



Лабораторная работа #1: хэширование

Выполнили: Смольникова Полина, Макридин Максим

Рассмотренные алгоритмы:

- хэширование цепочками 
- хэширование по методу открытой адресации
 - Линейное перехэширование
 - Двоичное перехэширование
 - Квадратичное перехэширование
- метод кукушки 

Алгоритмы написаны на C++

Юнит-тесты реализованы с помощью Google Tests Тестирование производительности производилось на таблице 4n размера относительно кол-ва используемых элементов размера n.

Данные производительности записывались в .csv -файлы и визуализировались в Python

[Ссылка на репозиторий с кодом, тестами и маленьким CI](#)

int-ы

$$h_{a,b}(x) = ((ax + b) \bmod 2^w) \div 2^{w-M}$$

w - размер машинного слова, $w = 32, 64, 128$

$m = 2^M$ - размер таблицы

$a \in \{0, 1, \dots, 2^w - 1\}$, нечетное

$b \in \{0, \dots, 2^{w-M} - 1\}$

In [232...

```
import csv
import matplotlib.pyplot as plt

def get_x_ydict_from_csv(filename: str):
    hash_insert_int = {}
    with open(filename) as csvfile:
        reader = csv.reader(csvfile, delimiter=',')
        for row in reader:
            key = row[0]
            hash_insert_int[key] = list(map(int, row[1:]))
            # print(len(row))
    keys = list(hash_insert_int.keys())
    indices = [i for i in range(len(hash_insert_int[keys[0]])
    return indices, hash_insert_int
# maximum = max([max(hash_insert_int[key]) for key in keys])
# maximum

def print_dict_performance():
    pass
```

Вставка int-ов

In [239...

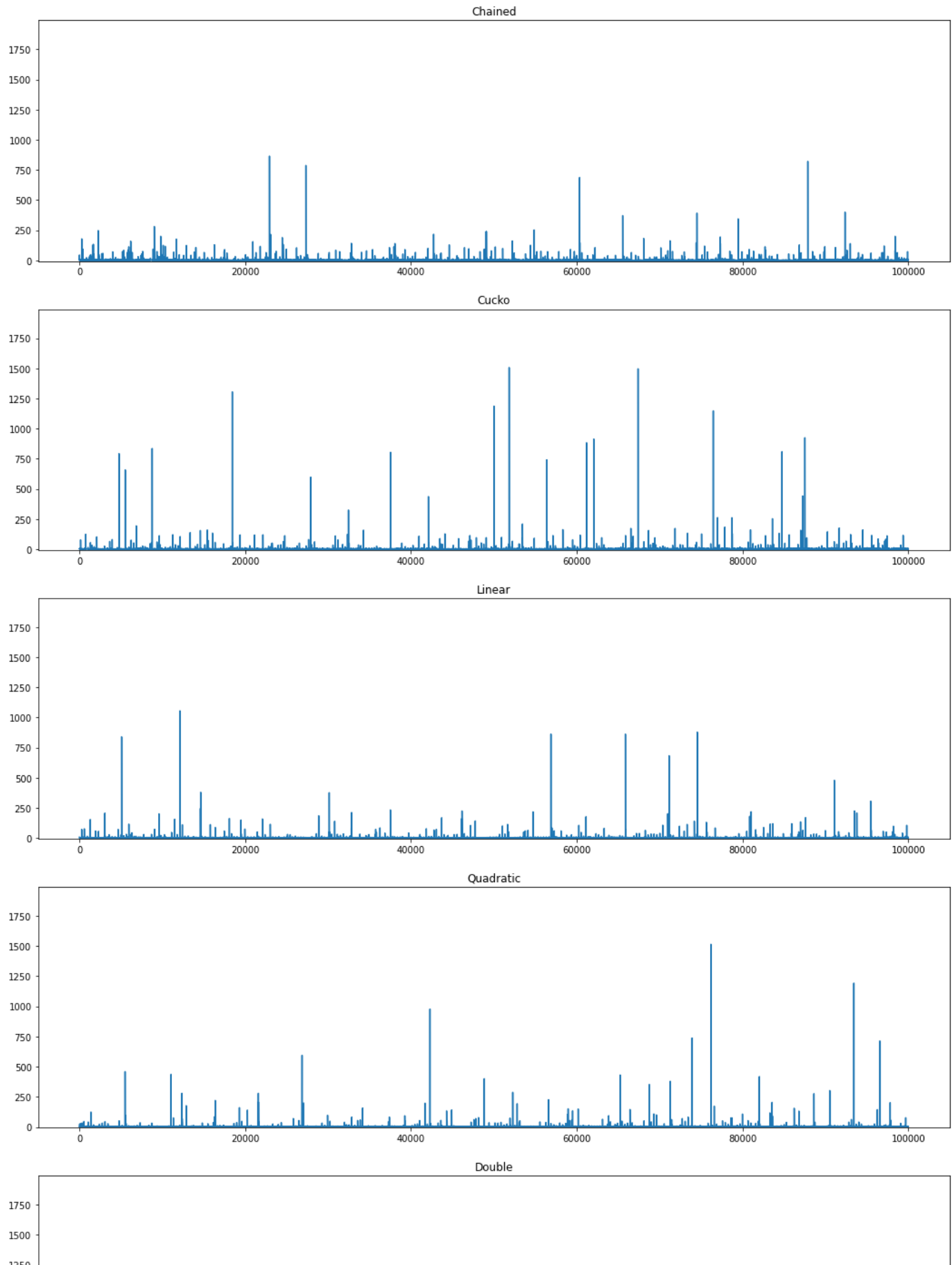
```
indices, data = get_x_ydict_from_csv('data/hash_insert_int.csv')
keys = list(data.keys())
maximum = 0
```

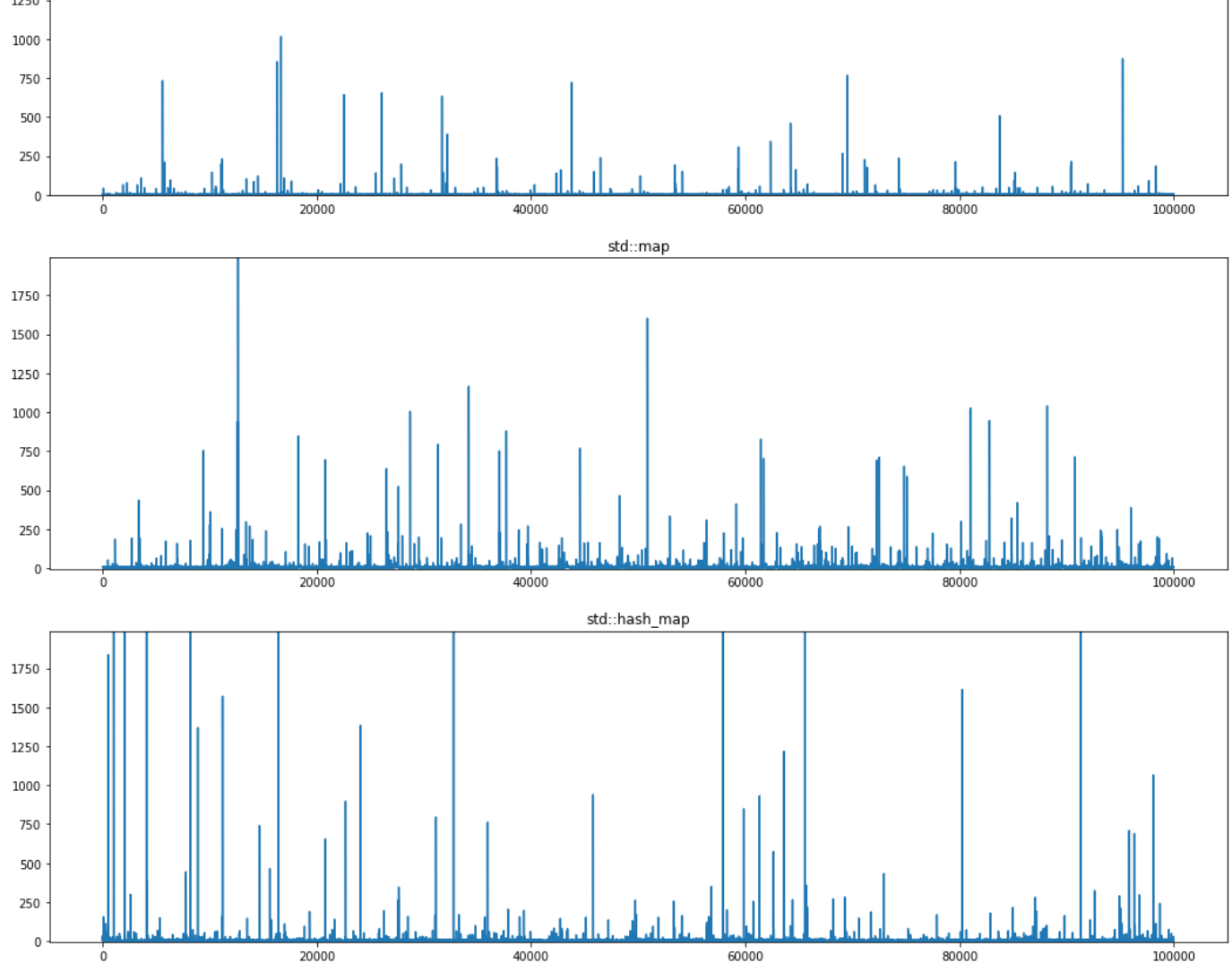
```

for key in keys:
    if max(data[key]) > maximum and key not in ['std::hash_map']:
        maximum = max(data[key])

fig, plots = plt.subplots(7, 1, figsize=(18, 40))
fig.patch.set_facecolor('xkcd:white')
for i in range(7):
    plots[i].set_title(keys[i])
    plots[i].plot(indices, data[keys[i]])
    bottom, top = plots[i].get_ylim()
    plots[i].set_ylim(-5, maximum)
plt.savefig('test.png')

```



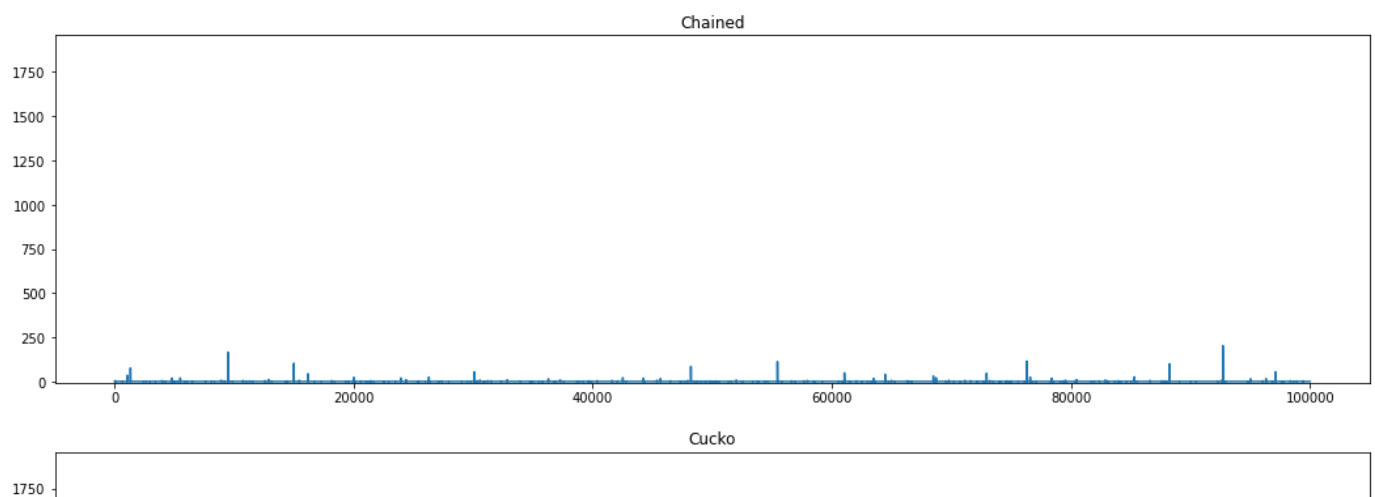


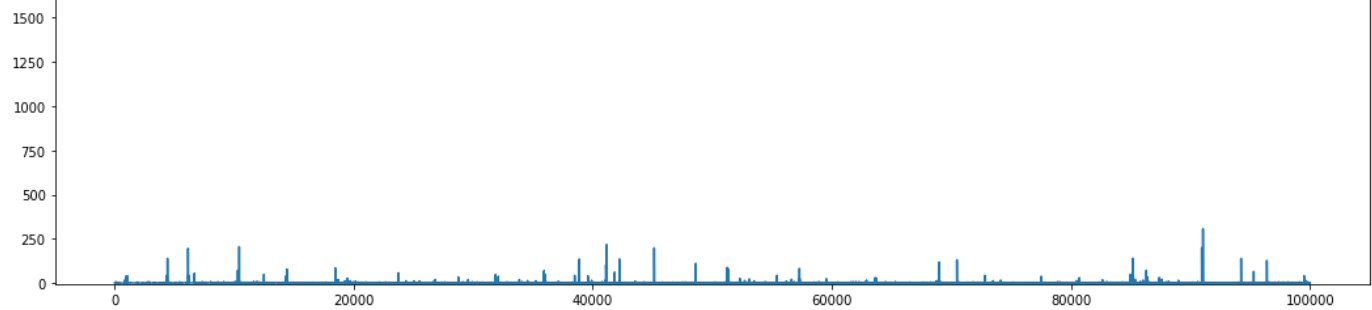
Поиск int-ов

In [240...

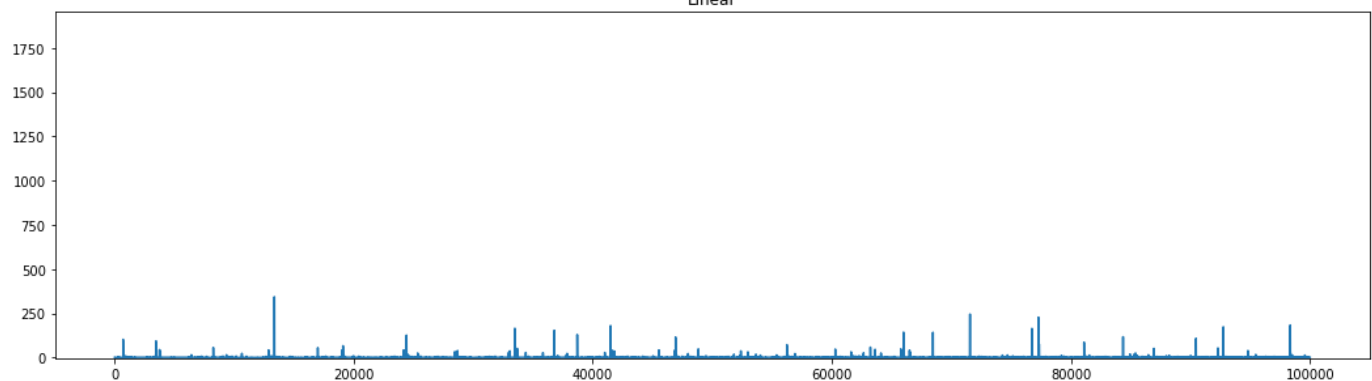
```
indices, data = get_x_ydict_from_csv('data/hash_find_int.csv')
keys = list(data.keys())
maximum = 0
for key in keys:
    if max(data[key]) > maximum and key not in []:
        maximum = max(data[key])

fig, plots = plt.subplots(7, 1, figsize=(18, 40))
fig.patch.set_facecolor('xkcd:white')
for i in range(7):
    plots[i].set_title(keys[i])
    plots[i].plot(indices, data[keys[i]])
    bottom, top = plots[i].get_ylim()
    plots[i].set_ylim(-5, maximum)
plt.savefig('test.png')
```

