ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ **«КАЗАНСКИЙ(ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ

ОТЧЕТ ПО ТЕМЕ:

**«Дерево квадрантов»**

Выполнил: Бекмансуров Илья, гр. 11-102

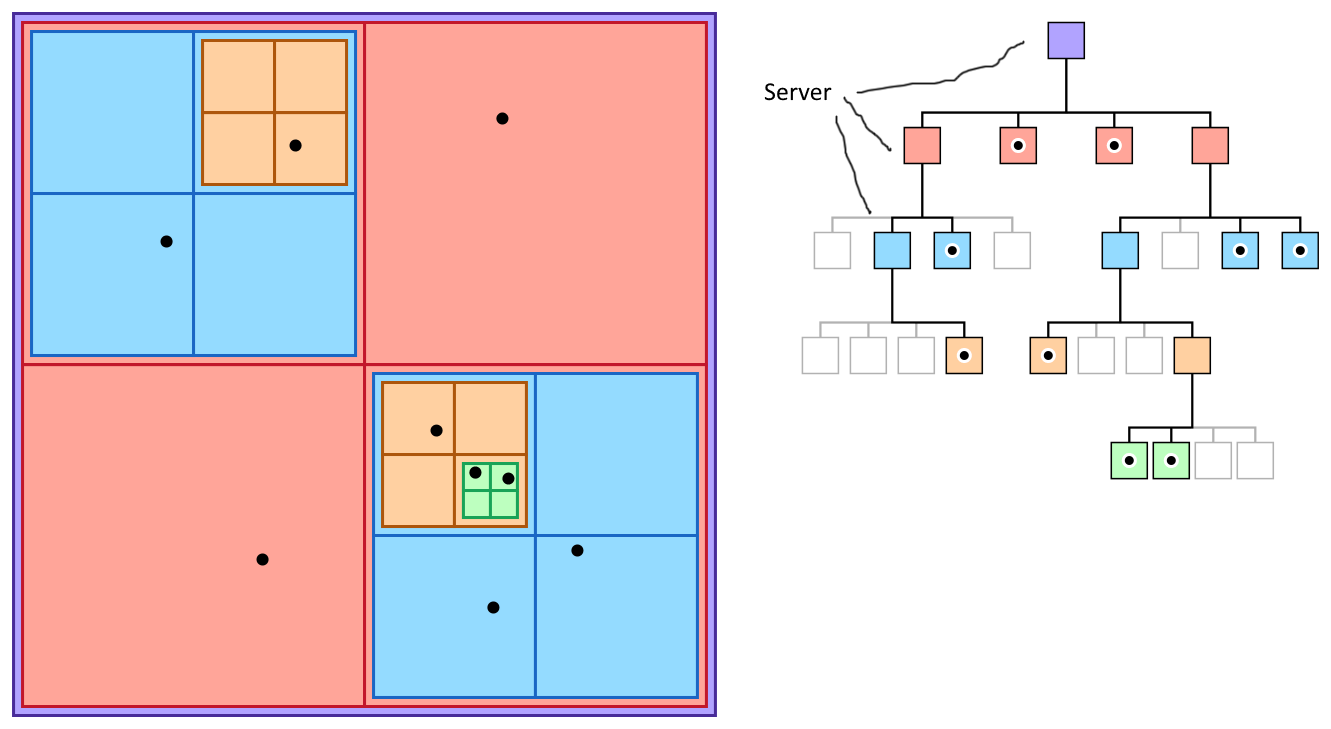
**Казань, 2022**

**Краткая историческая справка:**

Дерево квадрантов (также quadtree, квадродерево) считается одной из первых структур данных для данных высокой размерности (многомерных данных). Авторами дерева квадрантов являются американские учёные Рафаэль Финкель и Джон Бентли. Впервые эта структура данных была представлена и рассмотрена ими в работе «Quad Trees: A Data Structure for Retrieval on Composite Keys» в апреле 1974 года.

**Основной принцип устройства. Особенности**

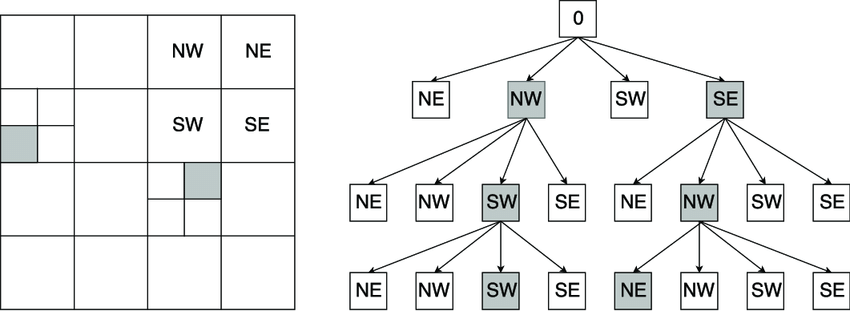
Дерево квадрантов — дерево, в котором у каждого внутреннего узла ровно 4 потомка.



Дерево квадрантов часто используется для рекурсивного разбиения двухмерного пространства по 4 квадранта (области), которые представляют собой квадраты, прямоугольники или имеют произвольную форму.

Изначально дерево квадрантов состоит из единичного узла (корня) – двухмерного пространства, в которое постепенно добавляются некоторые объекты (точки, отрезки, прямые, многоугольники и др.).

Одна из ключевых характеристик дерева квадрантов – максимально возможный объем каждого узла. Когда количество добавленных в корень объектов превысит это число, дерево рекурсивно разобьется на 4 равных подквадранта, и каждый объект будет помещен в один из них в зависимости от того, в какой области пространства он расположен. Далее, по мере добавления новых объектов, когда их количество в каком-либо подквадранте превысит его вместимость, он также рекурсивно разобьется на 4 равных подквадранта, в которые будут помещены соответствующие объекты. И так далее. Таким образом, образуется система вложенных областей пространства.

На плоскости удобно представлять расположение каждого из квадрантов как стороны света на карте: NW – северо-запад, NE – северо-восток, SW – юго-запад, SE – юго-восток.

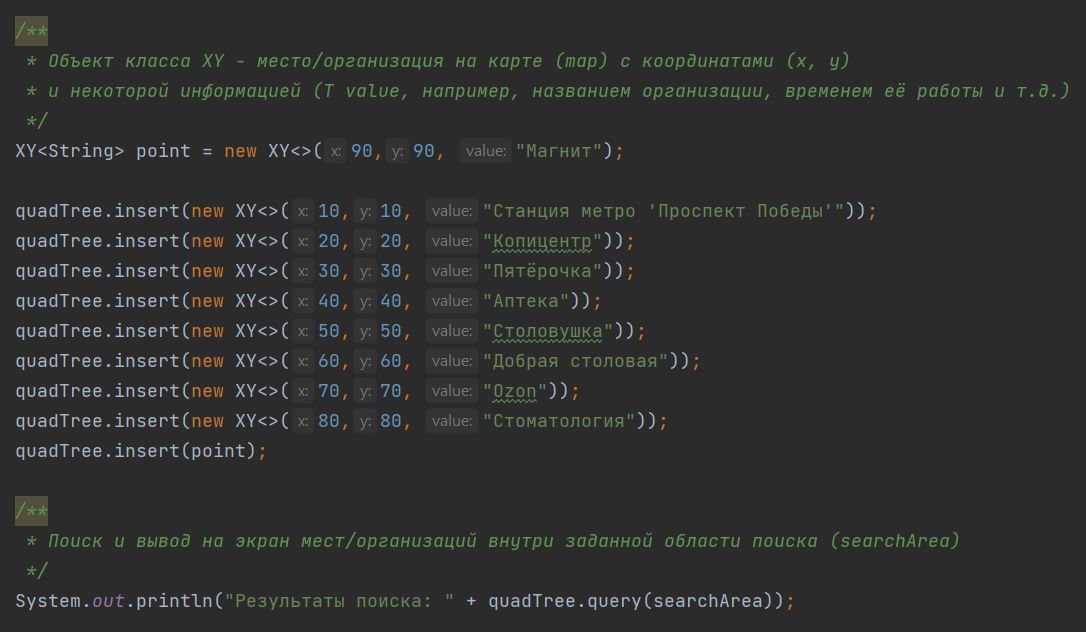
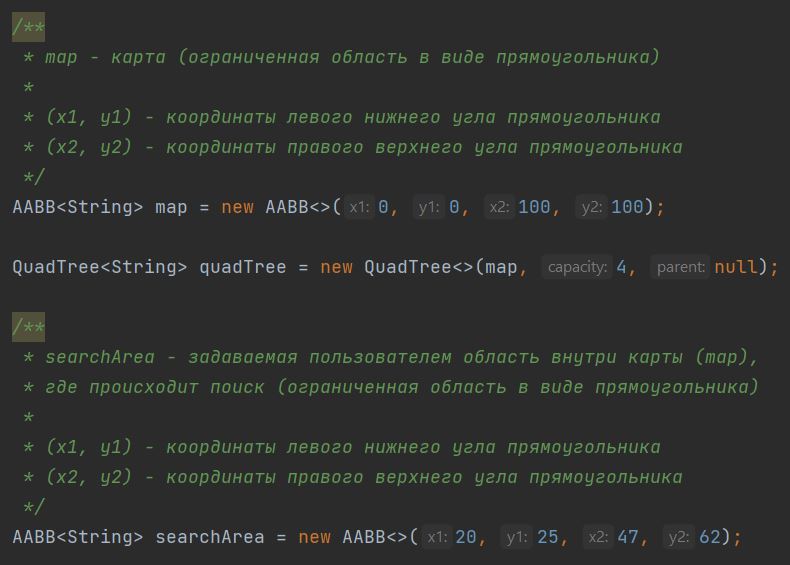
Как уже было отмечено, у каждого узла дерева квадрантов либо 4 потомка, либо их нет. В каждом узле хранятся его границы и вместимость, информация о родителе и точки, которые он содержит. В каждой точке, в свою очередь, хранятся координаты (x, y) и некоторое значение, например, имя.

