КДЗ №3

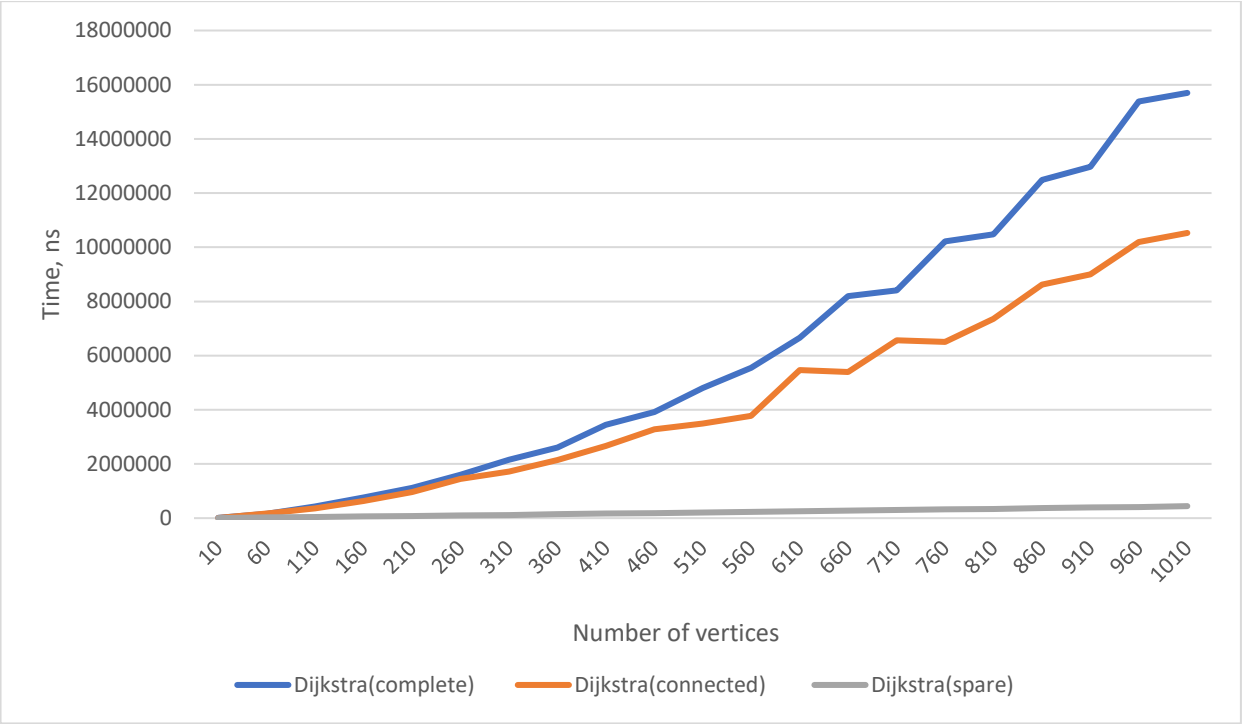
ОТЧЁТ

ПО ИССЛЕДОВАНИЮ

алгоритмов поиска кратчайшего пути в графе

Выполнил:
Студент БПИ219
Гладких Иван

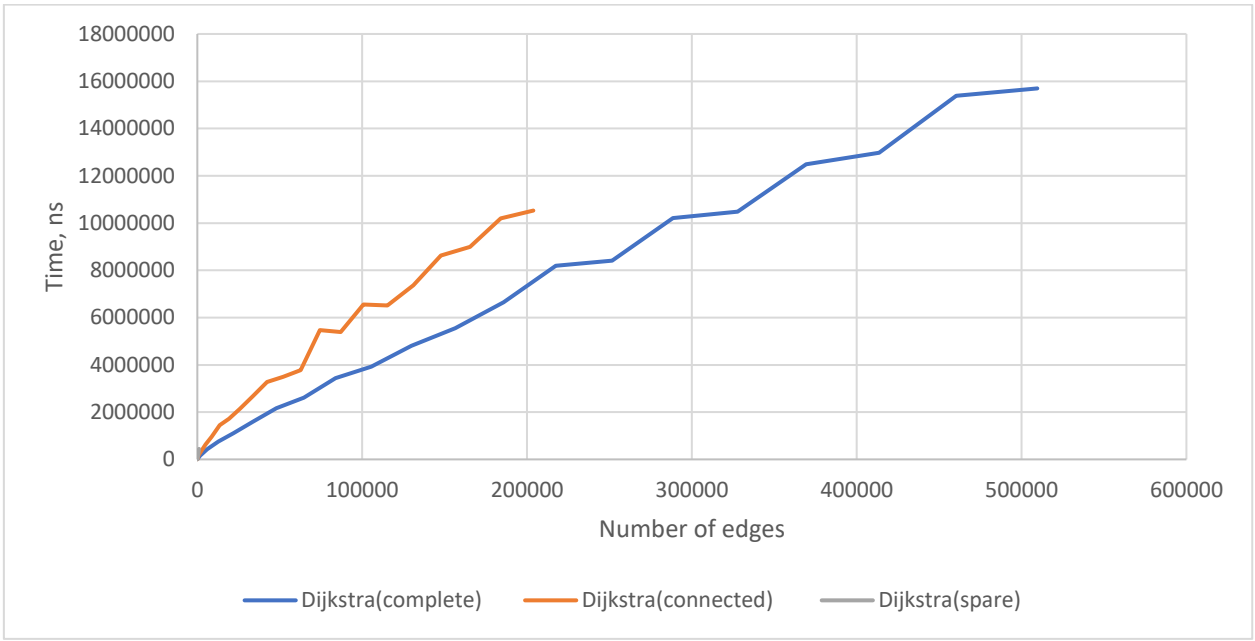
Алгоритм Дейкстры

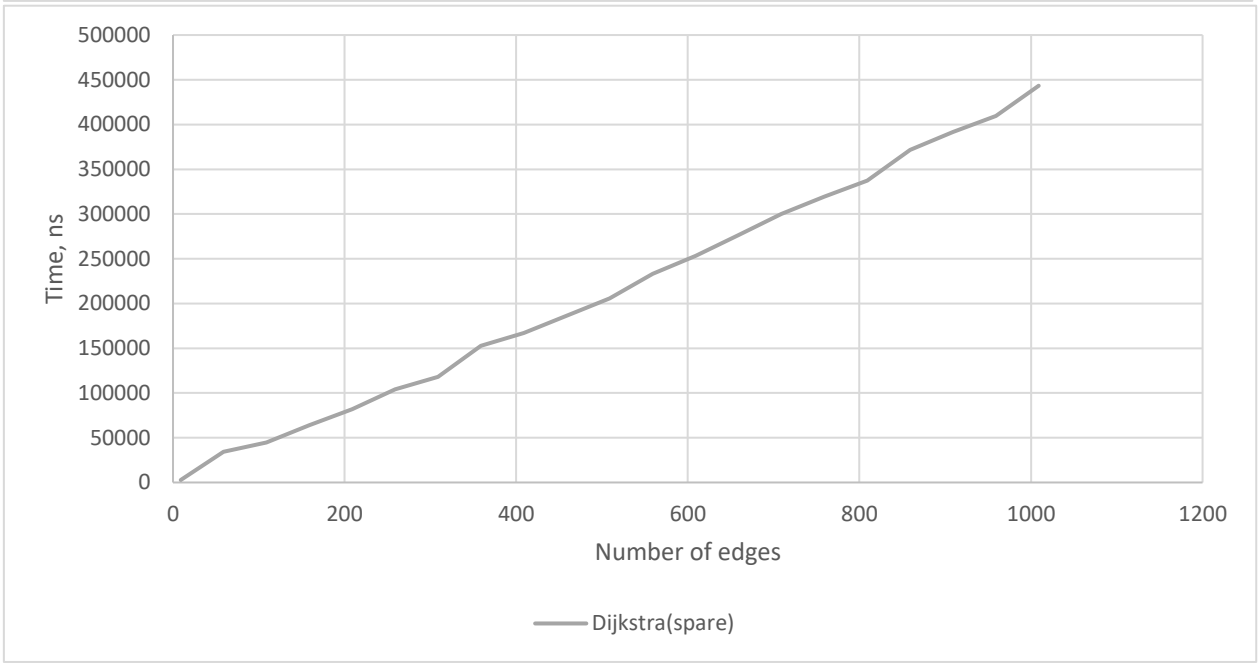


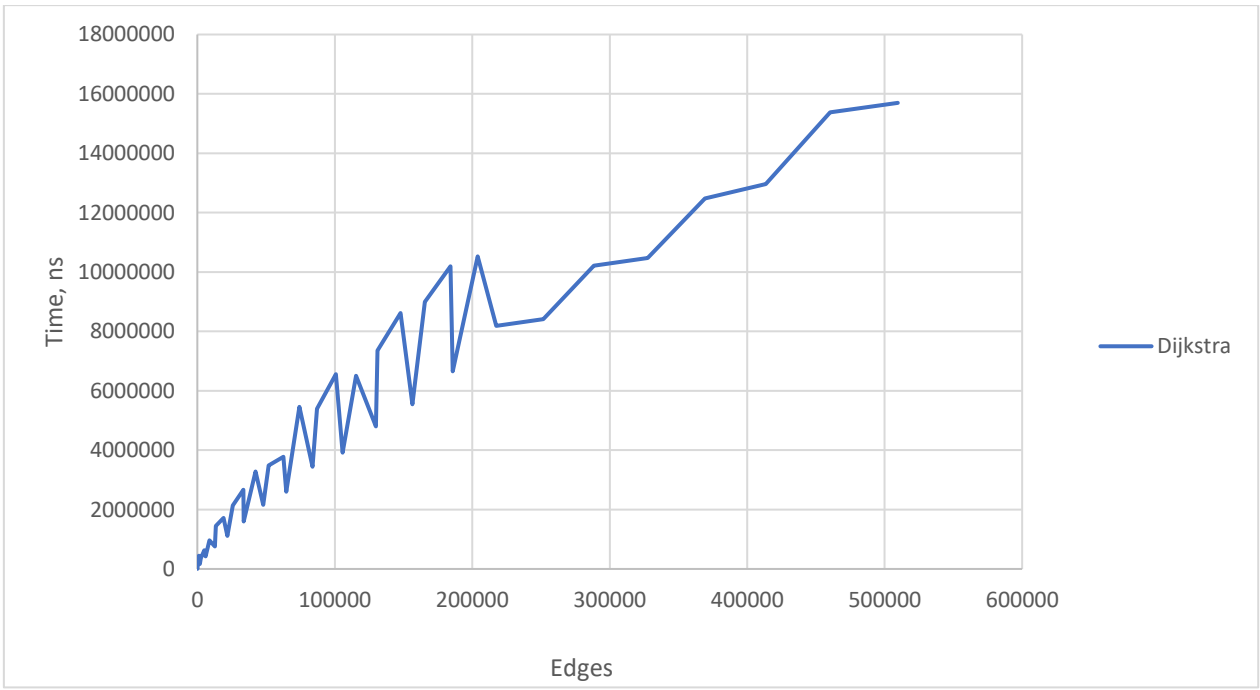
Согласно нашим теоретическим знаниям, алгоритм Дейкстры (на основе бинарной кучи) имеет сложность O(m*log n) и это можно увидеть на графике зависимости времени от числа вершин: быстрее всего алгоритм работает на разреженном графе, т.к. m ~ n (в данном случае это дерево ⇒ m = n – 1), из-за чего сложность получается O(n*log n). Хуже всего работает на полном графе, т.к. m ~ n^2, и сложность получается O(n^2 * log n). Чуть лучше на связном графе: он имеет коэффициент плотности C=0.4-0.5, то есть m = C * $m_{полн}$ ~ C*n^2 ~ n^2. Таким образом, получается та же асимптотическая сложность, но с меньшей константой.