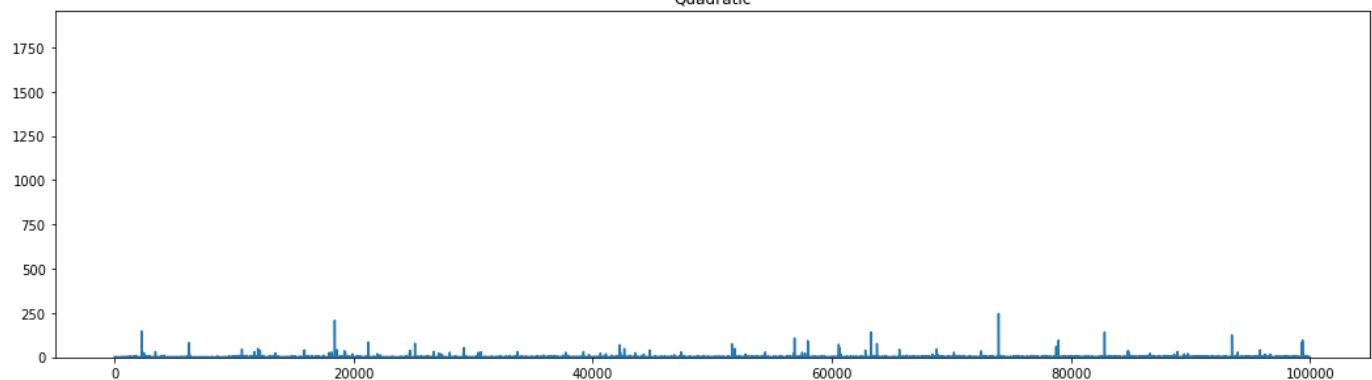




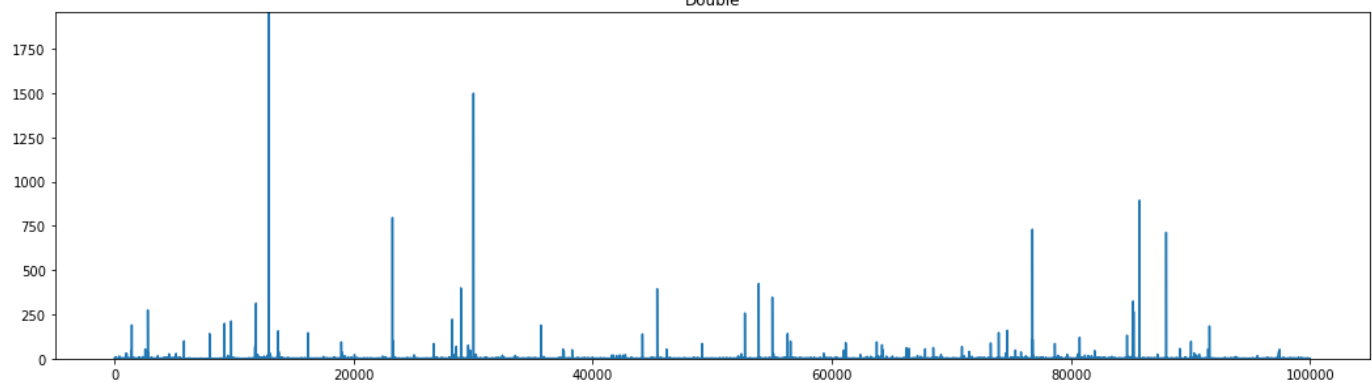
Linear



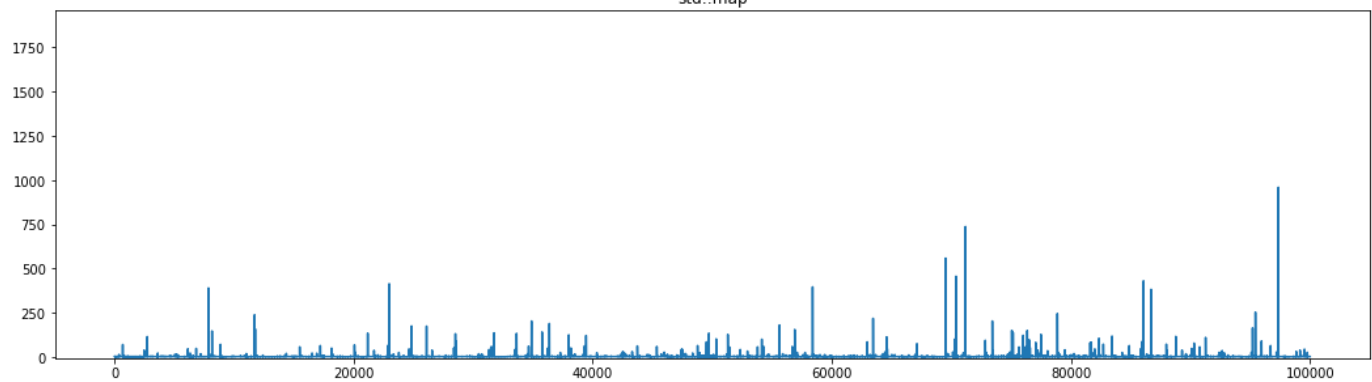
Quadratic



Double

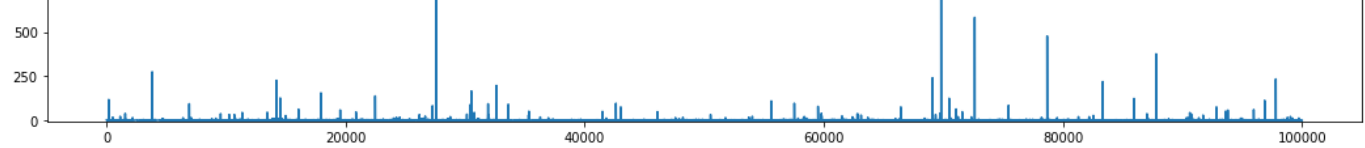


std::map



std::hash_map



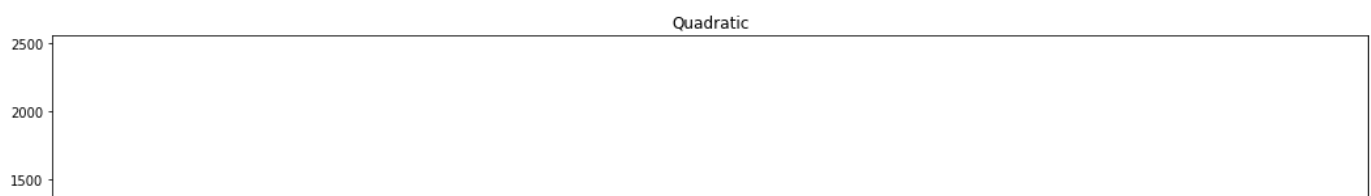
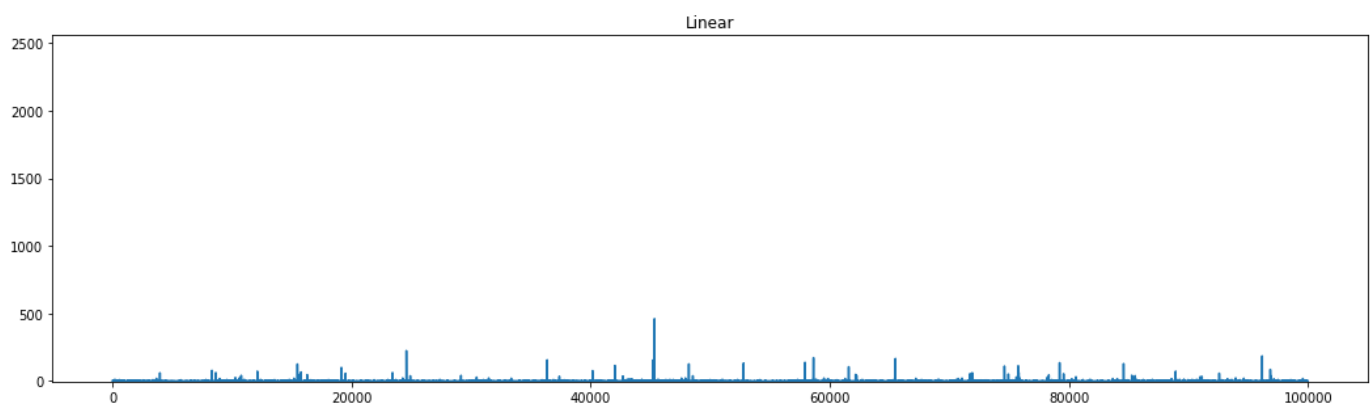
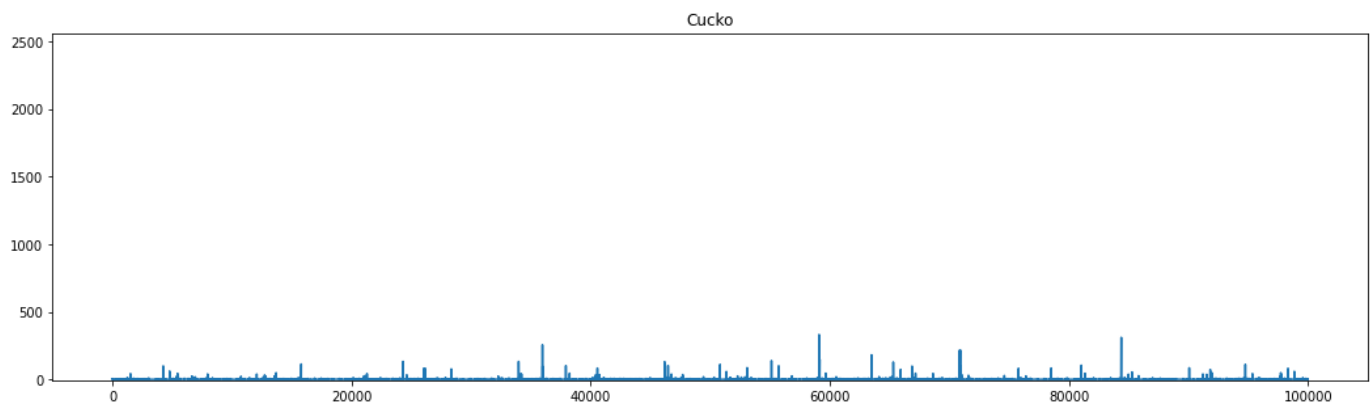
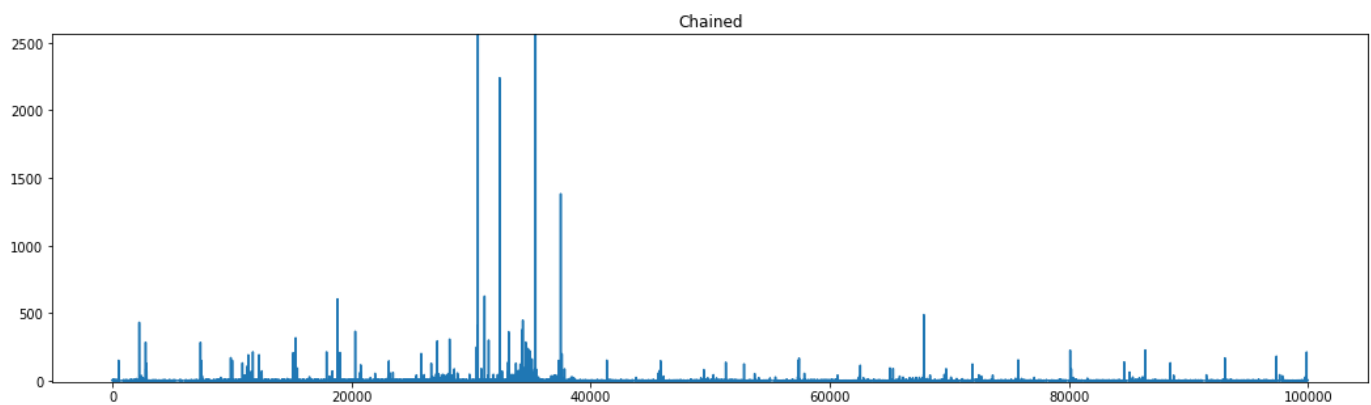


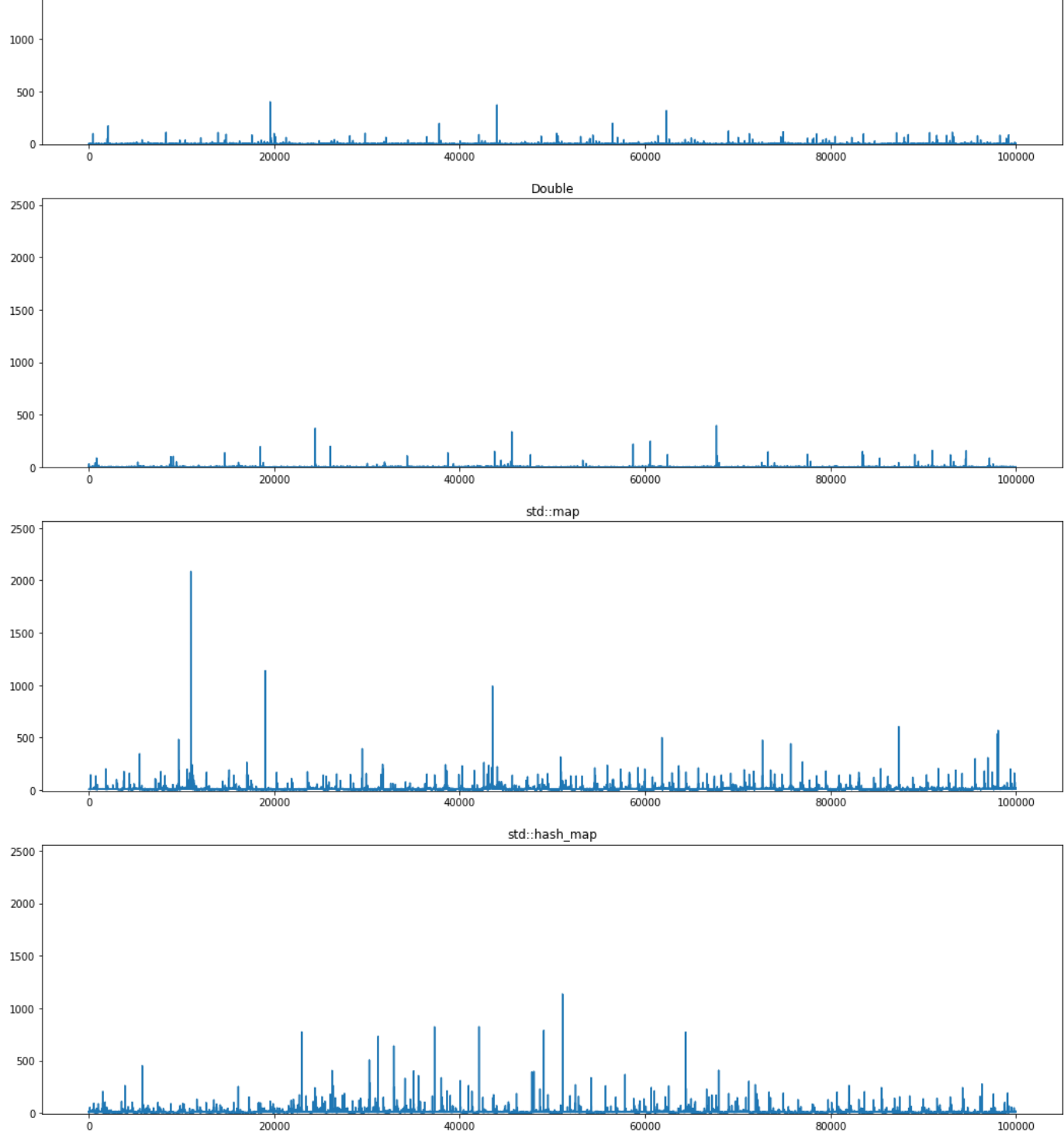
Удаление int-ов

In [244...]

```
indices, data = get_x_ydict_from_csv('data/hash_erase_int.csv')
keys = list(data.keys())
maximum = 0
for key in keys:
    if max(data[key]) > maximum and key not in []:
        maximum = max(data[key])

fig, plots = plt.subplots(7, 1, figsize=(18, 40))
fig.patch.set_facecolor('xkcd:white')
for i in range(7):
    plots[i].set_title(keys[i])
    plots[i].plot(indices, data[keys[i]])
    bottom, top = plots[i].get_ylim()
    plots[i].set_ylim(-5, maximum)
plt.savefig('test.png')
```





