) + '+' + ('-'*12) + '|\n')
    for msg, st in stats:
        f.write(f"| {repr(msg):50s} | {st:10} |\n")
    f.write('-' + ('-'*52) + '-' + ('-'*12) + '-\n')

data = [count for _, count in stats]
sum_itr = np.sum(data)
mean_val = np.mean(data)
variance = np.var(data)

# Computing confidence interval
q = 1 - ((1 - gamma) / 2)   # quantile
s = np.sqrt(variance)
scale_t = sp.t.ppf(q, N - 1)    # N - 1 degrees of freedom
confidence_interval = (mean_val - (scale_t * s / np.sqrt(N)), mean_val + (scale_t * s / np.sqrt(N)))

print(f"Sum of iterations: {sum_itr}")
print(f"Mean: {mean_val}")
print(f"Variance: {variance} -> Standart deviance: {s}")
print(f"Confidence interval: {confidence_interval}")

general_stats["1.2"] = [sum_itr, mean_val, variance, s, confidence_interval, stats.copy()]
```

```
Sum of iterations: 7827064
Mean: 62616.512
Variance: 3454251404.393856 -> Standart deviance: 58772.87983750546
Confidence interval: (52211.8200472472, 73021.20395275281)
```

Атака днів народжень 1

In [ ]:

```python
birthday_attack_1(gen_bm_sample([0, 0, 5, 0]), True)
```

```
Base message is: BondarPetro+3K`NOlexandrovych

BondarPetro+3K`NOlexandrovych       : d796286fcf4fc8abcc48f37df1b410ab  b83ca7c5

Collision found in 12423 iterations!
Msg: BondarPetro+3K`NOlexandrovych12423  : 87d9bd0dbd6dc15755f0640f3e1082fb     fae768e9
Col: BondarPetro+3K`NOlexandrovych656    : 7cd3511c81c9ceb73e7c7bb85a640256     fae768e9
```

Out[ ]:

```
12423
```

In [ ]:

```python
stats = []

with open(os.devnull, 'w') as f:
    sys.stdout = f
    for i in range(N):
        msg = gen_bm_sample(rand_count_vec())
        stats.append((msg, birthday_attack_1(msg, True, os.devnull)))
    sys.stdout = STD_OS

with open('birthday_attack_1_stats.txt', 'w') as f:
    f.write('-' + ('-'*52) + '-' + ('-'*12) + '-\n')
    f.write(f"| {'Start message':50s} | {'Iterations':10s} |\n")
    f.write('|' + ('-'*52) + '+' + ('-'*12) + '|\n')
    for msg, st in stats:
        f.write(f"| {repr(msg):50s} | {st:10} |\n")
    f.write('-' + ('-'*52) + '-' + ('-'*12) + '-\n')

data = [count for _, count in stats]
sum_itr = np.sum(data)
mean_val = np.mean(data)
variance = np.var(data)

# Computing confidence interval
q = 1 - ((1 - gamma) / 2)   # quantile
s = np.sqrt(variance)
scale_t = sp.t.ppf(q, N - 1)    # N - 1 degrees of freedom
confidence_interval = (mean_val - (scale_t * s / np.sqrt(N)), mean_val + (scale_t * s / np.sqrt(N)))

print(f"Sum of iterations: {sum_itr}")
print(f"Mean: {mean_val}")
print(f"Variance: {variance} -> Standart deviance: {s}")
print(f"Confidence interval: {confidence_interval}")

general_stats["2.1"] = [sum_itr, mean_val, variance, s, confidence_interval, stats.copy()]
```

```
Sum of iterations: 10933660
Mean: 87469.28
Variance: 1856544810.8576005 -> Standart deviance: 43087.6410454042
Confidence interval: (79841.38030735757, 95097.17969264243)
```

Атака днів народжень 2

In [ ]:

```python
birthday_attack_2(gen_bm_sample([0, 0, 0, 5]), True)
```

```
Base message is: BondarPetroOlexandrovychX<b<x

'BondarPetroOlexandrovychX<b<x'       : ee52113a11a2c50a2bcd78f0ead3675d        a21f93da