График зависимости времени от числа ребер показывает аналогичный результат – O(m * log m), но с разными константами.

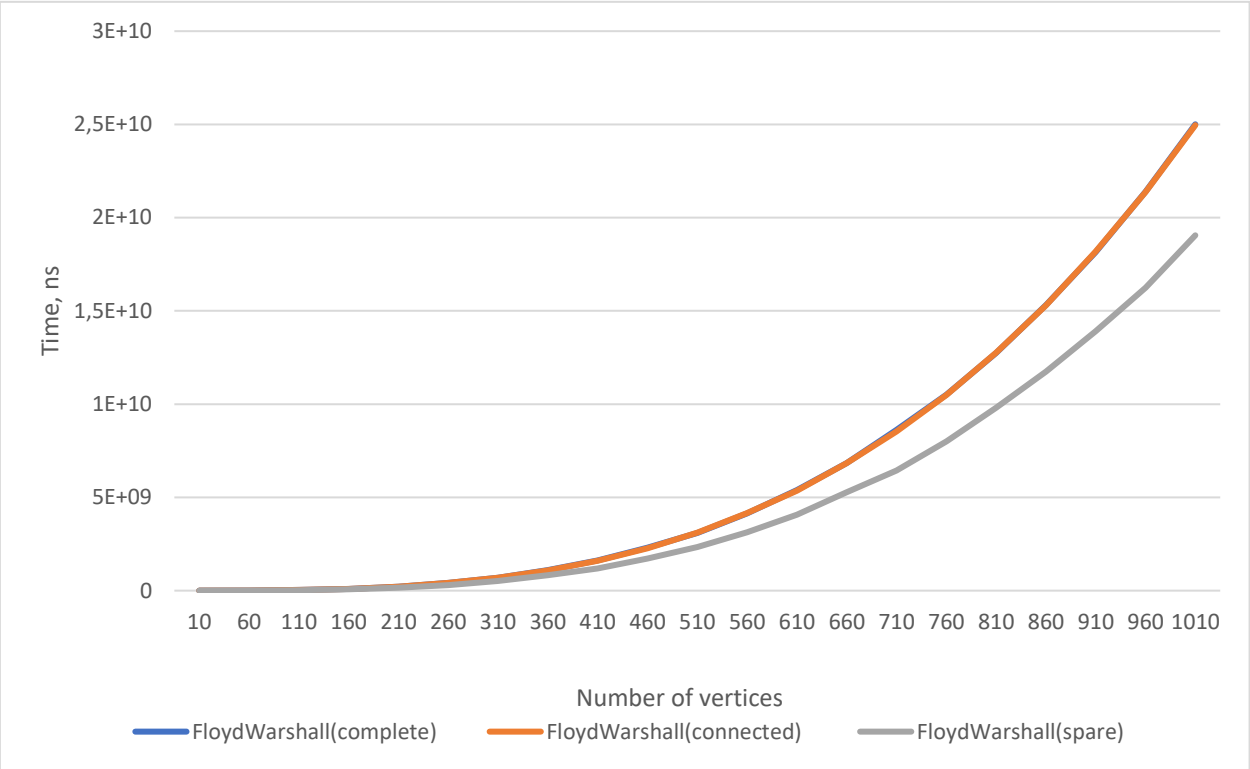
Колебания на последнем графике (где отсутствует разделение на типы графов) обусловлены чередованием значений из разных графов, что вызывает резкие скачки числа вершин.


18000000
16000000
14000000
12000000
10000000
8000000
6000000
4000000
2000000
0
Time, ns
0
100000
200000
300000
400000
500000
600000
Number of edges
Dijkstra(complete)
Dijkstra(connected)
Dijkstra(spare)


500000
450000
400000
350000
300000
250000
200000
150000
100000
50000
0
Time, ns
0
200
400
600
800
1000
1200
Number of edges
Dijkstra(spare)

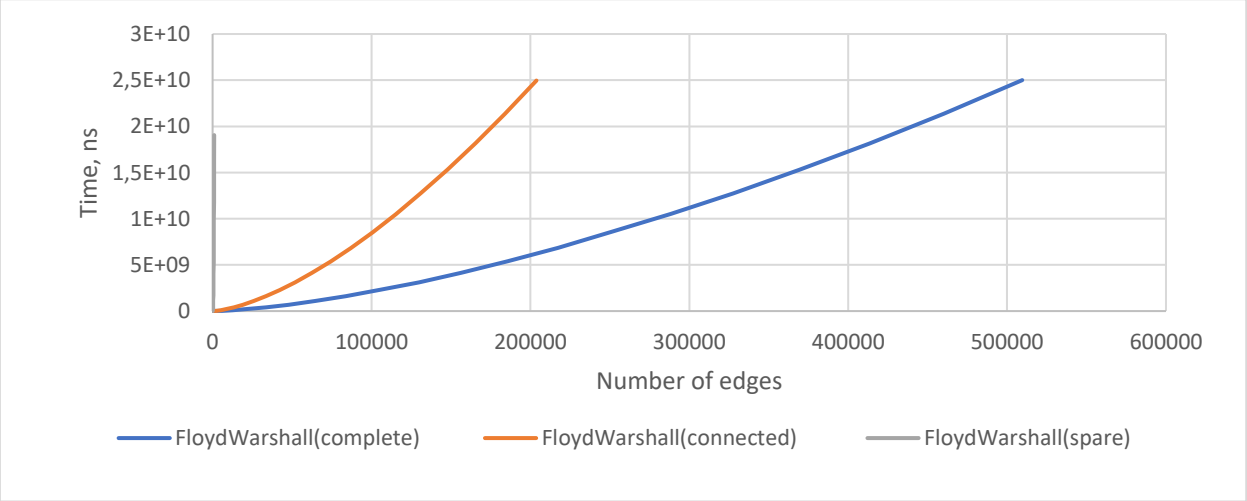

18000000
16000000
14000000
12000000
10000000
8000000
6000000
4000000
2000000
0
Time, ns
0
100000
200000
300000
400000
500000
600000
Edges
Dijkstra

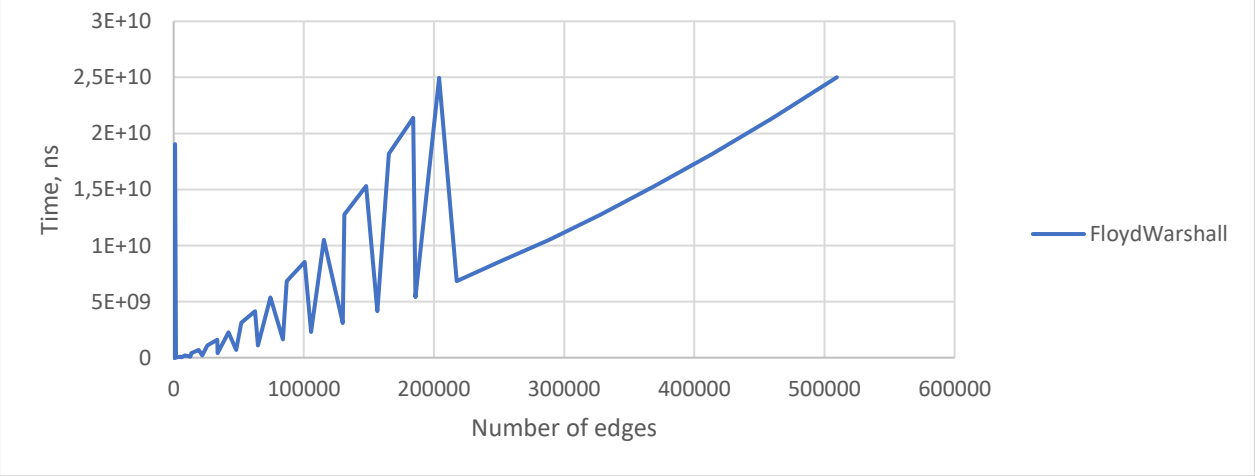
Алгоритм Флойда-Уоршелла



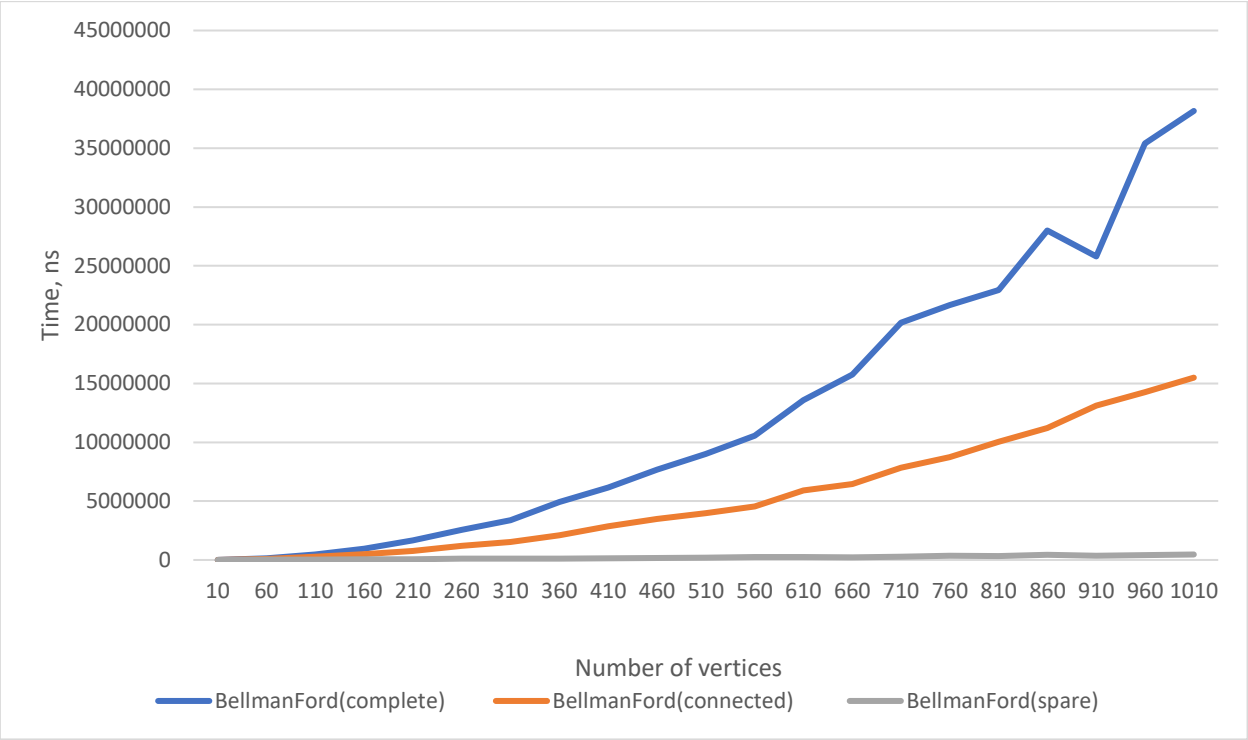
Алгоритм ищет расстояние от всех вершин до всех и имеет одинаковую асимптотическую сложность для всех графов – $O(n^3)$. В случае разреженного графа константа получается чуть меньше, т.к. для пары вершин, не имеющих общее ребро, (а таких подавляющее большинство) сравнение с минимумом не происходит.

Скачки в графике так же обслувлено чередованием разных графов – скачков числа вершин.



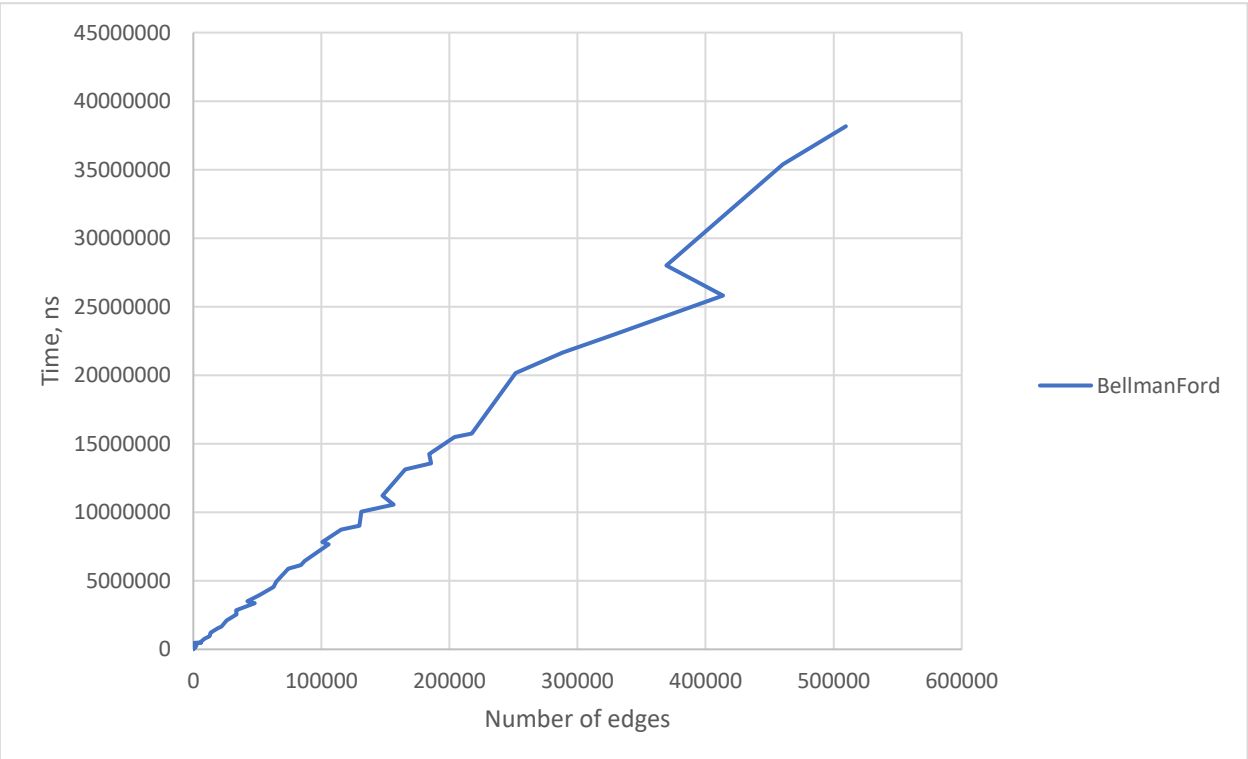


Алгоритм Беллмана-Форда

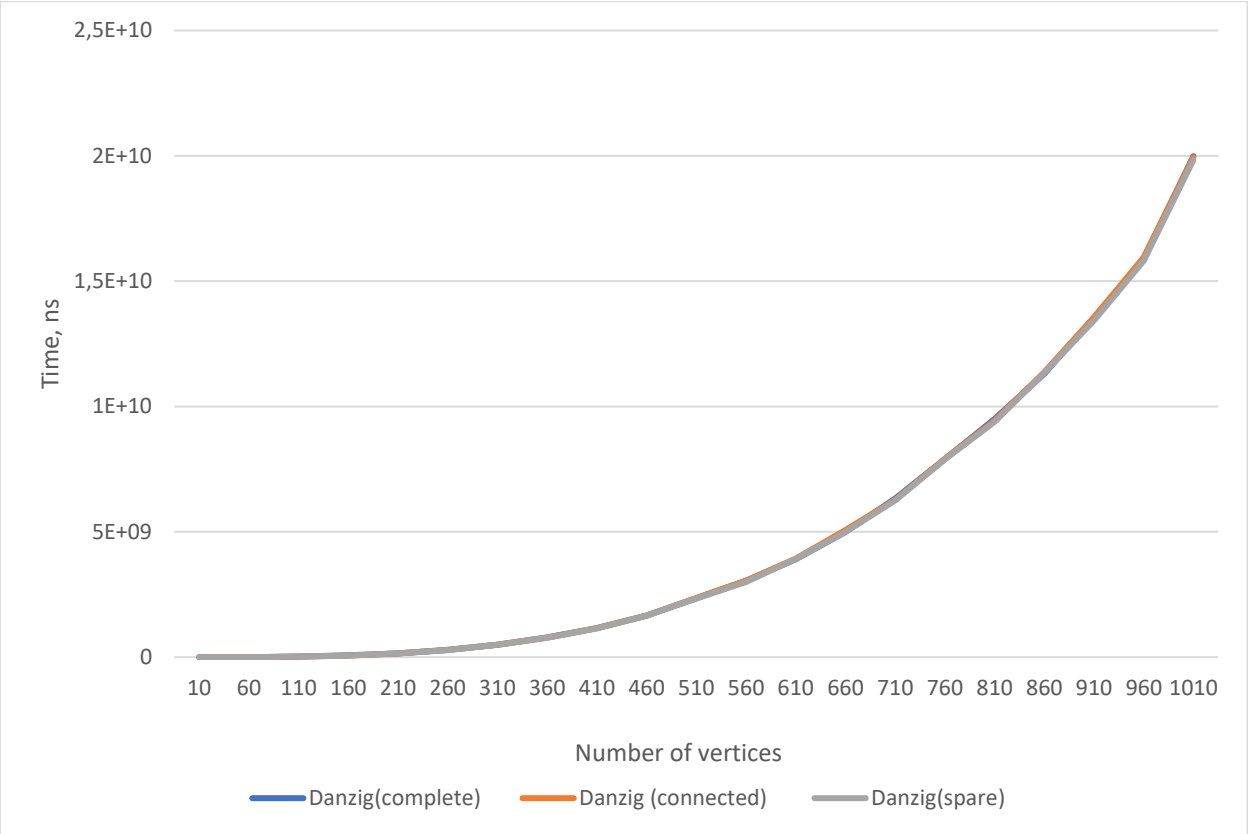


Алгоритм имеет асимптотическую сложность O(mn).
В случае разреженного графа m ~ n, из-за чего сложность приобретает вид $O(n^2)$ – лучший результат. Хуже всего алгоритм отработал на полном графе, т.к. m ~ n^2, сложность приобретает вид $O(n^3)$. В случае связного графа числе ребер всё еще пропорционально квадрату числа вершин, однако константа меньше, из-за чего на нем алгоритм и отрабатывает чуть лучше.

График зависимости времени от числа ребёр в данном случае колеблется меньше (в отличие от предыдущих алгоритмов), т.к. число ребер имеет бóльшее значение.

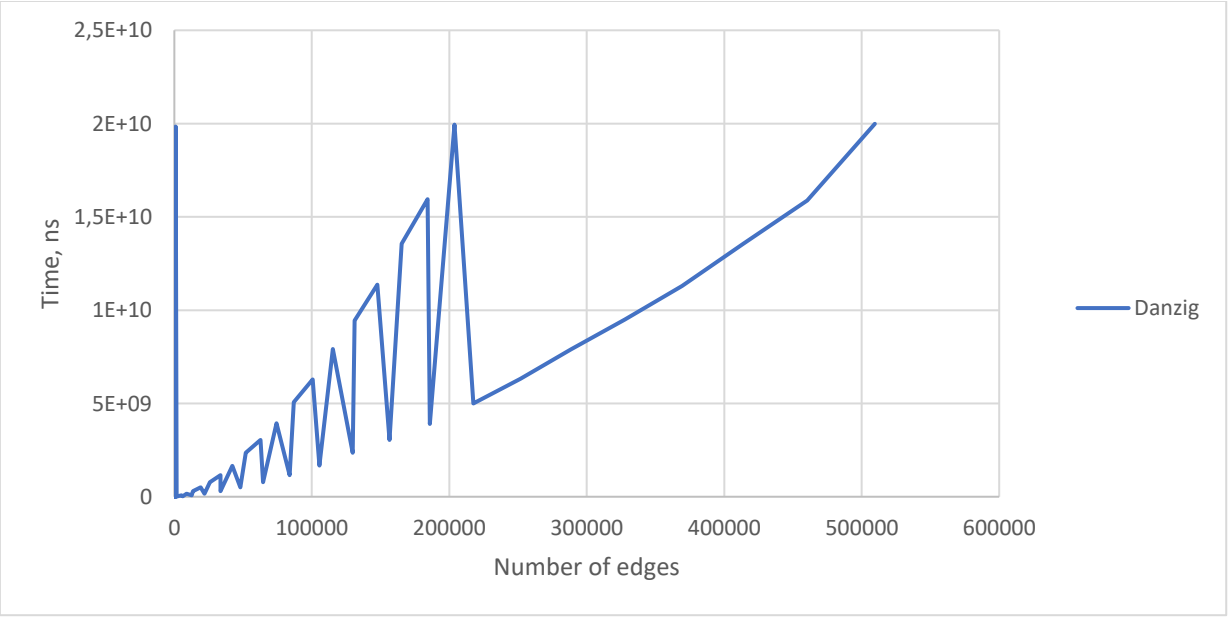


Алгоритм Данцига



Алгоритм Данцига так же как и алгоритм Флойда-Уоршелла вычисляет расстояние от всех вершин до всех остальных и имеет ту же асимптотическую сложность $O(n^3)$. Однако число ребёр на время работы не влияет (а значит и вид графа), т.к. вне зависимости от существования ребра между парой вершин происходит сравнение с минимумом.

График зависимости времени работы от числа ребер так колеблется, потому что время работы не зависит непосредственно от числа ребер, только от числа вершин, а значения времени чередуются для различных видов графа.

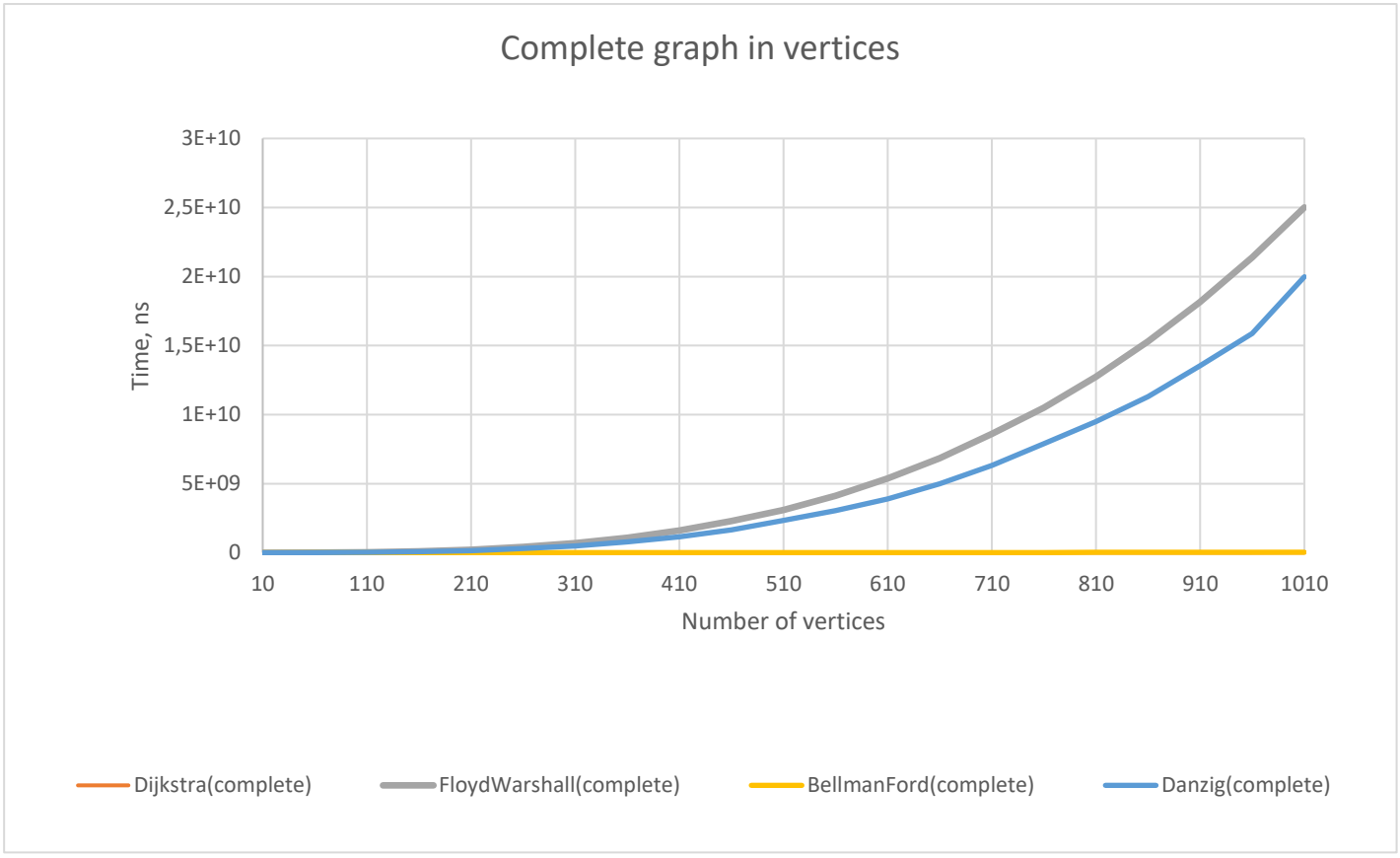


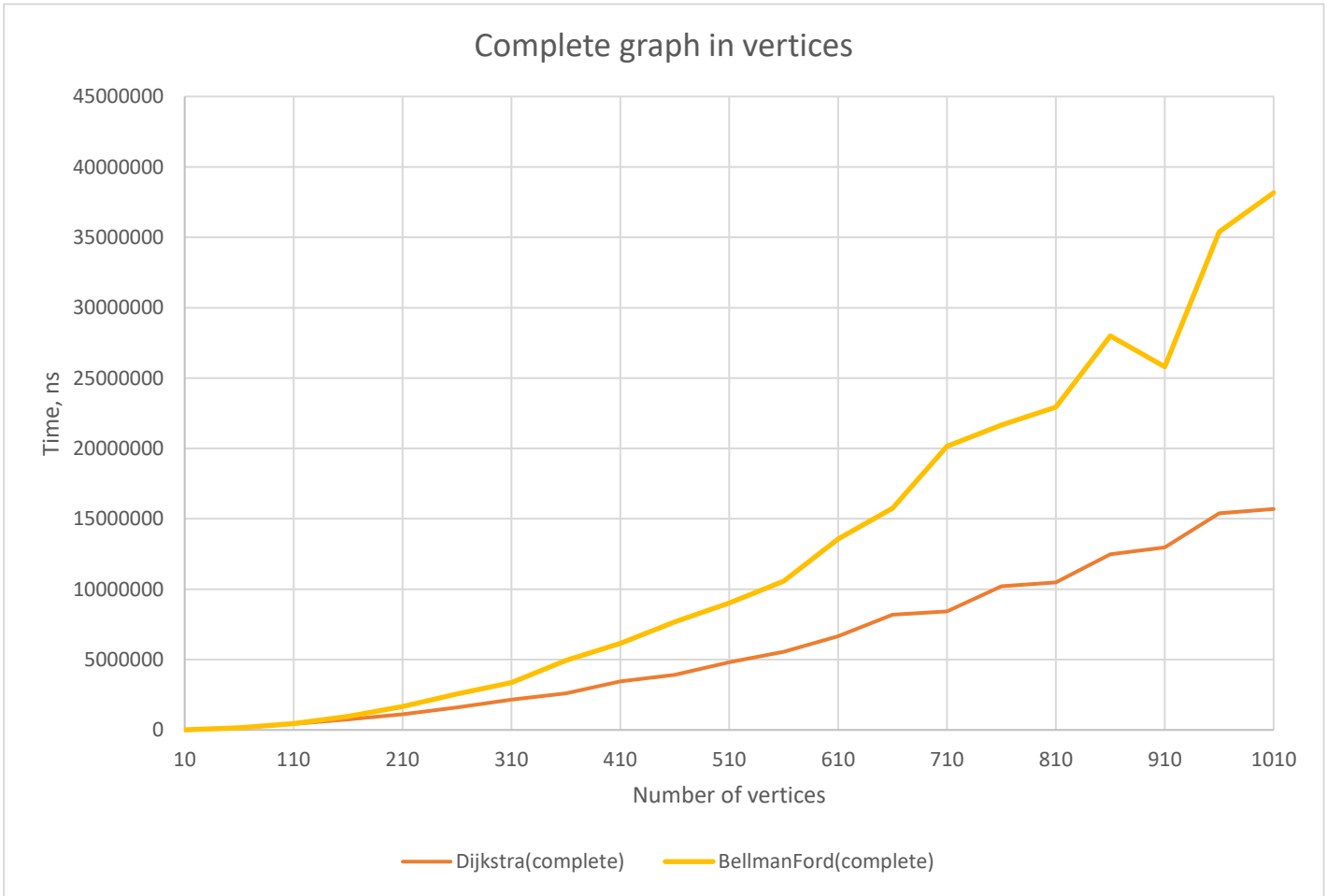
Полный граф

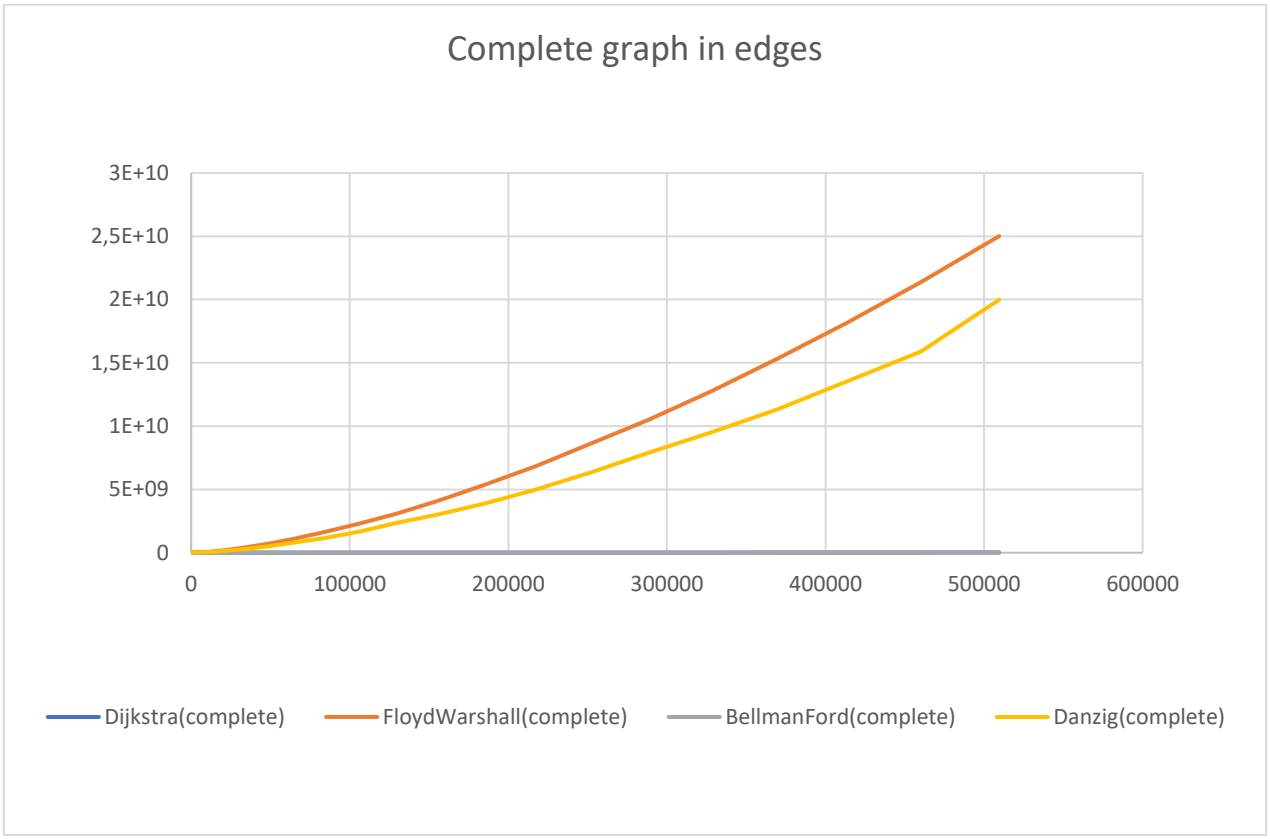
Подтверждая наши теоретические знания, быстрее всего на полном графе оказывается алгоритм Дейкстры, затем алгоритм Беллмана-Форда, т.к. алгоритм Дейкстры написан на основе бинарной кучи, он имеет асимптотическую сложность $O(n^2*\log n)$, Беллмана-Форда – $O(n^3)$, причем реализация алгоритма Дейкстры на основе массива так же имела бы сложность $O(n^3)$. Алгоритм Беллмана-Форда при этом имеет своё преимущество – возможность работы с отрицательными весами (и циклами), что является недопустимым в алгоритме Дейкстры: дойдя до вершины алгоритм Дейкстры фиксирует путь до неё как минимальный, и в дальнейшем он уже не сможет измениться, хотя с помощью ребёр отрицательного веса можно сократить путь; Беллман-Форд такой проблемы не имеет, т.к. по сути рассматривает все пути длины от 1 до n-1 и выбирает минимальный.

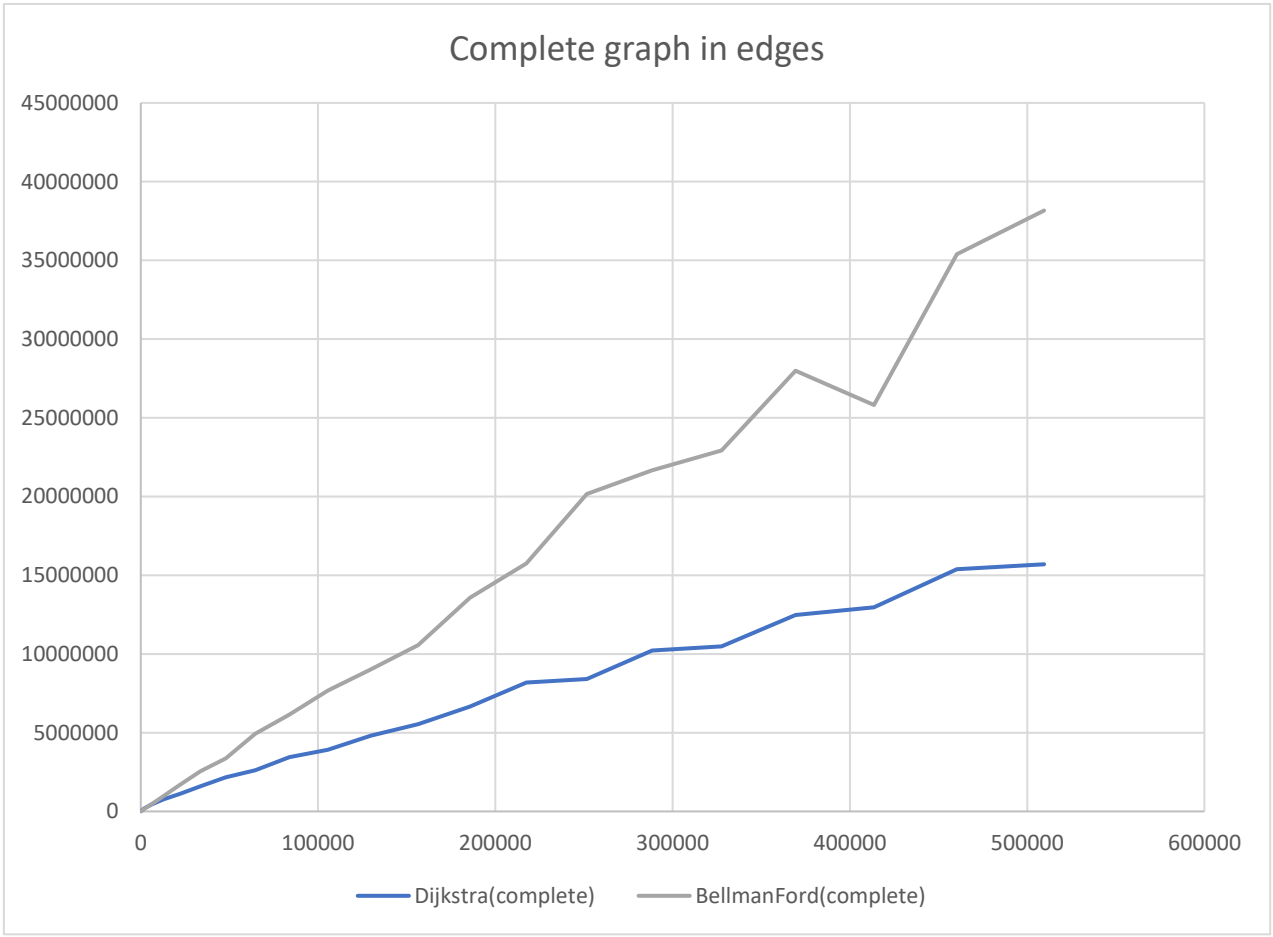
Очевидно, алгоритмы Флойда-Уоршелла и Данцига показали худшие результаты: они предназначены для поиска расстояния от всех вершин до всех, в отличие от алгоритмов Дейкстры и Беллмана-Форда, которые ищут расстояние от одной вершины до всех остальных. При этом алгоритм Данцига показал результат чуть лучше, т.к. на каждой из n итераций вычисляется i^2 значений матрицы (i – текущая итерация), а в алогритме Флойда-Уоршелла на каждой из n итераций обновляется вся матрица, т.е. n^2 значений. Таким образом, алгоритмы Флойда-Уоршелла и Данцига имеют одинаковую асимптотическую сложность, но алгоритм Данцига имеет меньшую константу. Хотя Беллман-Форд на полном графе имеет такую же сложность $O(n^3)$, на практике он работает гораздо быстрее, т.к. имеет условие досрочного выхода (если не найден ни один более оптимальный путь некоторой длины, значит таких путей большей длины тоже не будет).

Т.к. число ребер пропорционально числу вершин (в квадрате), резких скачков не происходит, поэтому графики зависимости времени работы от числа ребер не сильно колеблются.


Complete graph in vertices
3E+10
2,5E+10
2E+10
1,5E+10
1E+10
5E+09
0
Time, ns
10
110
210
310
410
510
610
710
810
910
1010
Number of vertices
Dijkstra(complete)
FloydWarshall(complete)
BellmanFord(complete)
Danzig(complete)

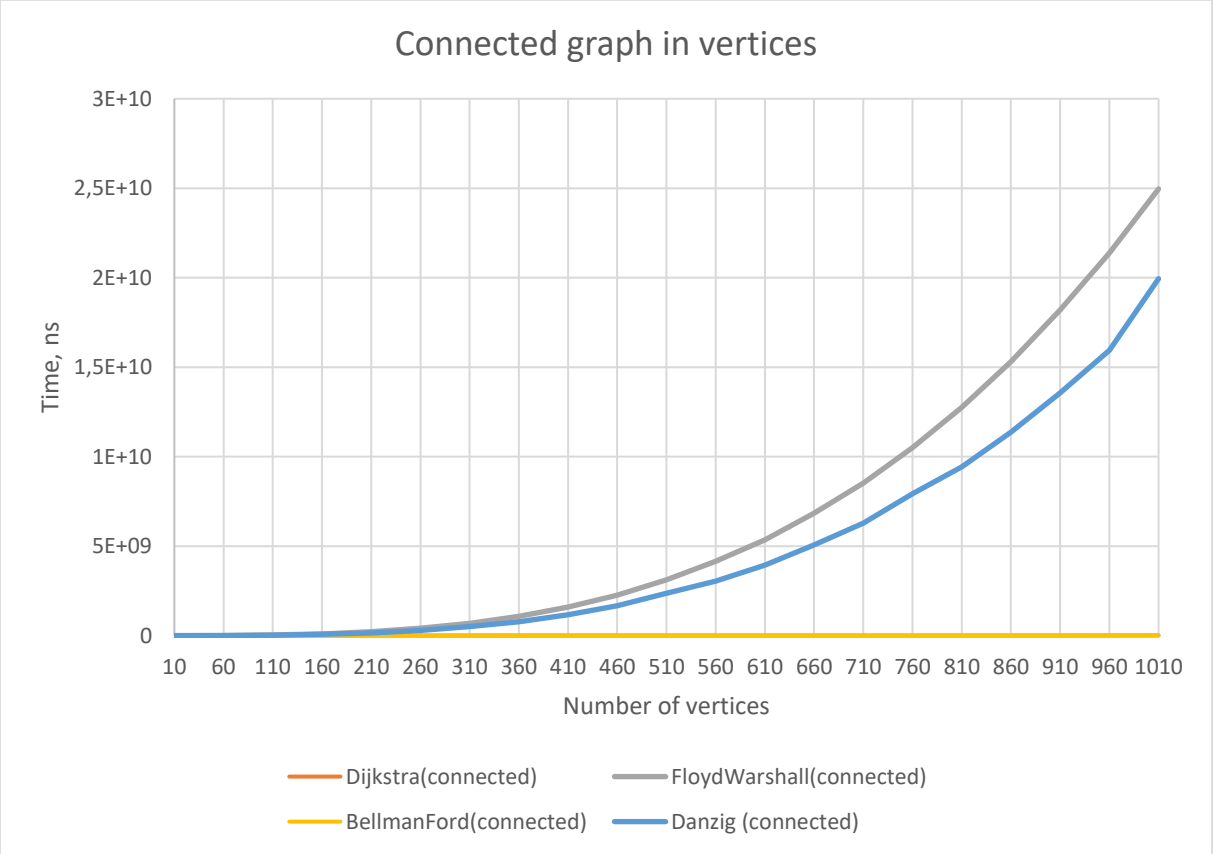

Complete graph in vertices
45000000
40000000
35000000
30000000
25000000
20000000
15000000
10000000
5000000
0
Time, ns
10
110
210
310
410
510
610
710
810
910
1010
Number of vertices
Dijkstra(complete)
BellmanFord(complete)

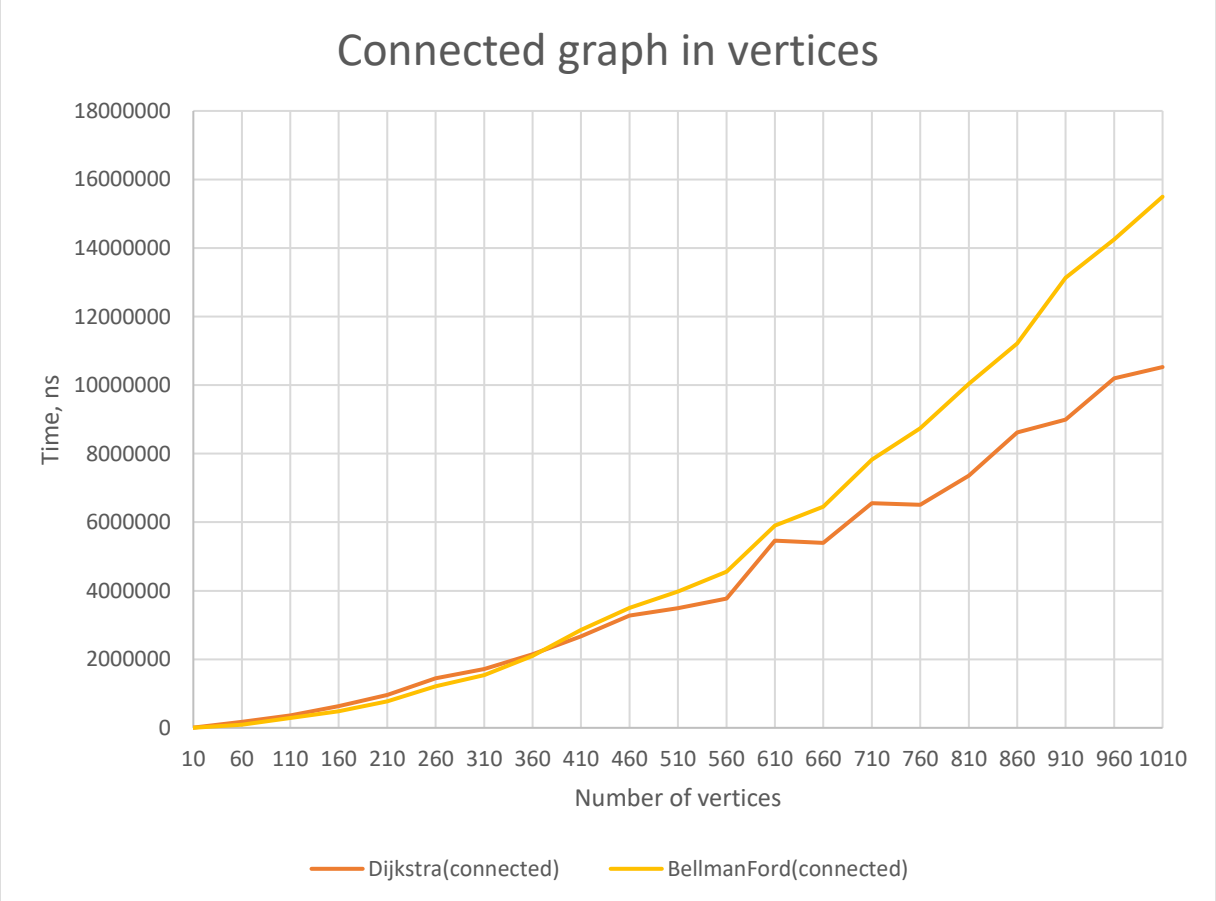

Complete graph in edges
3E+10
2,5E+10
2E+10
1,5E+10
1E+10
5E+09
0
0
100000
200000
300000
400000
500000
600000
Dijkstra(complete)
FloydWarshall(complete)
BellmanFord(complete)
Danzig(complete)

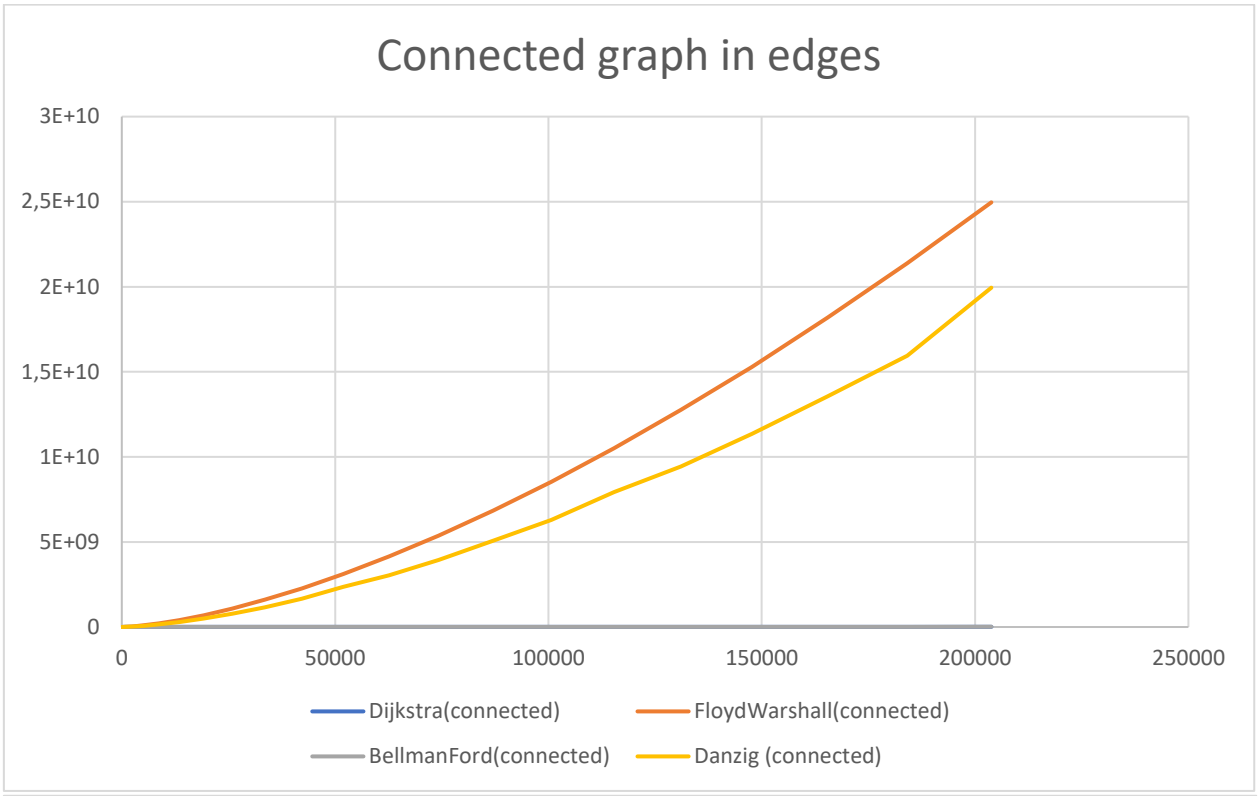

Complete graph in edges
45000000
40000000
35000000
30000000
25000000
20000000
15000000
10000000
5000000
0
0
100000
200000
300000
400000
500000
600000
Dijkstra(complete)
BellmanFord(complete)

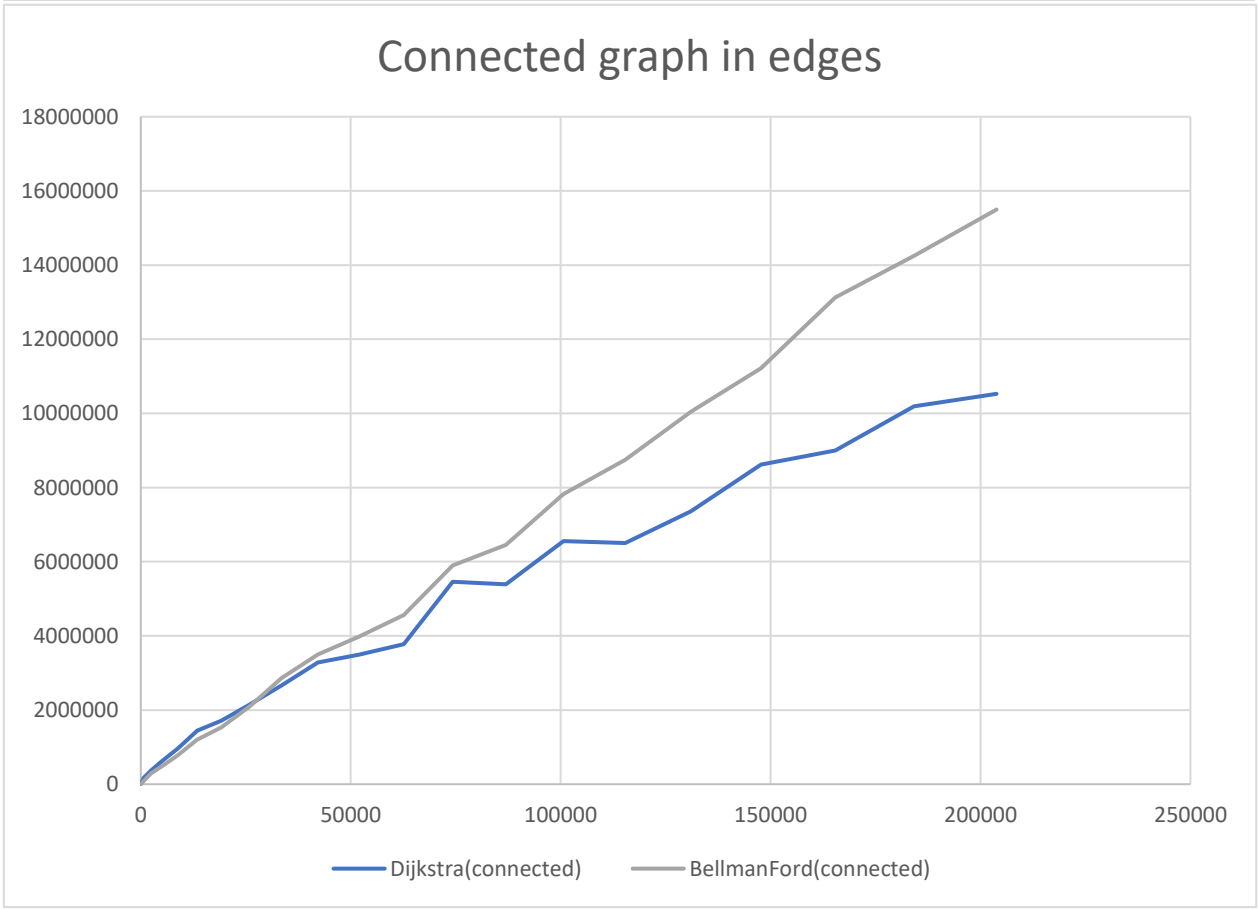
Связный граф

Результаты полученные при анализе связного графа практически аналогичны полному графу, т.к. число ребер пропорционально числу ребер в полном графе, но на небольших значениях количества вершин Беллман-Форд отрабатывает быстрее Дейкстры, т.к. уменьшение числа ребер в большей степени повлияло именно на алгоритм Беллмана-Форда (т.к. БФ ~ m, а Дейкстра ~ log m).



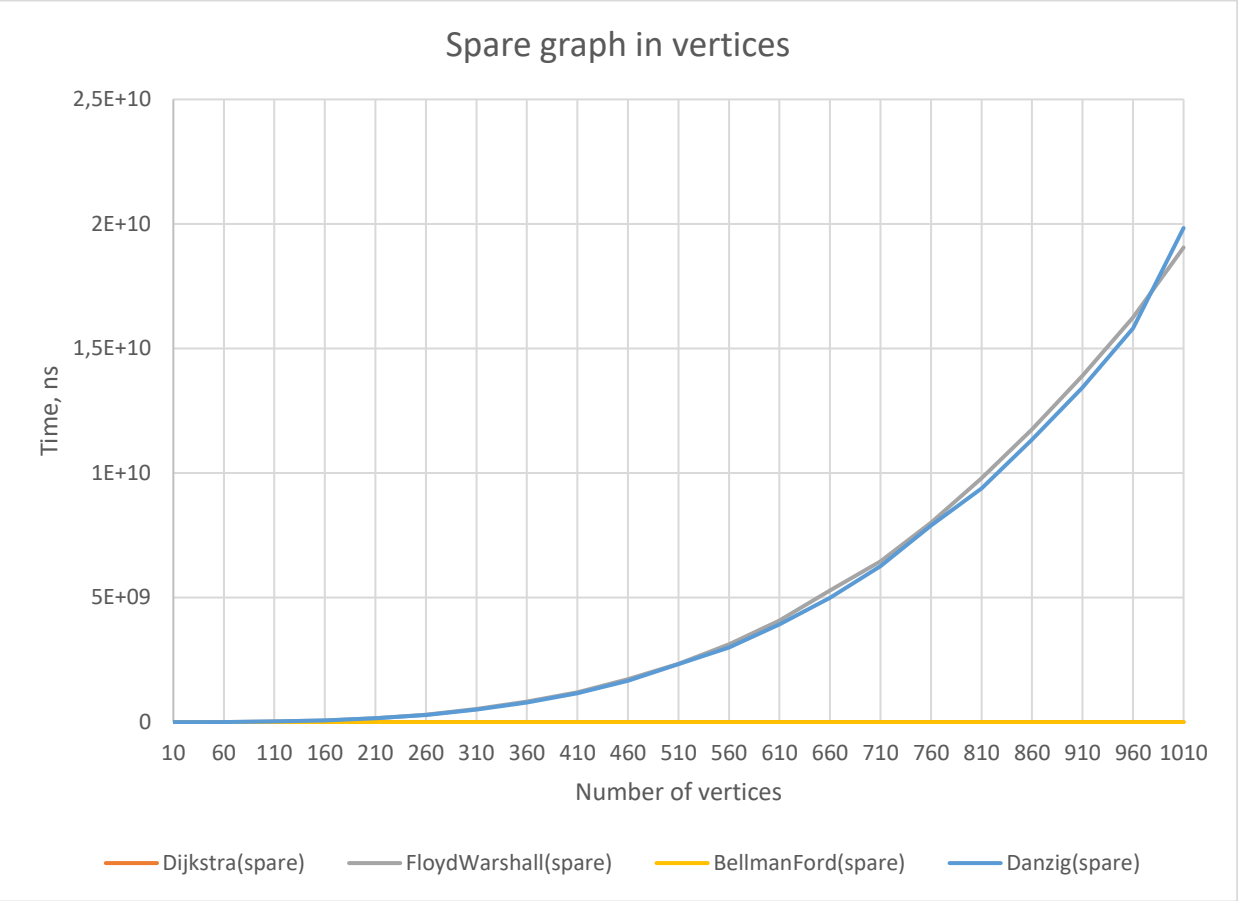


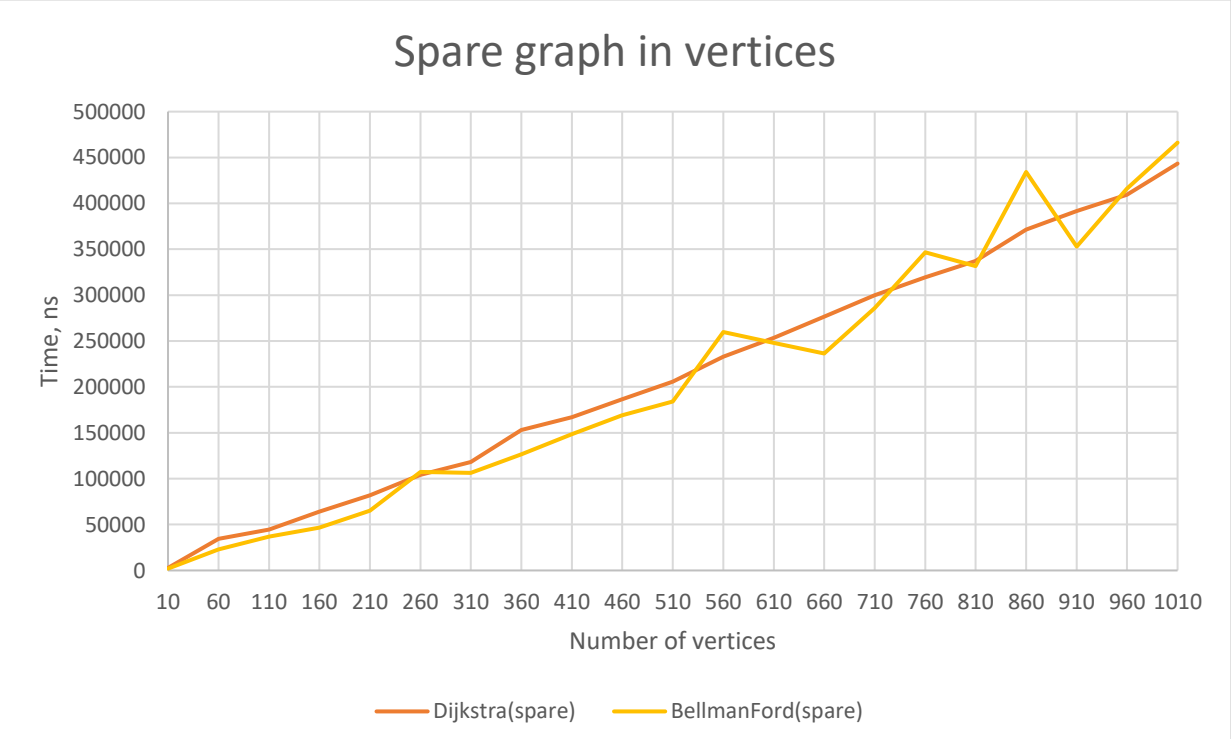

Connected graph in edges
3E+10
2,5E+10
2E+10
1,5E+10
1E+10
5E+09
0
0
50000
100000
150000
200000
250000
Dijkstra(connected)
FloydWarshall(connected)
BellmanFord(connected)
Danzig (connected)