std::string-и

$$x = (x_0, x_1, \dots, x_{s-1})$$

$$h_a(x) = (\sum (a^i x_i) \bmod p) \bmod m$$

$$a \in \{0, 1, \dots, p-1\}$$

p - большое простое число, m - размер таблицы

вставка std::string-ов

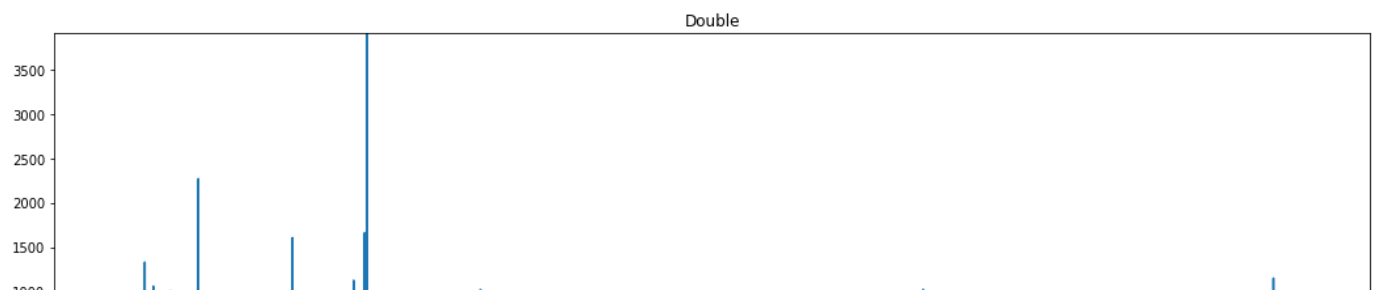
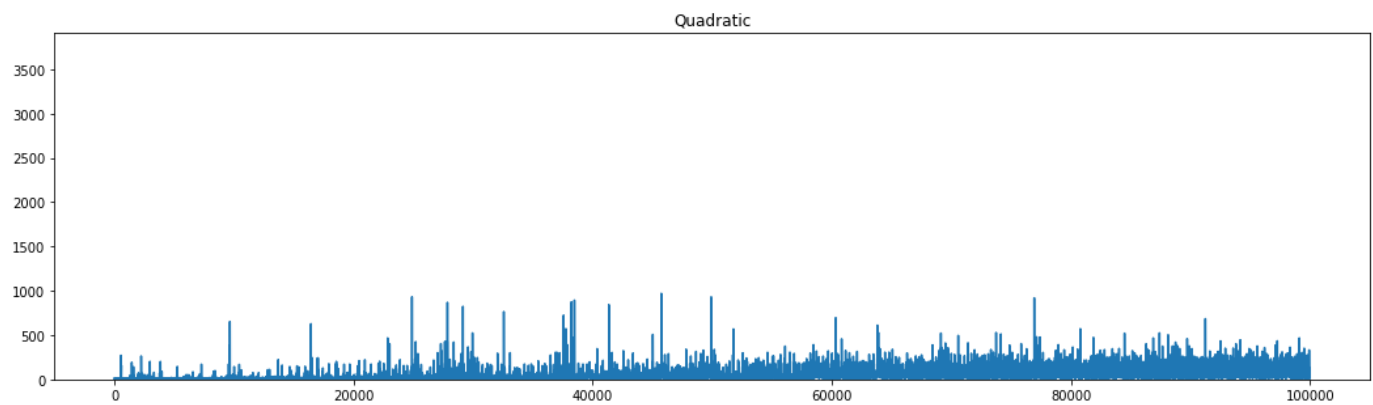
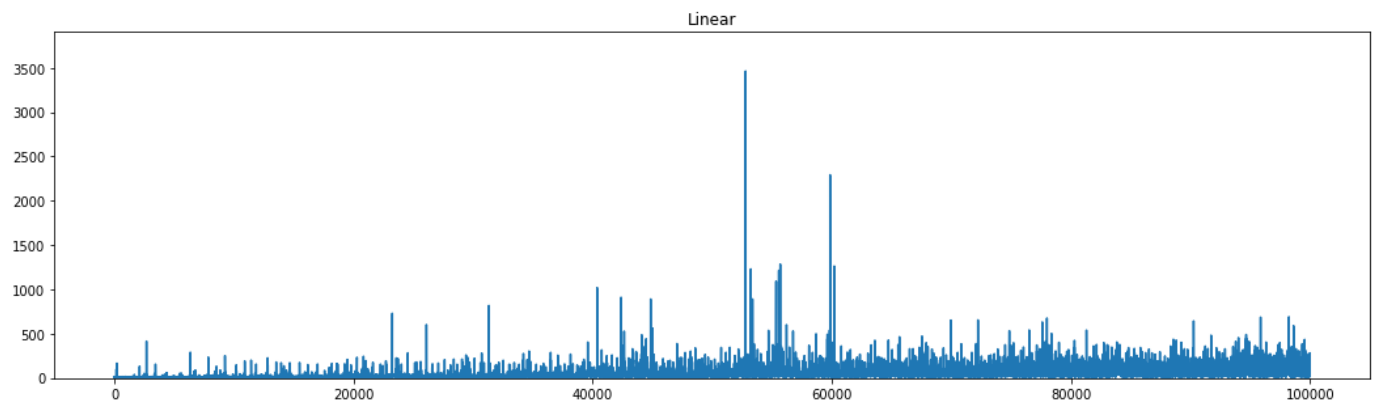
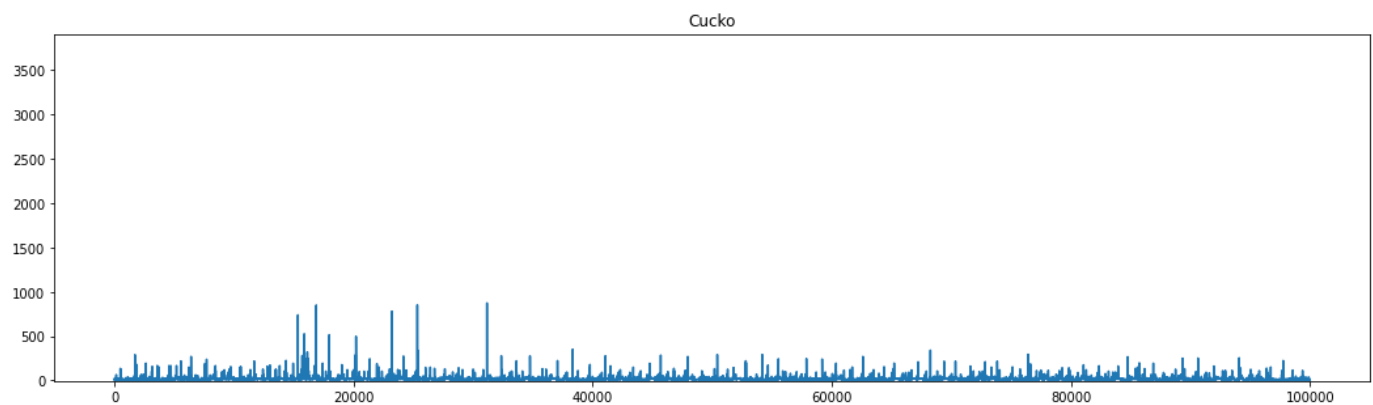
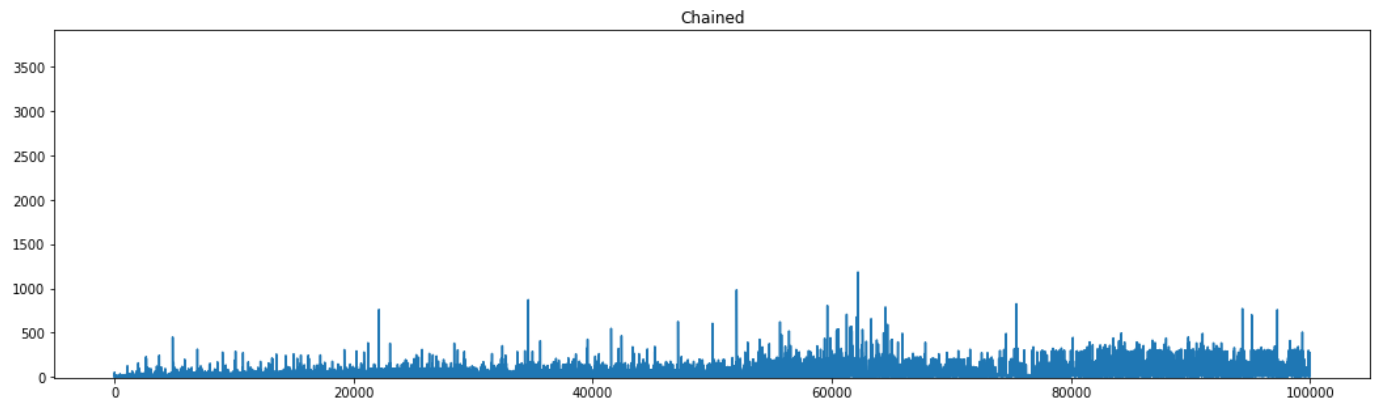
In [245..

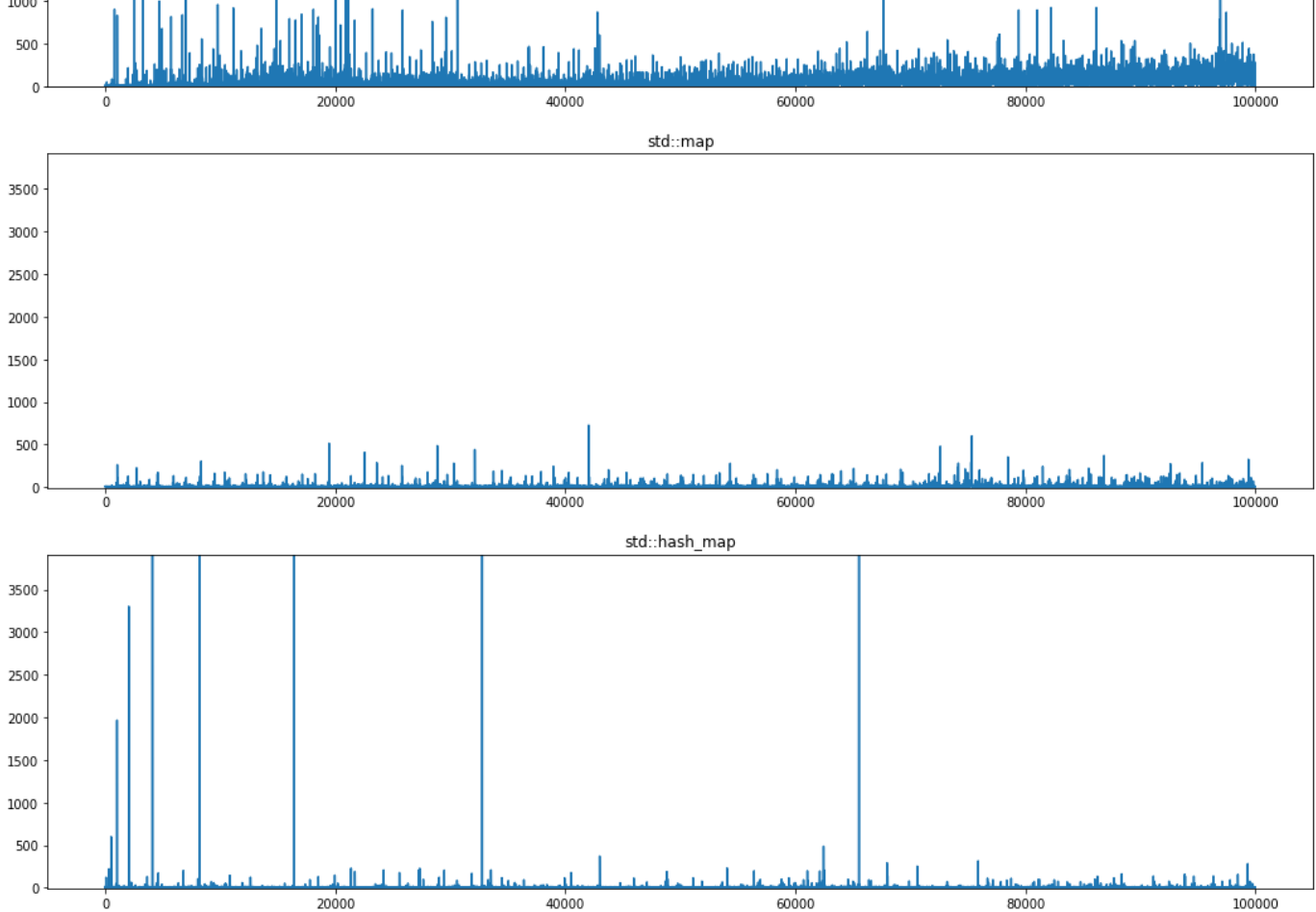
```
indices, data = get_x_ydict_from_csv('data/hash_insert_string.csv')
keys = list(data.keys())
maximum = 0
for key in keys:
    if max(data[key]) > maximum and key not in ['std::hash_map']:
        maximum = max(data[key])
```

```

fig, plots = plt.subplots(7, 1, figsize=(18, 40))
fig.patch.set_facecolor('xkcd:white')
for i in range(7):
    plots[i].set_title(keys[i])
    plots[i].plot(indices, data[keys[i]])
    bottom, top = plots[i].get_ylim()
    plots[i].set_ylim(-5, maximum)
plt.savefig('test.png')

```



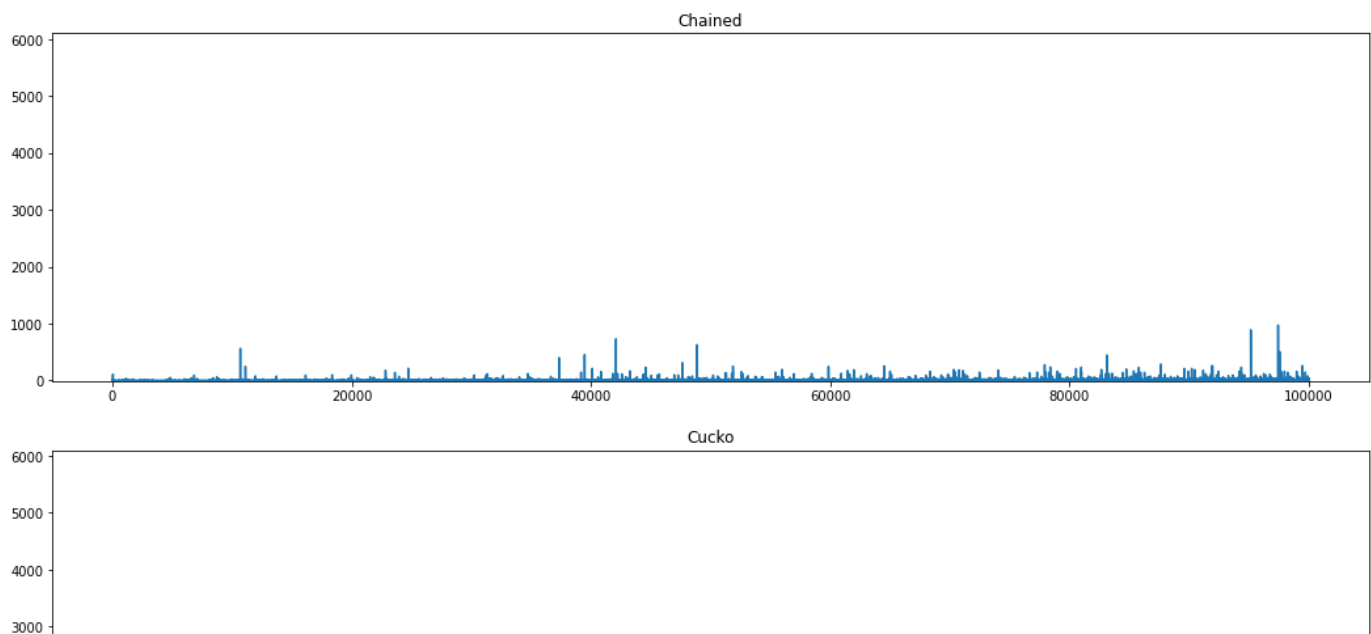


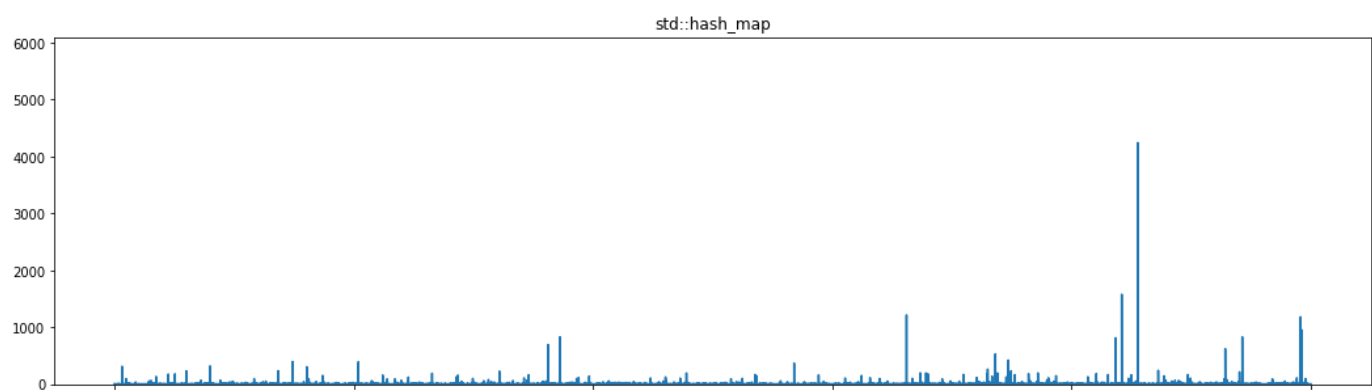
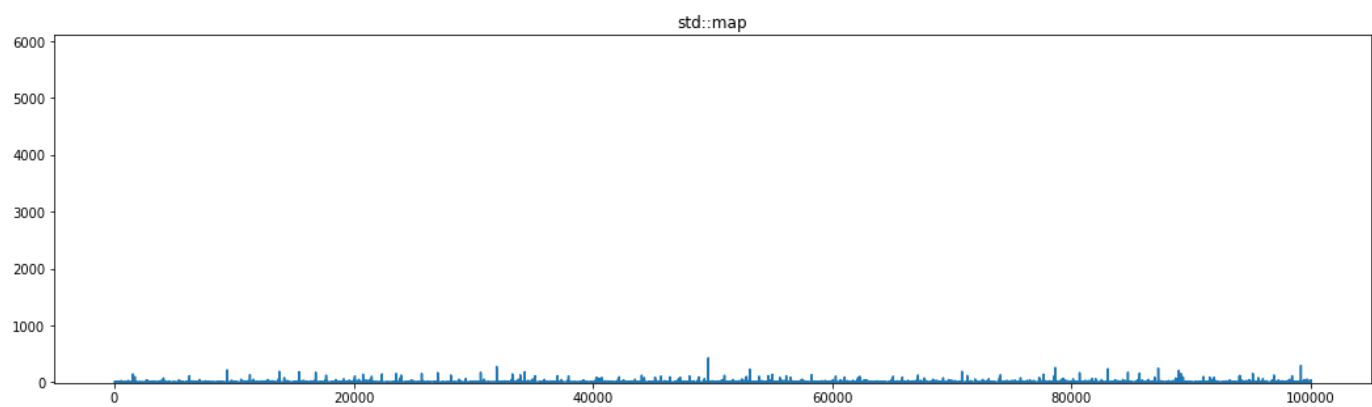
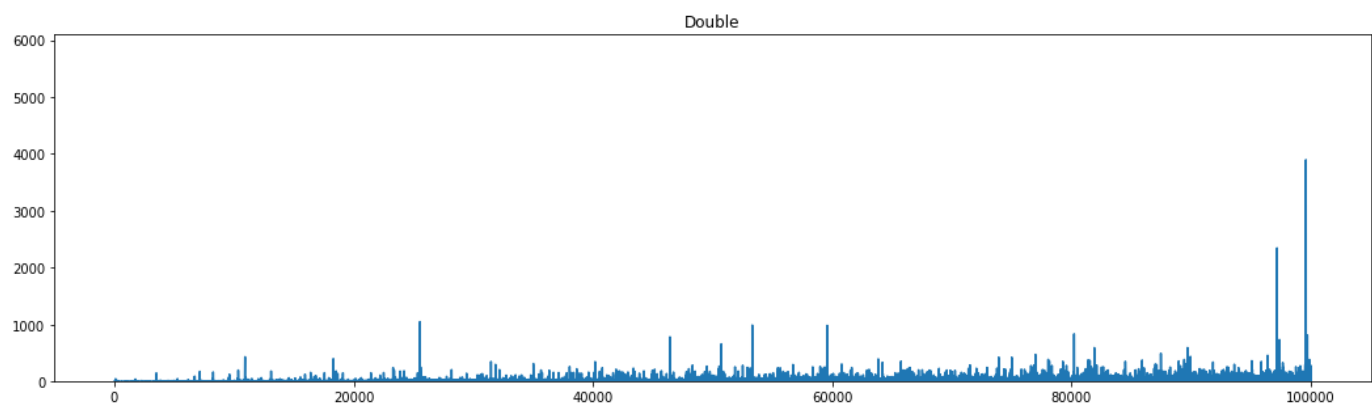
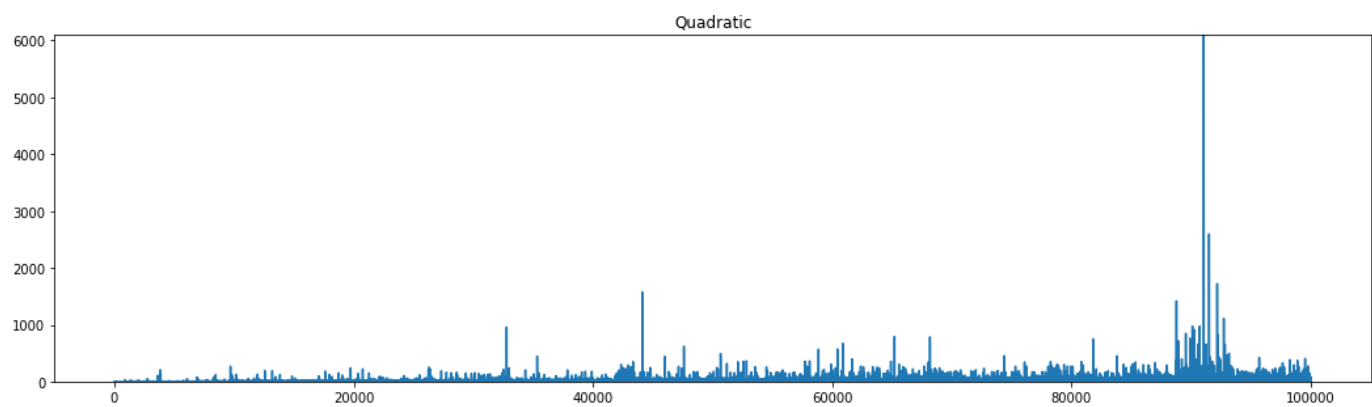
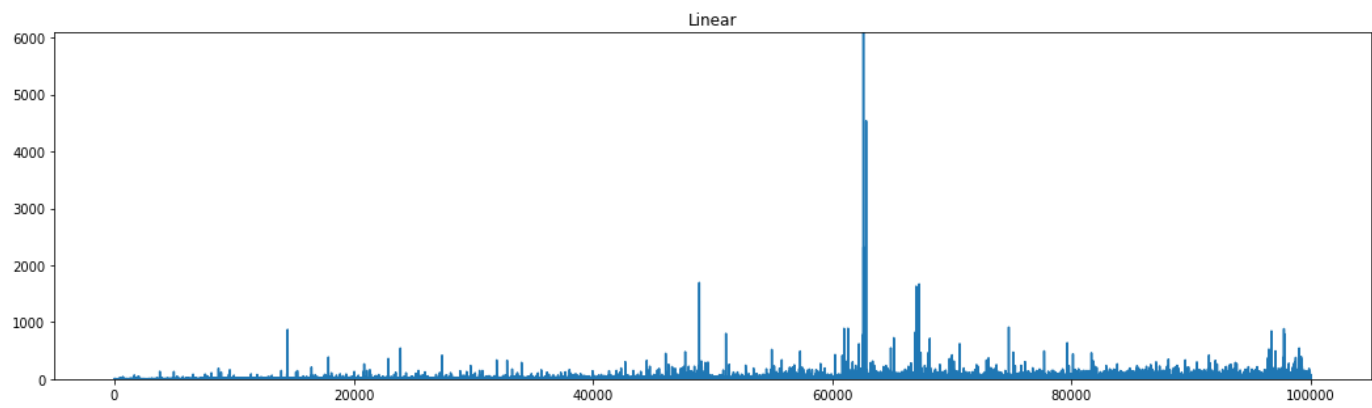
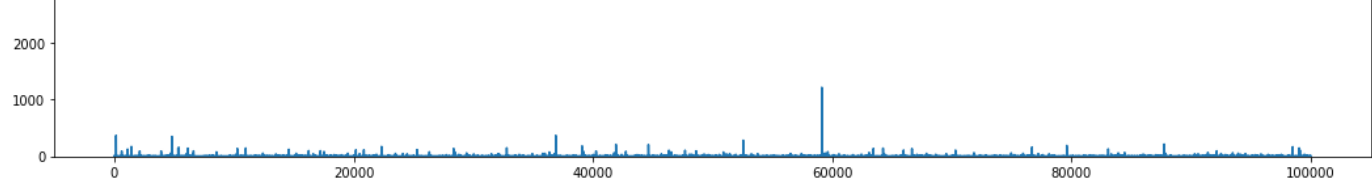
поиск std::string-ов