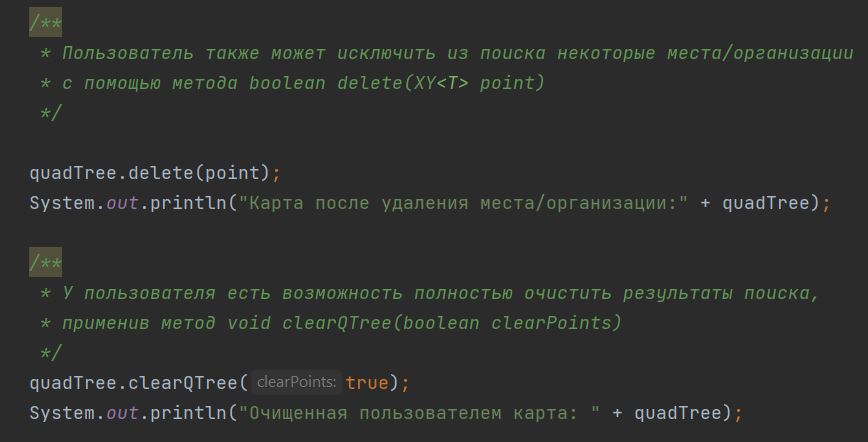
*Основные операции:*

* Вставка: *function insert(Point point)* – данный метод осуществляет рекурсивную вставку точки *point* в соответствующий квадрант дерева, осуществляя разбиение, если это необходимо (т.е. если количество объектов в одном квадранте превышает его вместимость).   
  Сначала происходит проверка, принадлежит ли объект дереву. Если нет, то он игнорируется. Затем проверяется, может ли объект поместиться в текущий узел. Если это так, то осуществляется вставка объекта в данный узел, иначе – узел разбивается на 4 квадранта, и лишь потом объект вставляется в тот квадрант, которому он принадлежит.
* Разбиение: *function subdivide()* – рассматриваемый метод создает 4-х потомков, которые делят исходный узел на 4 равных квадранта. Затем уже вставленные ранее в узел объекты перемещаются в соответствующие дочерние узлы.
* Вхождение в диапазон (поиск): *query(Region range)* – этот метод ищет все точки, расположенные внутри области *range*. Сначала происходит проверка, пересекаются ли границы узла с границами области поиска *range*. Если это не так, то можно пропустить этот узел со всеми его дочерними узлами, что значительно сокращает время поиска, так как нам не нужно проходится по узлам и объектам, которые заведомо не попадают в область поиска.

*Задача, в контексте которой реализована структура данных:*

Дерево квадрантов я решил использовать для реализации аналога навигатора с возможностью поиска мест/организаций (например, магазинов, столовых, аптек и т.д.) внутри заданной пользователем области.

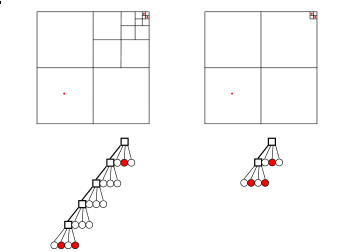
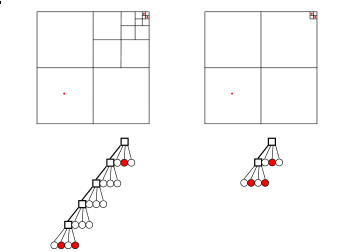




**Оценка временной сложности с обоснованием**

Пусть N – количество объектов в дереве квадрантов, P – максимальное количество объектов в одном узле.

*Поиск:* асимптотическая оценка – O(N/P), так как дерево может выродиться в список, и тогда до самой «худшей», удаленной от корня позиции придется спускаться N/P раз. К примеру, пусть имеется следующий набор точек *p* с координатами *(x, y)*: *p1(0, 0), p2(1, 1), p3(2, 2), p4(3, 3), …, pn(n-1, n-1)*. Каждый раз при вставке очередной точки, она добавляется в левый нижний угол (SE) предыдущего родительского узла (при условии, что координаты левого нижнего угла корневого узла *(0, 0)*). Таким образом, каждый узел имеет ровно одного потомка. Следовательно, дерево вырождается в связный список длины N.



