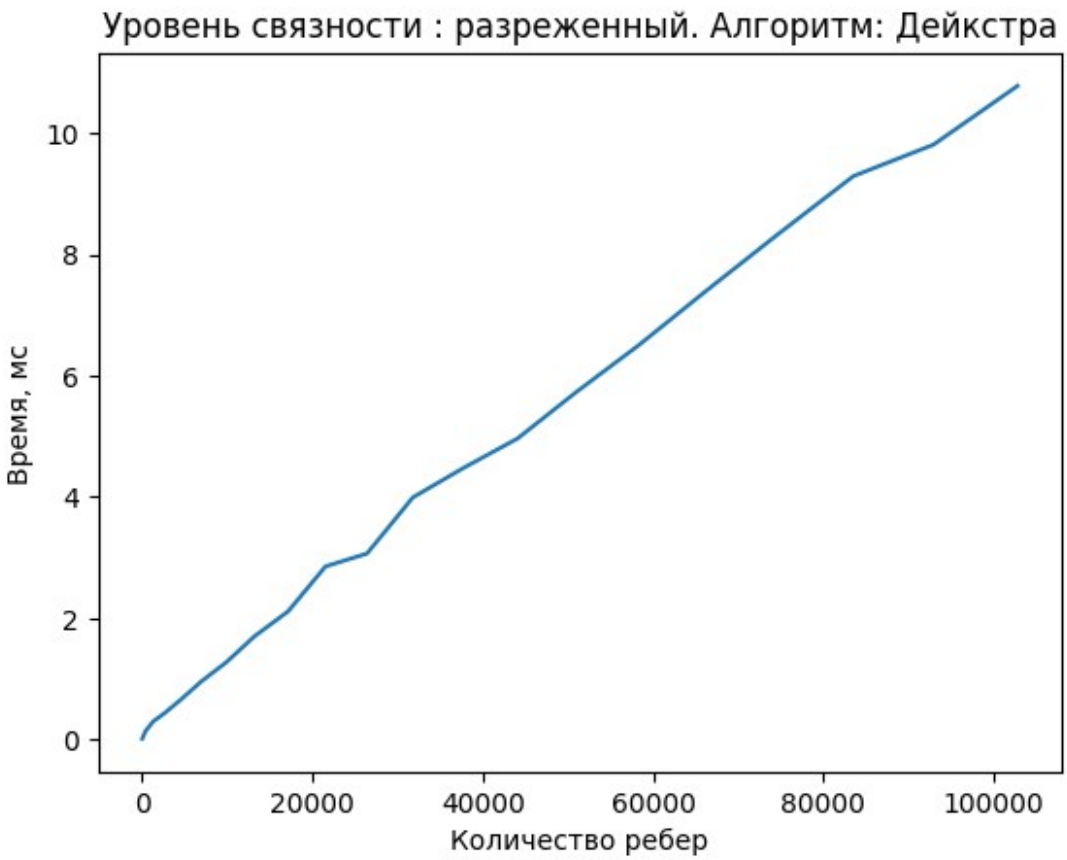
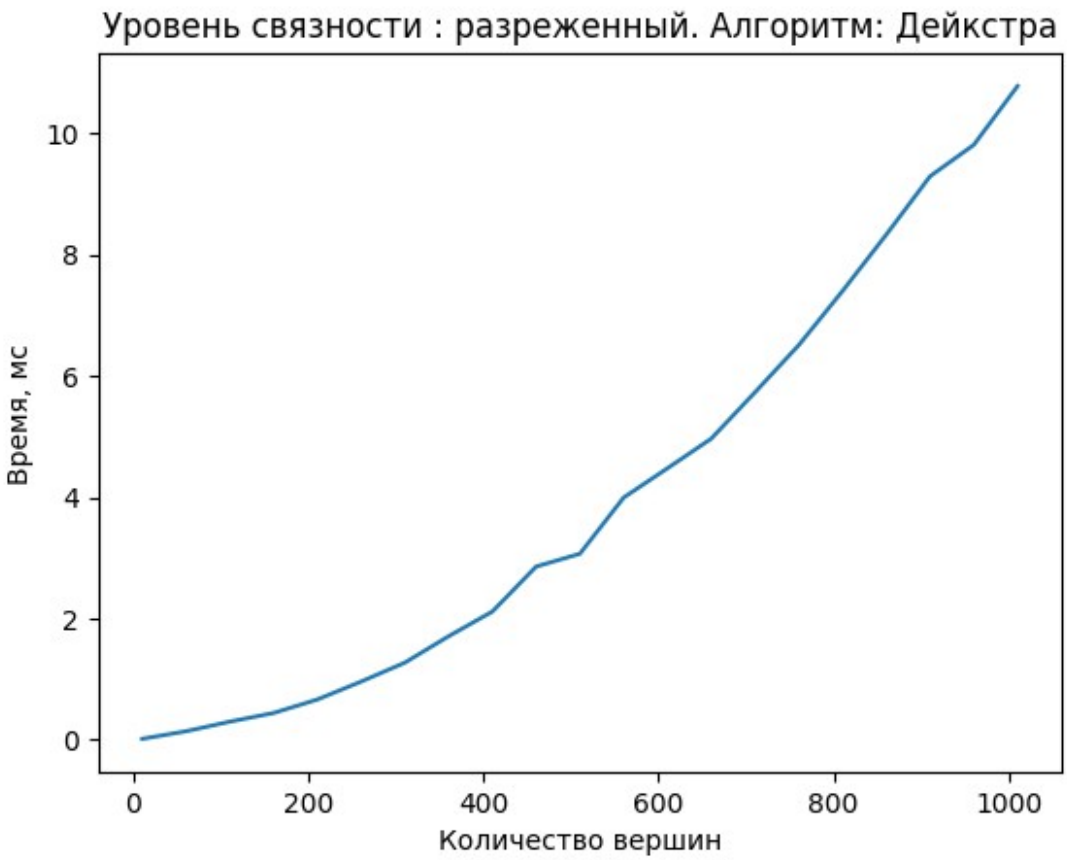
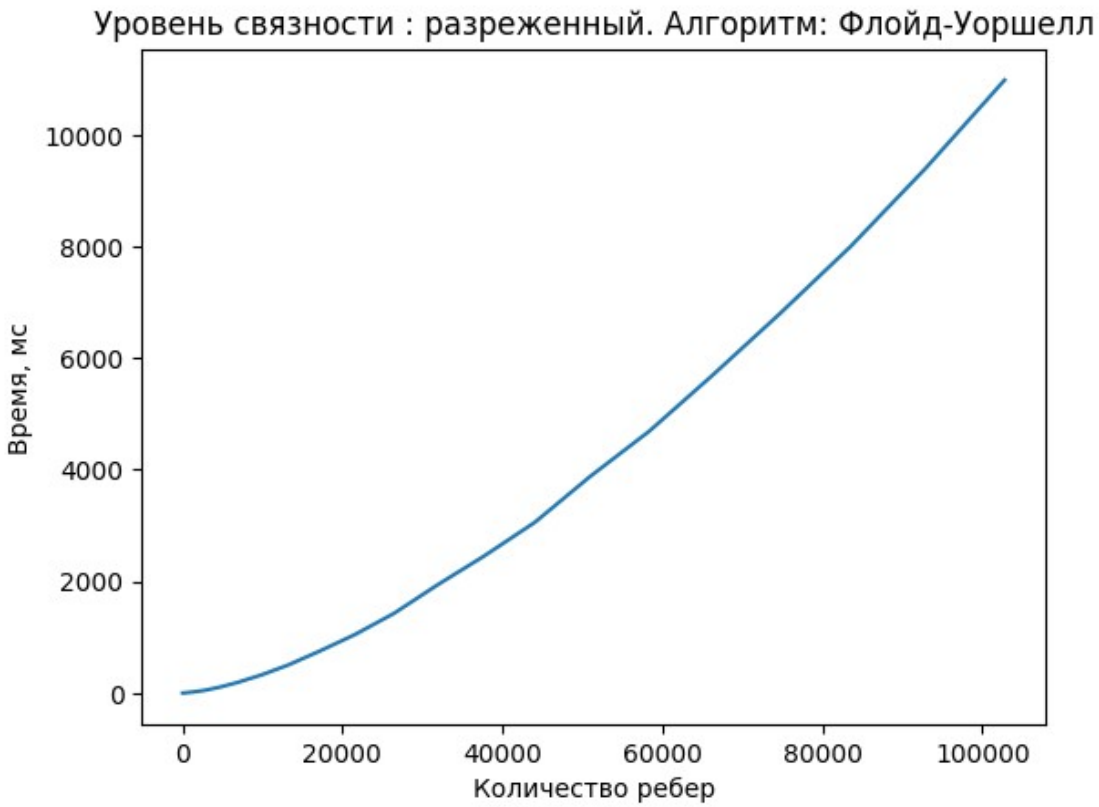
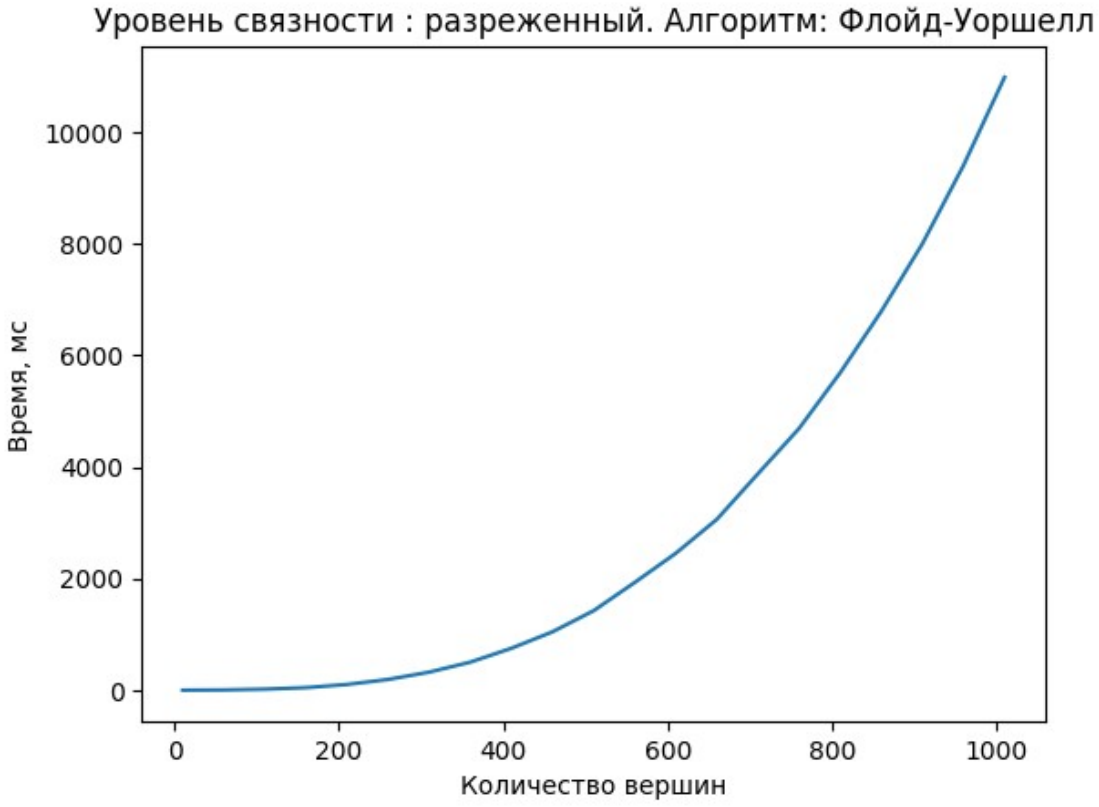
```
file_names = []
text_sizes = ['small', 'half', 'full']
sort_types = ['deikstra', 'floyd', 'ford']
types = ['edges', 'vertexes']

for text_size in text_sizes:
  for sort_type in sort_types:
    for typ in types:
      file_names.append(text_size + '_' + sort_type + '_' + typ)

import os.path
graph_data = {}
for k, file_name in enumerate(file_names):
  if os.path.isfile(file_name):
    with open(file_name, 'r') as file:
      print(file_name)
      text = file.read()[:-1]
      lines = [dot.split(',') for dot in text.split(';')]
      graph_data[file_name] = lines

small_deikstra_edges
small_deikstra_vertexes
small_floyd_edges
small_floyd_vertexes
small_ford_edges
small_ford_vertexes
half_deikstra_edges
half_deikstra_vertexes
half_floyd_edges
half_floyd_vertexes
half_ford_edges
half_ford_vertexes
full_deikstra_edges
full_deikstra_vertexes
full_floyd_edges
full_floyd_vertexes
full_ford_edges
full_ford_vertexes

from matplotlib import pyplot as plt
import numpy as np

def extract_data_from_name(name):
  paths = name.split('_')
  return {
      'text_size': paths[0],
      'sort_type': paths[1],
      'typ': paths[2],
  }

link_info = {
```