In [251...

```
indices, data = get_x_ydict_from_csv('data/hash_find_string.csv')
keys = list(data.keys())
maximum = 0
for key in keys:
    if max(data[key]) > maximum and key not in ['Linear']:
        maximum = max(data[key])

fig, plots = plt.subplots(7, 1, figsize=(18, 40))
fig.patch.set_facecolor('xkcd:white')
for i in range(7):
    plots[i].set_title(keys[i])
    plots[i].plot(indices, data[keys[i]])
    bottom, top = plots[i].get_ylim()
    plots[i].set_ylim(-5, maximum)
plt.savefig('test.png')
```



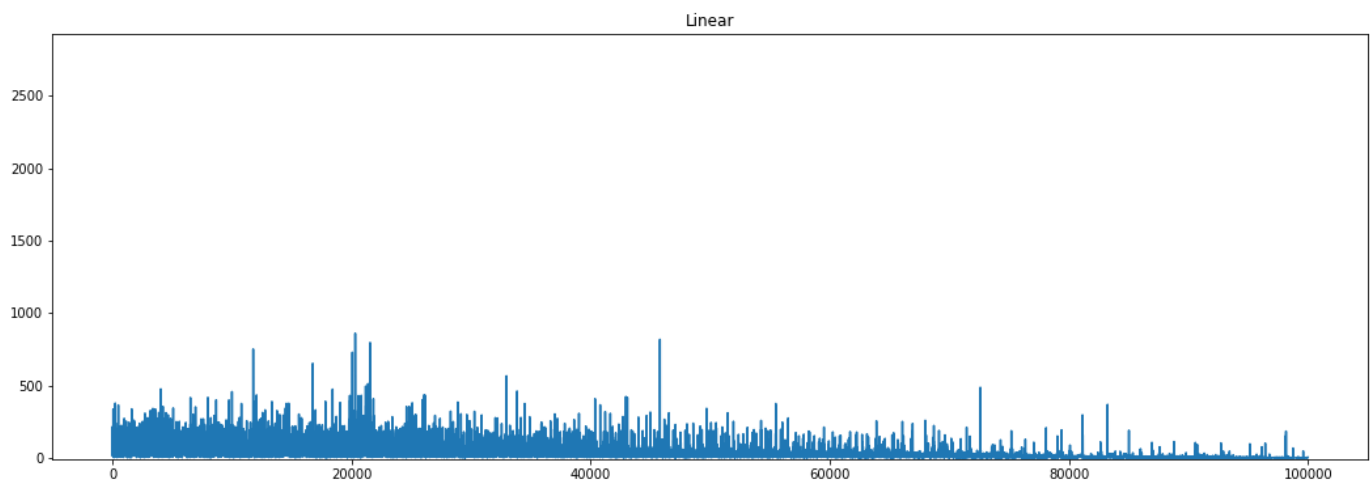
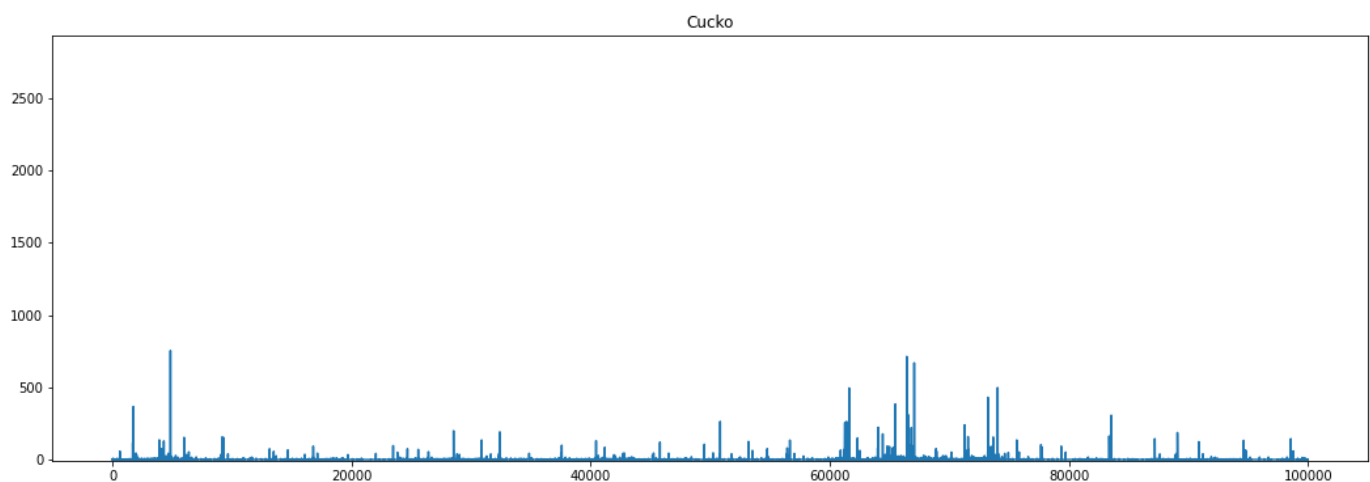
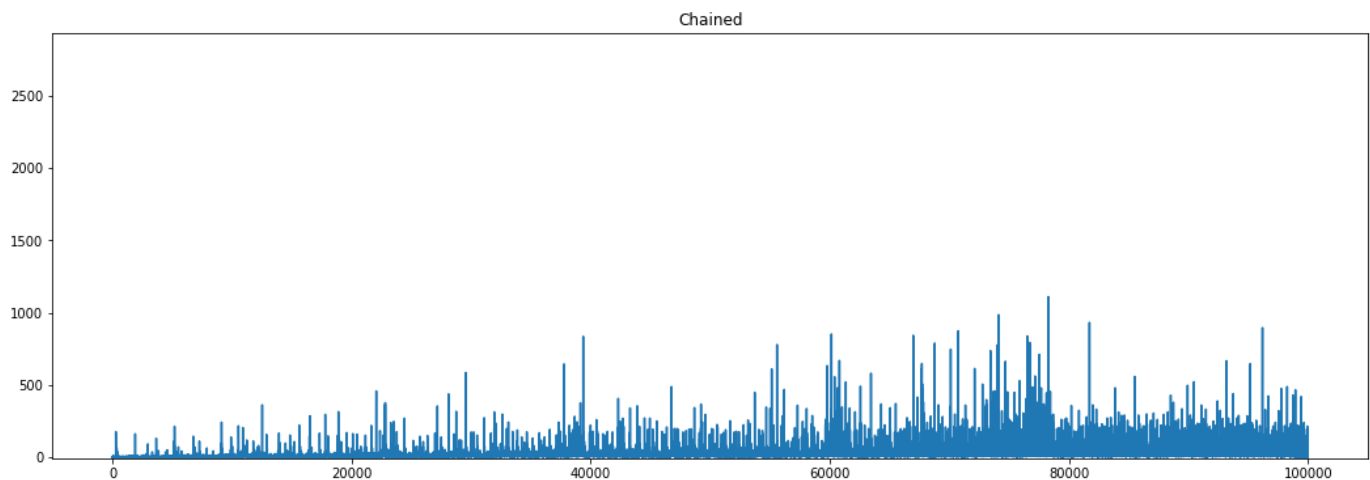


Удаление std::string-ов

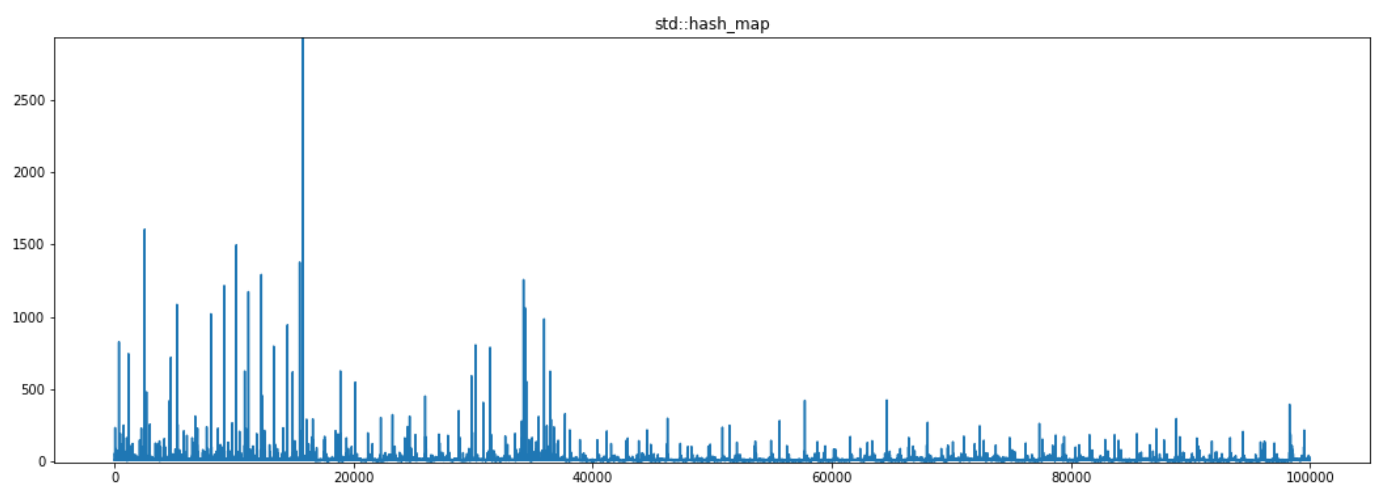
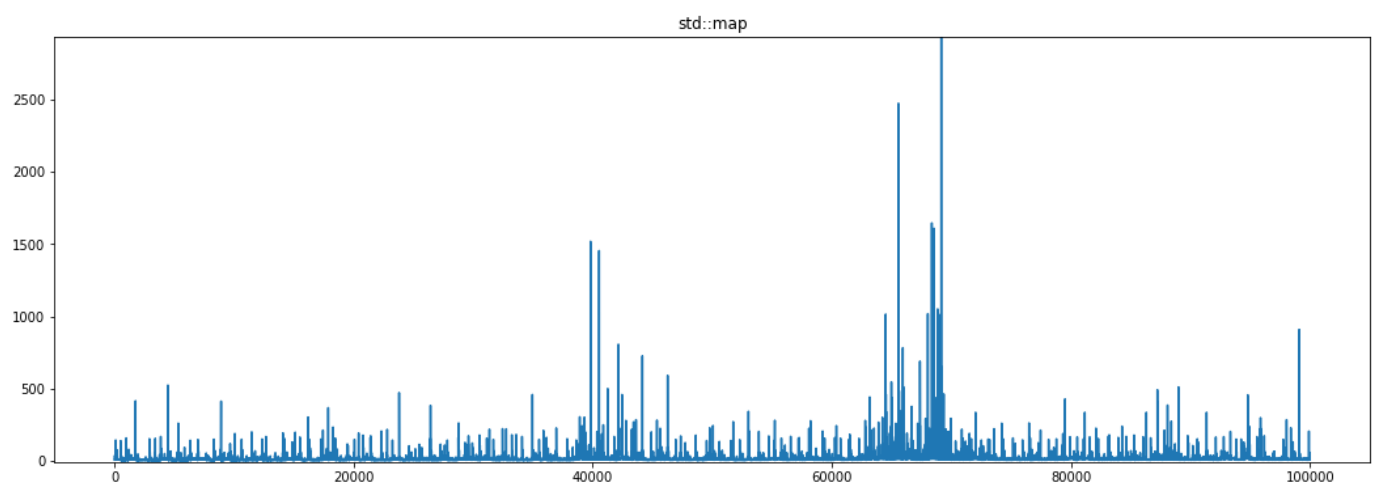
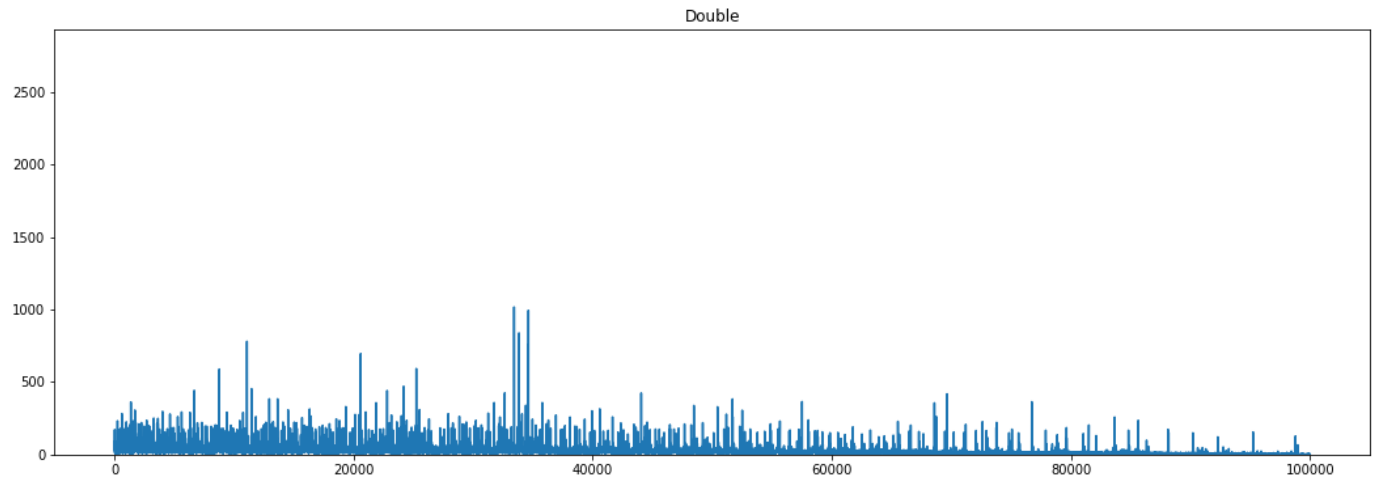
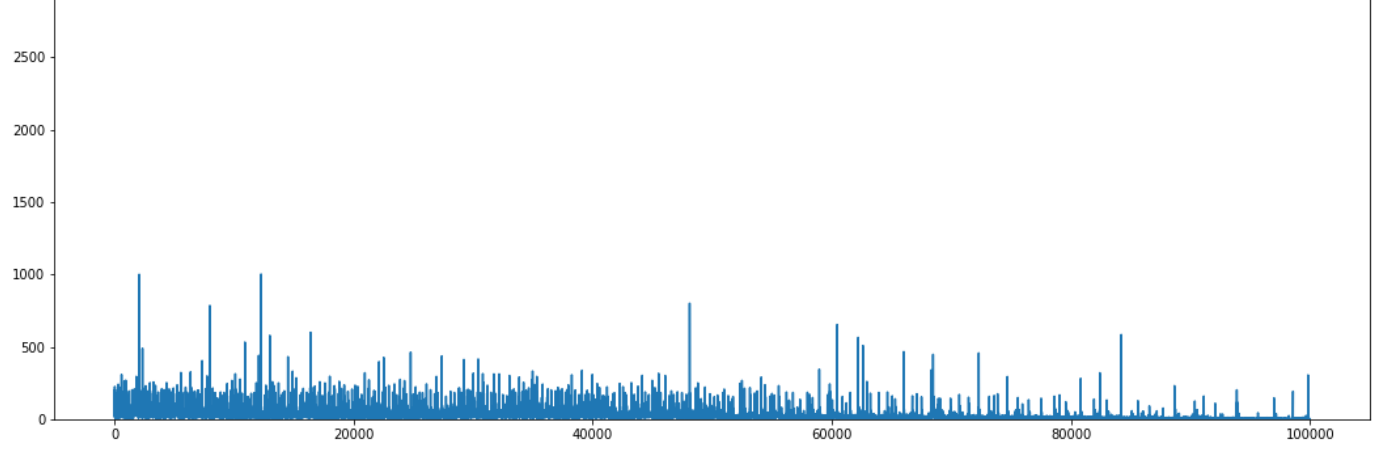
In [253...

```
indices, data = get_x_ydict_from_csv('data/hash_erase_string.csv')
keys = list(data.keys())
maximum = 0
for key in keys:
    if max(data[key]) > maximum and key not in ['std::map']:
        maximum = max(data[key])

fig, plots = plt.subplots(len(keys), 1, figsize=(18, 7*len(keys)))
fig.patch.set_facecolor('xkcd:white')
for i in range(len(keys)):
    plots[i].set_title(keys[i])
    plots[i].plot(indices, data[keys[i]])
    bottom, top = plots[i].get_ylim()
    plots[i].set_ylim(-5, maximum)
plt.savefig('test.png')
```