*Получится примерно такое дерево*

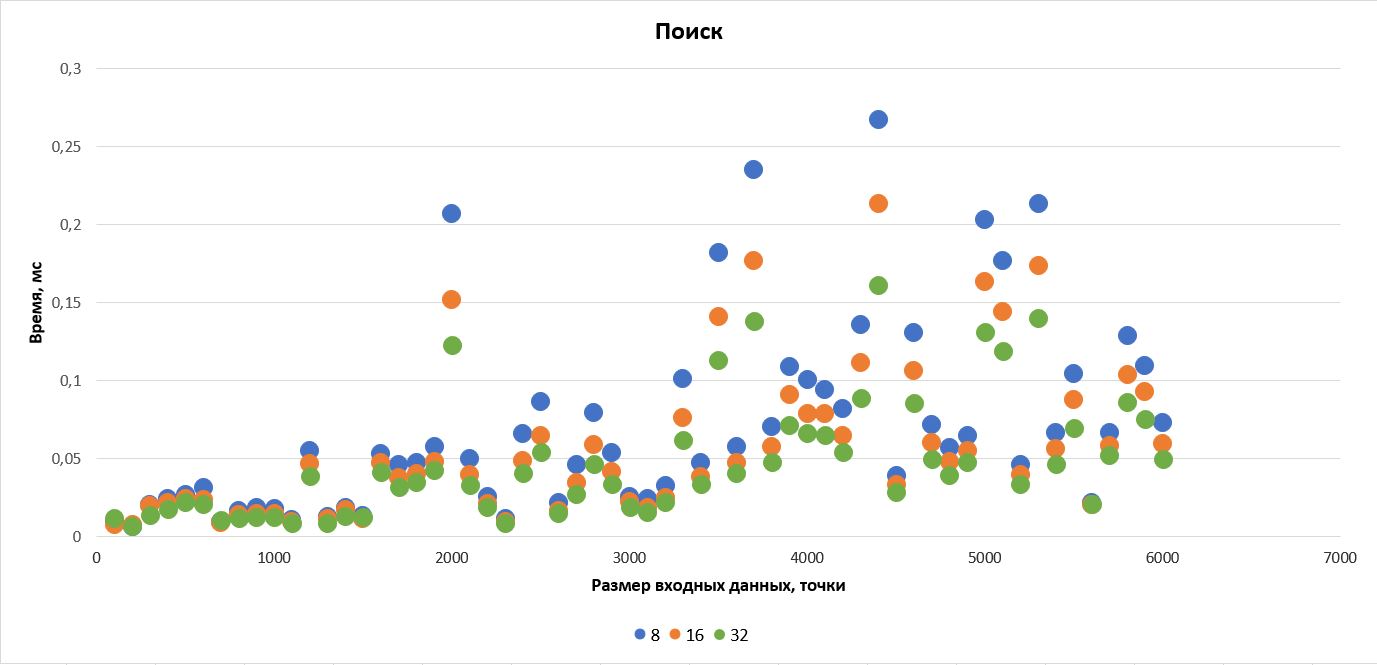
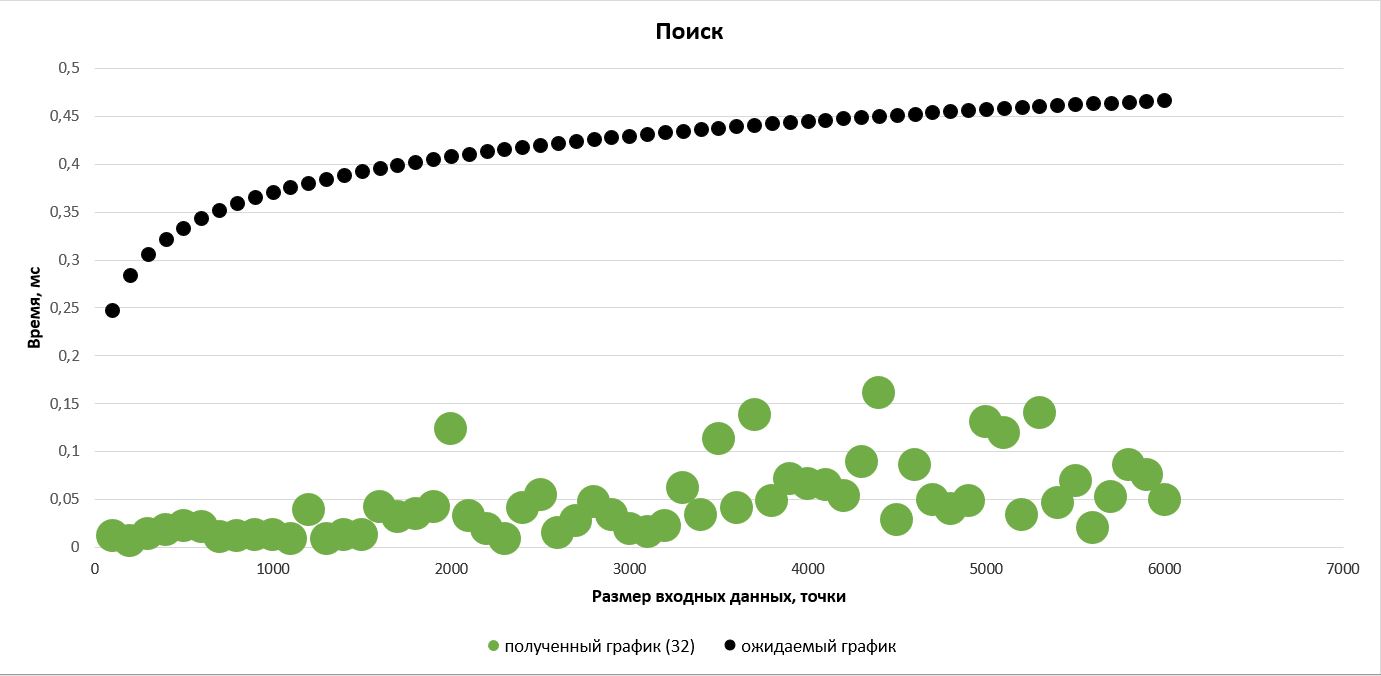
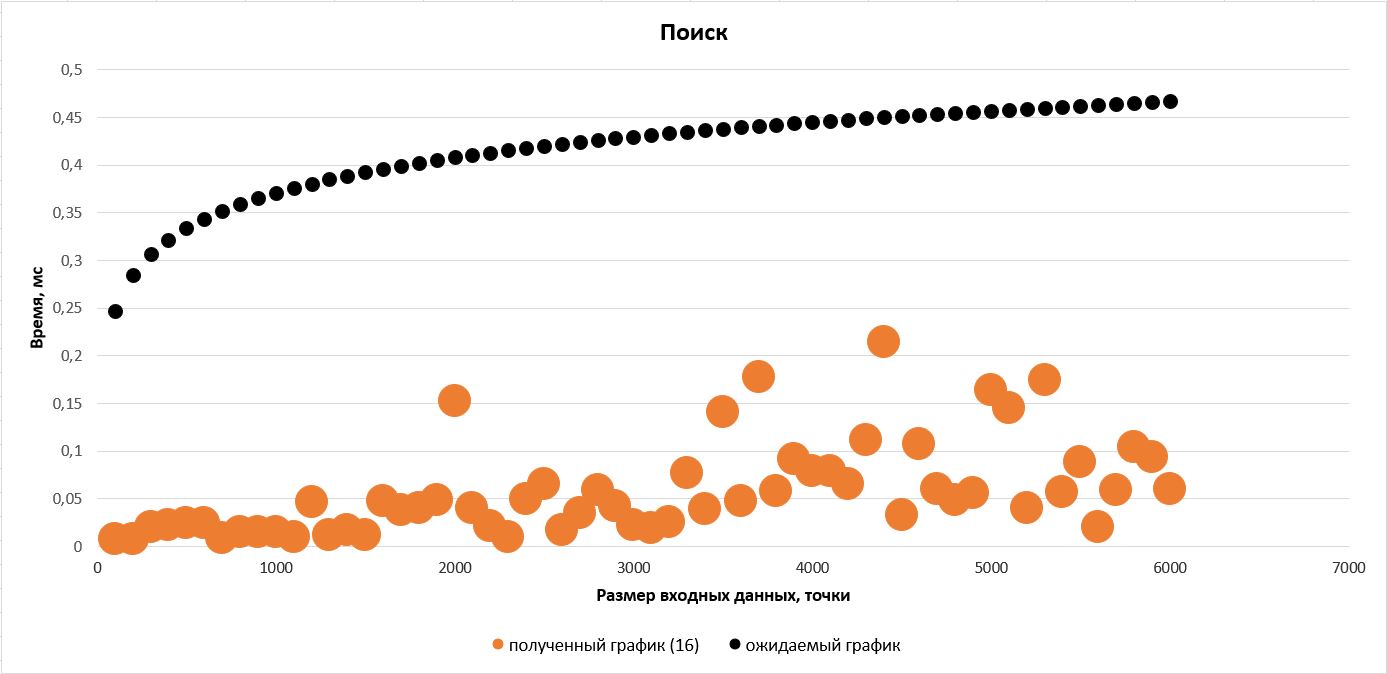
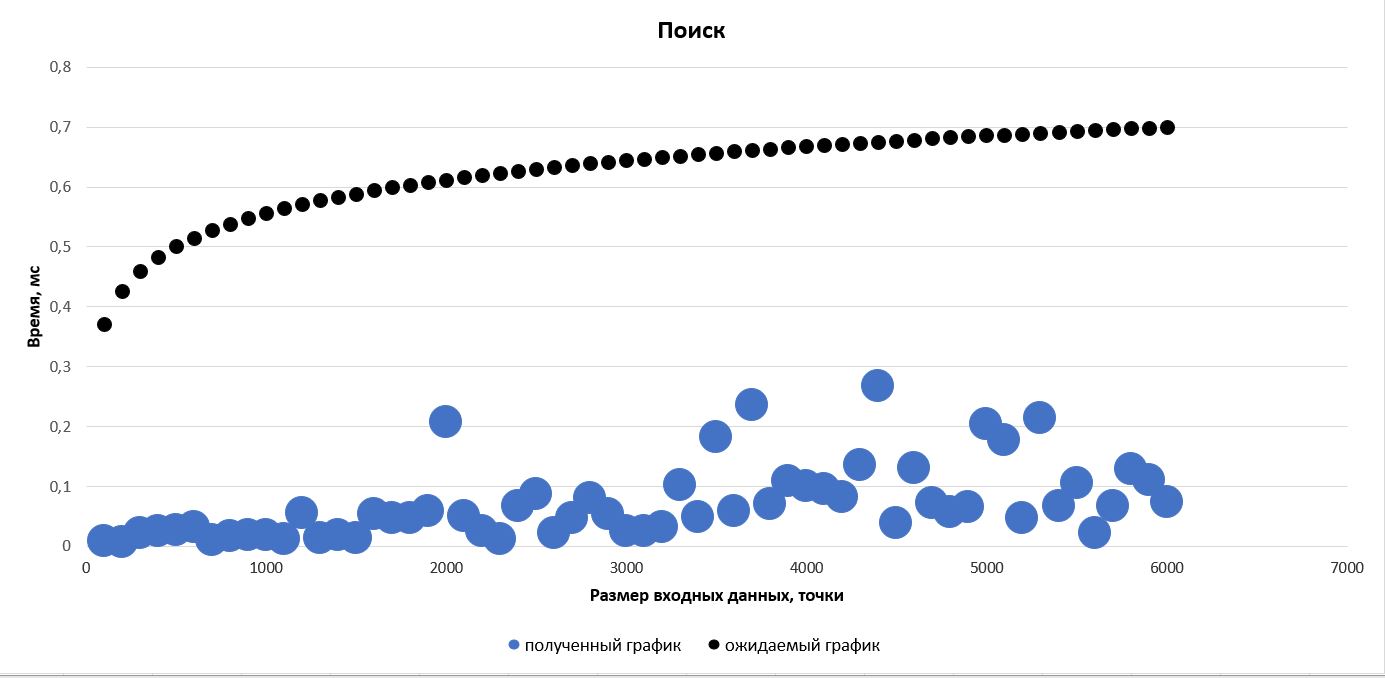
Однако амортизационно сложность поиска – ***O(logN)***, так как предполагается, что дерево более-менее сбалансировано и, следовательно, в список не вырождается.

