```
import pandas as pd
import matplotlib.pyplot as plt

data = pd.read_csv('../Output/data.csv', sep=';')
```

README:

В ходе работы мною были построены несколько графиков. Я не стал писать выводы для каждого из них, поскольку все они интерпретируют теоретические предположения о времени работы алгоритмов. Проанализировав все из них можно сделать выводы, описанные в самом низу этого cell'a. Графики зависимости времени работы определенного алгоритма от числа ребер (самые последние) имеют не самый лучший вид из-за того, что связи с генерацией числа вершин по условию не предусмотрено. Хорошая кривая получилась только для полного графа, поскольку число его ребер коррелирует с числом вершин сильнее, чем у двух других типов графов.

Особенности реализации:

- Для избежания выбросов я замерял время работы каждого из алгоритмов для соответствующего графа 5 раз, полученная сумма значений времени работы усредняется.

Выводы:

- Алгоритм Дейкстры (на приоритетной очереди) - наилучший результат среди всех анализируемых алгоритмов, так как его сложность - O(E*logV)
- Алгоритм Беллмана-Форда занимает второе место по производительности, поскольку его теоретическая сложность - O(VE)
- Алгоритм Флойда-Уоршелла будет работать дольше всех, поскольку асимптотическая сложность = O(V^3)

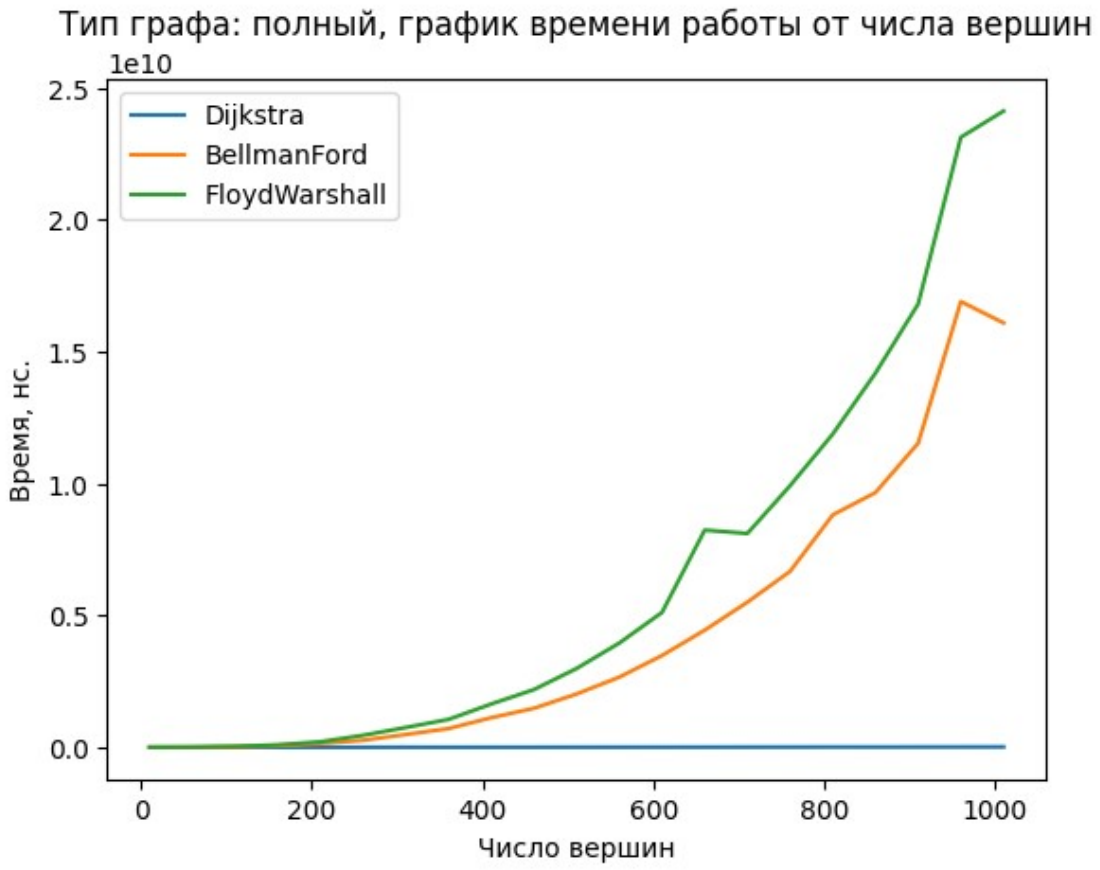
Теоретические знания о сложностях алгоритмов полностью подтверждаются полученными результатами, отраженными на графиках