Quadratic



Real-life std::string-и

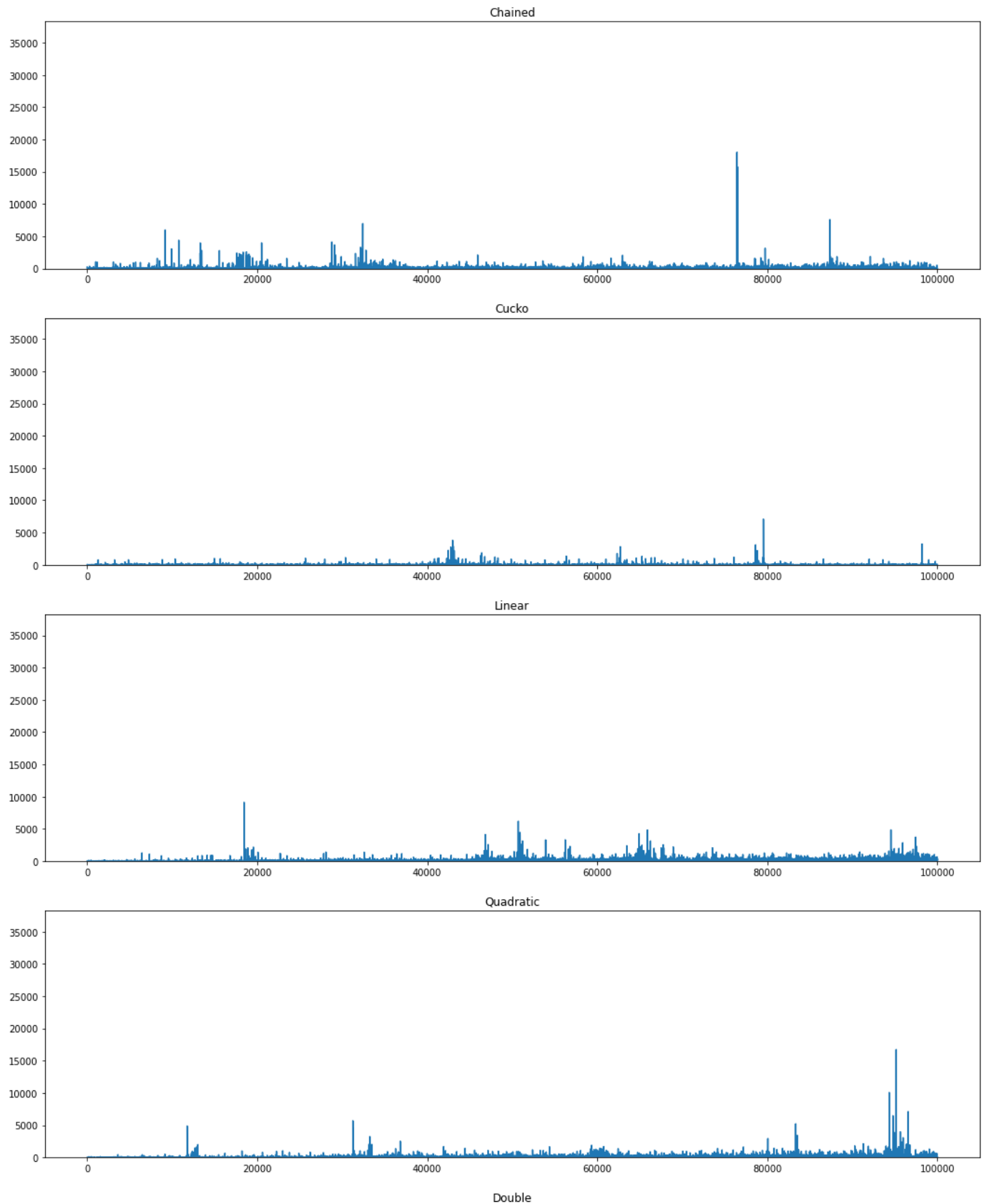
Просто словарь отсортированных в алфавитном порядке слов

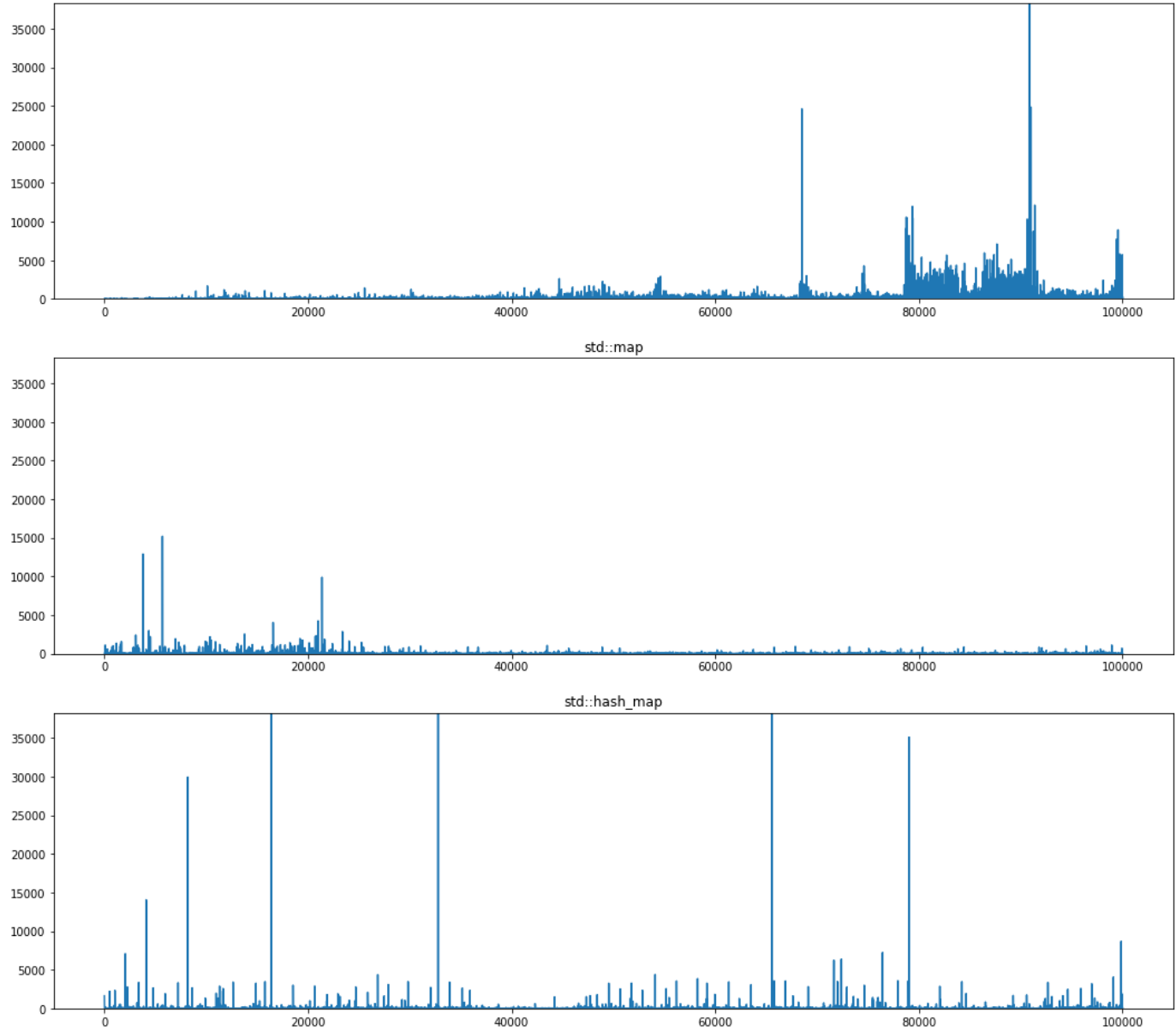
Вставка

In [257...

```
indices, data = get_x_ydict_from_csv('data/hash_insert_real_data.csv')
keys = list(data.keys())
maximum = 0
for key in keys:
    if max(data[key]) > maximum and key not in ['std::hash_map']:
        maximum = max(data[key])

fig, plots = plt.subplots(7, 1, figsize=(18, 40))
fig.patch.set_facecolor('xkcd:white')
for i in range(7):
    plots[i].set_title(keys[i])
    plots[i].plot(indices, data[keys[i]])
    bottom, top = plots[i].get_ylim()
    plots[i].set_ylim(-5, maximum)
plt.savefig('test.png')
```



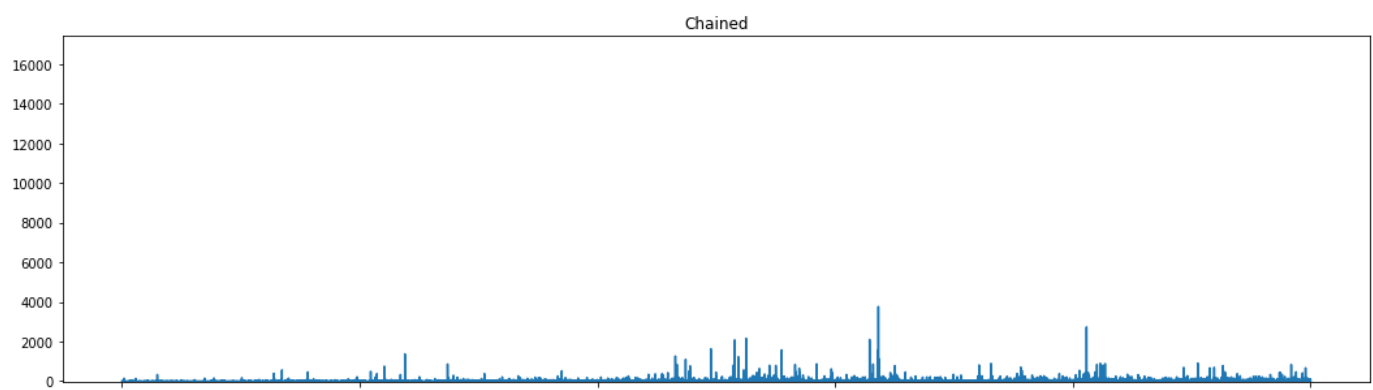


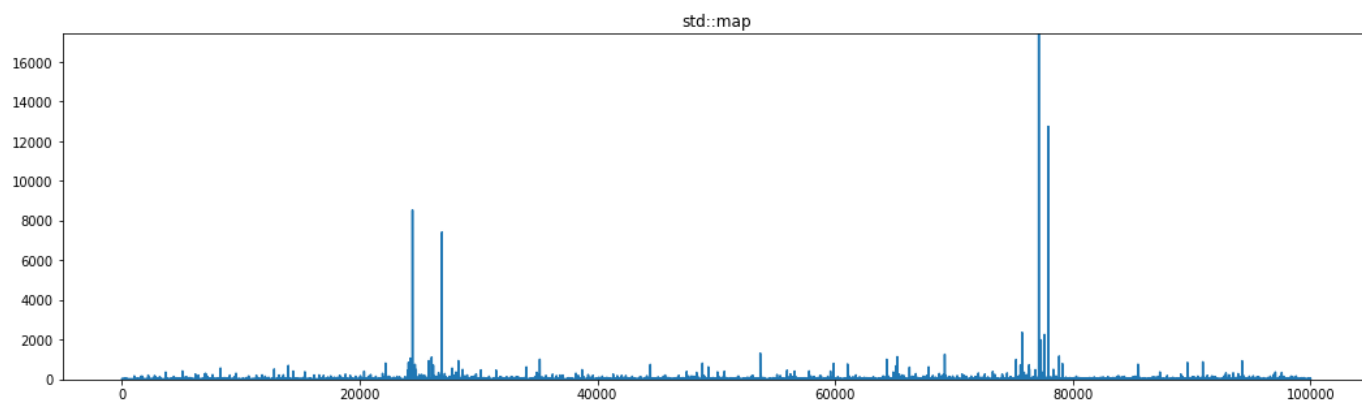
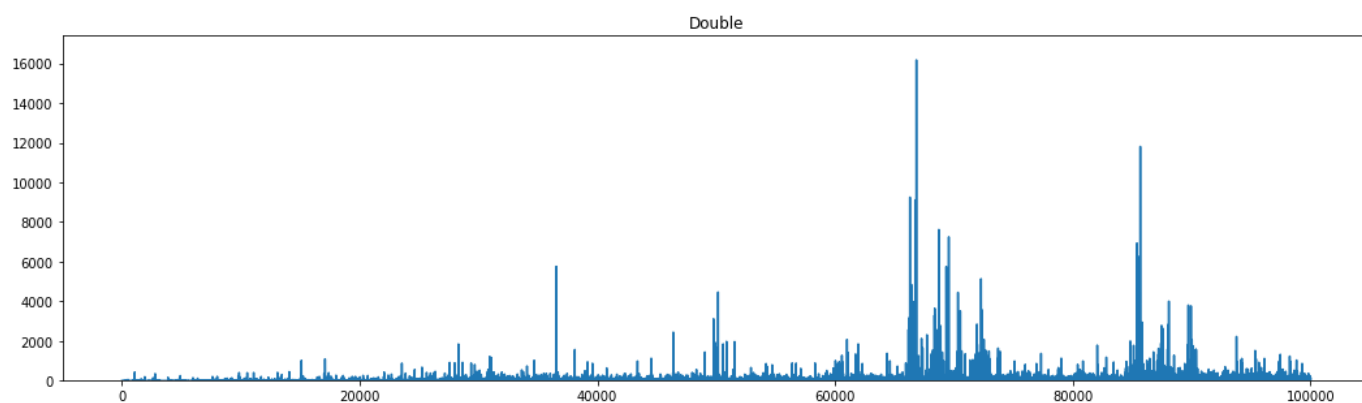
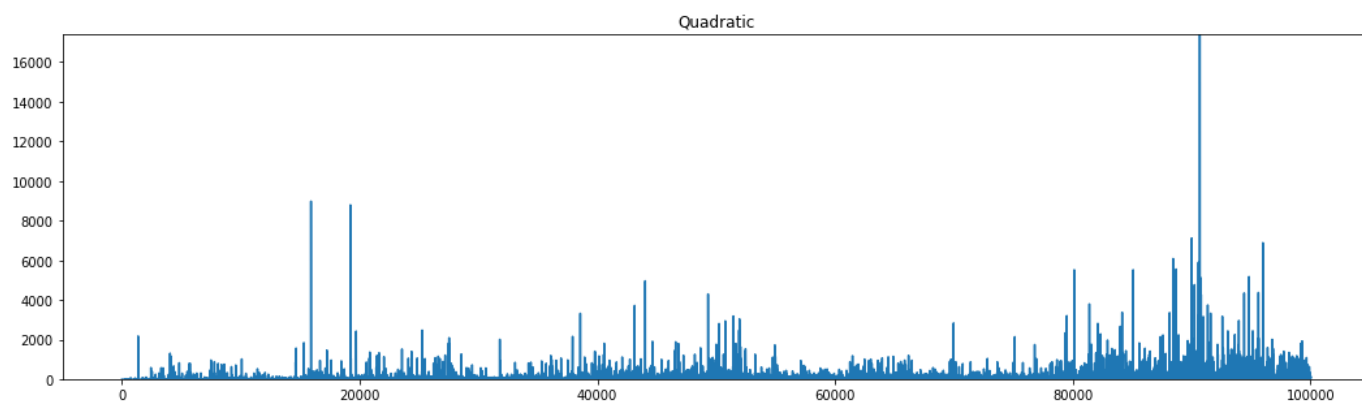
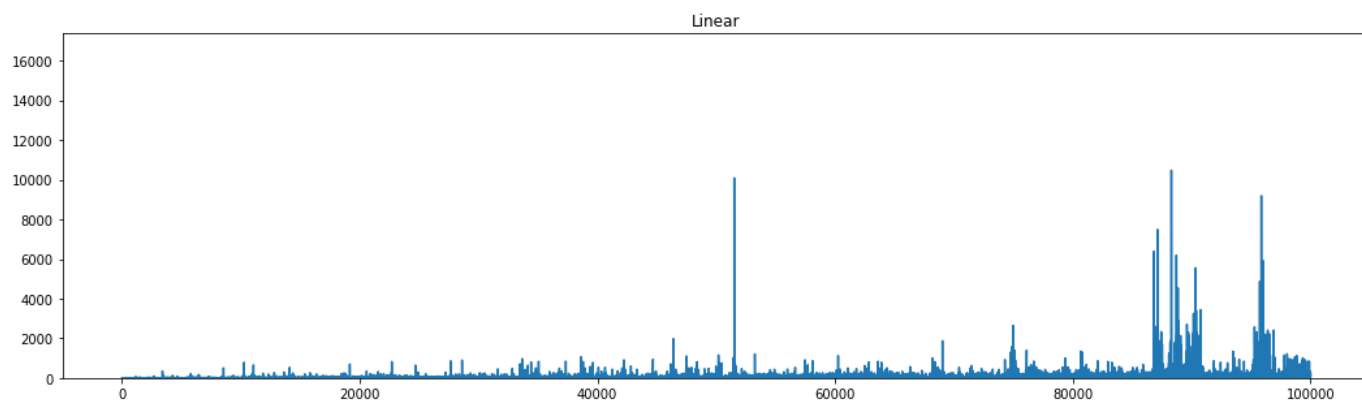
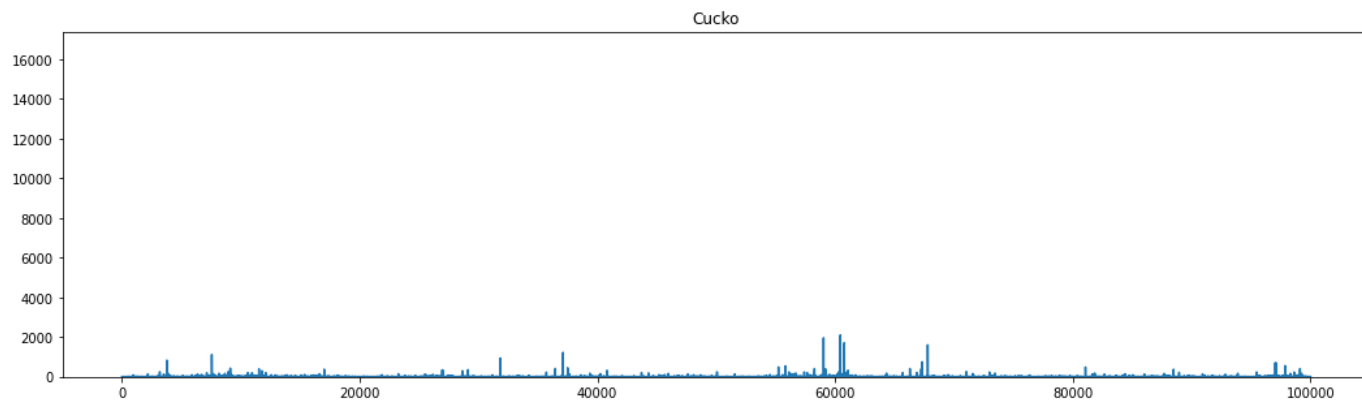
Поиск