Collision found in 48812 iterations!
Msg: 'BhjlaMbeZ\\oOFex?-f;gv<c{X<b<{'      : 95a6a019006127e2ab7fc54767aac99e   556af8ea
Col: '"oWMmrP\x0csr+7loqanMr#vkMhX<b_x'    : 2a3194980ff33b4c7ef722fcd0aa0e10   556af8ea
```

Out[ ]: 48812

In [ ]:

```
stats = []

with open(os.devnull, 'w') as f:
    sys.stdout = f
    for i in range(N):
        msg = gen_bm_sample(rand_count_vec())
        stats.append((msg, birthday_attack_2(msg, True, os.devnull)))
    sys.stdout = STD_OS

with open('birthday_attack_2_stats.txt', 'w') as f:
    f.write('-' + ('-'*52) + '-' + ('-'*12) + '-\n')
    f.write(f"| {'Start message':50s} | {'Iterations':10s} |\n")
    f.write('|' + ('-'*52) + '+' + ('-'*12) + '|\n')
    for msg, st in stats:
        f.write(f"| {repr(msg):50s} | {st:10} |\n")
    f.write('-' + ('-'*52) + '-' + ('-'*12) + '-\n')

data = [count for _, count in stats]
sum_itr = np.sum(data)
mean_val = np.mean(data)
variance = np.var(data)

# Computing confidence interval
q = 1 - ((1 - gamma) / 2)   # quantile
s = np.sqrt(variance)
scale_t = sp.t.ppf(q, N - 1)    # N - 1 deqrees of freedom
confidence_interval = (mean_val - (scale_t * s / np.sqrt(N)), mean_val + (scale_t * s / np.sqrt(N)))

print(f"Sum of iterations: {sum_itr}")
print(f"Mean: {mean_val}")
print(f"Variance: {variance} -> Standart deviance: {s}")
print(f"Confidence interval: {confidence_interval}")

general_stats["2.2"] = [sum_itr, mean_val, variance, s, confidence_interval, stats.copy()]
```

```
Sum of iterations: 9907192
Mean: 79257.536
Variance: 1544131682.712704 -> Standart deviance: 39295.440991452226
Confidence interval: (72300.97780863625, 86214.09419136374)
```

Візуалізація результатів

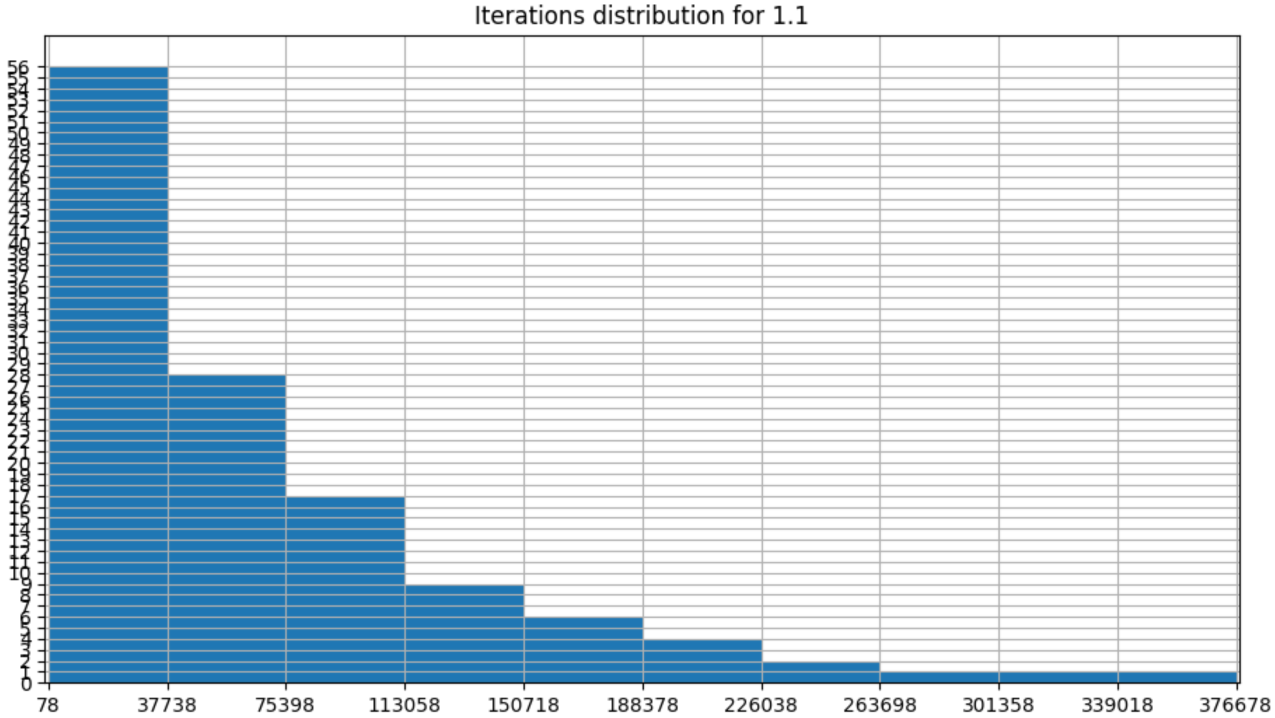
In [ ]:

```
def plot_hist(data_array, atk):
    plt.figure(figsize=(11, 6))
    n, bins, _ = plt.hist(data_array)
    print(n)

    plt.title(f"Iterations distribution for {atk}")
    plt.grid(True)
    plt.yticks(range(int(np.max(n)) + 1))
    plt.xlim([np.min(data_array) - 1000, np.max(data_array) + 1000])
    plt.xticks(bins)
    plt.show()
```

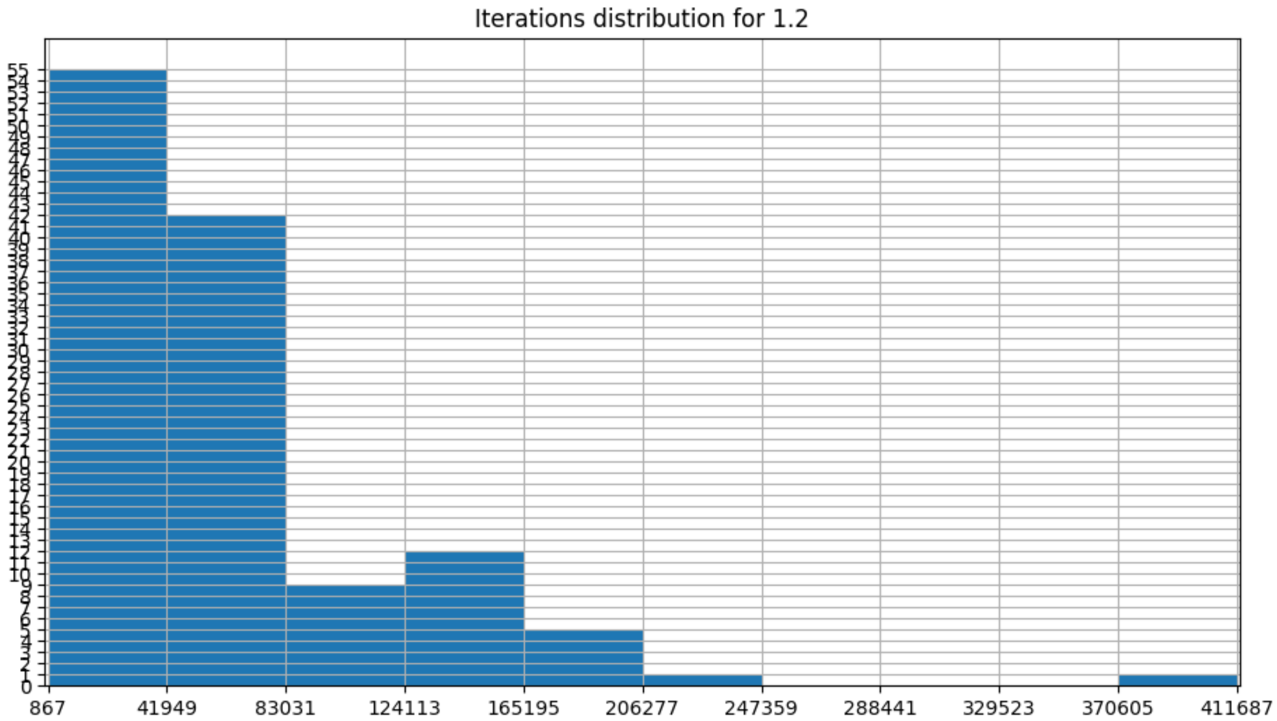
In [ ]:

```
print(f"Number of iterations: {N}", end='\n\n')
for key in general_stats:
    print('Attack ' + key, end=':\n')
    print(f'General number of iterations: {general_stats[key][0]}')
    print(f'Mean value: {general_stats[key][1]}')
    print(f'Variance: {general_stats[key][2]}')
    print(f'Standart deviance: {general_stats[key][3]}')
    print(f'Confidence interval: {general_stats[key][4]}')
    print()

    plot_hist([count for _, count in general_stats[key][5]], key)
```

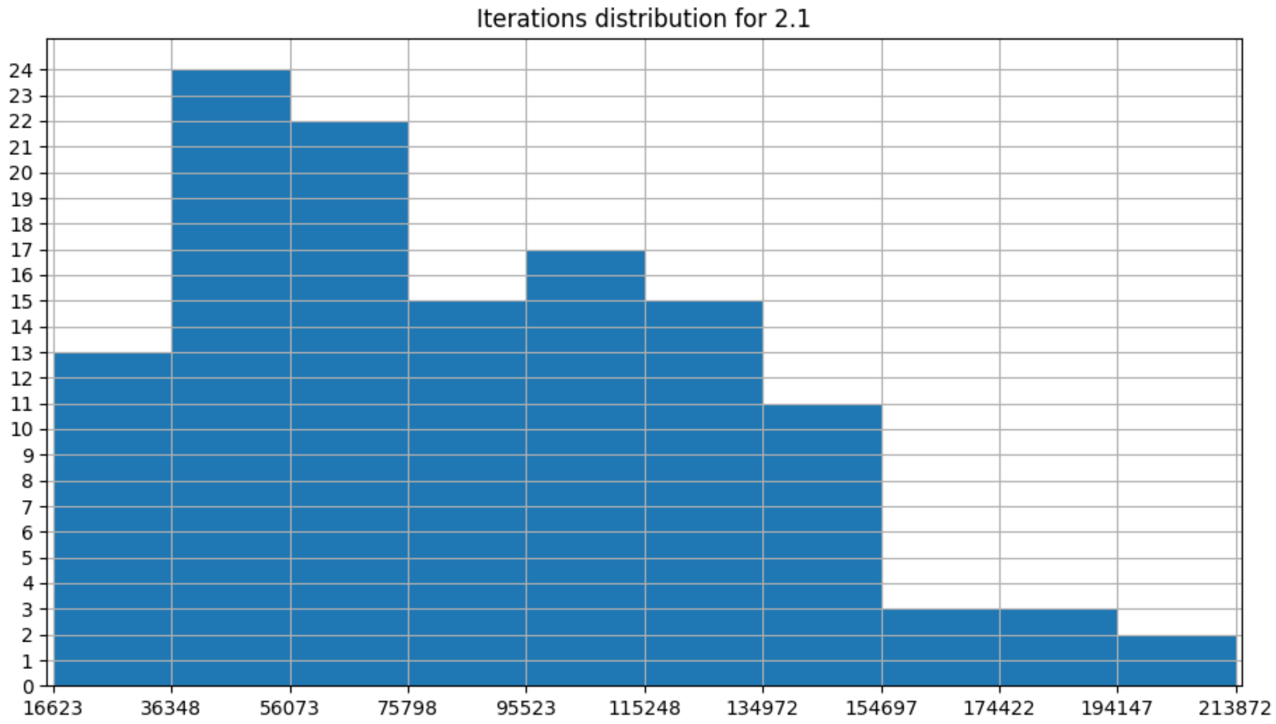
```
Number of iterations: 125

Attack 1.1:
General number of iterations: 8350721
Mean value: 66805.768
Variance: 4928728861.362176
Standart deviance: 70204.9062485107
Confidence interval: (54377.239312836595, 79234.2966871634)

[56. 28. 17.  9.  6.  4.  2.  1.  1.  1.]
```



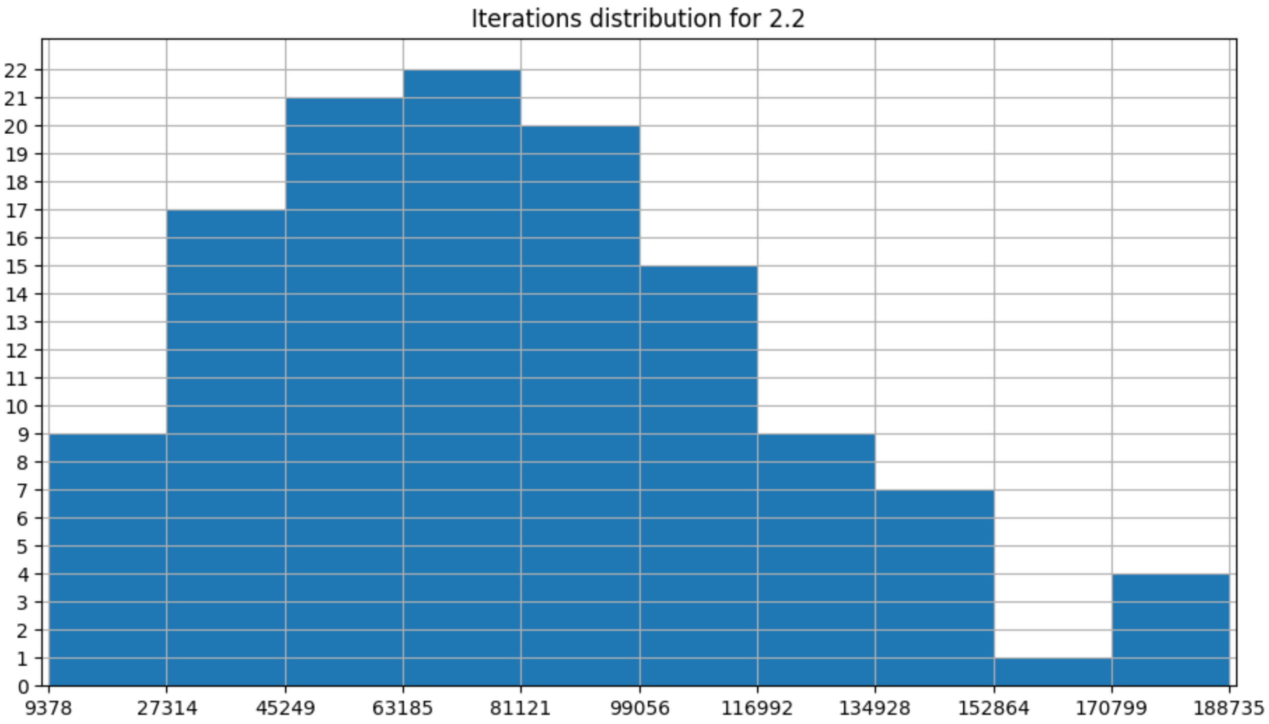
```
Attack 1.2:
General number of iterations: 7827064
Mean value: 62616.512
Variance: 3454251404.393856
Standart deviance: 58772.87983750546
Confidence interval: (52211.8200472472, 73021.20395275281)

[55. 42.  9. 12.  5.  1.  0.  0.  0.  1.]
```



```
Attack 2.1:
General number of iterations: 10933660
Mean value: 87469.28
Variance: 1856544810.8576005
Standart deviance: 43087.6410454042
Confidence interval: (79841.38030735757, 95097.17969264243)

[13. 24. 22. 15. 17. 15. 11.  3.  3.  2.]
```



```
Attack 2.2:
General number of iterations: 9907192
Mean value: 79257.536
Variance: 1544131682.712704
Standart deviance: 39295.440991452226
Confidence interval: (72300.97780863625, 86214.09419136374)

[ 9. 17. 21. 22. 20. 15.  9.  7.  1.  4.]
```


Iterations distribution for 2.2
22
21
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
9378
27314
45249
63185
81121
99056
116992
134928
152864
170799
188735