*Вставка:* для амортизационного анализа рассмотрим начальное и конечное положение каждого из N вставляемых объектов. Пусть h(T) – конечная высота дерева квадрантов T, h(i) – конечная позиция (глубина) объекта i, t(i) – позиция объекта i в момент, когда он был вставлен. При вставке i-ый объект посетит t(i) узлов дерева, пока не найдет корректный для вставки. Стоимость этой операции – t(i). Как мы уже знаем, когда число объектов в узле превышает его вместимость, он разбивается на 4 подквадранта, и, следовательно, глубина каждого объекта увеличивается на 1. Таким образом, в процессе вставки всех N объектов из-за того, что узлы дерева разбиваются, объект i переместится h(i) – t(i) раз. Получается, чтобы достичь своей конечной позиции h(i) объект i накапливает сложность O(t(i) + h(i) – t(i)) = O(h(i)) ~ O(logN) (так как высота дерева = ***O(logN)***).

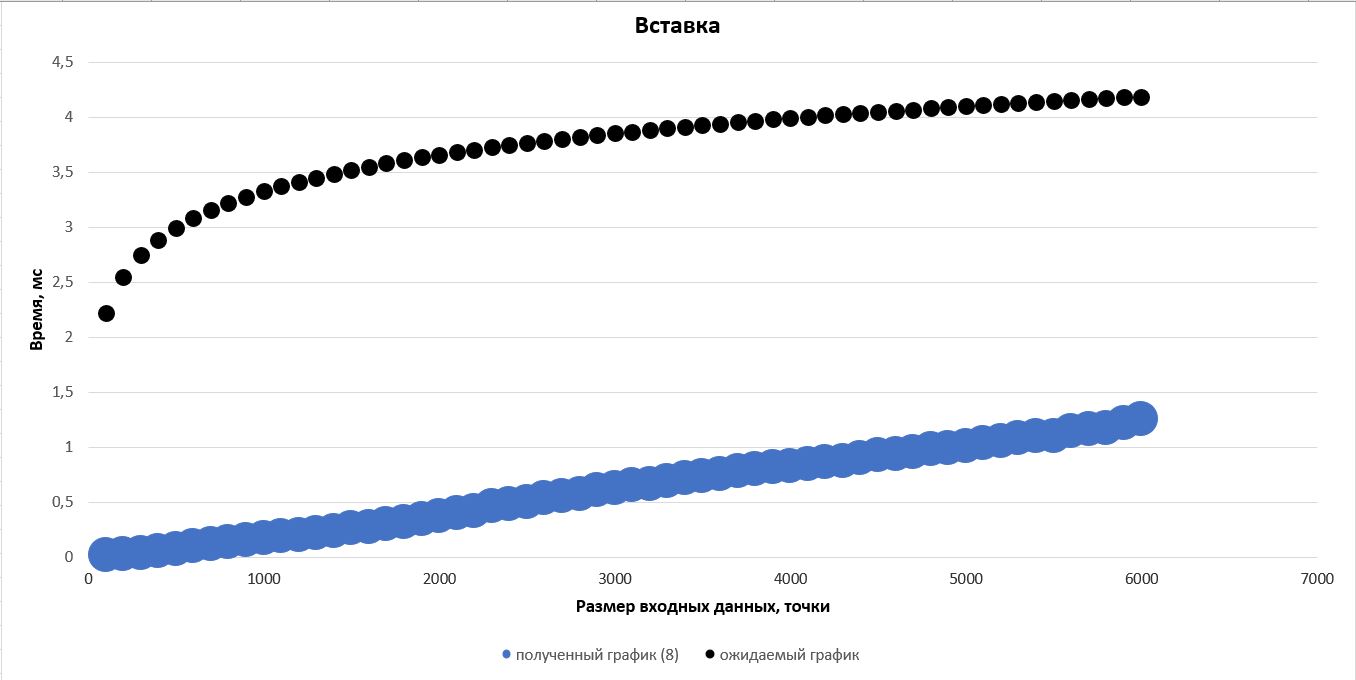
*Удаление:* ***O(logN)*** – аналогично вставке: чтобы добраться до i-го элемента нам потребуется O(logN) времени.

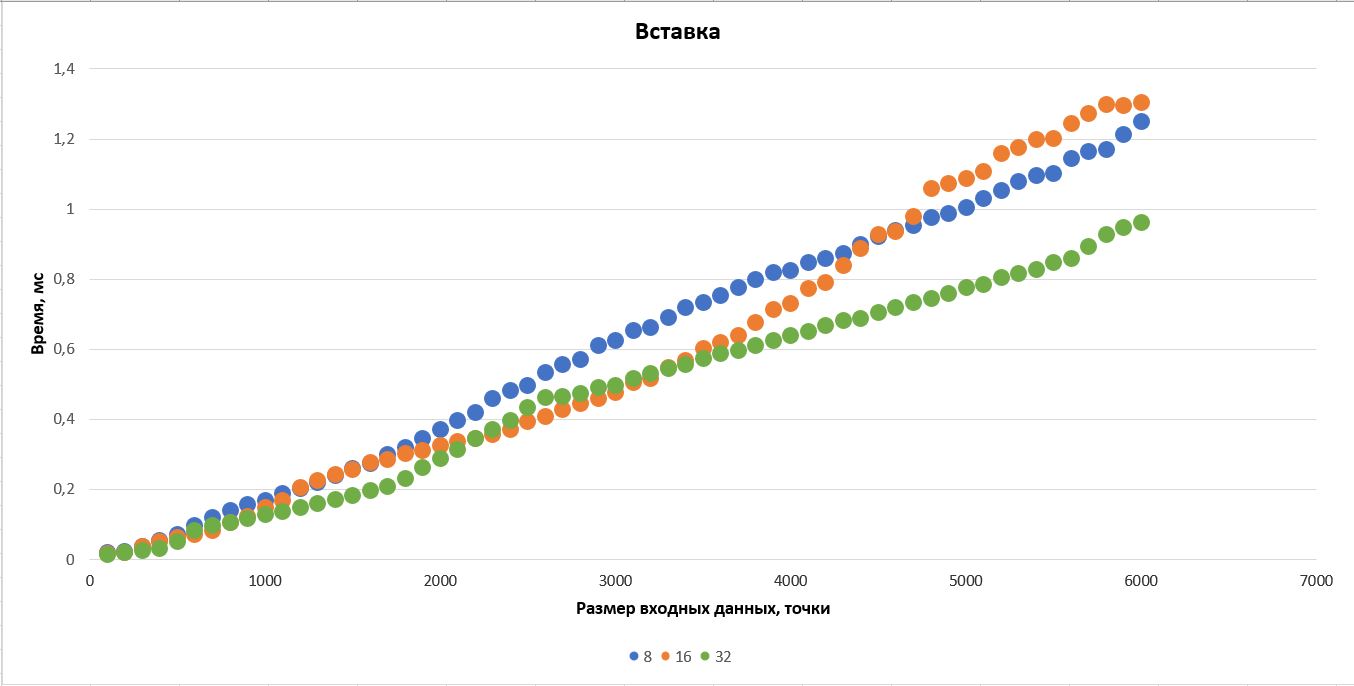
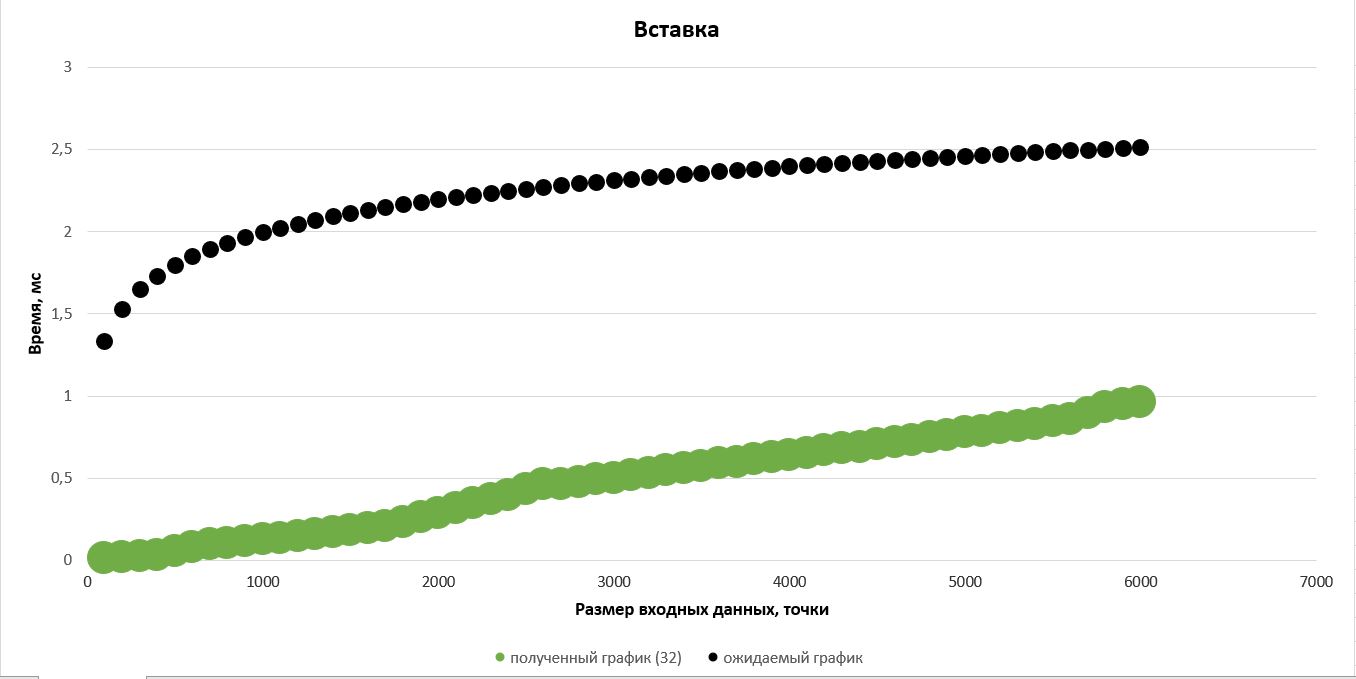
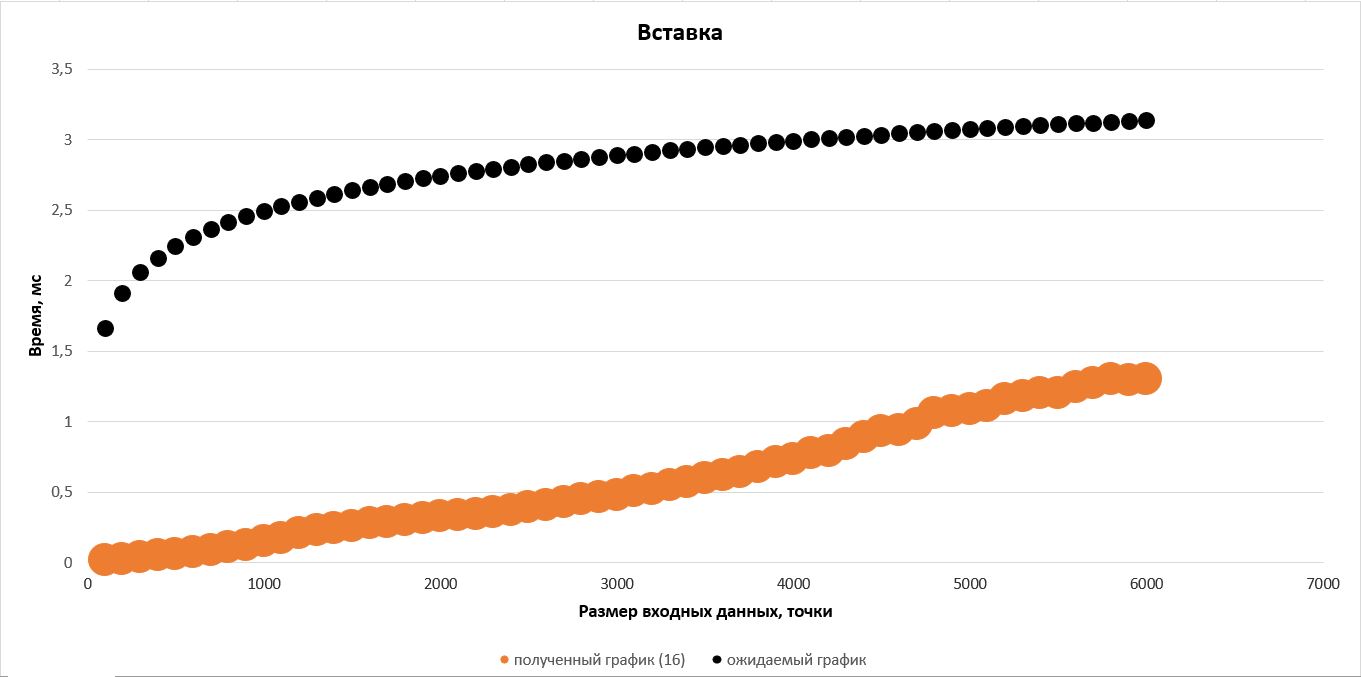
**Графики**

*Search (query)*

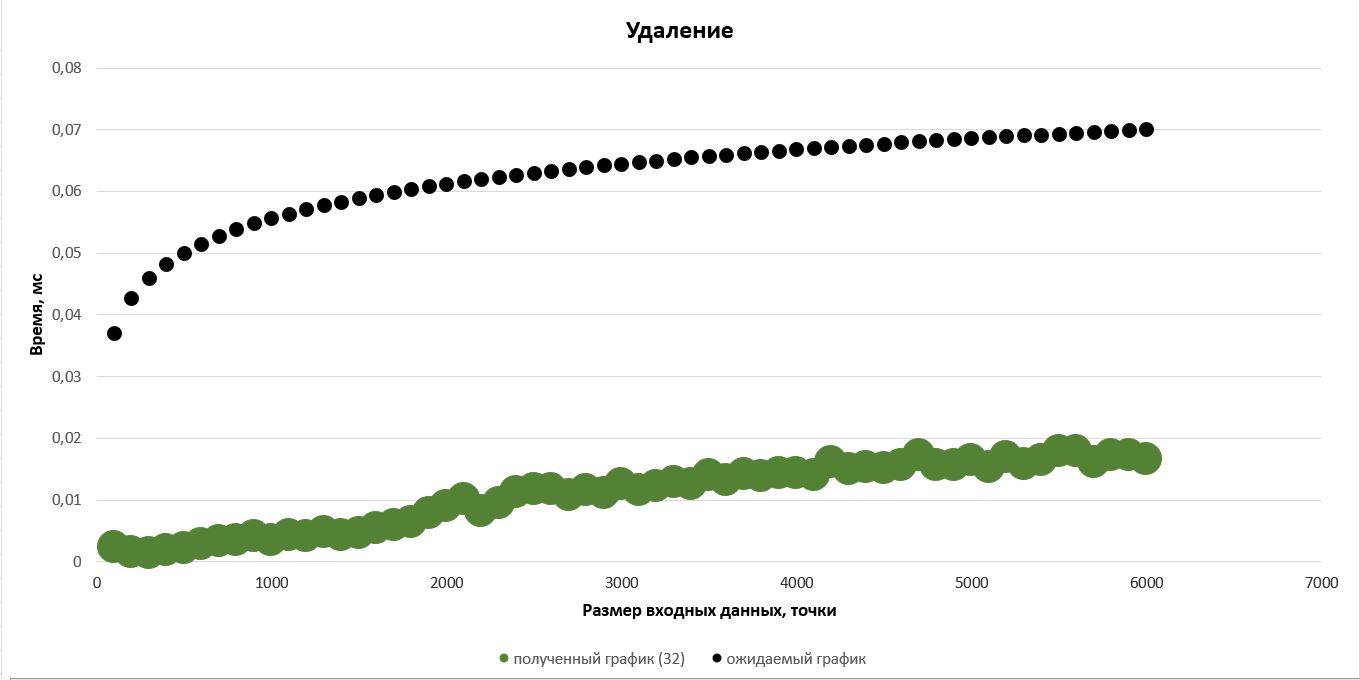
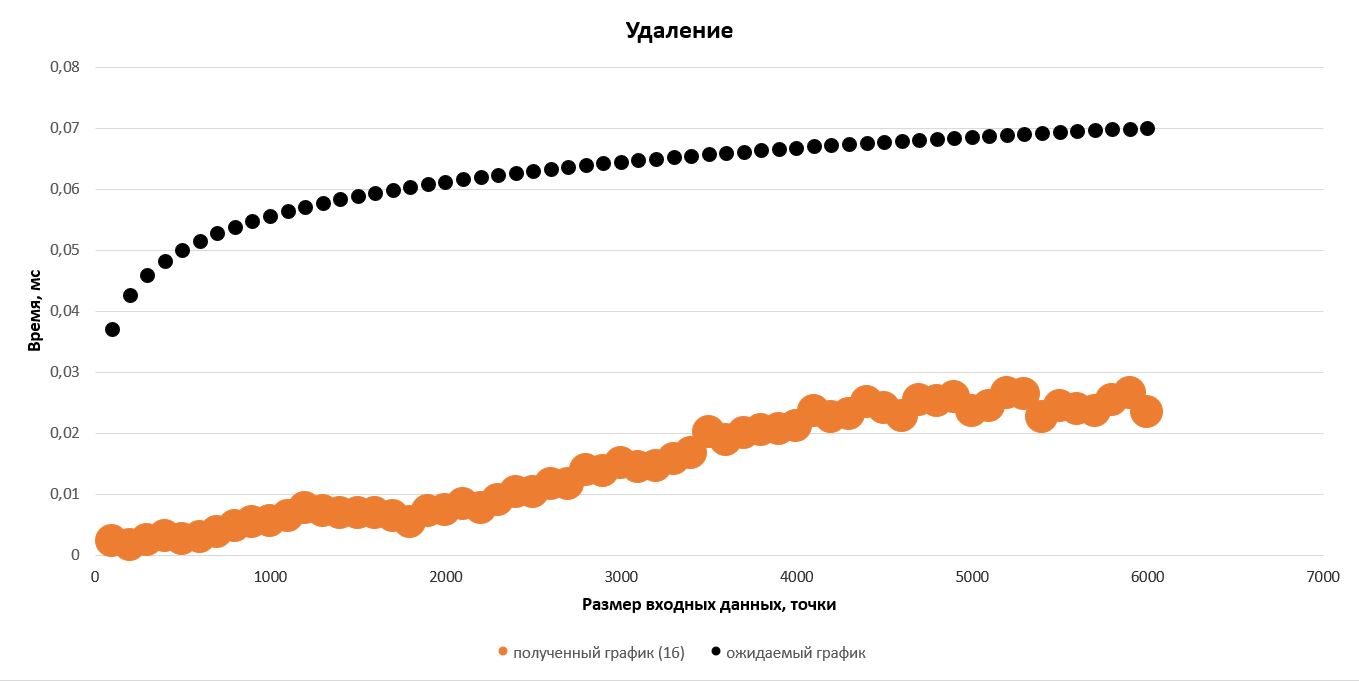
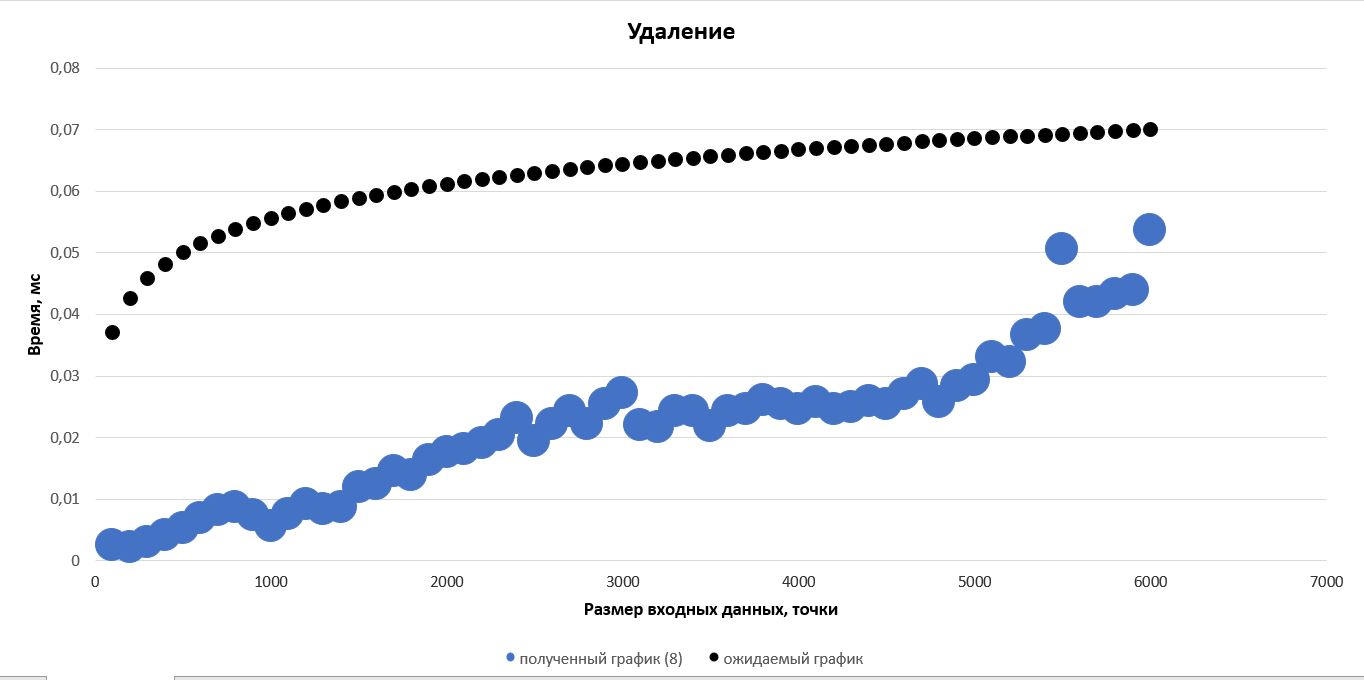


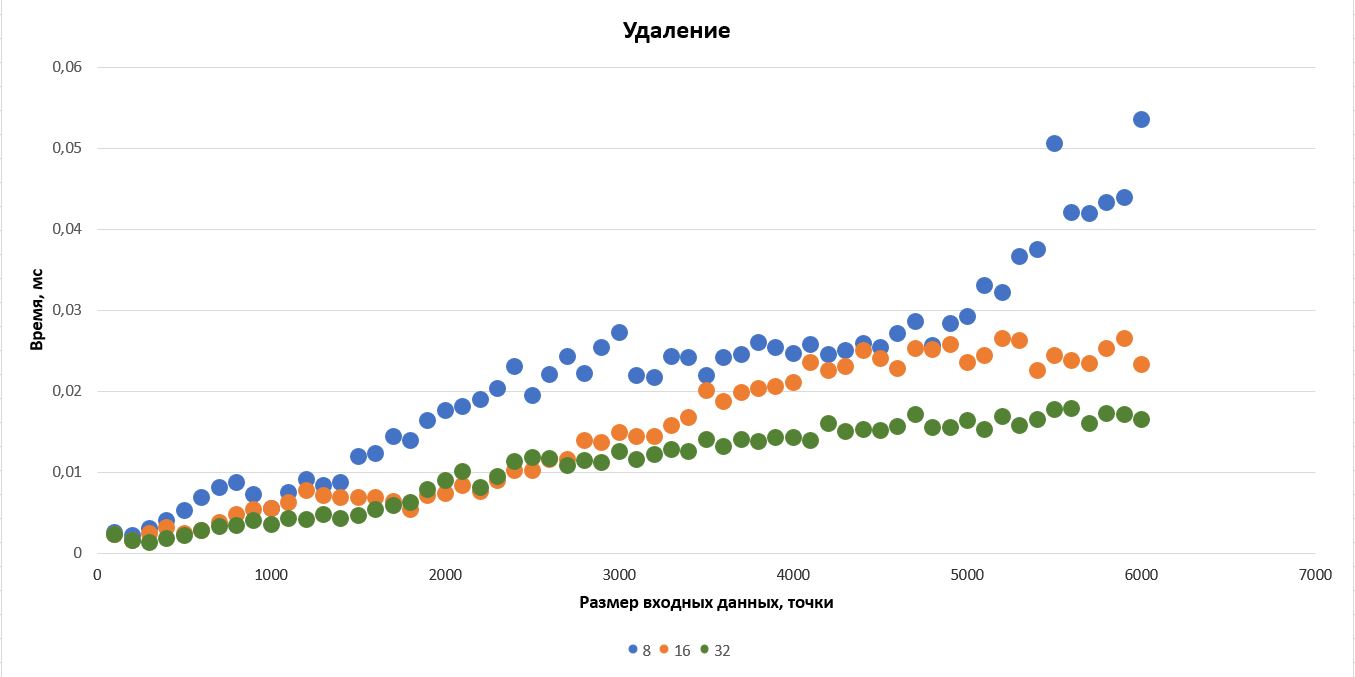
*Insert*





*Remove*





Как мы можем заметить, построенные графики отличаются от ожидаемых в теории. Это можно обосновать тем, что, так как данные на вход генерировались произвольно / рандомно, могло произойти так, что объекты, подаваемые на вход, были расположены очень близко друг к другу, и, следовательно, каждый раз добавлялись в один и тот же квадрант, что в свою очередь привело к тому, что дерево превратилось в вырожденный список. Поэтому на некоторых наборах данных сложность алгоритма превышает ожидаемую сложность. На других наборах данных объекты могли быть сгенерированы так, что построенное дерево получилось хорошо сбалансированным, и поэтому ожидаемое время алгоритма оказалось меньше предполагаемого в теории.

**Выводы. Плюсы и минусы структуры данных, ее применимость**

*Применимость:*

* Сollision detection algorithms (обнаружение столкновений – обнаружение пересечений между собой двух и более объектов) в GameDev’е, симуляции физических процессов
* Сжатие изображений, где каждый узел содержит «средний» цвет каждого дочернего узла. Чем глубже продвигаемся по дереву, тем более детальное изображение получаем
* Определение скрытой поверхности (процесс определения того, какие поверхности и части поверхностей можно увидеть под определенным углом обзора) в 3D моделировании, VR и GameDev’е
* Вычисления, связанные с многомерными полями (в вычислительной гидродинамике, электромагнетизме)
* Симуляция игры «Жизнь»
* Пространственные базы данных – базы данных, оптимизированные для хранения и выполнения запросов к данным о пространственных объектах, представленных некоторыми абстракциями: точка, линия, многоугольник и им подобных
* Хранение данных для табличных или матричных вычислений.

*Плюсы:*

* Простая в реализации и эффективная в использовании структура данных
* Быстро строится и значительно уменьшает количество перебираемых объектов, что увеличивает скорость и производительность алгоритмов, приложений, использующих рассматриваемую структуру данных

*Минусы:*

* Не константное время основных операций, что предоставляют другие структуры данных
* Деревья квадрантов не всегда является лучшей структурой данных для collision detection. Например, вместо них могут быть использованы разреженная сетка (sparse grid), Zomorodian and Edelsbrunner’s algorithm, которые предоставляют лучшую производительность

**Список использованной литературы (источников)**

1. <https://ru.wikipedia.org/wiki/Дерево_квадрантов>
2. *Raphael Finkel and J.L. Bentley.* Quad Trees: A Data Structure for Retrieval on Composite Keys – Acta Informatica: journal. – 1974
3. *Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf.* Computational Geometry — 2nd revised. — Springer-Verlag, 2000
4. *Hanan Samet; Robert Webbe.* Storing a Collection of Polygons Using Quadtrees – ACM Transactions on Graphics: journal. – 1985

**Ссылка на репозиторий**

[**https://github.com/bekmnsrw/aisd-spring-2022/tree/main/TermProject**](https://github.com/bekmnsrw/aisd-spring-2022/tree/main/TermProject)

(структура данных, тесты, таблицы, графики и сам отчет в репозитории)

**Приложение**

*Insert*

|  |  |  |  |
| --- | --- | --- | --- |
| **Размер входных данных (количество точек)** | **Среднее время выполнения операции (в мс),  если вместимость квадранта = 8** | **Среднее время выполнения операции (в мс),  если вместимость квадранта = 16** | **Среднее время выполнения операции (в мс),  если вместимость квадранта = 32** |
| **100** | 0,019105801 | 0,018252924 | 0,01454598 |
| **200** | 0,0227269 | 0,019541634 | 0,01928278 |
| **300** | 0,0381203 | 0,036968822 | 0,025650755 |
| **400** | 0,053648899 | 0,052972351 | 0,032846157 |
| **500** | 0,073102501 | 0,061849983 | 0,052769329 |
| **600** | 0,096583 | 0,072994507 | 0,082117729 |
| **700** | 0,1210271 | 0,083912949 | 0,096240791 |
| **800** | 0,139640793 | 0,105496933 | 0,105316109 |
| **900** | 0,157652298 | 0,12414772 | 0,11599794 |
| **1000** | 0,168413406 | 0,147571167 | 0,128485718 |
| **1100** | 0,187308197 | 0,169281982 | 0,136408997 |
| **1200** | 0,202935898 | 0,20594545 | 0,147578094 |
| **1300** | 0,220382294 | 0,225721008 | 0,159743805 |
| **1400** | 0,238613907 | 0,241835098 | 0,171009155 |
| **1500** | 0,260022491 | 0,256501617 | 0,18177684 |
| **1600** | 0,275004791 | 0,277622803 | 0,19632103 |
| **1700** | 0,301357697 | 0,286149017 | 0,209448883 |
| **1800** | 0,321144012 | 0,301978973 | 0,231367599 |
| **1900** | 0,345786896 | 0,312630829 | 0,263823608 |
| **2000** | 0,372328705 | 0,325056671 | 0,287906403 |
| **2100** | 0,396621704 | 0,335837921 | 0,314989868 |
| **2200** | 0,41961319 | 0,344500732 | 0,345527832 |
| **2300** | 0,458722809 | 0,357260406 | 0,37059729 |
| **2400** | 0,481556396 | 0,372217316 | 0,397760376 |
| **2500** | 0,495701294 | 0,3930755 | 0,434022064 |
| **2600** | 0,532597717 | 0,407226746 | 0,461341797 |
| **2700** | 0,55655188 | 0,427714508 | 0,465084137 |
| **2800** | 0,57128833 | 0,445497711 | 0,473886597 |
| **2900** | 0,611500793 | 0,460835419 | 0,492150513 |
| **3000** | 0,626503601 | 0,477882416 | 0,498168213 |
| **3100** | 0,652541199 | 0,504450195 | 0,516141418 |
| **3200** | 0,662966309 | 0,515993591 | 0,52983136 |
| **3300** | 0,692098572 | 0,547687927 | 0,546683594 |
| **3400** | 0,719124084 | 0,56679541 | 0,557686829 |
| **3500** | 0,734452026 | 0,60105957 | 0,574139221 |
| **3600** | 0,752895813 | 0,620315491 | 0,587661316 |
| **3700** | 0,776866882 | 0,6402453 | 0,59759729 |
| **3800** | 0,79996698 | 0,677781982 | 0,611496399 |
| **3900** | 0,818205811 | 0,713668701 | 0,624665527 |
| **4000** | 0,824488403 | 0,731502319 | 0,638272034 |
| **4100** | 0,846633301 | 0,774203308 | 0,652125244 |
| **4200** | 0,860264221 | 0,790465027 | 0,668572937 |
| **4300** | 0,874677124 | 0,838282227 | 0,681878845 |
| **4400** | 0,898208313 | 0,887360229 | 0,689136963 |
| **4500** | 0,922872925 | 0,928595276 | 0,706307617 |
| **4600** | 0,938233521 | 0,936999512 | 0,719142334 |
| **4700** | 0,952149719 | 0,978421326 | 0,732528687 |
| **4800** | 0,977480896 | 1,058776733 | 0,745700195 |
| **4900** | 0,988400391 | 1,072523682 | 0,758301331 |
| **5000** | 1,004968628 | 1,085974487 | 0,777076843 |
| **5100** | 1,031277954 | 1,108370239 | 0,786306458 |
| **5200** | 1,052204834 | 1,15941394 | 0,803829224 |
| **5300** | 1,079856445 | 1,176272461 | 0,814975037 |
| **5400** | 1,095589355 | 1,197293701 | 0,828990906 |
| **5500** | 1,101784058 | 1,2027854 | 0,846898499 |
| **5600** | 1,143717773 | 1,244031128 | 0,858394958 |
| **5700** | 1,163870972 | 1,271552246 | 0,892233582 |
| **5800** | 1,17033606 | 1,298809204 | 0,928412903 |
| **5900** | 1,212915894 | 1,294626709 | 0,947109314 |
| **6000** | 1,25023584 | 1,303317017 | 0,962549927 |

*Remove*

|  |  |  |  |
| --- | --- | --- | --- |
| **Размер входных данных (количество точек)** | **Среднее время выполнения операции (в мс),  если вместимость квадранта = 8** | **Среднее время выполнения операции (в мс),  если вместимость квадранта = 16** | **Среднее время выполнения операции (в мс),  если вместимость квадранта = 32** |
| **100** | 0,00248831 | 0,00226462 | 0,002320995 |
| **200** | 0,002228437 | 0,001511596 | 0,001552485 |
| **300** | 0,003034088 | 0,002386893 | 0,001321227 |
| **400** | 0,004070114 | 0,003113403 | 0,001765435 |
| **500** | 0,00524106 | 0,002475294 | 0,002188487 |
| **600** | 0,006891608 | 0,002819328 | 0,002754399 |
| **700** | 0,00815749 | 0,003739115 | 0,003241513 |
| **800** | 0,008722107 | 0,004718292 | 0,003395389 |
| **900** | 0,007289898 | 0,005385295 | 0,004027083 |
| **1000** | 0,00555621 | 0,005544022 | 0,00348083 |
| **1100** | 0,007491305 | 0,00630851 | 0,004303966 |
| **1200** | 0,009075751 | 0,0076921 | 0,004106198 |
| **1300** | 0,008379895 | 0,007109938 | 0,004706579 |
| **1400** | 0,008743187 | 0,006814607 | 0,004326113 |
| **1500** | 0,011973879 | 0,006846422 | 0,004629211 |
| **1600** | 0,012336075 | 0,006893786 | 0,005386311 |
| **1700** | 0,014431378 | 0,006389278 | 0,005934249 |
| **1800** | 0,013928315 | 0,005340688 | 0,00628058 |
| **1900** | 0,016345724 | 0,007090714 | 0,007872557 |
| **2000** | 0,017639357 | 0,0073722 | 0,008929136 |
| **2100** | 0,018115301 | 0,008317824 | 0,010050707 |
| **2200** | 0,019033403 | 0,007608315 | 0,008138607 |
| **2300** | 0,020359015 | 0,008970977 | 0,009490703 |
| **2400** | 0,02305879 | 0,010218012 | 0,011280107 |
| **2500** | 0,019441914 | 0,010244997 | 0,011775784 |
| **2600** | 0,022064896 | 0,011554742 | 0,011678962 |
| **2700** | 0,024327995 | 0,011511487 | 0,010790209 |
| **2800** | 0,022165398 | 0,013855188 | 0,011466795 |
| **2900** | 0,02543709 | 0,013646719 | 0,011133105 |
| **3000** | 0,027221003 | 0,014946475 | 0,012572209 |
| **3100** | 0,021988598 | 0,014355206 | 0,011510778 |
| **3200** | 0,021733128 | 0,014453702 | 0,012175626 |
| **3300** | 0,024267582 | 0,015748899 | 0,012797585 |
| **3400** | 0,024225708 | 0,016739828 | 0,012530972 |
| **3500** | 0,021882113 | 0,020054001 | 0,014049554 |
| **3600** | 0,024227106 | 0,018780136 | 0,013225819 |
| **3700** | 0,024596312 | 0,019895699 | 0,014073212 |
| **3800** | 0,026056417 | 0,02038612 | 0,013822201 |
| **3900** | 0,025447939 | 0,020571115 | 0,014235321 |
| **4000** | 0,024633993 | 0,021107821 | 0,014268594 |
| **4100** | 0,025726706 | 0,023486614 | 0,013909511 |
| **4200** | 0,024595758 | 0,022520201 | 0,016049088 |
| **4300** | 0,024976786 | 0,023051493 | 0,014966288 |
| **4400** | 0,02595513 | 0,024978809 | 0,015330402 |
| **4500** | 0,025362024 | 0,024039209 | 0,015175763 |
| **4600** | 0,027078573 | 0,022803116 | 0,015580209 |
| **4700** | 0,028616503 | 0,025291899 | 0,017154985 |
| **4800** | 0,025682592 | 0,025203815 | 0,015521047 |
| **4900** | 0,028390127 | 0,025827436 | 0,015561884 |
| **5000** | 0,029273979 | 0,023586302 | 0,016353901 |
| **5100** | 0,033072403 | 0,024405891 | 0,015293514 |
| **5200** | 0,032145313 | 0,026502399 | 0,016848 |
| **5300** | 0,036595783 | 0,026256895 | 0,015814889 |
| **5400** | 0,037537006 | 0,022584101 | 0,016486307 |
| **5500** | 0,050580074 | 0,024366228 | 0,017791702 |
| **5600** | 0,042030991 | 0,0238239 | 0,017856081 |
| **5700** | 0,04192403 | 0,023477297 | 0,016006107 |
| **5800** | 0,043337982 | 0,025274927 | 0,017294312 |
| **5900** | 0,043881744 | 0,026528969 | 0,01716349 |
| **6000** | 0,053578793 | 0,023307774 | 0,016504108 |

*Search (query)*

|  |  |  |  |
| --- | --- | --- | --- |
| **Размер входных данных (количество точек)** | **Среднее время выполнения операции (в мс),  если вместимость квадранта = 8** | **Среднее время выполнения операции (в мс),  если вместимость квадранта = 16** | **Среднее время выполнения операции (в мс),  если вместимость квадранта = 32** |
| **100** | 0,008759406 | 0,007784897 | 0,01191489 |
| **200** | 0,00646508 | 0,007700027 | 0,006749518 |
| **300** | 0,020893024 | 0,01987331 | 0,014061298 |
| **400** | 0,024555689 | 0,022108883 | 0,018017714 |
| **500** | 0,027279728 | 0,024127108 | 0,022294783 |
| **600** | 0,031320173 | 0,023998205 | 0,020931087 |
| **700** | 0,009502394 | 0,00898392 | 0,010898494 |
| **800** | 0,016472813 | 0,014409904 | 0,011730216 |
| **900** | 0,018363016 | 0,01473412 | 0,012740204 |
| **1000** | 0,018317305 | 0,014810825 | 0,012816585 |
| **1100** | 0,011267411 | 0,009581768 | 0,008633489 |
| **1200** | 0,05535881 | 0,046699299 | 0,038826012 |
| **1300** | 0,012927794 | 0,011662786 | 0,008493298 |
| **1400** | 0,018883617 | 0,017347099 | 0,013227415 |
| **1500** | 0,013829788 | 0,01172109 | 0,012589299 |
| **1600** | 0,053605972 | 0,047550915 | 0,041727089 |
| **1700** | 0,046463104 | 0,037944096 | 0,032122803 |
| **1800** | 0,047465981 | 0,040190186 | 0,035011894 |
| **1900** | 0,058105206 | 0,048147404 | 0,042566509 |
| **2000** | 0,207314392 | 0,152098618 | 0,123128395 |
| **2100** | 0,049897797 | 0,039564121 | 0,032839378 |
| **2200** | 0,025815792 | 0,021334877 | 0,018770515 |
| **2300** | 0,011888009 | 0,009893203 | 0,009019594 |
| **2400** | 0,066248695 | 0,049099697 | 0,041038788 |
| **2500** | 0,086443024 | 0,065060852 | 0,054481213 |
| **2600** | 0,021663816 | 0,016990826 | 0,014993303 |
| **2700** | 0,046160915 | 0,034540916 | 0,027560471 |
| **2800** | 0,079432831 | 0,059372063 | 0,04686013 |
| **2900** | 0,054150829 | 0,041895271 | 0,033570992 |
| **3000** | 0,025845921 | 0,022478258 | 0,018984308 |
| **3100** | 0,024424601 | 0,01870439 | 0,015560098 |
| **3200** | 0,032482525 | 0,025355413 | 0,022351311 |
| **3300** | 0,101423492 | 0,076637215 | 0,062031898 |
| **3400** | 0,04772131 | 0,038737629 | 0,033539814 |
| **3500** | 0,182157822 | 0,141149002 | 0,113298813 |
| **3600** | 0,057780323 | 0,047569 | 0,040547916 |
| **3700** | 0,235402115 | 0,177033569 | 0,138325546 |
| **3800** | 0,070670326 | 0,057552597 | 0,048073696 |
| **3900** | 0,109018097 | 0,091364395 | 0,071316689 |
| **4000** | 0,100586784 | 0,079147247 | 0,066175087 |
| **4100** | 0,094529495 | 0,078929001 | 0,064986168 |
| **4200** | 0,081957321 | 0,065067818 | 0,054016903 |
| **4300** | 0,136015793 | 0,111823982 | 0,088645599 |
| **4400** | 0,267277985 | 0,213612732 | 0,161300217 |
| **4500** | 0,039201733 | 0,033104698 | 0,028861521 |
| **4600** | 0,130719635 | 0,106614075 | 0,08562841 |
| **4700** | 0,072179497 | 0,060190166 | 0,049697598 |
| **4800** | 0,057003677 | 0,048282001 | 0,039637577 |
| **4900** | 0,064912987 | 0,05535281 | 0,048145191 |
| **5000** | 0,203529099 | 0,163596802 | 0,131003098 |
| **5100** | 0,176836288 | 0,144545807 | 0,119130402 |
| **5200** | 0,046317394 | 0,040144691 | 0,034012383 |
| **5300** | 0,213434906 | 0,174171417 | 0,140000839 |
| **5400** | 0,066881798 | 0,056810085 | 0,046252987 |
| **5500** | 0,104355721 | 0,0881716 | 0,069656158 |
| **5600** | 0,021735403 | 0,020411818 | 0,020642857 |
| **5700** | 0,066990044 | 0,058545101 | 0,052380325 |
| **5800** | 0,129297211 | 0,104250389 | 0,086398315 |
| **5900** | 0,110023613 | 0,093099083 | 0,075315132 |
| **6000** | 0,073073738 | 0,059970394 | 0,04963921 |