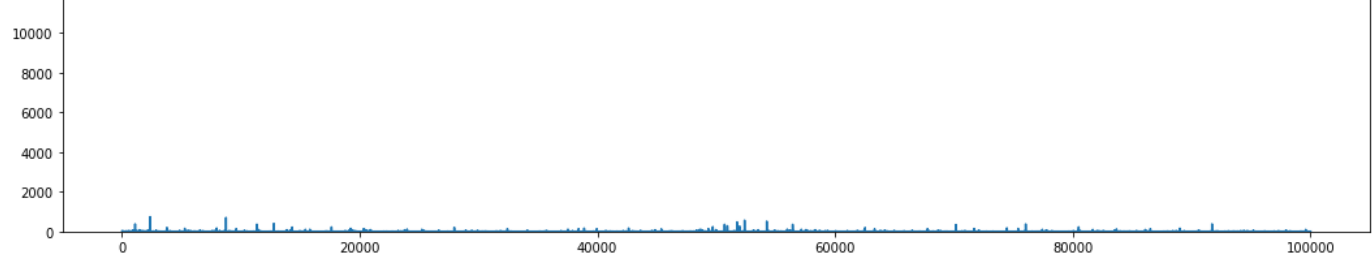
In [259...

```
indices, data = get_x_ydict_from_csv('data/hash_find_real_data.csv')
keys = list(data.keys())
maximum = 0
for key in keys:
    if max(data[key]) > maximum and key not in ['std::map']:
        maximum = max(data[key])

fig, plots = plt.subplots(7, 1, figsize=(18, 40))
fig.patch.set_facecolor('xkcd:white')
for i in range(7):
    plots[i].set_title(keys[i])
    plots[i].plot(indices, data[keys[i]])
    bottom, top = plots[i].get_ylim()
    plots[i].set_ylim(-5, maximum)
plt.savefig('test.png')
```





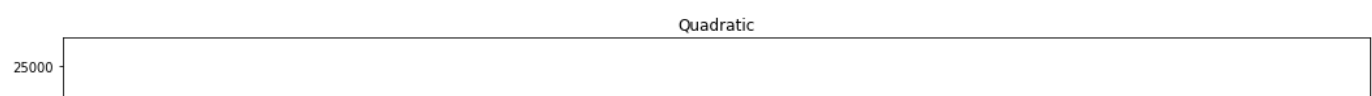
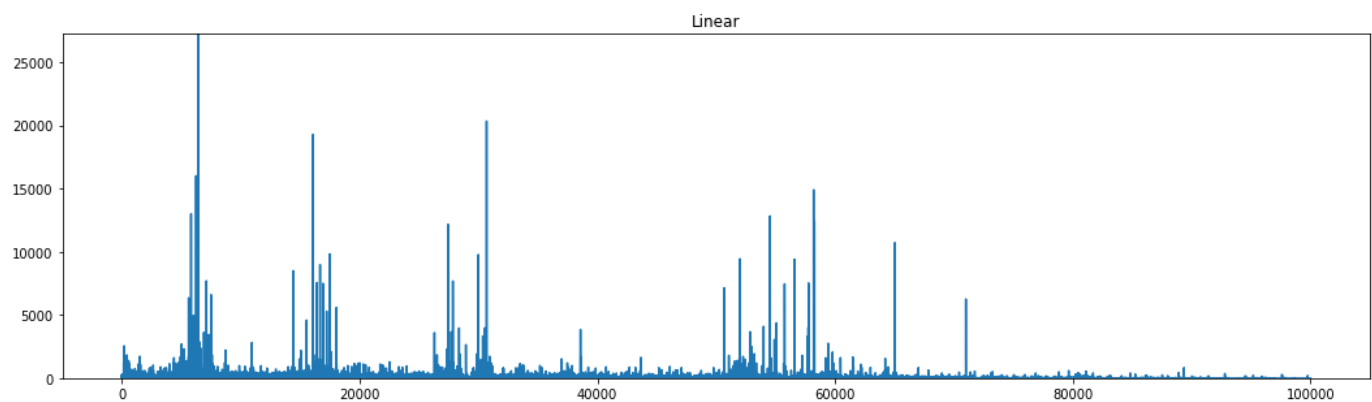
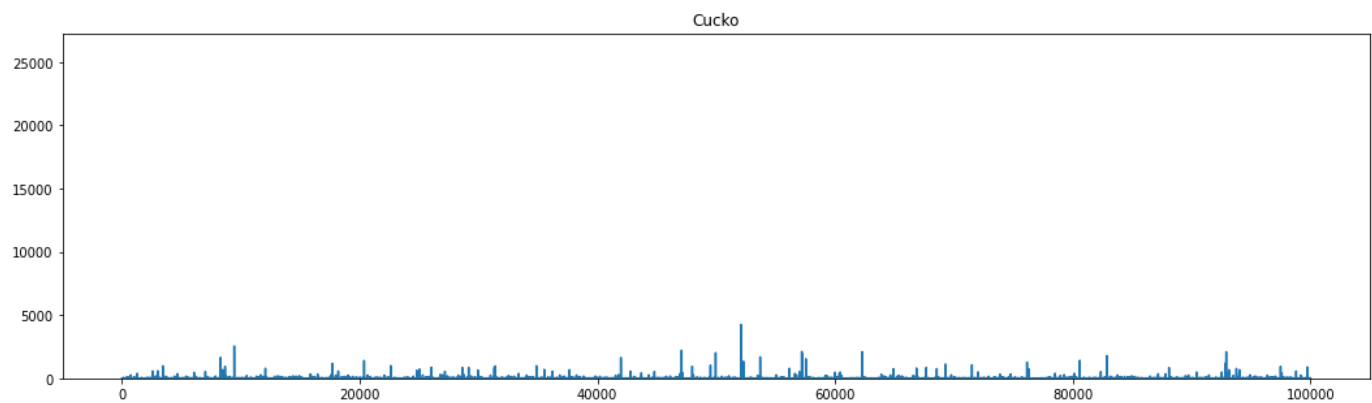
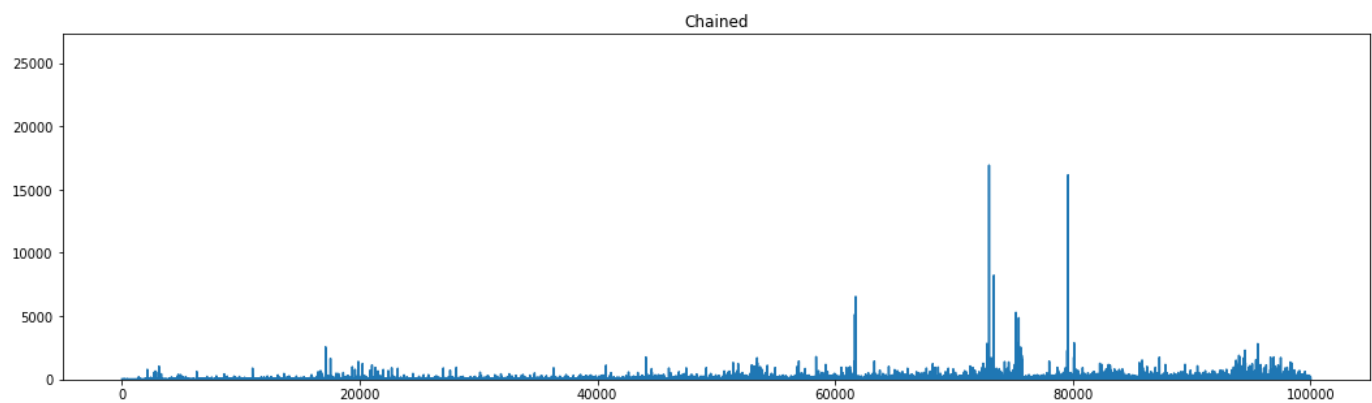


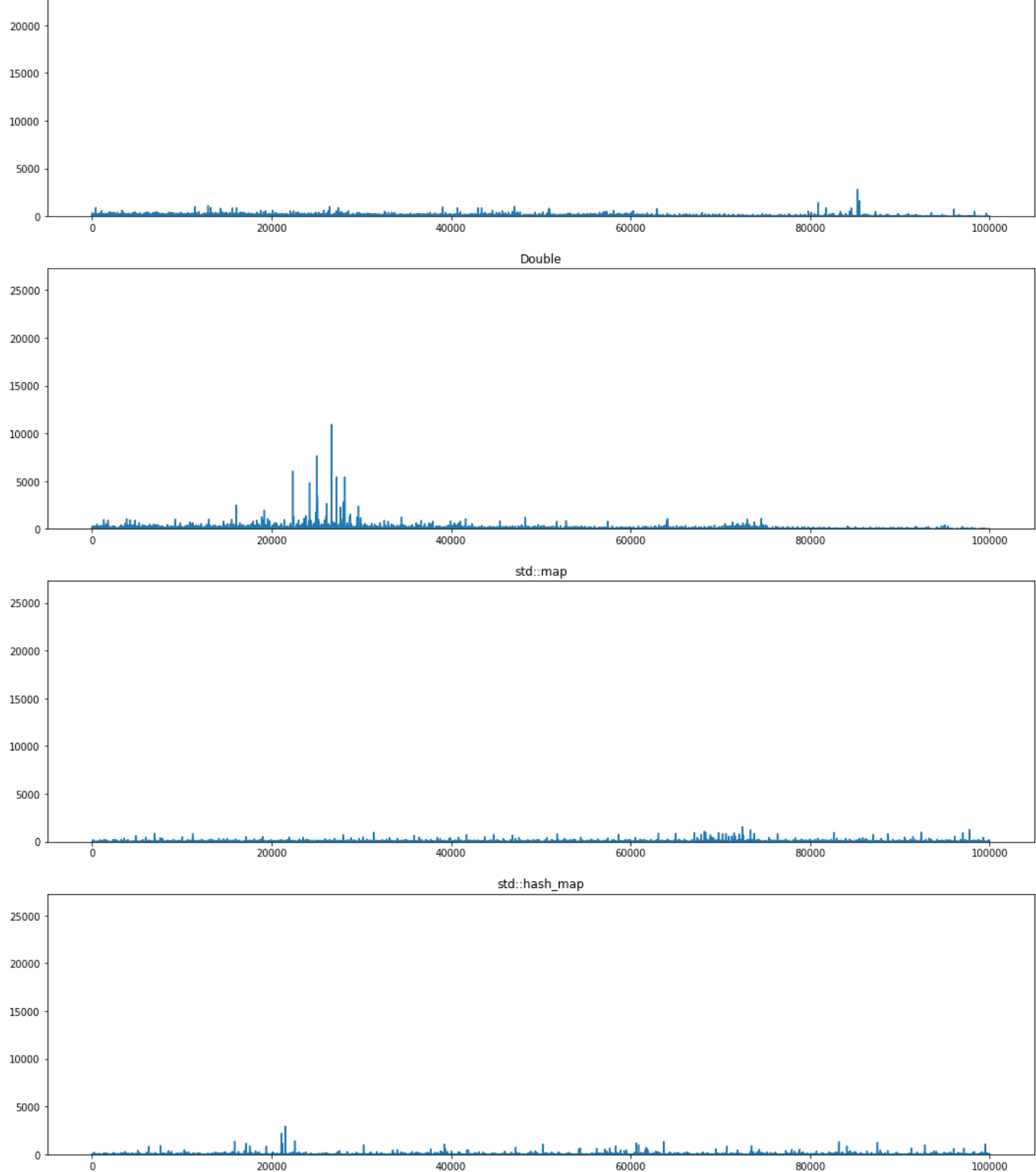
Удаление

In [260...

```
indices, data = get_x_ydict_from_csv('data/hash_erase_real_data.csv')
keys = list(data.keys())
maximum = 0
for key in keys:
    if max(data[key]) > maximum and key not in []:
        maximum = max(data[key])

fig, plots = plt.subplots(7, 1, figsize=(18, 40))
fig.patch.set_facecolor('xkcd:white')
for i in range(7):
    plots[i].set_title(keys[i])
    plots[i].plot(indices, data[keys[i]])
    bottom, top = plots[i].get_ylim()
    plots[i].set_ylim(-5, maximum)
plt.savefig('test.png')
```