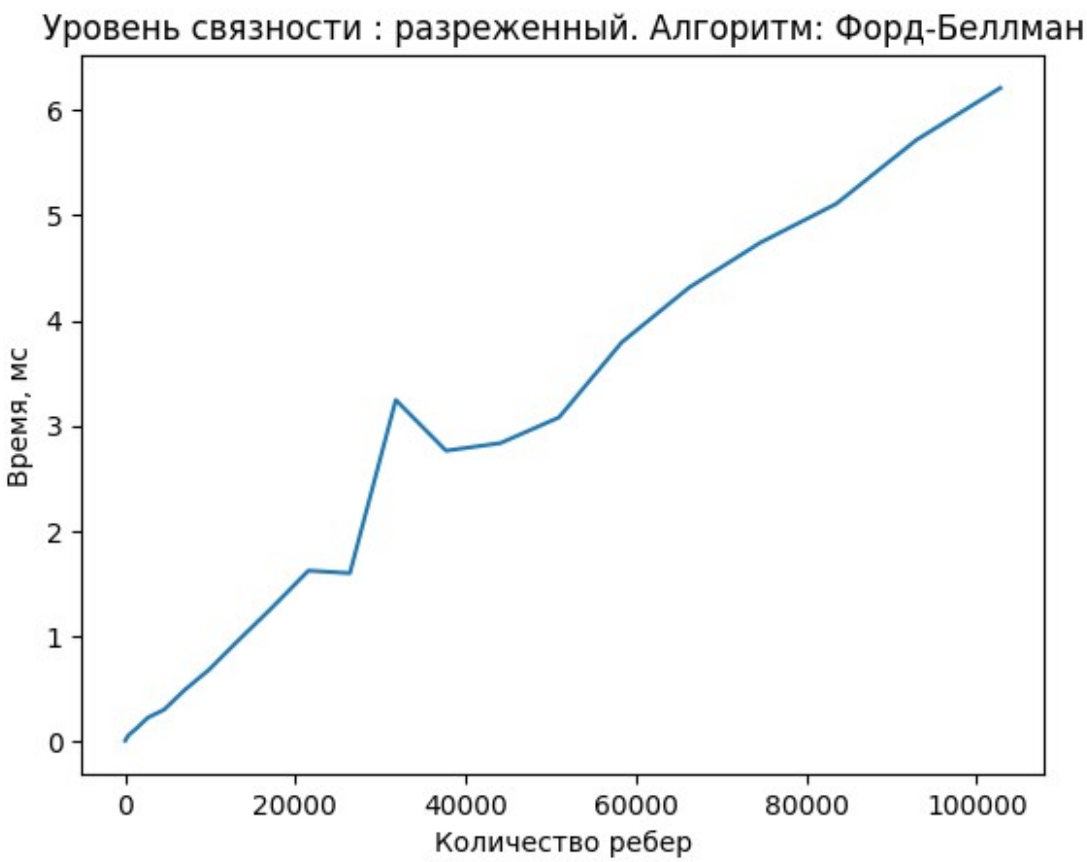
```
    'small': 'разреженный',
    'half': 'обычный',
    'full': 'полный'
}
algo_names = {
    'deikstra' : 'Дейкстра',
    'floyd': 'Флойд-Уоршелл',
    'ford':  'Форд-Беллман',
}

def create_title(data):

  return f"Уровень связности : {link_info[data['text_size']]}.
Алгоритм: {algo_names[data['sort_type']]}"

type_names = {
    'edges': 'ребер',
    'vertexes': 'вершин',
}
for k, v in graph_data.items():
  XX = [line[1] for line in v]
  YY = [line[0] for line in v]

  title_data = extract_data_from_name(k)
  ax = plt.subplot()
  ax.plot(np.asarray(XX, int), np.asarray(YY, float))
  ax.set_title(create_title(title_data))
  t = type_names[title_data['typ']]
  ax.set_xlabel(f'Количество ' + t)
  ax.set_ylabel('Время, мс')
  plt.show()
```

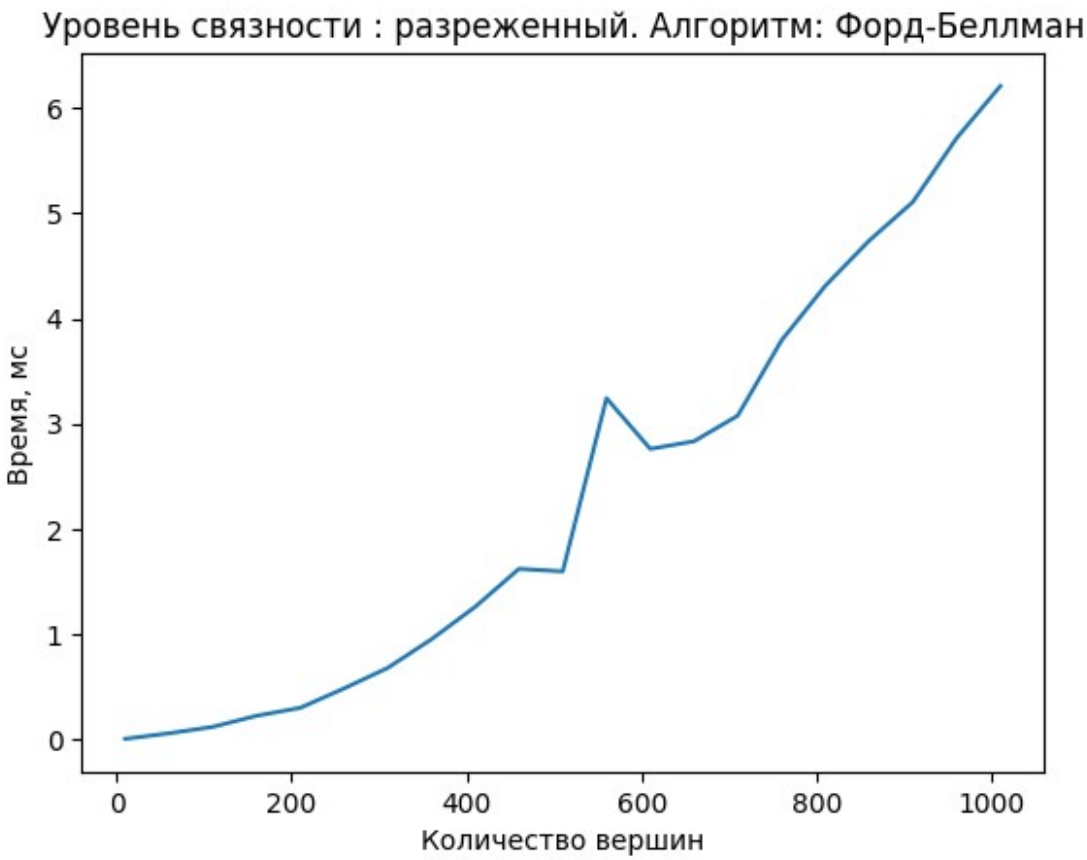

Уровень связности : разреженный. Алгоритм: Дейкстра
Время, мс
0
2
4
6
8
10
0
20000
40000
60000
80000
100000
Количество ребер