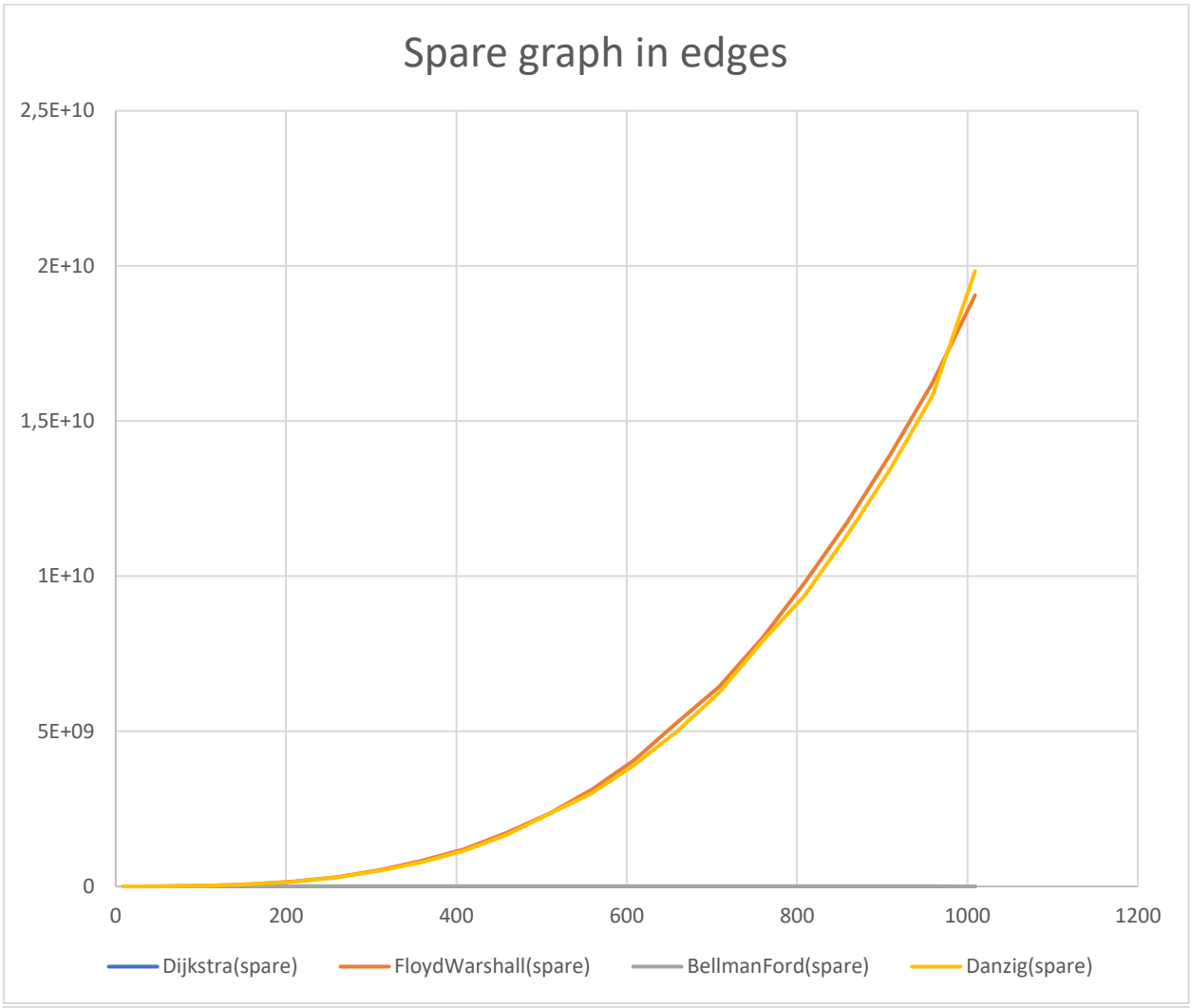

Connected graph in edges
18000000
16000000
14000000
12000000
10000000
8000000
6000000
4000000
2000000
0
0
50000
100000
150000
200000
250000
Dijkstra(connected)
BellmanFord(connected)

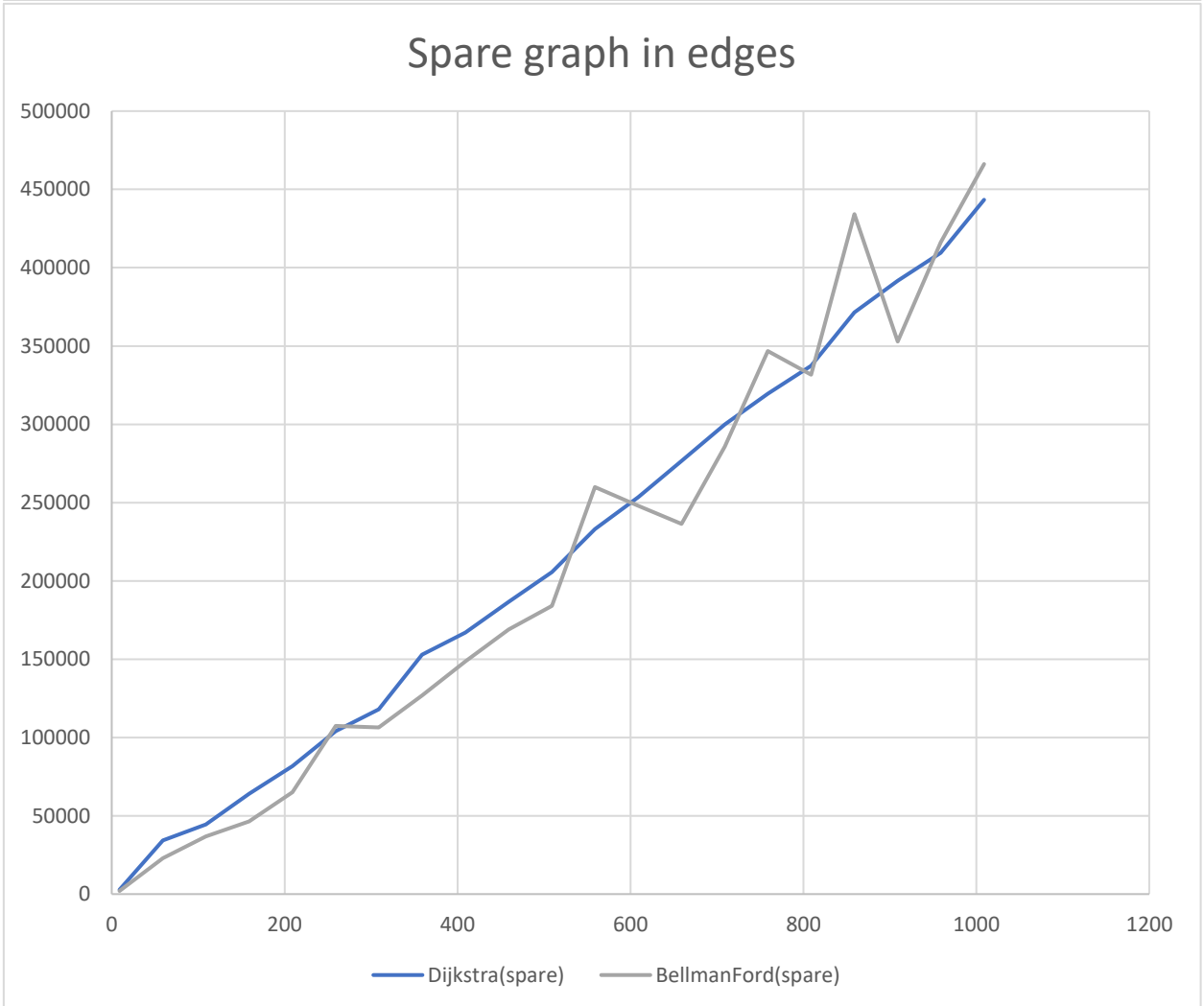
Разреженный граф

На разреженном графе m ~ n, по сравнению со связным графом в предыдущем пункте число ребер ещё уменьшилось. Т.к. на Беллмана-Форда это влияет сильней, чем на Дейкстру, они практически сравнялись по времени работы (колебания можно объяснить случайностью досрочного выхода). В связи с тем, что в разреженном графе подавляющее число ребер отсутствует (по сравнению с полным графом), в алгоритме Флойда-Уоршелла происходит минимальное число сравнений с минимумом, из-за чего он равняется с алгоритмом Данцига.






Spare graph in edges
2,5E+10
2E+10
1,5E+10
1E+10
5E+09
0
0
200
400
600
800
1000
1200
Dijkstra(spare)
FloydWarshall(spare)
BellmanFord(spare)
Danzig(spare)


Spare graph in edges
500000
450000
400000
350000
300000
250000
200000
150000
100000
50000
0
0
200
400
600
800
1000
1200
Dijkstra(spare)
BellmanFord(spare)