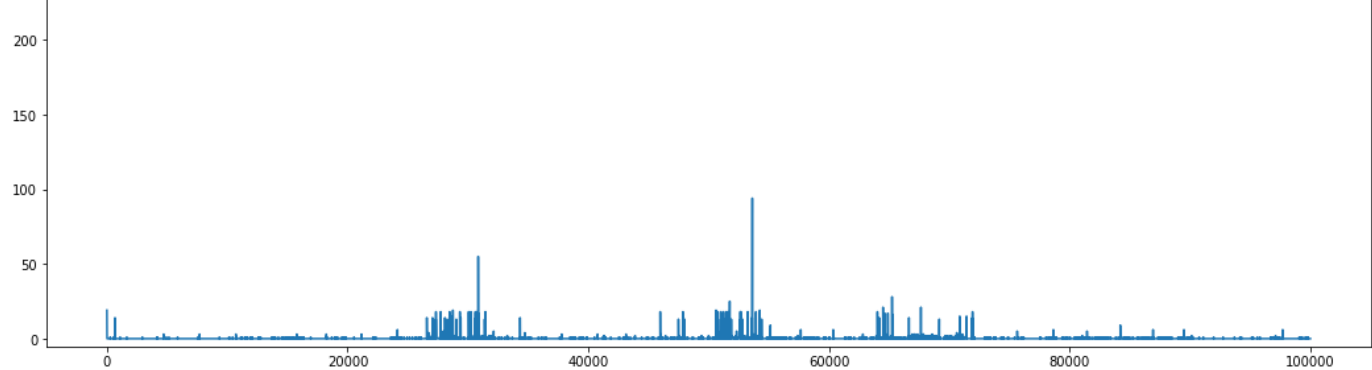


Вставка в кукушку различной степени

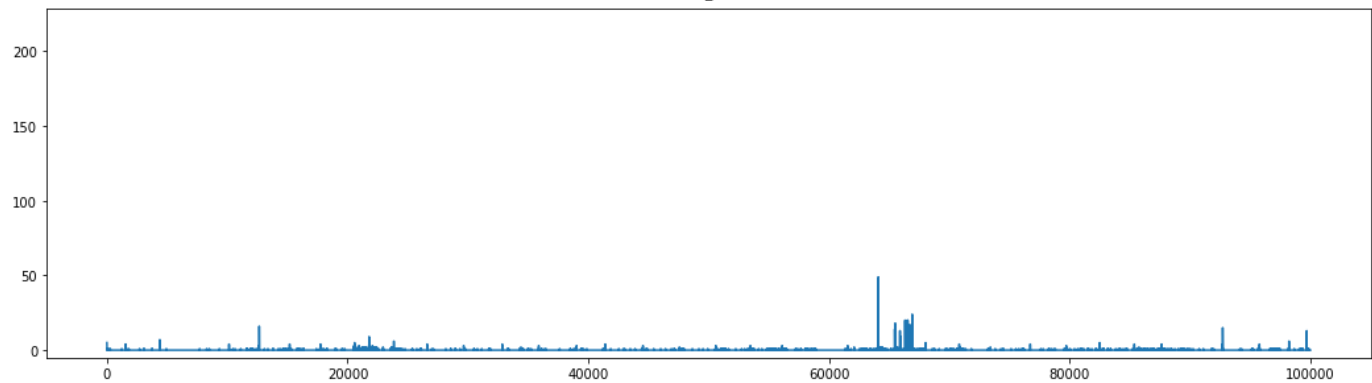
In [261...

```
indices, data = get_x_ydict_from_csv('data/hash_insert_real_data_cuckoo_degrees.csv')
keys = list(data.keys())
maximum = 0
for key in keys:
    if max(data[key]) > maximum and key not in []:
        maximum = max(data[key])

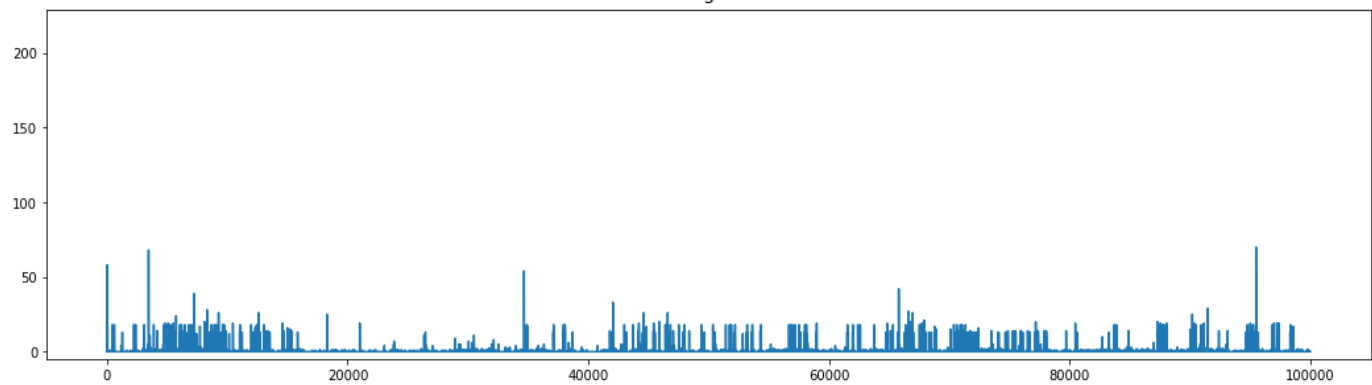
fig, plots = plt.subplots(7, 1, figsize=(18, 40))
fig.patch.set_facecolor('xkcd:white')
for i in range(len(keys)):
    plots[i].set_title(keys[i])
    plots[i].plot(indices, data[keys[i]])
    bottom, top = plots[i].get_ylim()
    plots[i].set_ylim(-5, maximum)
plt.savefig('test.png')
```

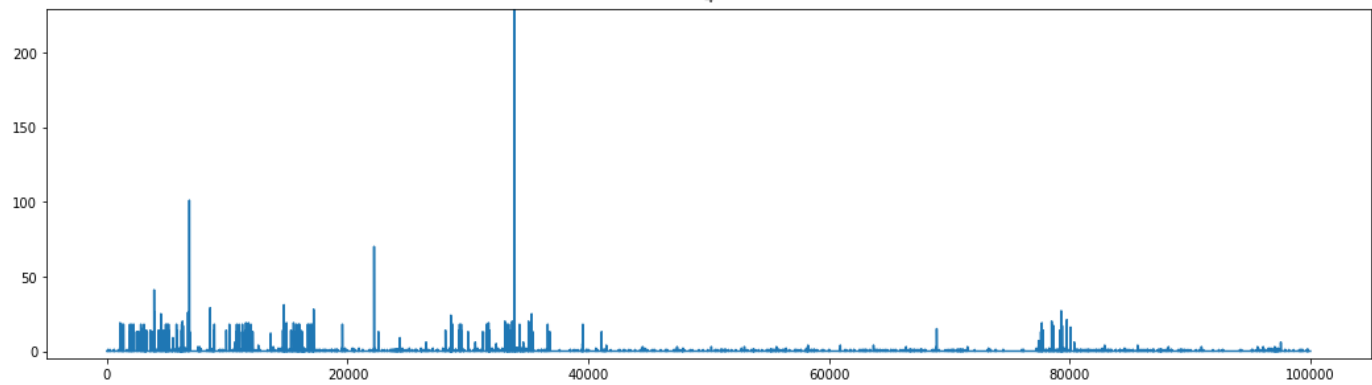
2



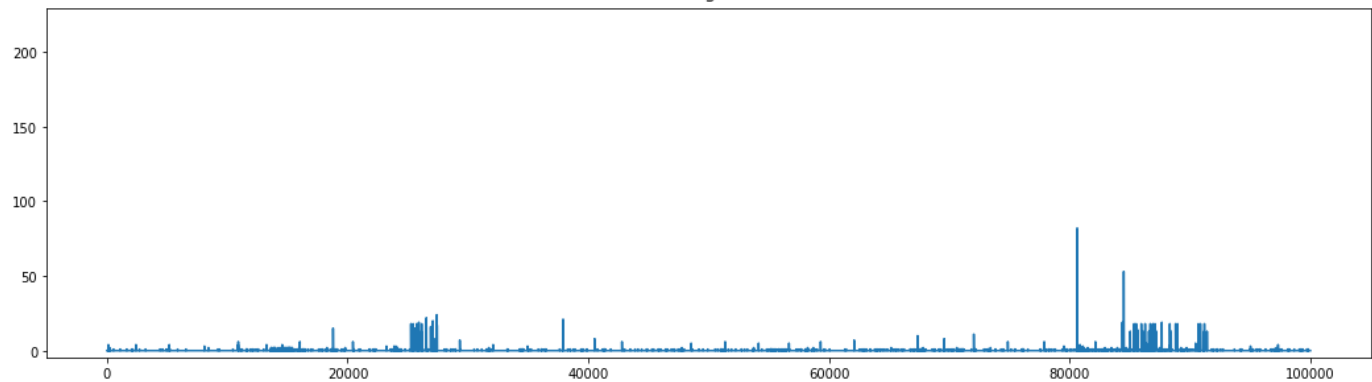
3



4

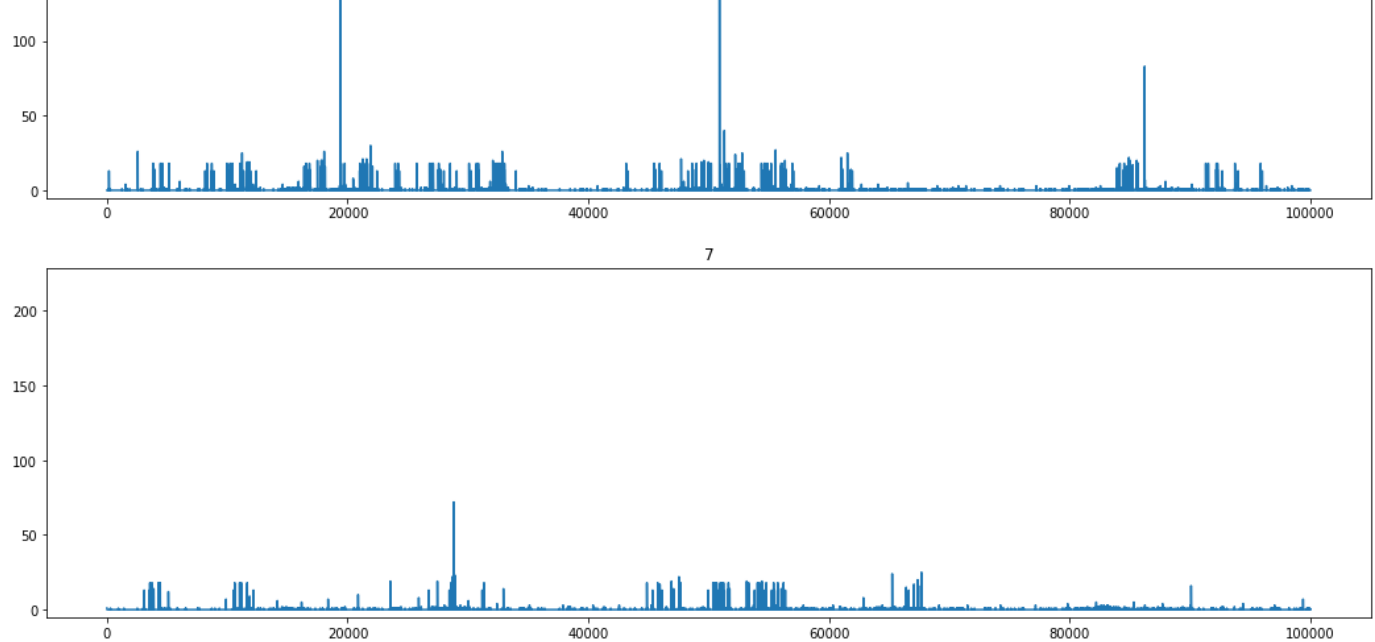


5



6



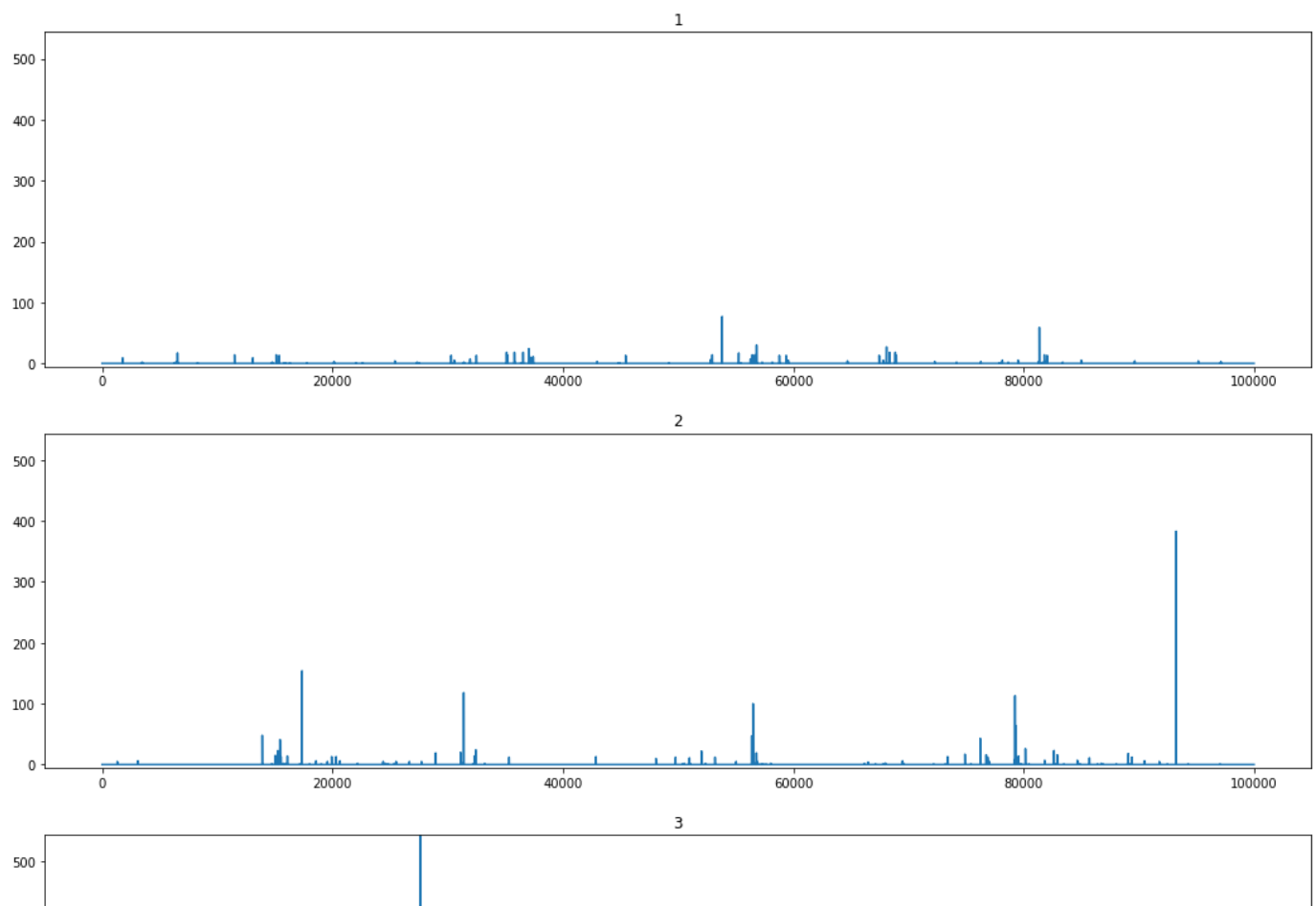


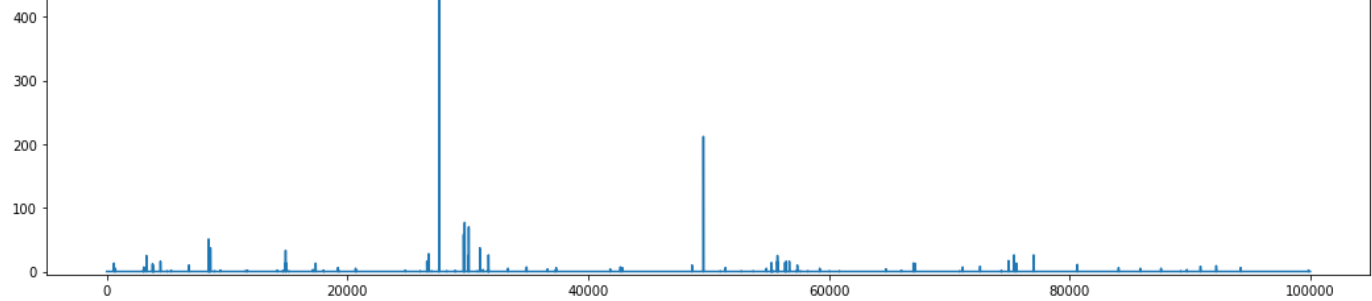
Чтение из кукушки различной степени

In [263...