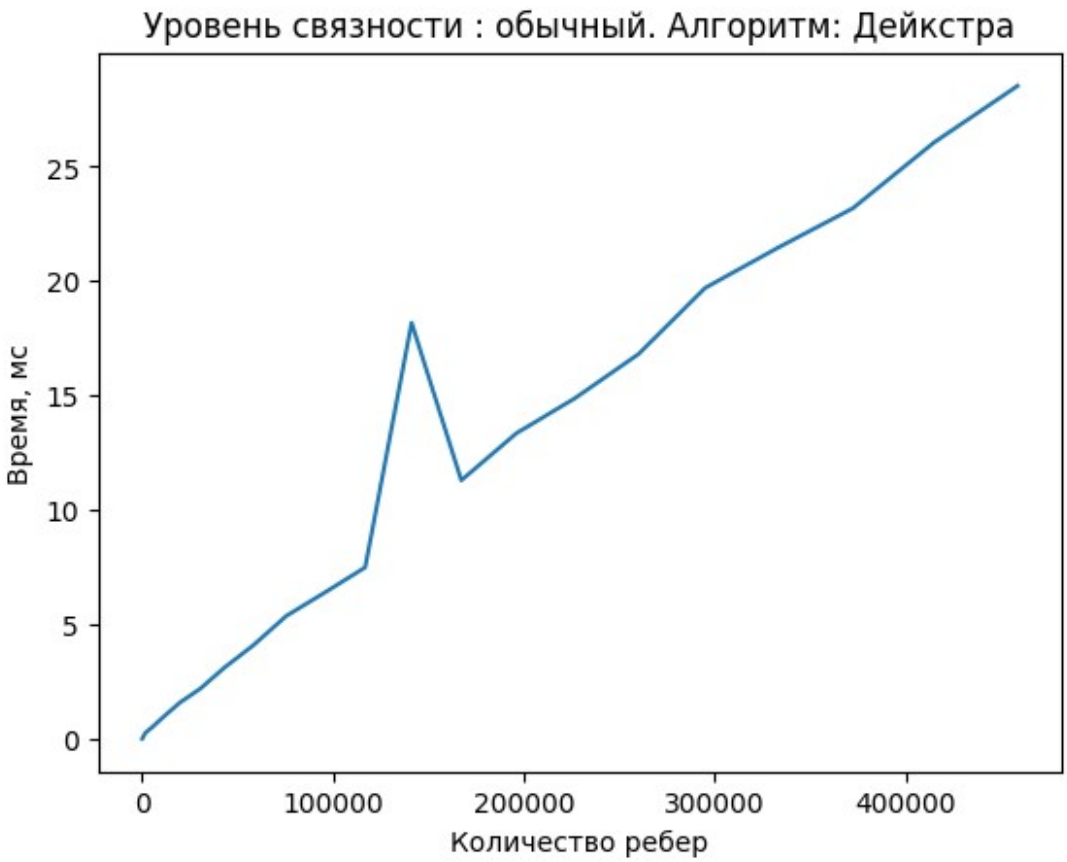

Уровень связности : разреженный. Алгоритм: Дейкстра
Время, мс
0
2
4
6
8
10
0
200
400
600
800
1000
Количество вершин

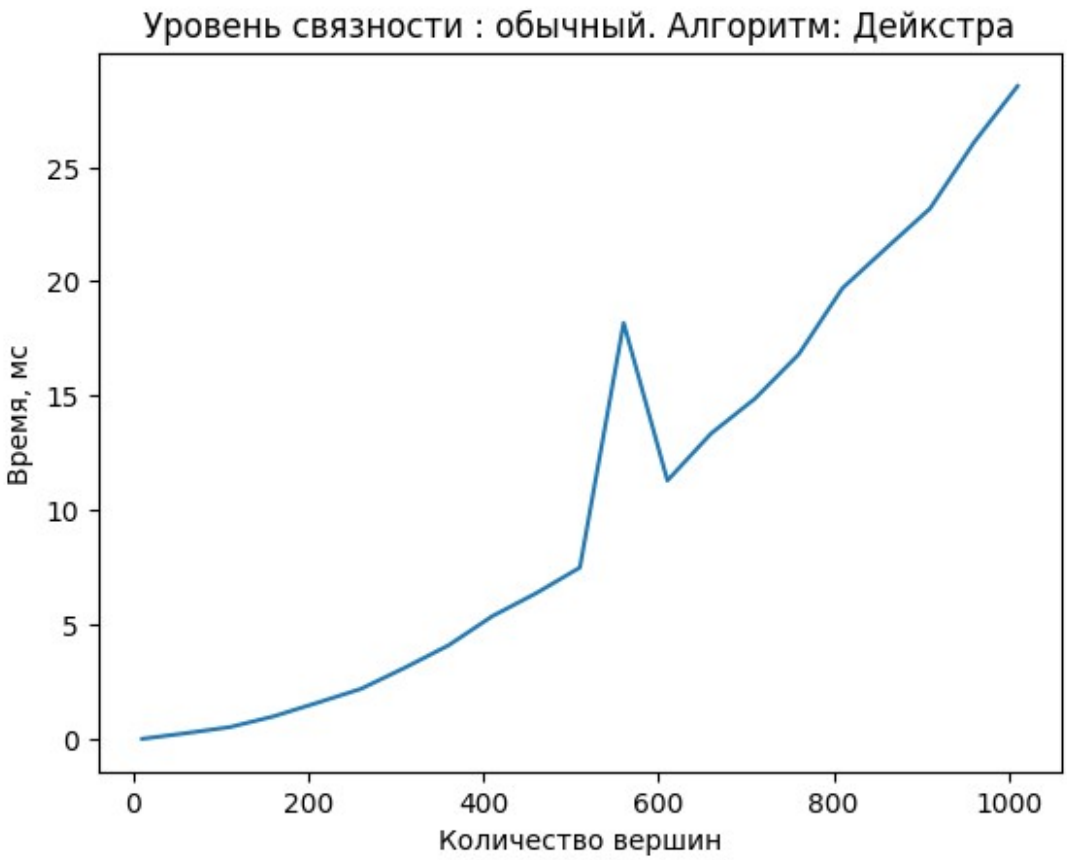

Уровень связности : разреженный. Алгоритм: Флойд-Уоршелл
Время, мс
0
2000
4000
6000
8000
10000
0
20000
40000
60000
80000
100000
Количество ребер

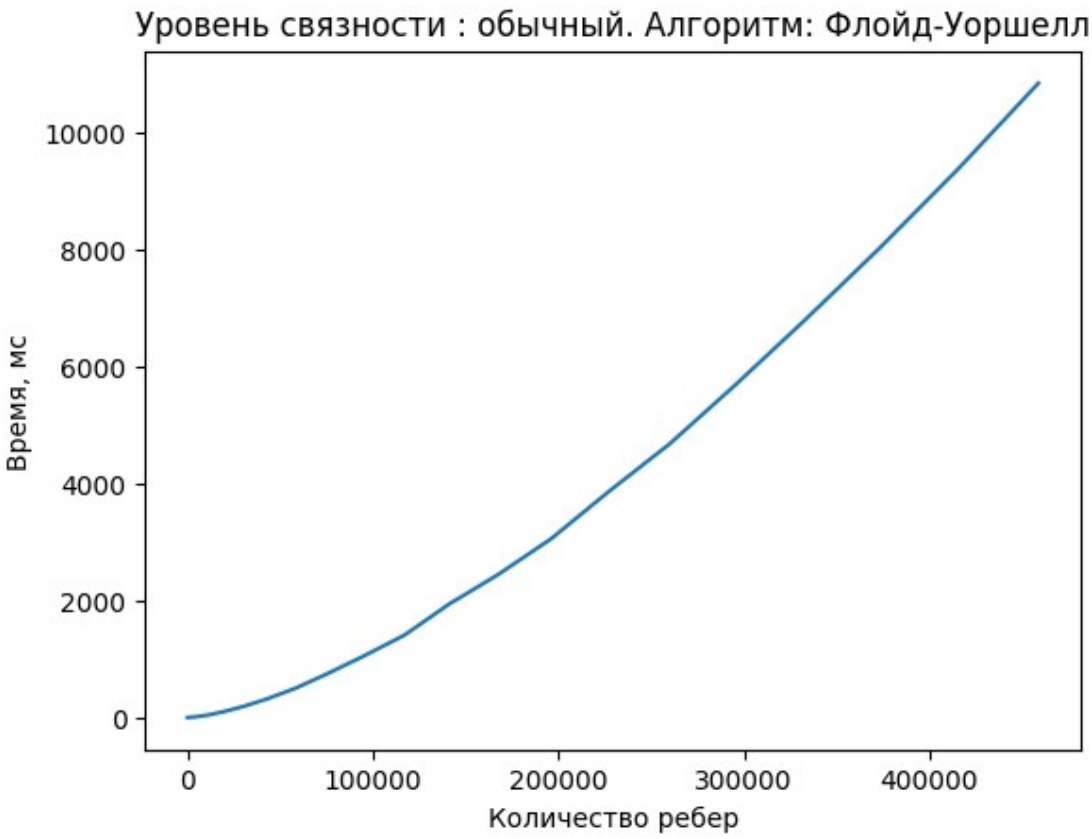

Уровень связности : разреженный. Алгоритм: Флойд-Уоршелл
Время, мс
0
2000
4000
6000
8000
10000
0
200
400
600
800
1000
Количество вершин

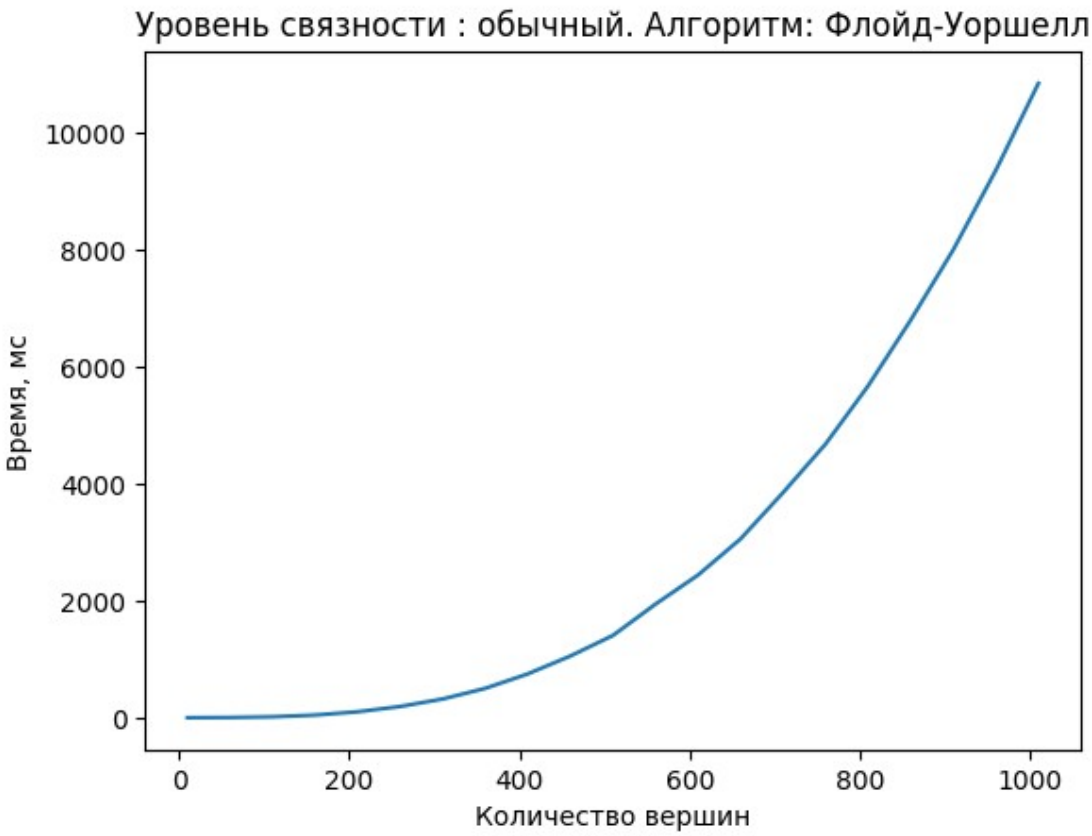

Уровень связности : разреженный. Алгоритм: Форд-Беллман
Время, мс
0
1
2
3
4
5
6
0
20000
40000
60000
80000
100000
Количество ребер

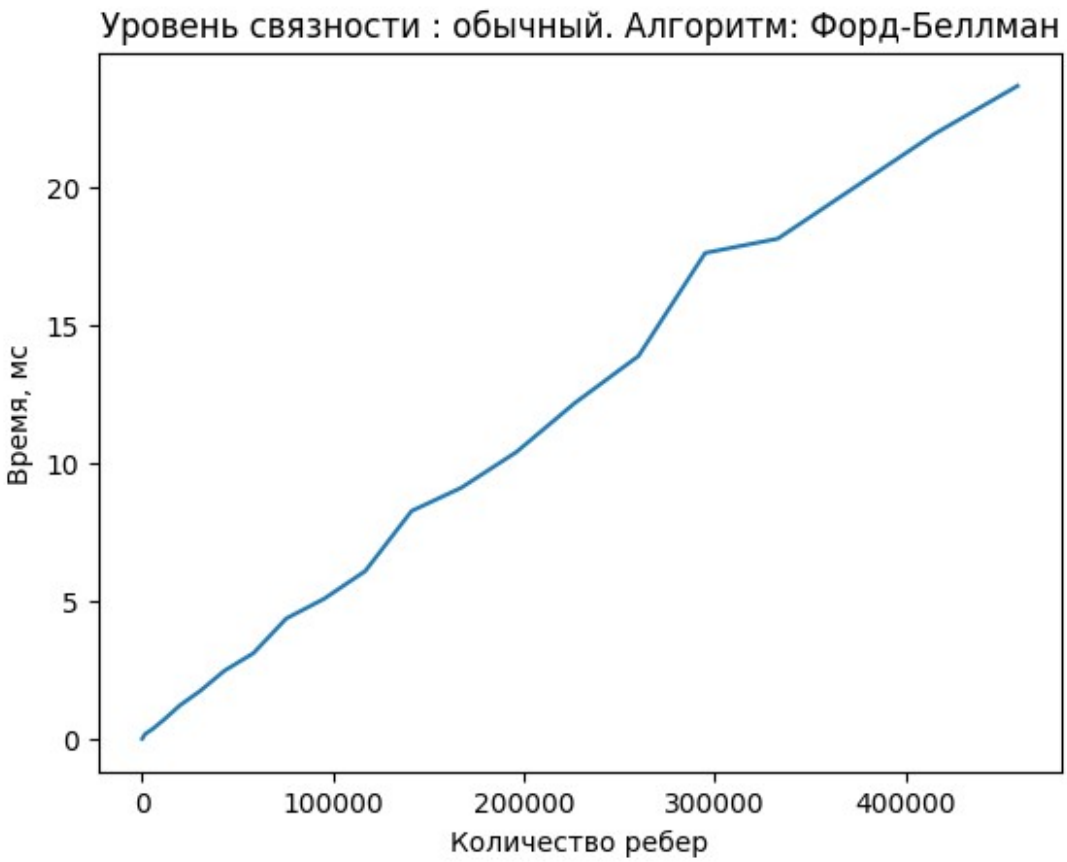

Уровень связности : разреженный. Алгоритм: Форд-Беллман
Время, мс
0
1
2
3
4
5
6
0
200
400
600
800
1000
Количество вершин

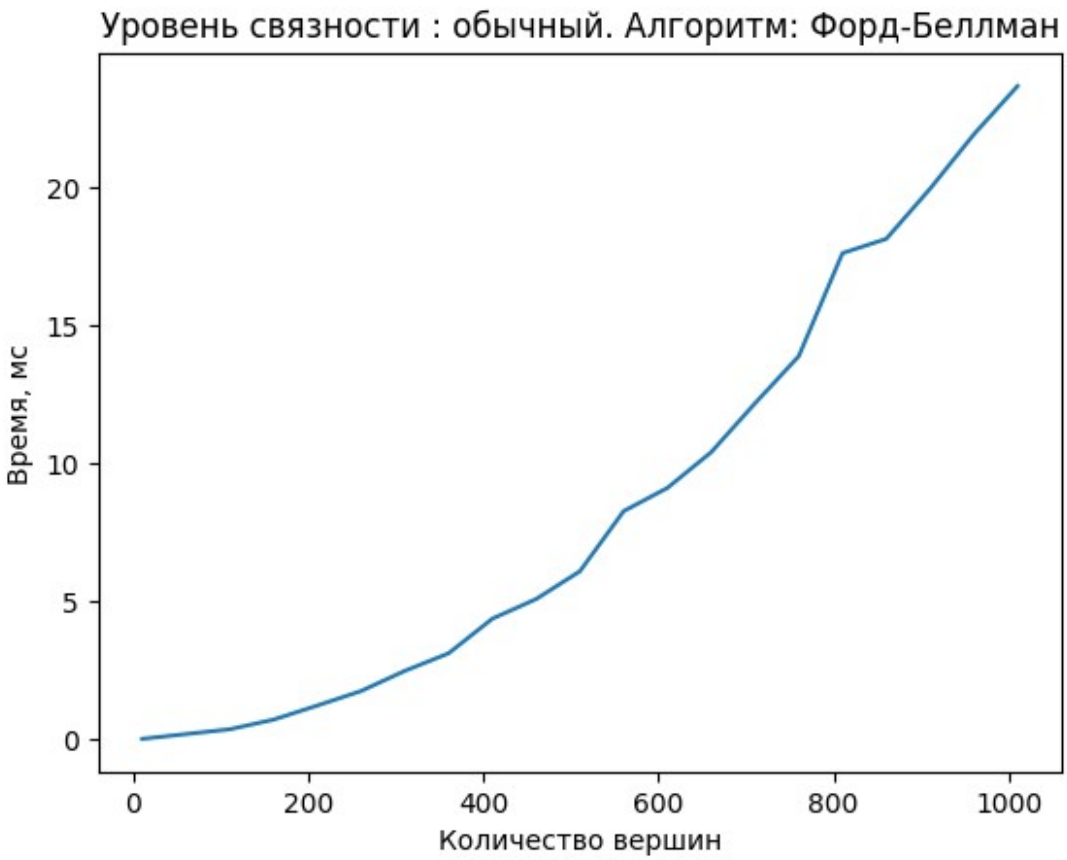

Уровень связности : обычный. Алгоритм: Дейкстра
Время, мс
0
5
10
15
20
25
0
100000
200000
300000
400000
Количество ребер

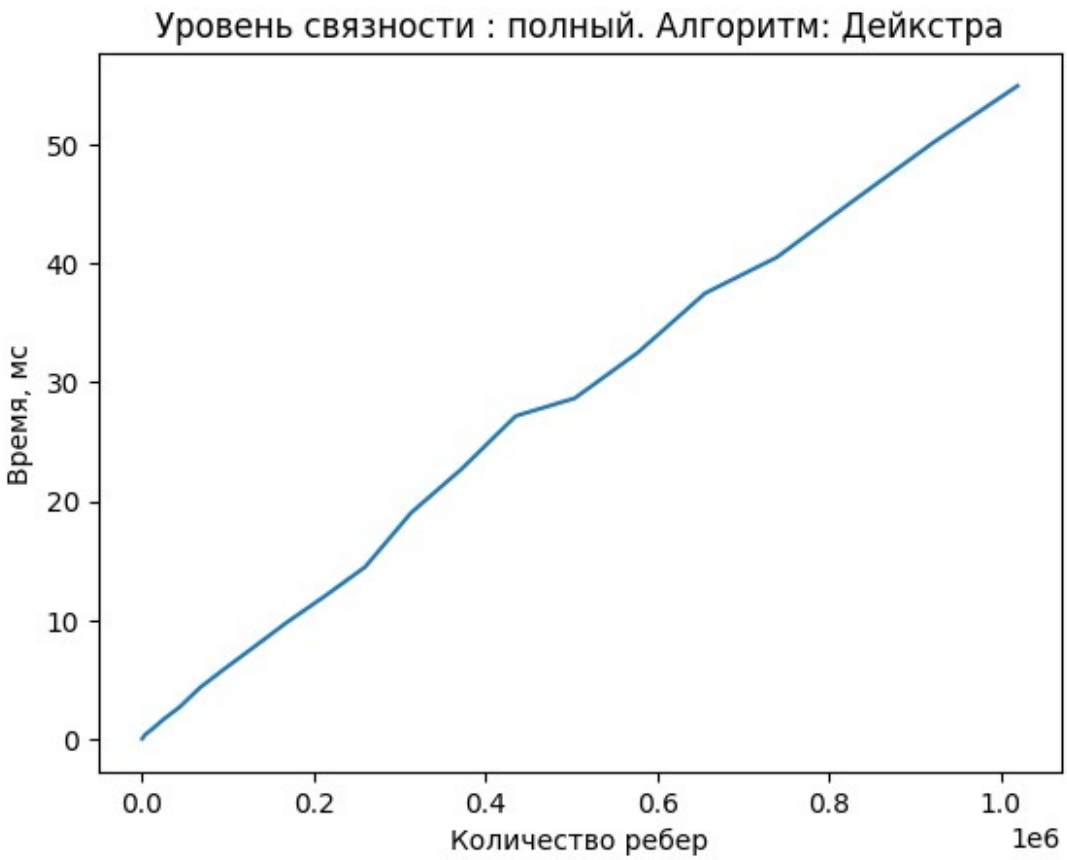

Уровень связности : обычный. Алгоритм: Дейкстра
Время, мс
0
5
10
15
20
25
0
200
400
600
800
1000
Количество вершин

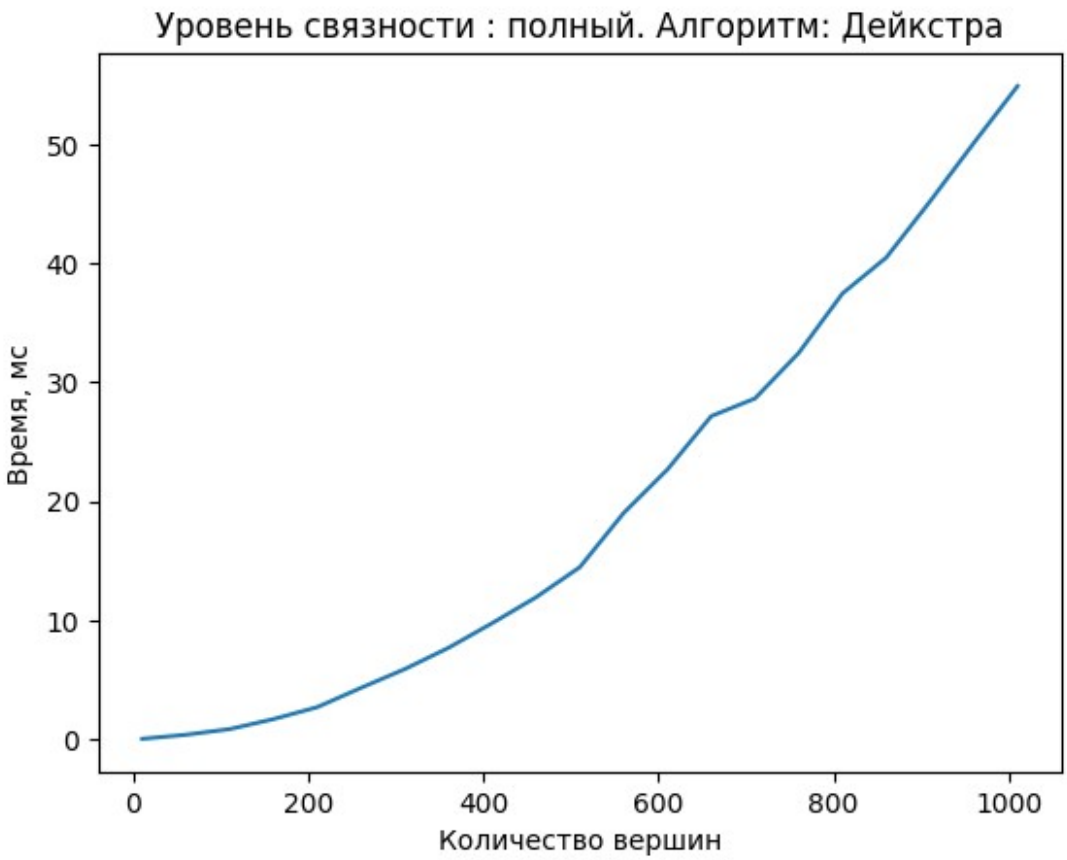

Уровень связности : обычный. Алгоритм: Флойд-Уоршелл
Время, мс
0
2000
4000
6000
8000
10000
0
100000
200000
300000
400000
Количество ребер

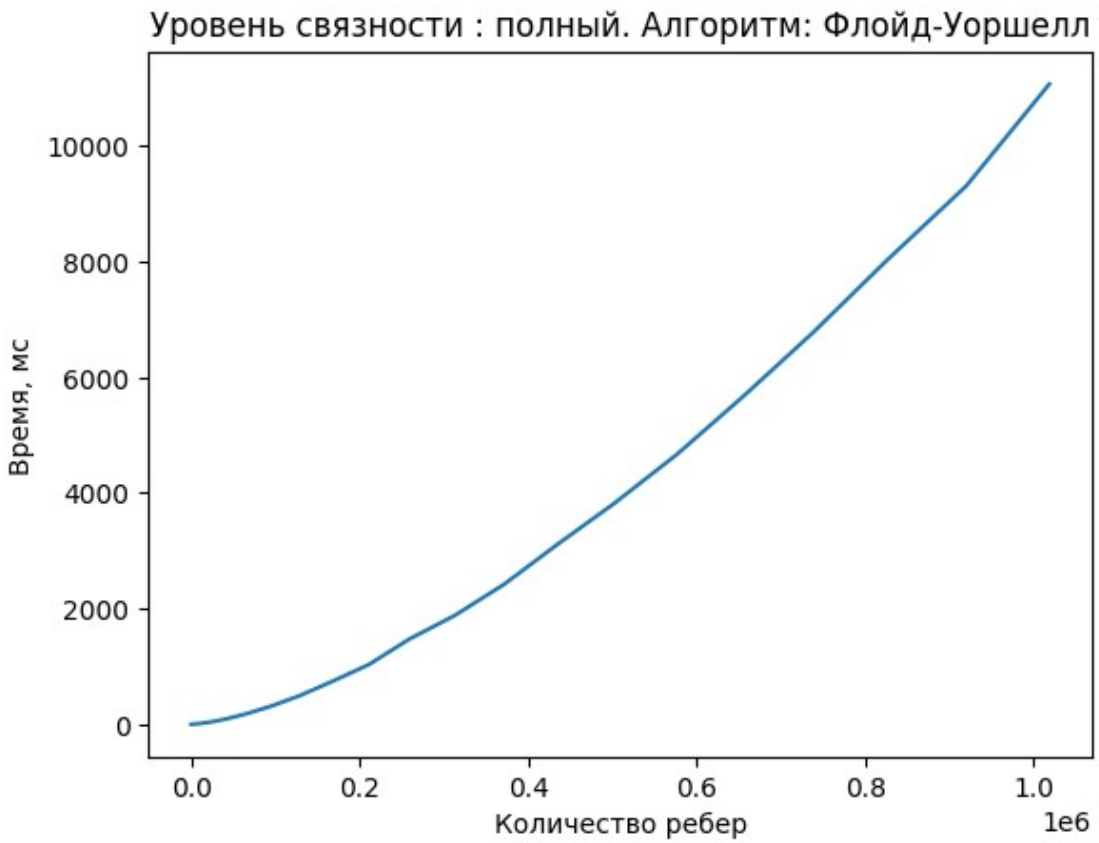

Уровень связности : обычный. Алгоритм: Флойд-Уоршелл
Время, мс
0
2000
4000
6000
8000
10000
0
200
400
600
800
1000
Количество вершин

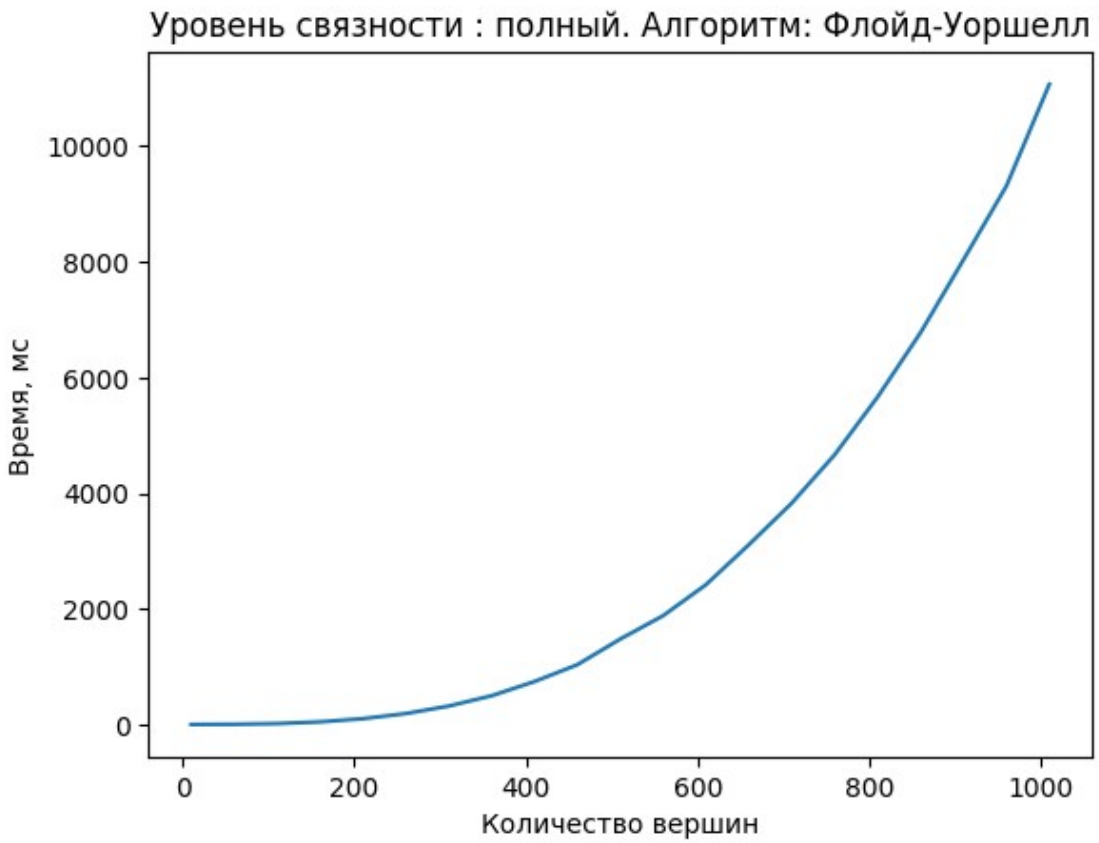

Уровень связности : обычный. Алгоритм: Форд-Беллман
Время, мс
0
5
10
15
20
0
100000
200000
300000
400000
Количество ребер

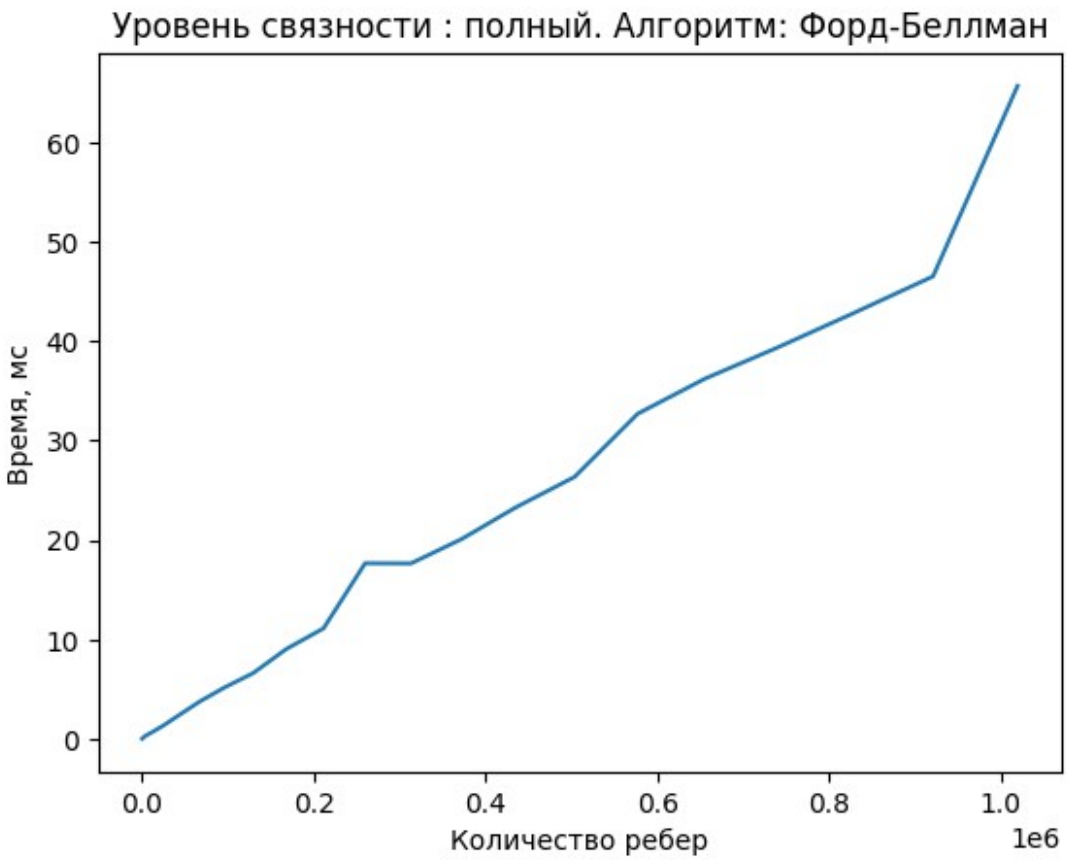

Уровень связности : обычный. Алгоритм: Форд-Беллман
Время, мс
0
5
10
15
20
0
200
400
600
800
1000
Количество вершин

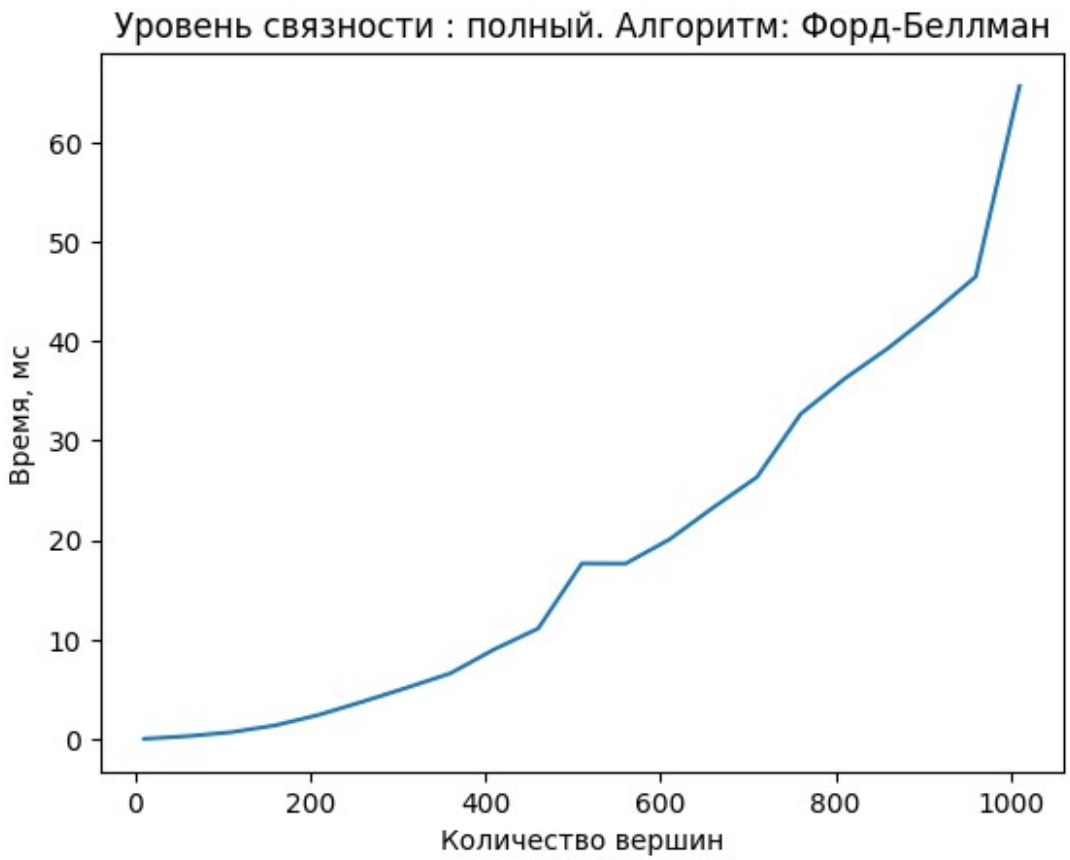

Уровень связности : полный. Алгоритм: Дейкстра
Время, мс
0
10
20
30
40
50
0.0
0.2
0.4
0.6
0.8
1.0
1e6
Количество ребер


Уровень связности : полный. Алгоритм: Дейкстра
Время, мс
0
10
20
30
40
50
0
200
400
600
800
1000
Количество вершин


Уровень связности : полный. Алгоритм: Флойд-Уоршелл
Время, мс
10000
8000
6000
4000
2000
0
0.0
0.2
0.4
0.6
0.8
1.0
Количество ребер
1e6

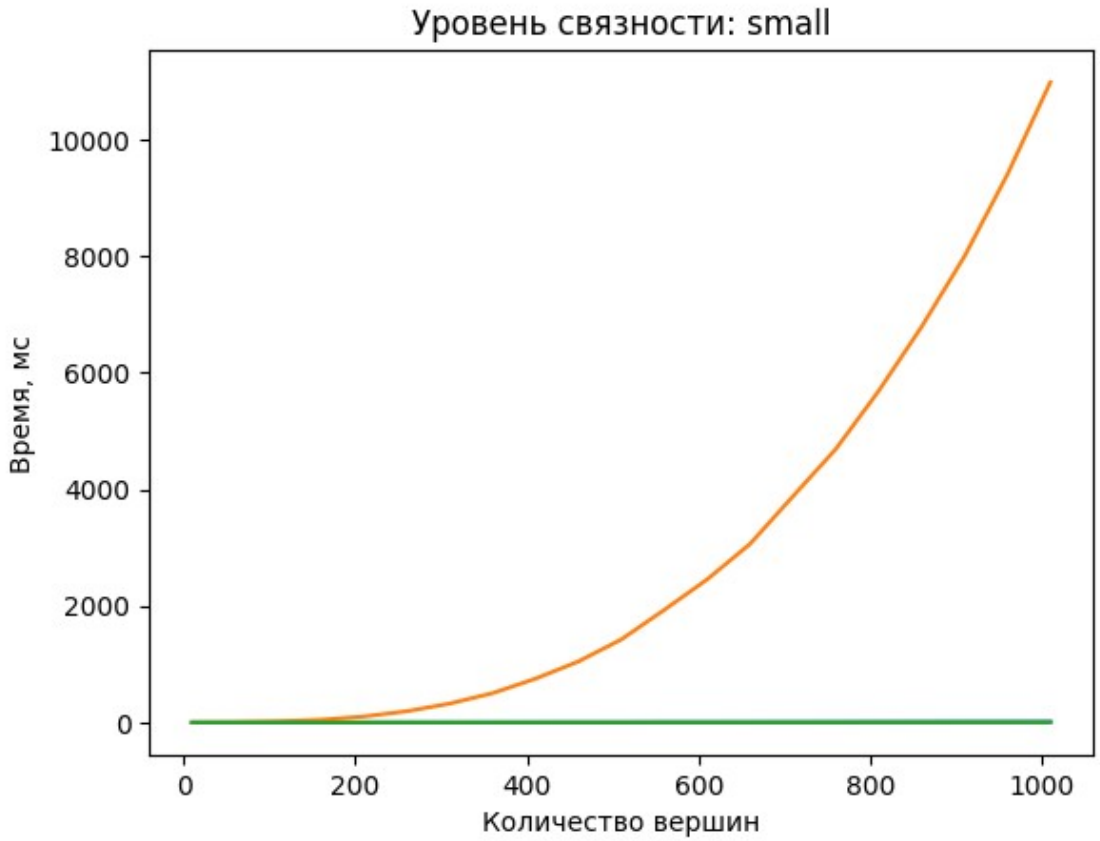

Уровень связности : полный. Алгоритм: Флойд-Уоршелл
Время, мс
0
2000
4000
6000
8000
10000
0
200
400
600
800
1000
Количество вершин


Уровень связности : полный. Алгоритм: Форд-Беллман
Время, мс
0
10
20
30
40
50
60
0.0
0.2
0.4
0.6
0.8
1.0
1e6
Количество ребер

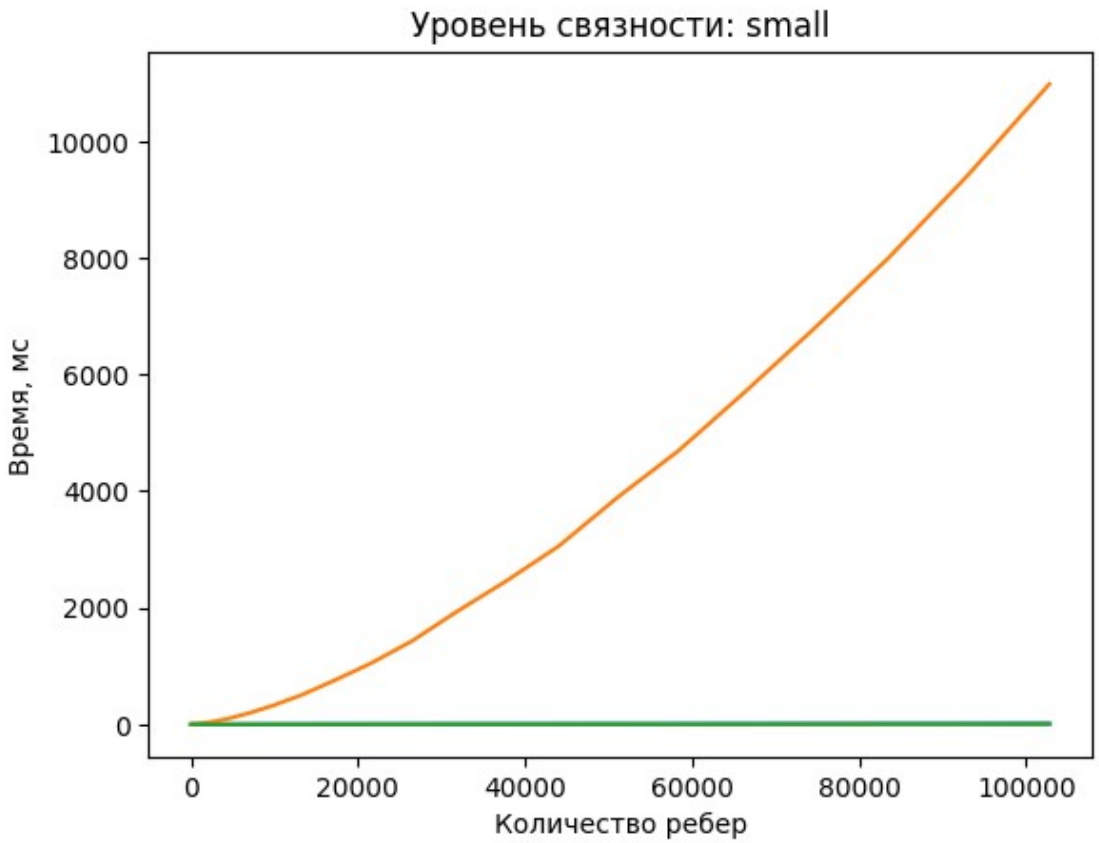



```
data_sorted = {}

for i in ['small', 'full', 'half']:
  data_sorted[i] = {}
  for j in ['vertexes', 'edges']:
    data_sorted[i][j] = {}

for k, v in graph_data.items():
  title_data = extract_data_from_name(k)
  data_sorted[title_data['text_size']][title_data['typ']][k] = v

for i in data_sorted.values():
  for j in i.values():
    for k, v in j.items():
      print(k)
      XX = [line[1] for line in v]
      YY = [line[0] for line in v]

      title_data = extract_data_from_name(k)
      ax = plt.subplot()
      ax.plot(np.asarray(XX, int), np.asarray(YY, float))
    ax.set_title('Уровень связности: ' + title_data['text_size'])
    t = type_names[title_data['typ']]
```

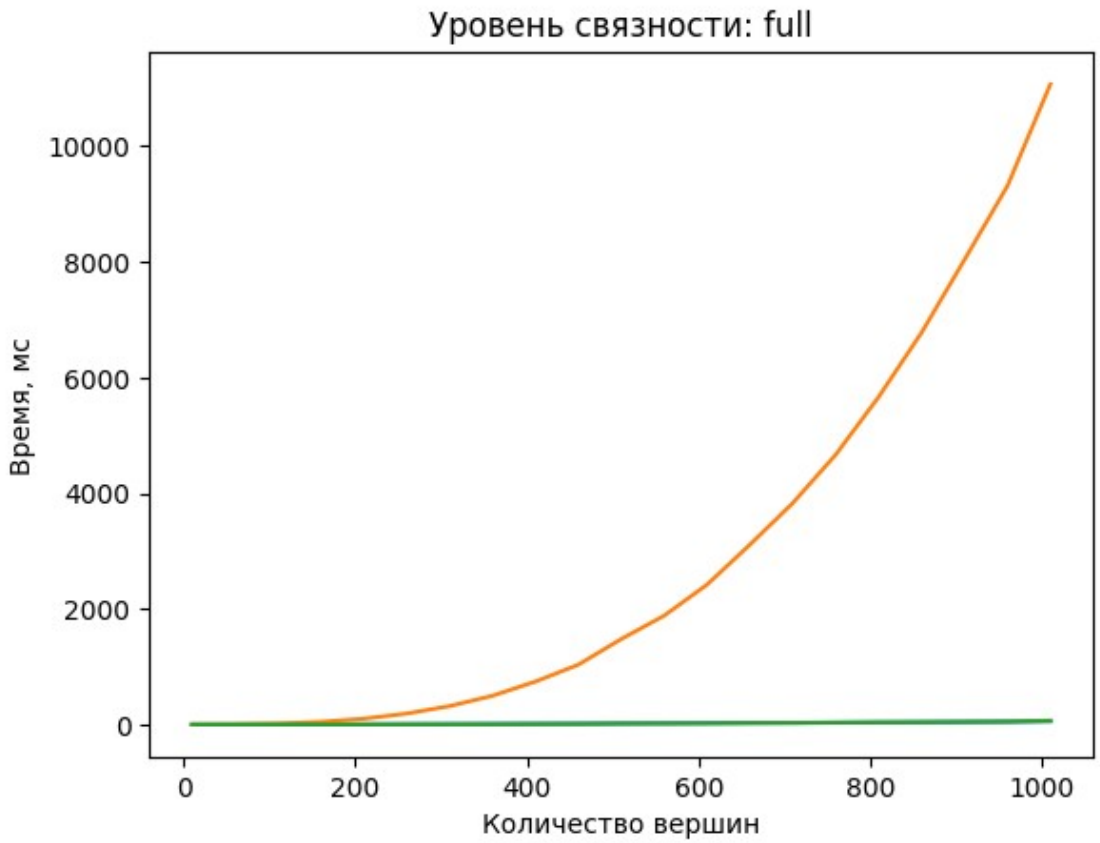
```
    ax.set_xlabel(f'Количество ' + t)
    ax.set_ylabel('Время, мс')
    plt.show()

small_deikstra_vertexes
small_floyd_vertexes
small_ford_vertexes
```



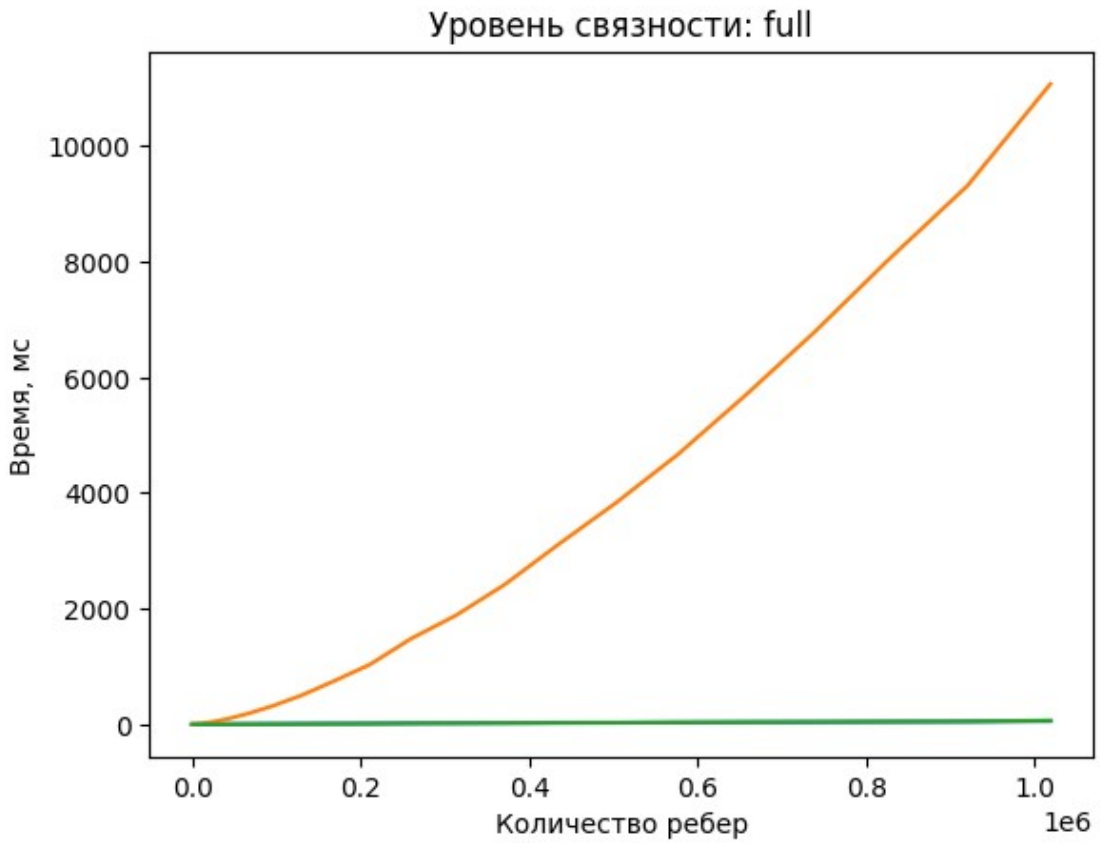

```
small_deikstra_edges
small_floyd_edges
small_ford_edges
```



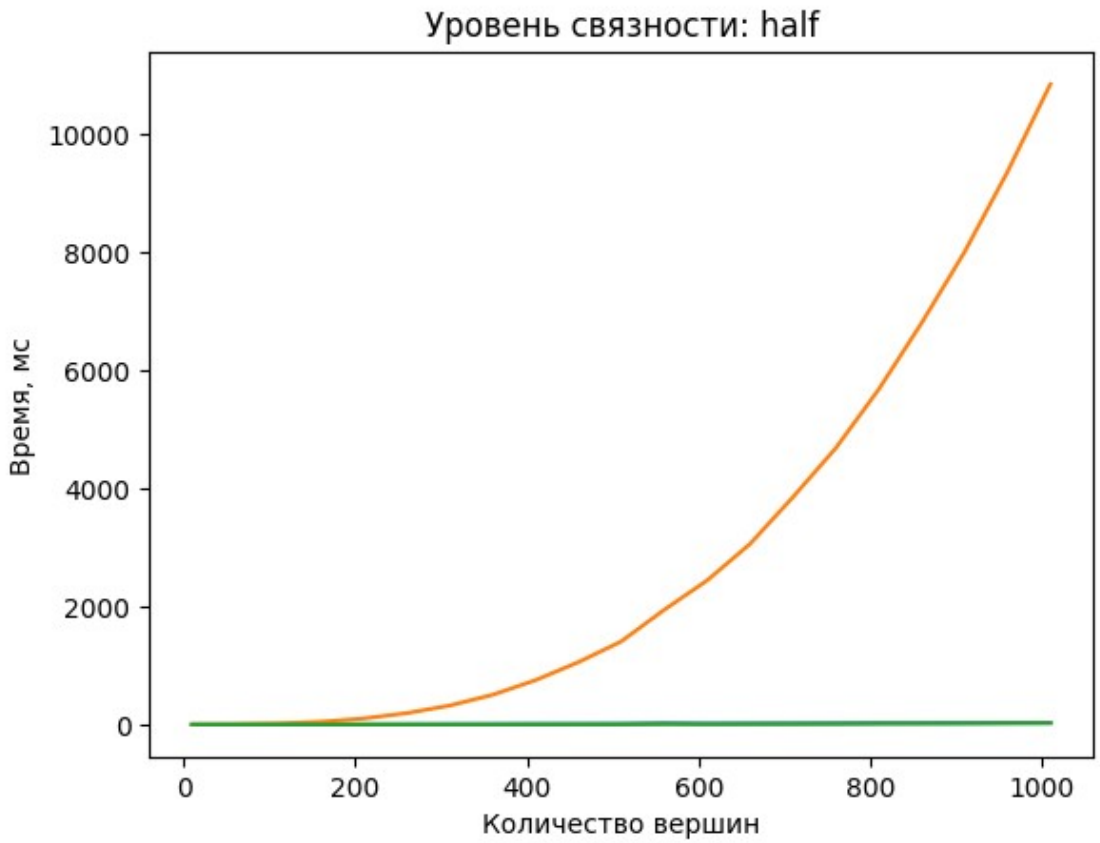
full_deikstra_vertexes
full_floyd_vertexes
full_ford_vertexes



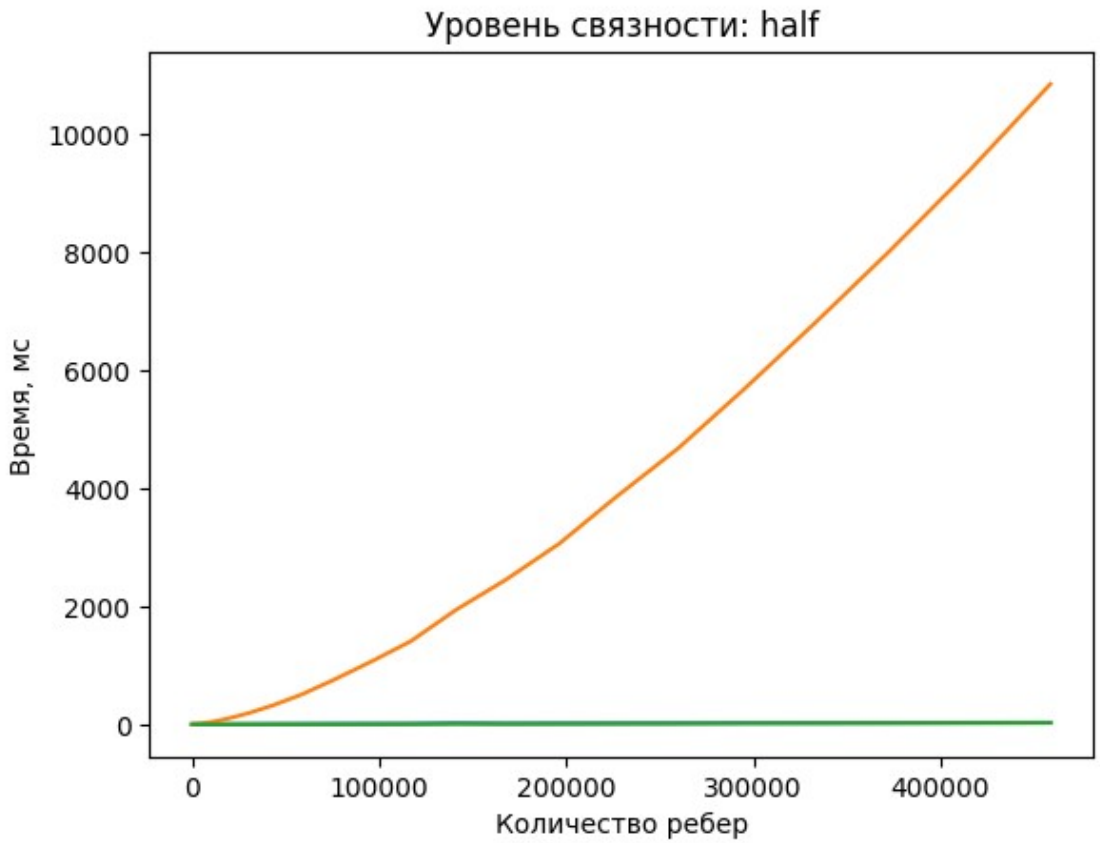
full_deikstra_edges
full_floyd_edges
full_ford_edges



half_deikstra_vertexes
half_floyd_vertexes
half_ford_vertexes



half_deikstra_edges
half_floyd_edges
half_ford_edges



Отмечу, что линий на каждом графике все-таки три. Просто две из них сливаются

Сложность алгоритмов

Теоретическая асимптотика (n - кол-во вершин, m - кол-во ребер):

Дейкстра: O(n^2 + m)

Флойд-Уоршелл: O(n^3)

Беллман-Форд: O(n*m)

Сделать вывод об асимптотике алгоритмов по полученным графикам тяжело, так как графики нерепрезентативны.

Ведь на каждом из них вместе с ростом числа вершин, растет число ребер и наоборот.

Нелинейный вид таких графиков, как например, график для алгоритма Беллмана-Форда для полносвязного графа объясняется как раз этим. С ростом количества вершин, количество ребер при этом растет как

квадрат. Из этого получаем, что сложность в целом растет как куб. Примерно это мы и видим на графике.

Также, например, нерепрезентативным становится график для алгоритма Флойда-Уоршелла (полносвязный). Казалось бы, для подтверждения гипотезы мы хотим увидеть независимость сложности от количества ребер, но сложность растет линейно. На самом деле это объясняется тем, что количество вершин при этом растет как sqrt(m), а значит сложность растет как m^(3/2).

Таким образом, по созданным графикам нельзя сделать выводы об асимптотике алгоритмов. Считаю важным отметить, что я не имел возможности выбрать данные для построения графиков, поэтому не несу ответственность за невозможность их создания репрезентативными.

Применимость алгоритмов

Алгоритм Беллмана-Форда эффективен для слабосвязных графов, а также может использоваться для графов с ребрами отрицательного веса.

Алгоритм Дейкстры эффективен для сильносвязных графов.

Алгоритм Флойда-Уоршелла эффективен для поиска путей между всеми вершинами. Также может использоваться для графов с ребрами отрицательного веса.