Агрегированные графики зависимости

Тип графа: полный, время работы от числа вершин

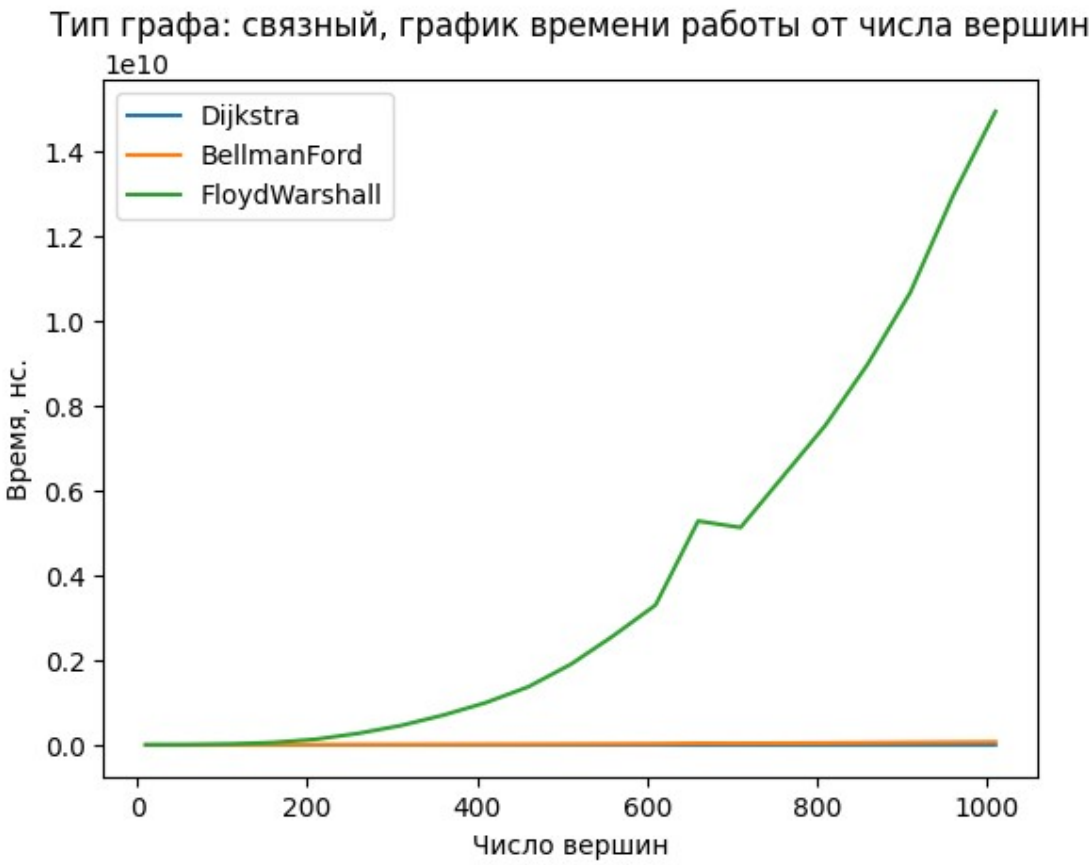
```
complete = data[(data["GraphType"] == "Complete")]
complete_algorithms = complete["Algorithm"].unique()
for algorithm in complete_algorithms:
    current = complete[complete["Algorithm"] == algorithm]
    plt.plot(current["VertexAmount"], current["Time"],
label=algorithm)
```

```
plt.title('Тип графа: полный, график времени работы от числа вершин')
plt.xlabel('Число вершин')
plt.ylabel('Время, нс.')
plt.legend()
plt.show()
```



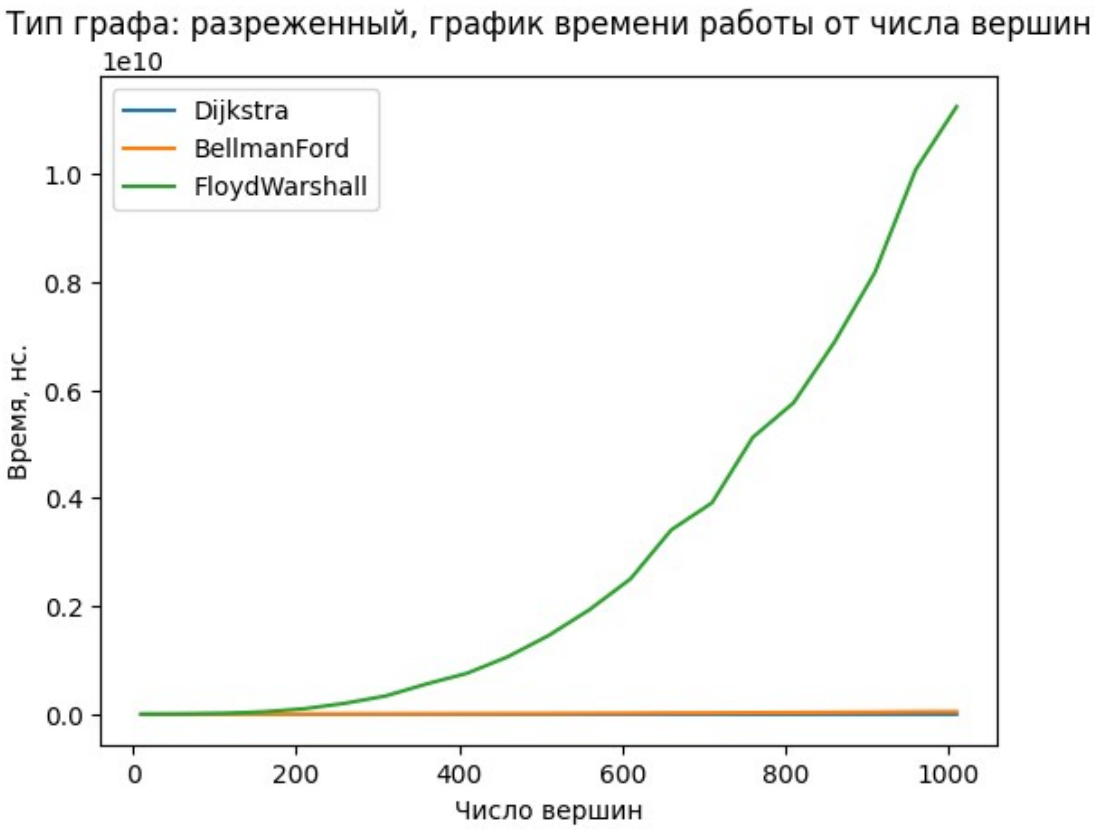
Тип графа: связный, время работы от числа вершин

```
connected = data[(data["GraphType"] == "Connected")]
connected_algorithms = connected["Algorithm"].unique()
for algorithm in connected_algorithms:
    current = connected[connected["Algorithm"] == algorithm]
    plt.plot(current["VertexAmount"], current["Time"],
label=algorithm)

plt.title('Тип графа: связный, график времени работы от числа вершин')
plt.xlabel('Число вершин')
plt.ylabel('Время, нс.')
plt.legend()
plt.show()
```



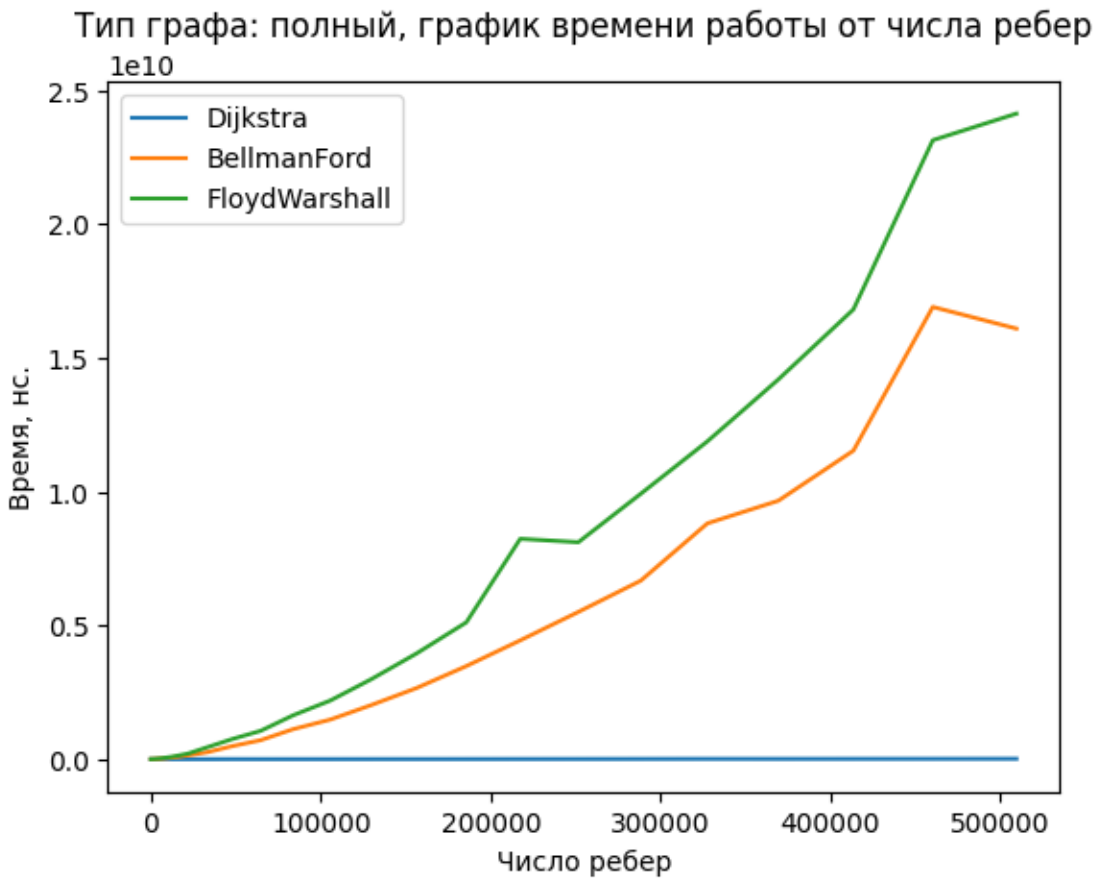
Тип графа: разреженный, время работы от числа вершин

```
sparse = data[(data["GraphType"] == "Sparse")]
sparse_algorithms = sparse["Algorithm"].unique()
for algorithm in sparse_algorithms:
    current = sparse[sparse["Algorithm"] == algorithm]
    plt.plot(current["VertexAmount"], current["Time"],
label=algorithm)

plt.title('Тип графа: разреженный, график времени работы от числа
вершин')
plt.xlabel('Число вершин')
plt.ylabel('Время, нс.')
plt.legend()
plt.show()
```



Тип графа: полный, время работы от числа ребер

```
complete = data[(data["GraphType"] == "Complete")]
complete_algorithms = complete["Algorithm"].unique()
for algorithm in complete_algorithms:
    current = complete[complete["Algorithm"] == algorithm]
    plt.plot(current["EdgeAmount"], current["Time"], label=algorithm)

plt.title('Тип графа: полный, график времени работы от числа ребер')
plt.xlabel('Число ребер')
plt.ylabel('Время, нс.')
plt.legend()
plt.show()
```

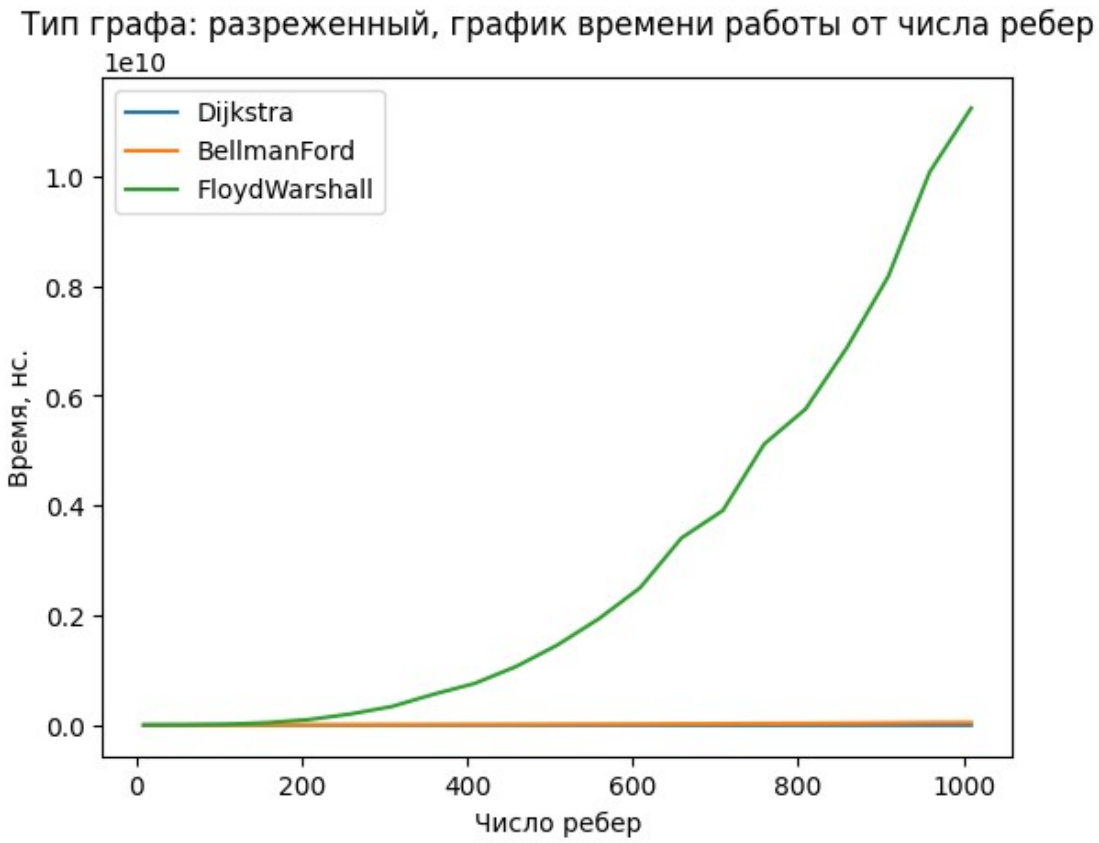


Тип графа: связный, время работы от числа ребер

```
connected = data[(data["GraphType"] == "Connected")]
connected_algorithms = connected["Algorithm"].unique()
for algorithm in connected_algorithms:
    current = connected[connected["Algorithm"] == algorithm]
    plt.plot(current["EdgeAmount"], current["Time"], label=algorithm)

plt.title('Тип графа: связный, график времени работы от числа ребер')
plt.xlabel('Число ребер')
plt.ylabel('Время, нс.')
plt.legend()
plt.show()
```

Тип графа: разреженный, время работы от числа ребер

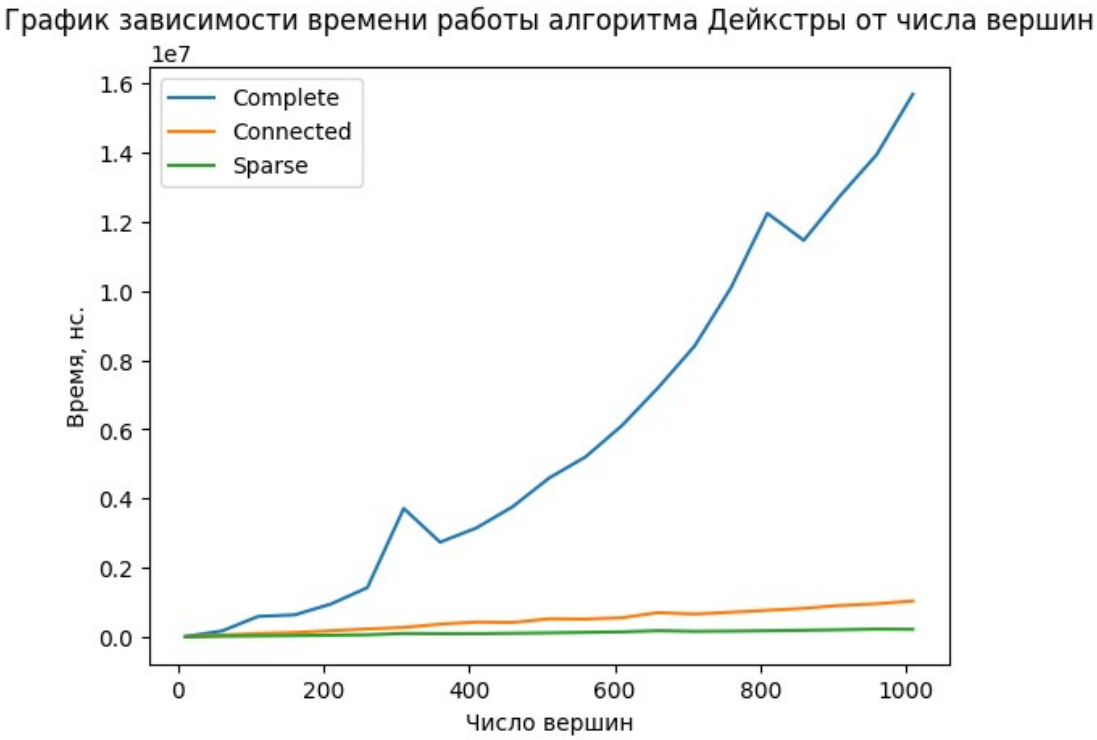
```
sparse = data[(data["GraphType"] == "Sparse")]
sparse_algorithms = sparse["Algorithm"].unique()
for algorithm in sparse_algorithms:
    current = sparse[sparse["Algorithm"] == algorithm]
    plt.plot(current["EdgeAmount"], current["Time"], label=algorithm)

plt.title('Тип графа: разреженный, график времени работы от числа
ребер')
plt.xlabel('Число ребер')
plt.ylabel('Время, нс.')
plt.legend()
plt.show()
```



Графики времени работы каждого алгоритма для отдельных типов графов

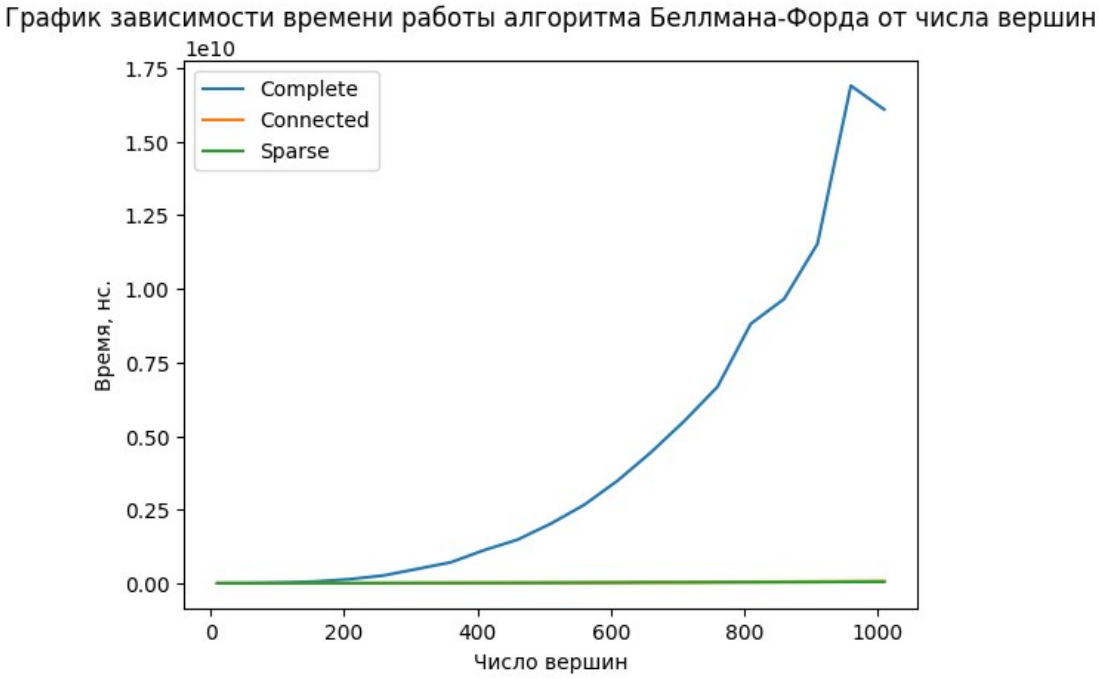
Алгоритм Дейкстры, время работы от числа вершин

```
dijkstra = data[(data["Algorithm"] == "Dijkstra")]
graph_types = dijkstra["GraphType"].unique()
for type_d in graph_types:
    current = dijkstra[dijkstra["GraphType"] == type_d]
    plt.plot(current["VertexAmount"], current["Time"], label=type_d)

plt.title('График зависимости времени работы алгоритма Дейкстры от числа вершин')
plt.xlabel('Число вершин')
plt.ylabel('Время, нс.')
plt.legend()
plt.show()
```



Алгоритм Беллмана-Форда, время работы от числа вершин

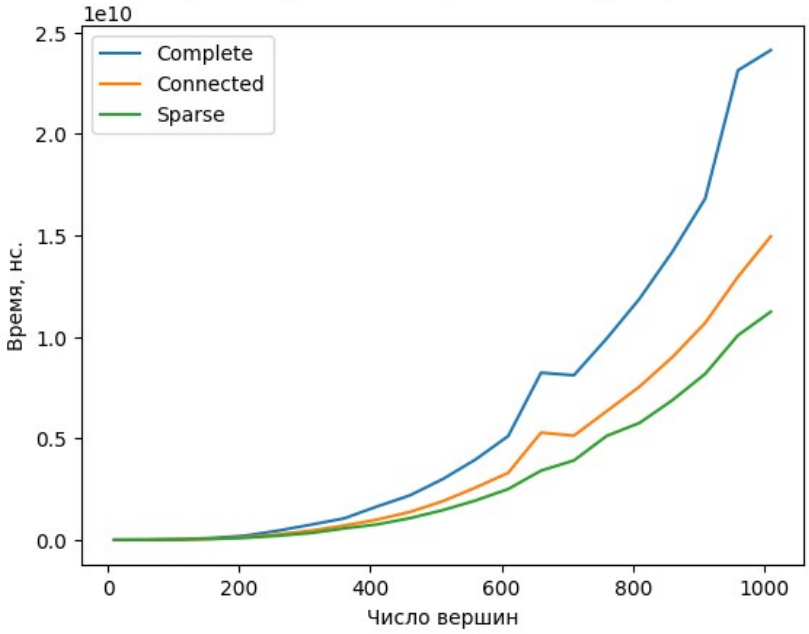
```
bellman_ford = data[(data["Algorithm"] == "BellmanFord")]
graph_types = bellman_ford["GraphType"].unique()
for type_bf in graph_types:
    current = bellman_ford[bellman_ford["GraphType"] == type_bf]
    plt.plot(current["VertexAmount"], current["Time"], label=type_bf)

plt.title('График зависимости времени работы алгоритма Беллмана-Форда
от числа вершин')
plt.xlabel('Число вершин')
plt.ylabel('Время, нс.')
plt.legend()
plt.show()
```



Алгоритм Флойда-Уоршелла, время работы от числа вершин

```
floyd_warshall = data[(data["Algorithm"] == "FloydWarshall")]
graph_types = floyd_warshall["GraphType"].unique()
for type_fw in graph_types:
    current = floyd_warshall[floyd_warshall["GraphType"] == type_fw]
    plt.plot(current["VertexAmount"], current["Time"], label=type_fw)

plt.title('График зависимости времени работы алгоритма Флойда-Уоршелла
от числа вершин')
plt.xlabel('Число вершин')
plt.ylabel('Время, нс.')
plt.legend()
plt.show()
```

График зависимости времени работы алгоритма Флойда-Уоршелла от числа вершин

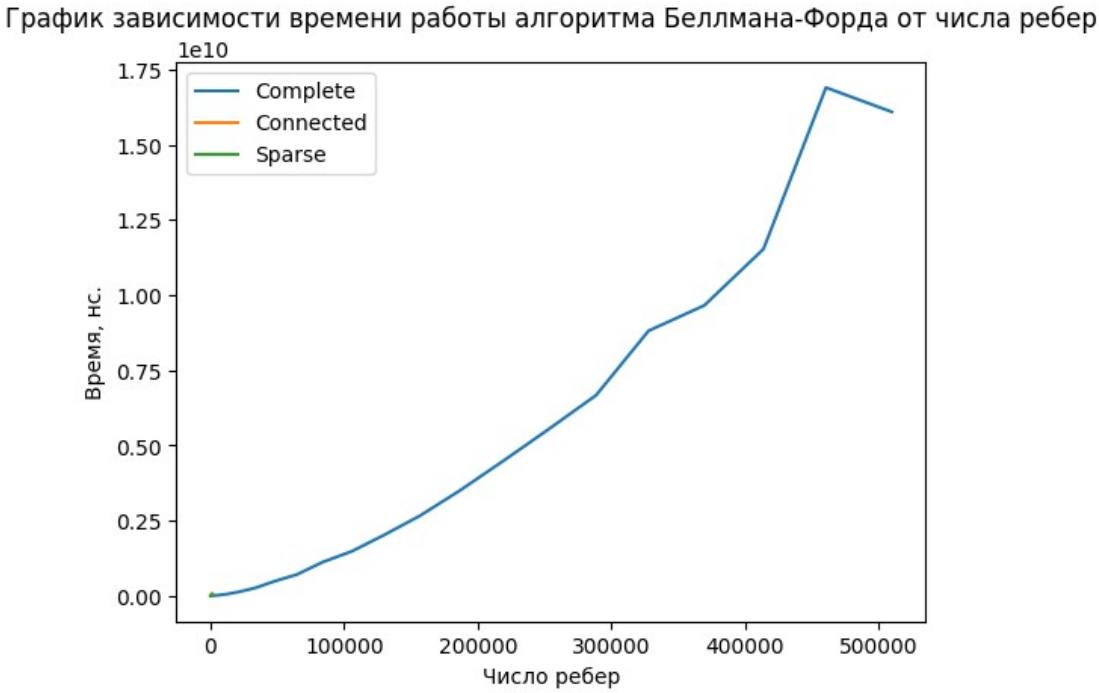


Алгоритм Дейкстры, время работы от числа ребер

```
dijkstra = data[(data["Algorithm"] == "Dijkstra")]
graph_types = dijkstra["GraphType"].unique()
for type_d in graph_types:
    current = dijkstra[dijkstra["GraphType"] == type_d]
    plt.plot(current["EdgeAmount"], current["Time"], label=type_d)

plt.title('График зависимости времени работы алгоритма Дейкстры от числа ребер')
plt.xlabel('Число ребер')
plt.ylabel('Время, нс.')
plt.legend()
plt.show()
```

Алгоритм Беллмана-Форда, время работы от числа ребер

```
bellman_ford = data[(data["Algorithm"] == "BellmanFord")]
graph_types = bellman_ford["GraphType"].unique()
for type_bf in graph_types:
    current = bellman_ford[bellman_ford["GraphType"] == type_bf]
    plt.plot(current["EdgeAmount"], current["Time"], label=type_bf)

plt.title('График зависимости времени работы алгоритма Беллмана-Форда
от числа ребер')
plt.xlabel('Число ребер')
plt.ylabel('Время, нс.')
plt.legend()
plt.show()
```



Алгоритм Флойда-Уоршелла, время работы от числа ребер

```
floyd_warshall = data[(data["Algorithm"] == "FloydWarshall")]
graph_types = floyd_warshall["GraphType"].unique()
for type_fw in graph_types:
    current = floyd_warshall[floyd_warshall["GraphType"] == type_fw]
    plt.plot(current["EdgeAmount"], current["Time"], label=type_fw)

plt.title('График зависимости времени работы алгоритма Флойда-Уоршелла
от числа ребер')
plt.xlabel('Число ребер')
plt.ylabel('Время, нс.')
plt.legend()
plt.show()
```

График зависимости времени работы алгоритма Флойда-Уоршелла от числа ребер