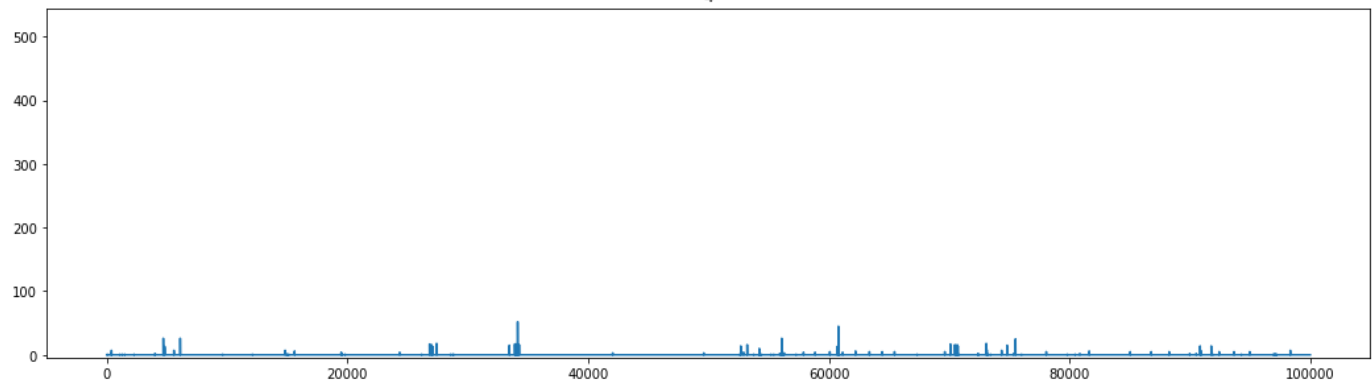
```
indices, data = get_x_ydict_from_csv('data/hash_find_real_data_cuckoo_degrees.csv')
keys = list(data.keys())
maximum = 0
for key in keys:
    if max(data[key]) > maximum and key not in []:
        maximum = max(data[key])

fig, plots = plt.subplots(7, 1, figsize=(18, 40))
fig.patch.set_facecolor('xkcd:white')
for i in range(len(keys)):
    plots[i].set_title(keys[i])
    plots[i].plot(indices, data[keys[i]])
    bottom, top = plots[i].get_ylim()
    plots[i].set_ylim(-5, maximum)
plt.savefig('test.png')
```

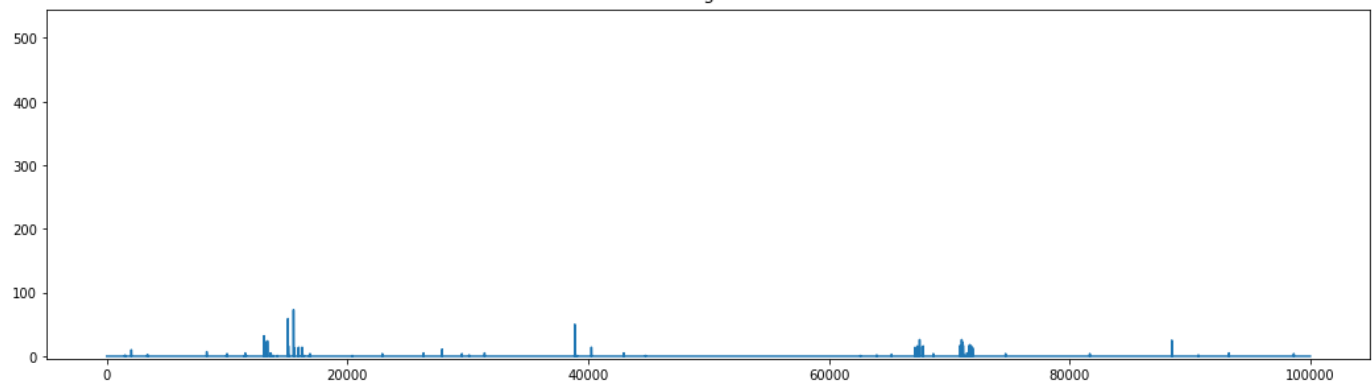




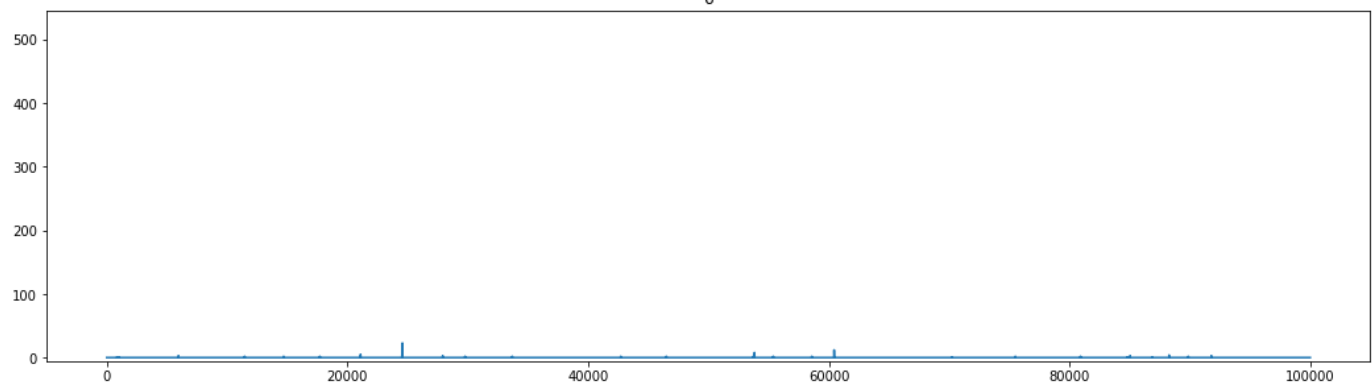
4



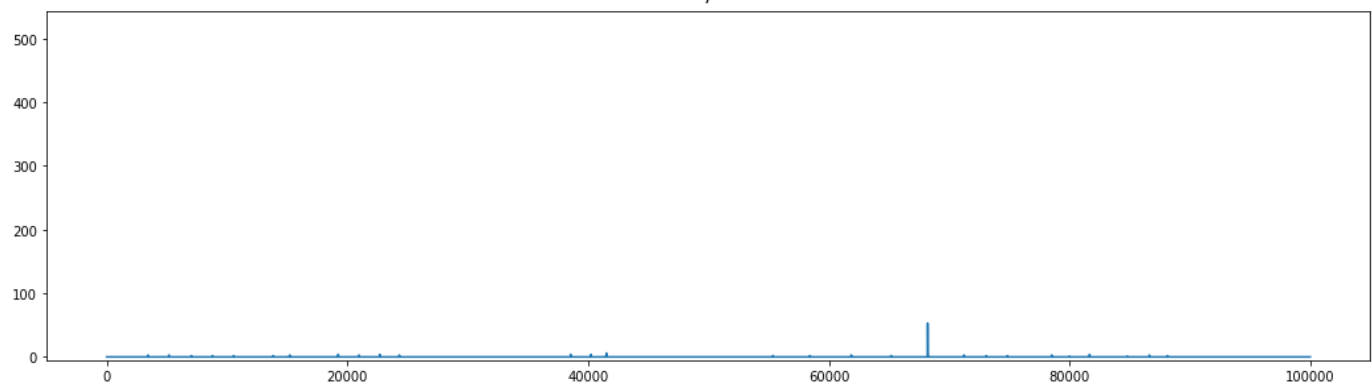
5



6



7



Удаление из кукушки различной степени

In [264...

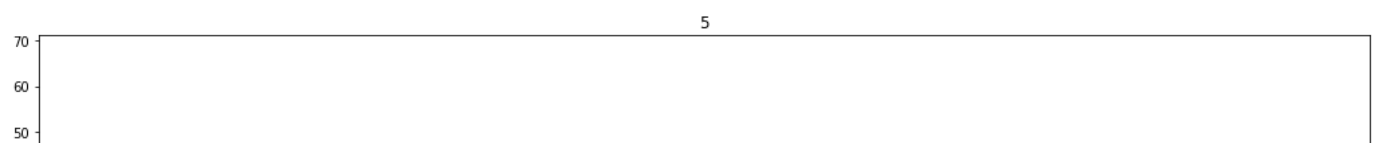
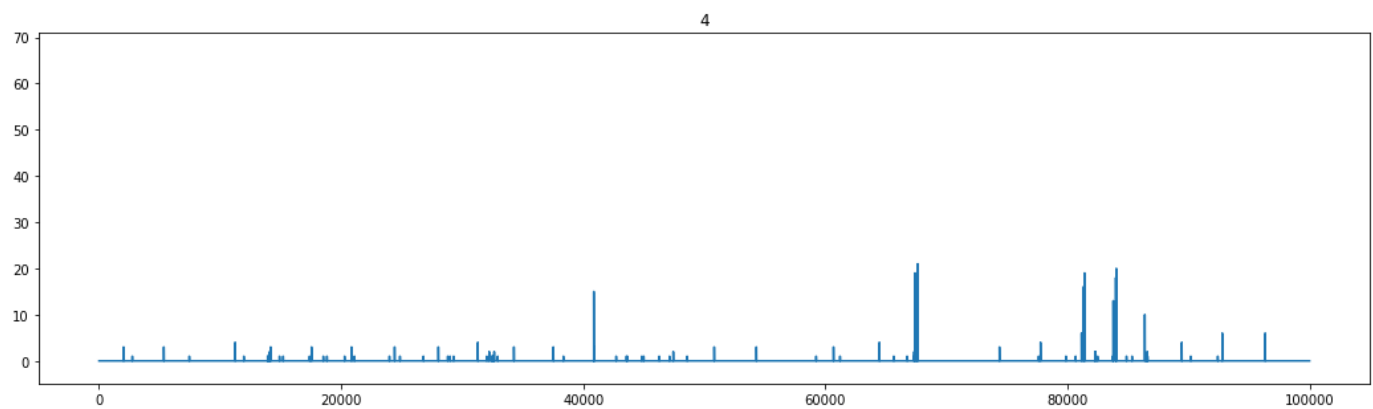
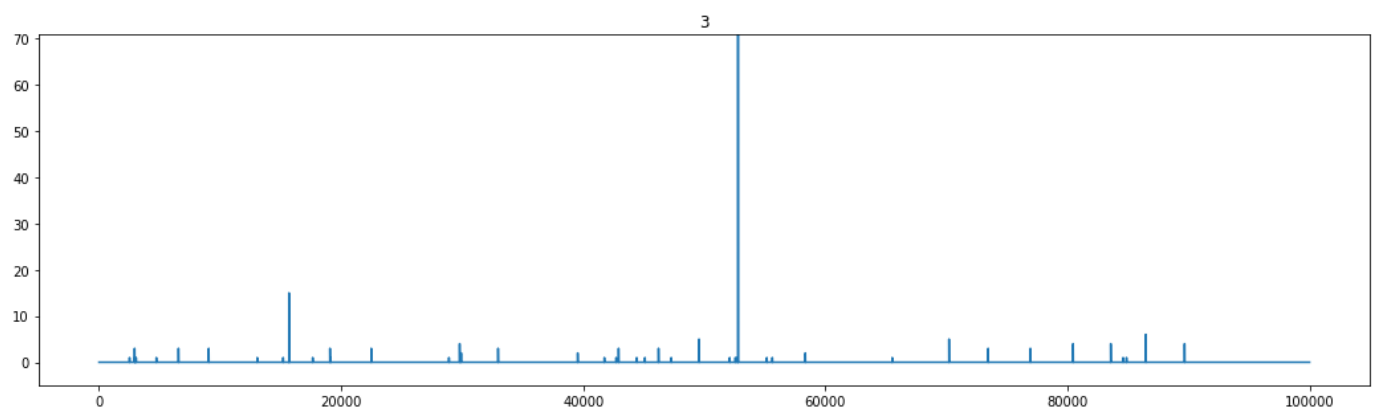
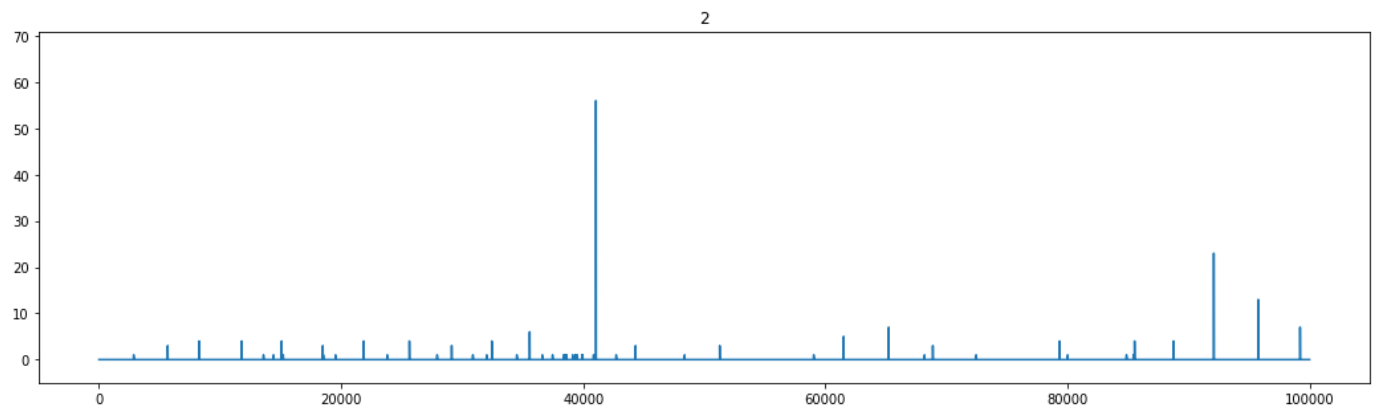
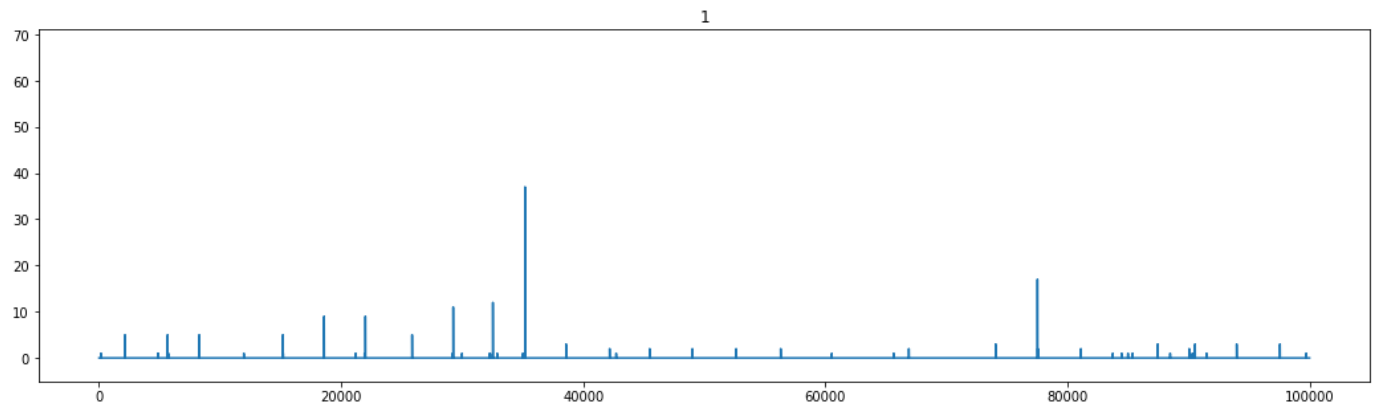
```
indices, data = get_x_ydict_from_csv('data/hash_erase_real_data_cuckoo_degrees.csv')
keys = list(data.keys())
maximum = 0
```

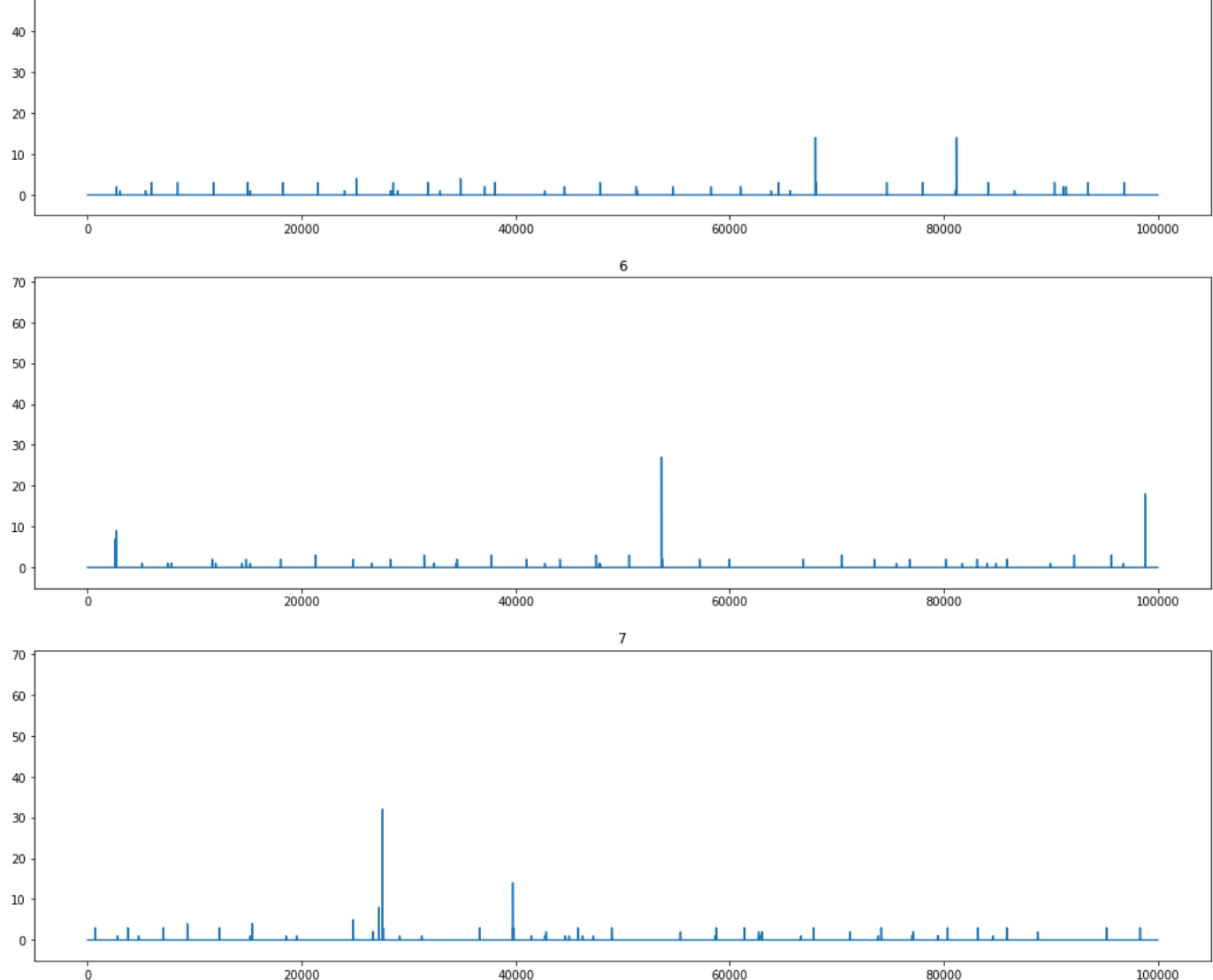
```

for key in keys:
    if max(data[key]) > maximum and key not in []:
        maximum = max(data[key])

fig, plots = plt.subplots(7, 1, figsize=(18, 40))
fig.patch.set_facecolor('xkcd:white')
for i in range(len(keys)):
    plots[i].set_title(keys[i])
    plots[i].plot(indices, data[keys[i]])
    bottom, top = plots[i].get_ylim()
    plots[i].set_ylim(-5, maximum)
plt.savefig('test.png')

```





Выводы

- `std::hash_map` не зря в `std`, он крайне хорош при вставке и поиске
- `Cuckoo` ожидаемо прекрасно показывает себя на поиске и удалении
- Степень полинома в используемой в `Cuckoo` хэш-функции конкретно для наших real-life данных оказалась не особо значимой, но в среднем, кажется, с увеличением степени полинома вероятность возникновения "выброса" снижается
- Хэшировать строки - ужасно медленно, если есть способ этого избежать в практических задачах - этого нужно избегать
- Писать на C++ очень больно, нужно учить Rust, хочу обратно в Питон :(

In []: