

# **Средства анализа нагрузки на структуры хранения данных**



**Выполнил :            Потапов Д. Р.**  
**Руководитель:       Селезнев К. Е.**

# Актуальность работы

---

Потребность эффективно хранить  
и обрабатывать информацию

Огромное количество различных  
модулей хранения информации  
и их комбинаций



Необходимость анализа  
работы контейнеров  
для выбора оптимального

# Примеры необходимости оптимального контейнера

---

- индекс базы данных (если структура индекса хорошо оптимизирована под поиск, то происходит сильное ускорение работы);
- системы с ограниченными ресурсами (если структуры данных оптимальны, то система работает быстро).

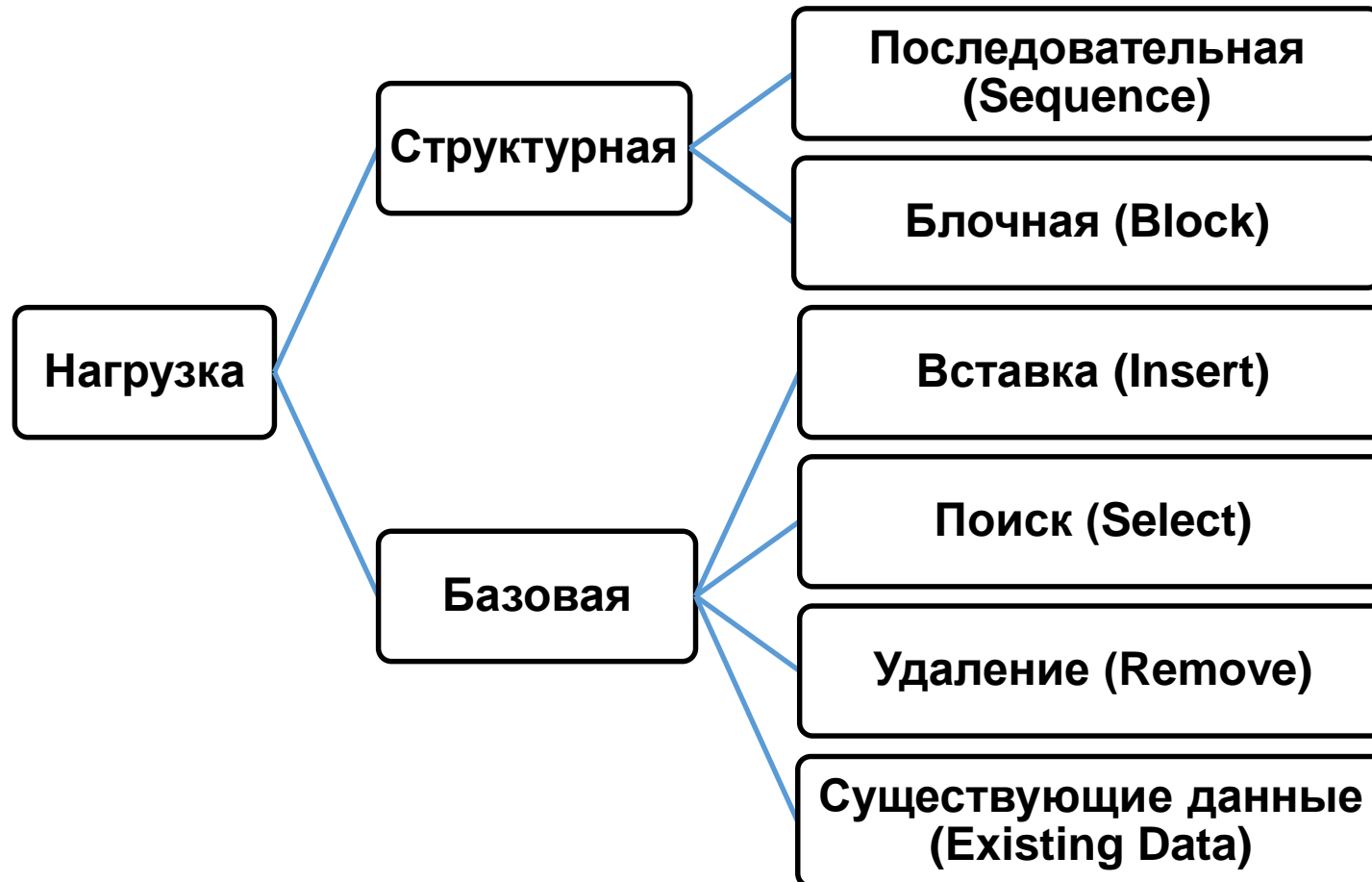
# Постановка задачи

---

Разработать программное обеспечение, в котором должны быть реализованы:

1. Визуализация нагрузки на одном контейнере.
2. Визуализация сравнения нескольких контейнеров.
3. Алгоритм получения линейного классификатора для адаптивного выбора оптимального способа хранения данных в зависимости от нагрузки.

# Модель нагрузки



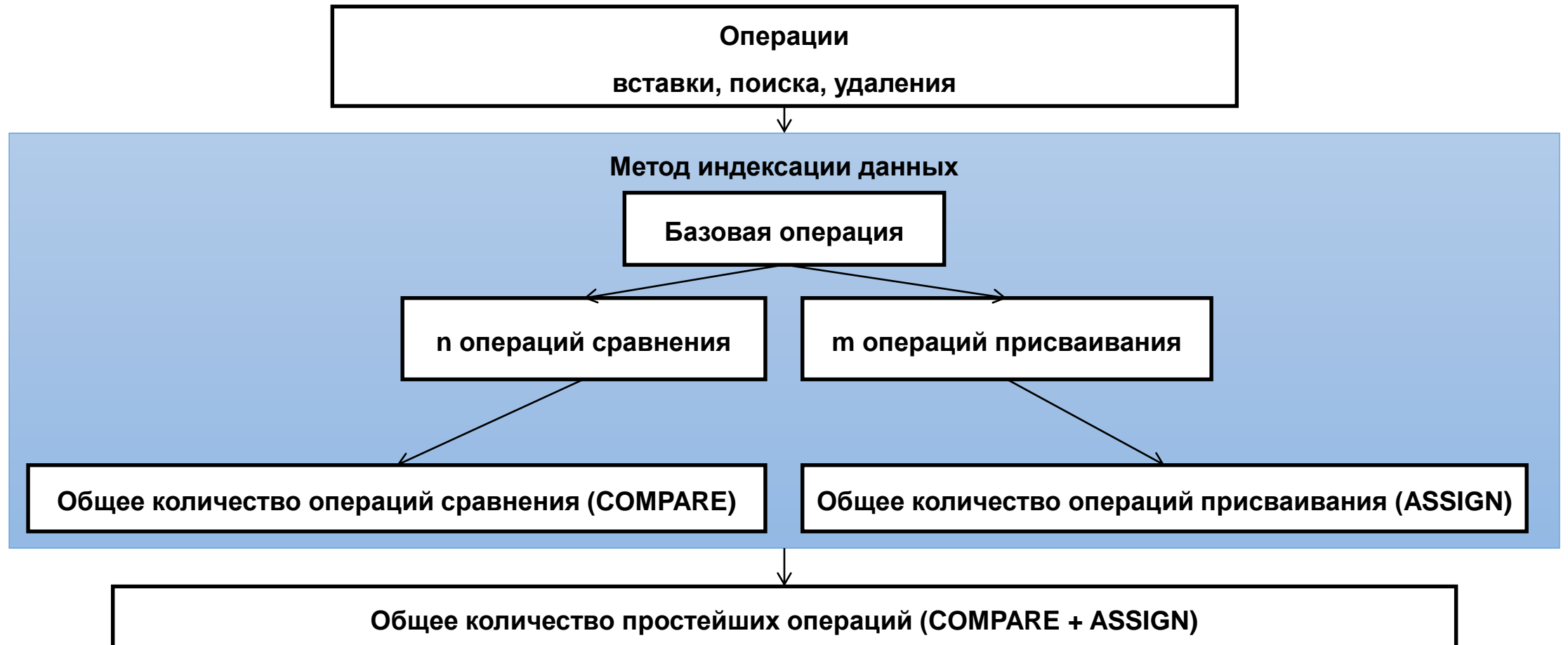
# Методы индексации данных

---

Каждый метод индексации данных должен предоставлять базовый интерфейс для работы с контейнером. Такой интерфейс должен содержать три базовых операции:

1. Поиск (Select). Операция чтения данных из контейнера.
2. Вставка (Insert). Операция вставки данных.
3. Удаление (Remove). Операция удаления данных.

# Способ оценки метода индексации данных



# Общая схема работы программы

---

1. Берется фиксированное число, равное сумме операций.
2. На основе этого числа строится плоскость, в которой при отсечении положительными осями координат образуется треугольник.
3. Для каждой точки этого треугольника определяются координаты в трехмерном пространстве IRS.

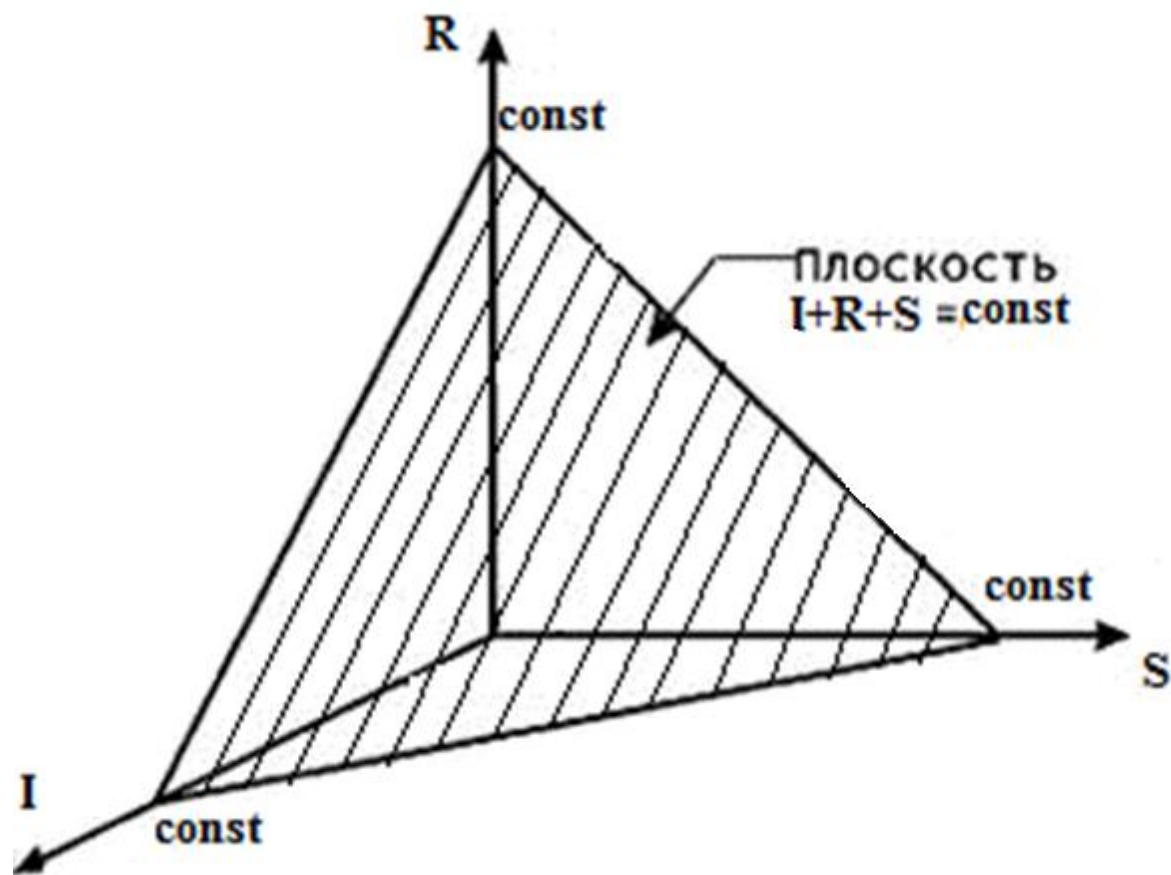


# Общая схема работы программы

---

4. По полученным координатам строится нагрузка, в которой количество каждой базовой операции совпадает с координатой по соответствующей оси.
5. Полученная нагрузка применяется к контейнеру (или контейнерам).
6. На основе полученных значений количества простейших операций определяется цвет пикселя.

# Преобразование координат



# Преобразование координат

Переход от системы экранных координат  $d=(x, y, z)$  к системе координат  $IRS=(insert, remove, select)$ .

Справедливо следующее  $X_{IRS} = M \cdot X_d$ , где  $M$  – матрица перехода.

Окончательно получаем матрицу перехода

$$M = M_{Remove}(35, 264) * M_{Select}(-45) * T(const * \sqrt{2}/2, 0, 0).$$

# Определение цвета пикселя для одного контейнера

---

Для этого используется метод линейного градиента:

- определяется минимальное и максимальное значение среди всех пикселей. Потом получаем коэффициент пикселя из диапазона [0;1] по формуле

$$Ratio = (value - min) / (max - min);$$

- в итоге цвет определяется по формуле:

$$Color = colorMax * ratio + colorMin * (1 - ratio).$$

# Определение цвета пикселя для нескольких контейнеров

---

1. Для каждого контейнера задается свой цвет.
2. Для каждого пикселя ищется минимальное количество операций (по всем тестируемым контейнерам).
3. Выбирается цвет хранилища, соответствующего индексу минимального числа из пункта 2.

# Классификация

---

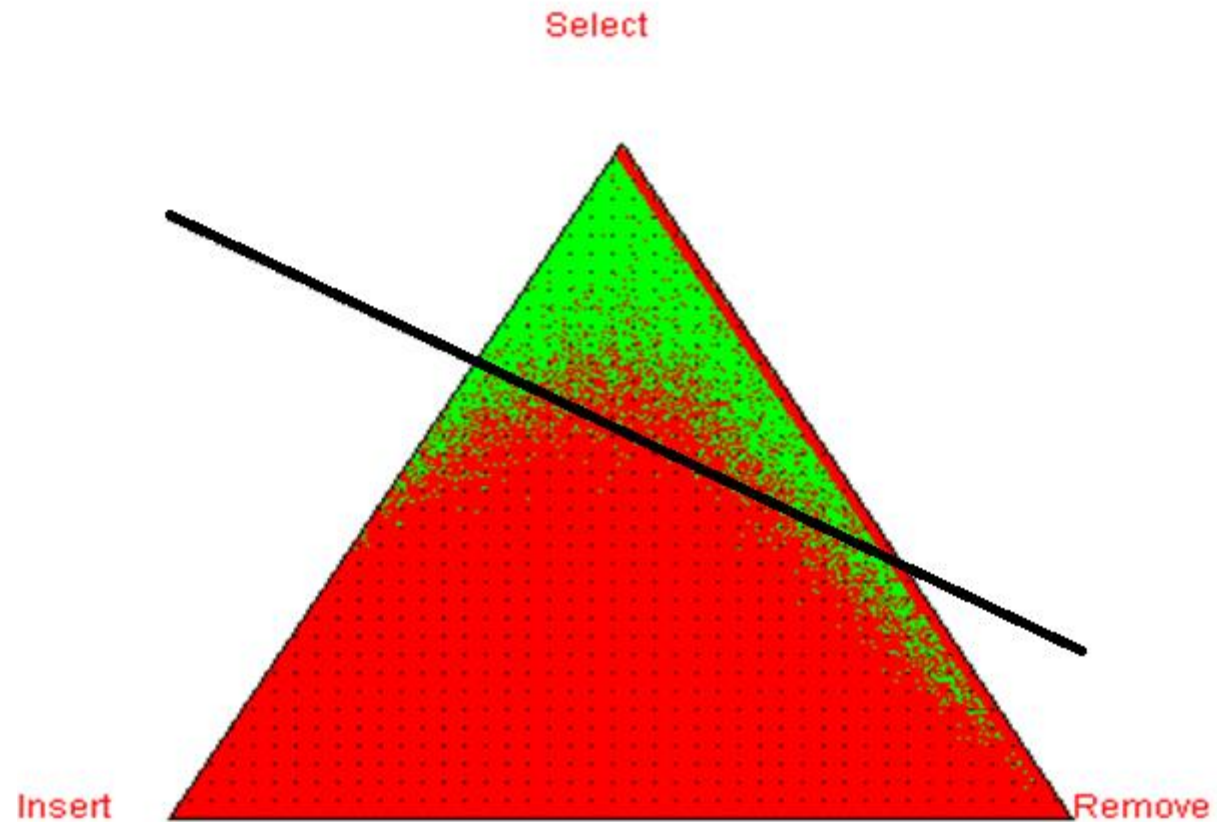
Признак, по которому можно сказать, что определенная точка в пространстве IRS лучше подходит для одного контейнера, чем для другого.

Простейшим примером классификатора является прямая, по одну сторону от которой оптимальным будет являться один контейнер, а по другую – второй.

Метод опорных векторов (SVM) с линейным ядром.

# Классификация

---



# Средства реализации

---

1. Среда разработки Eclipse Luna Service Release 2 (4.4.2).
2. Язык программирования Java 8.
3. Библиотека для работы с xml-файлами jdom2.
4. Библиотека логирования log4j.
5. Библиотека для решения задачи SVM libsvm.



# Требования к аппаратному и программному обеспечению

---

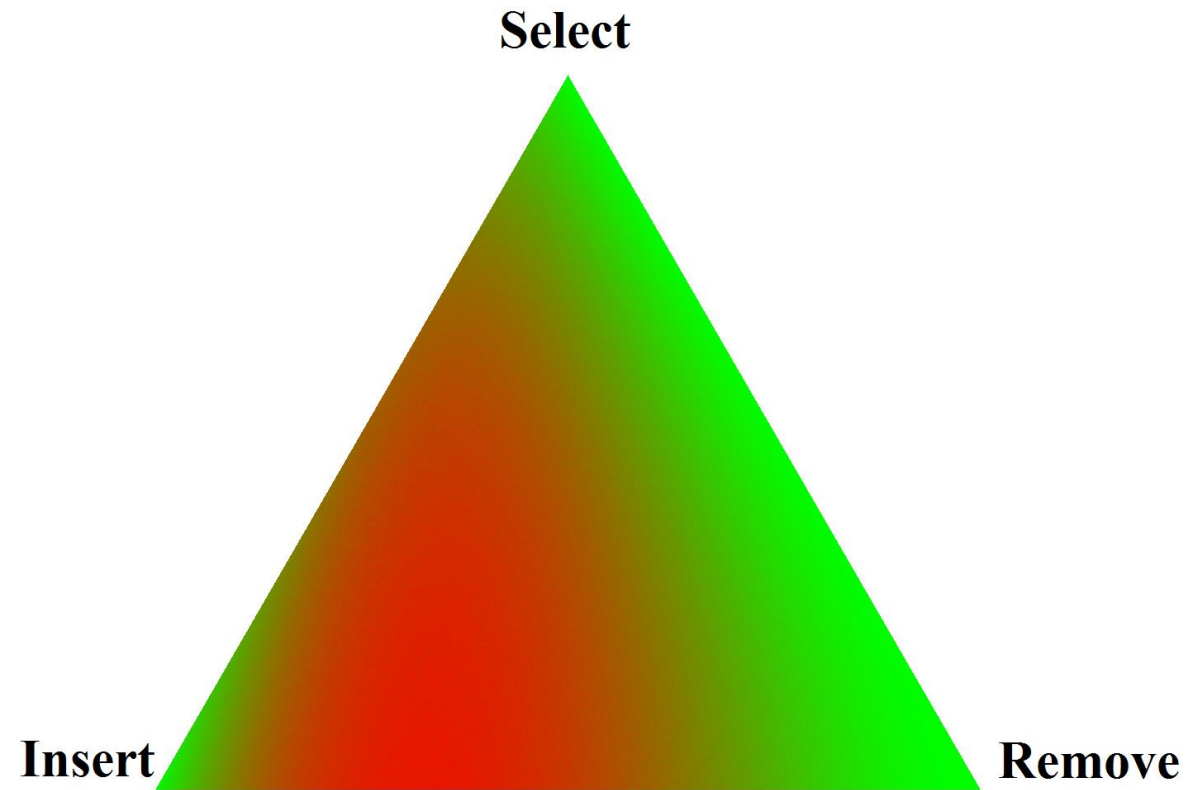
1. Процессор не ниже Pentium IV 2.6 GHz.
2. Оперативная память размером не менее 1024 Мб.
3. Не менее 1 Гб свободного дискового пространства.
4. Операционная система Windows XP и выше.
5. Виртуальная машина Java версии 1.8.

# Интерфейс пользователя

```
Properties Problems Type Hierarchy Search Console
<terminated> Main (2) [Java Application] C:\Program Files\Java\jdk1.8.0_45\bin\javaw.exe (13 июня 2015 г., 19:58:55)
***Программа для визуализации нагрузки на хранилища***
Размер выходного файла изображения в пикселях(от 10 до 2000):100
Название выходного файла изображения:SortedVsSimple
Класс хранилища:SortedList
Добавить к хранилищу параметры? (y/n)n
Добавить еще одно хранилище? (y/n)y
Класс хранилища:SimpleList
Добавить к хранилищу параметры? (y/n)n
Добавить еще одно хранилище? (y/n)n
13.06.2015 19:59:21 INFO com.vsu.amm.visualization.Vizualizator: Создаем файл с
именем SortedVsSimple и размером 100*100 пикселей.
13.06.2015 19:59:21 INFO com.vsu.amm.visualization.Vizualizator: Началась класс
ификация.
13.06.2015 19:59:23 INFO com.vsu.amm.visualization.Vizualizator: Классификация
выполнена успешно.
Уравнение разделяющей плоскости: 1304051.0*x+1301069.0*y+1307530.0z*+0.0=0
13.06.2015 19:59:24 INFO com.vsu.amm.visualization.Vizualizator: Файл с именем
SortedVsSimple и размером 100*100 пикселей успешно создан.
```

# Результат работы программы

---

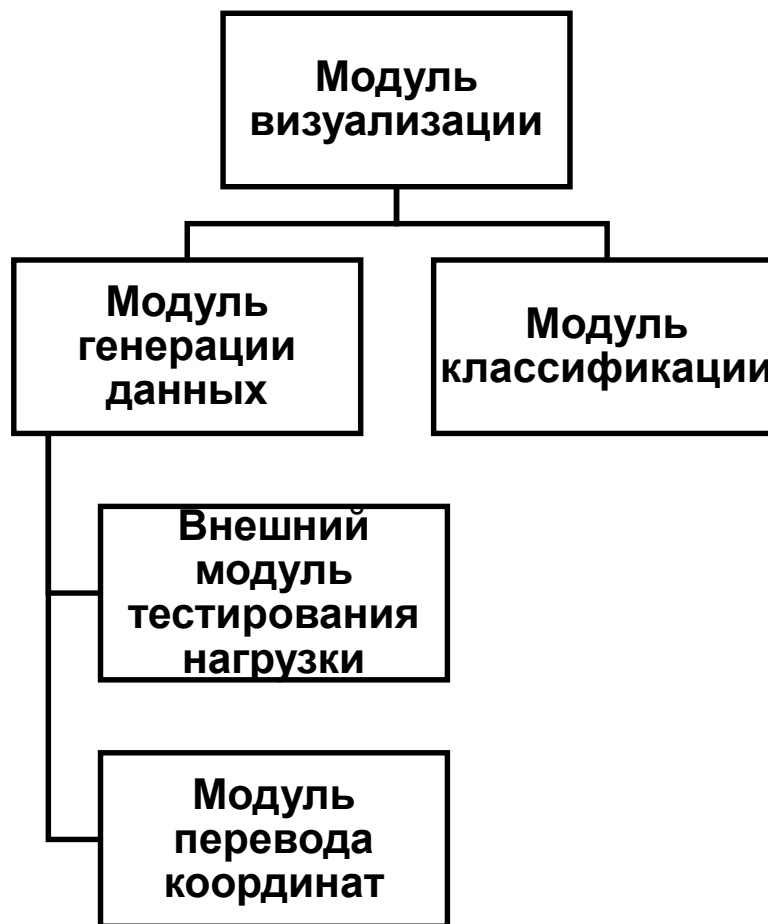


# Общая схема работы программы

---

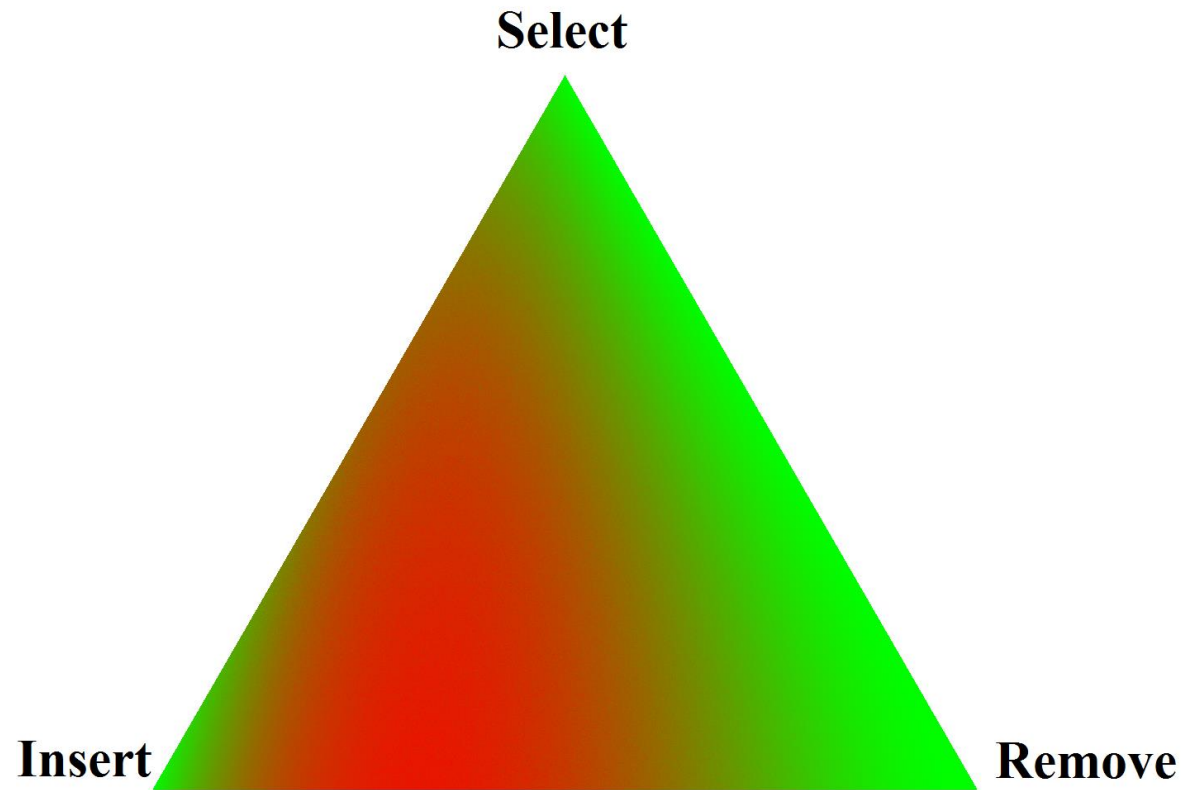


# Общая архитектура программы



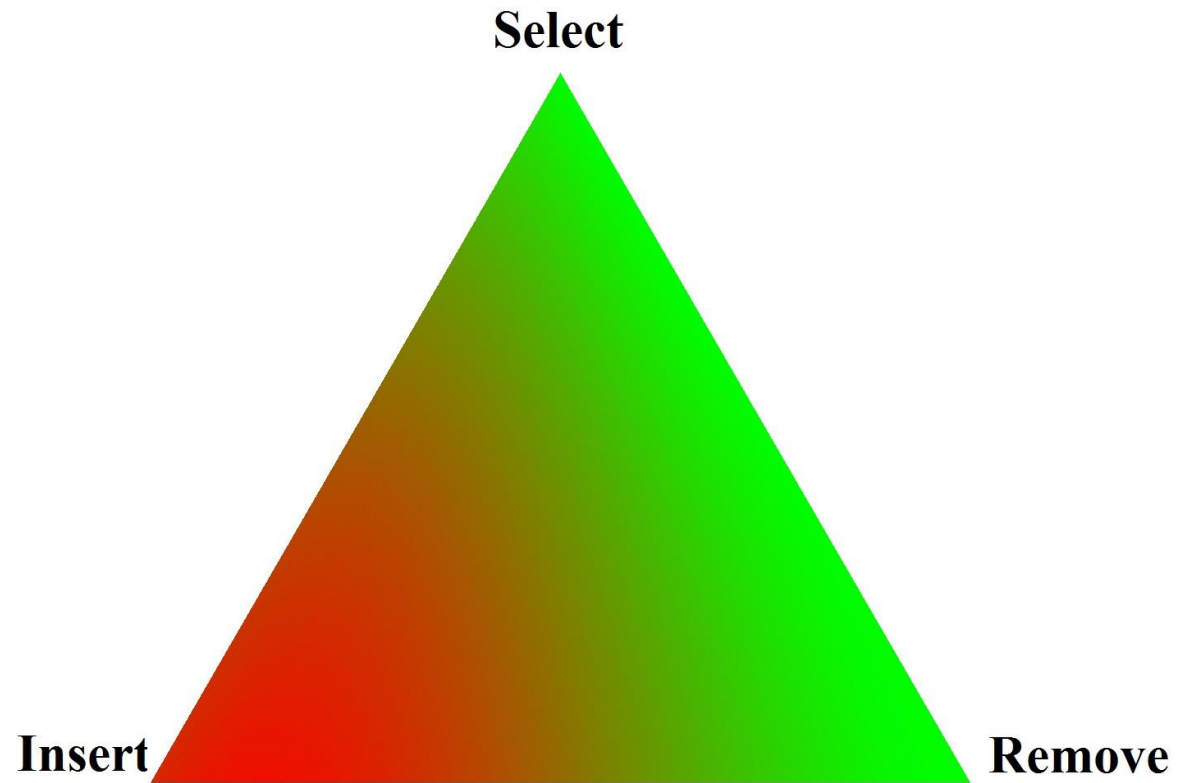
# Тестирование простого списка

---



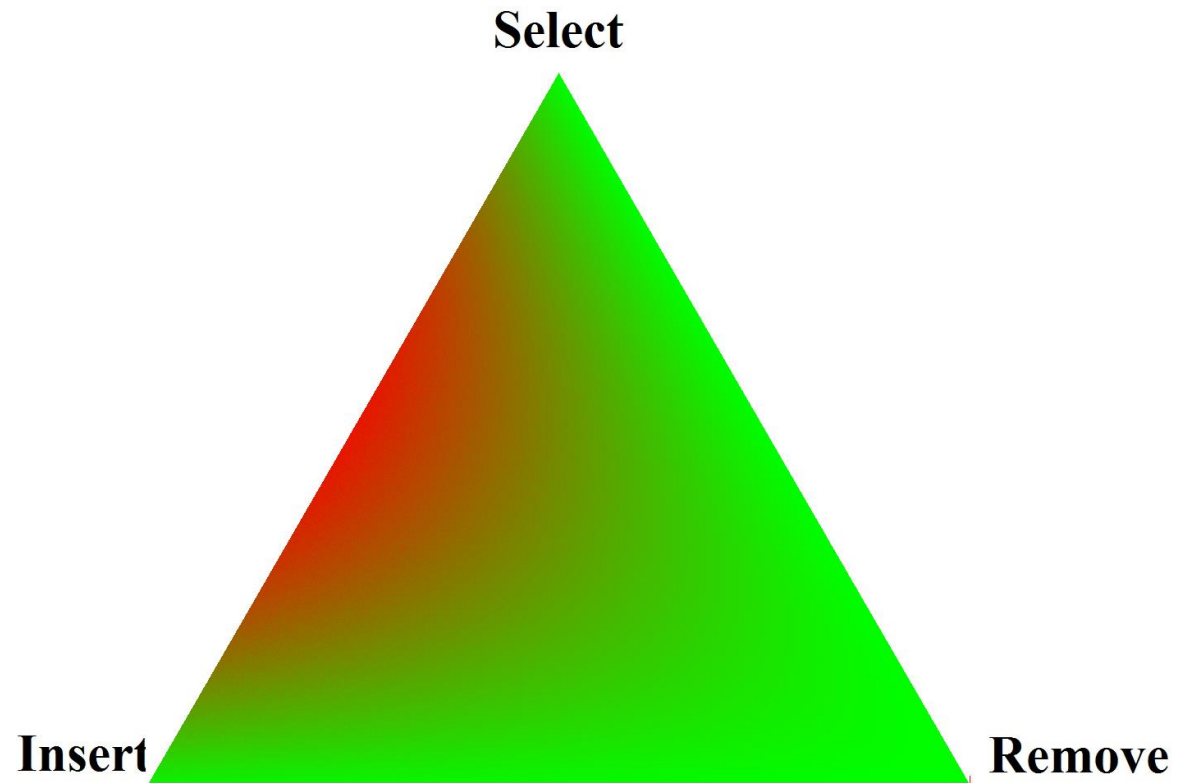
# Тестирование сортированного списка

---



# Тестирование В-дерева

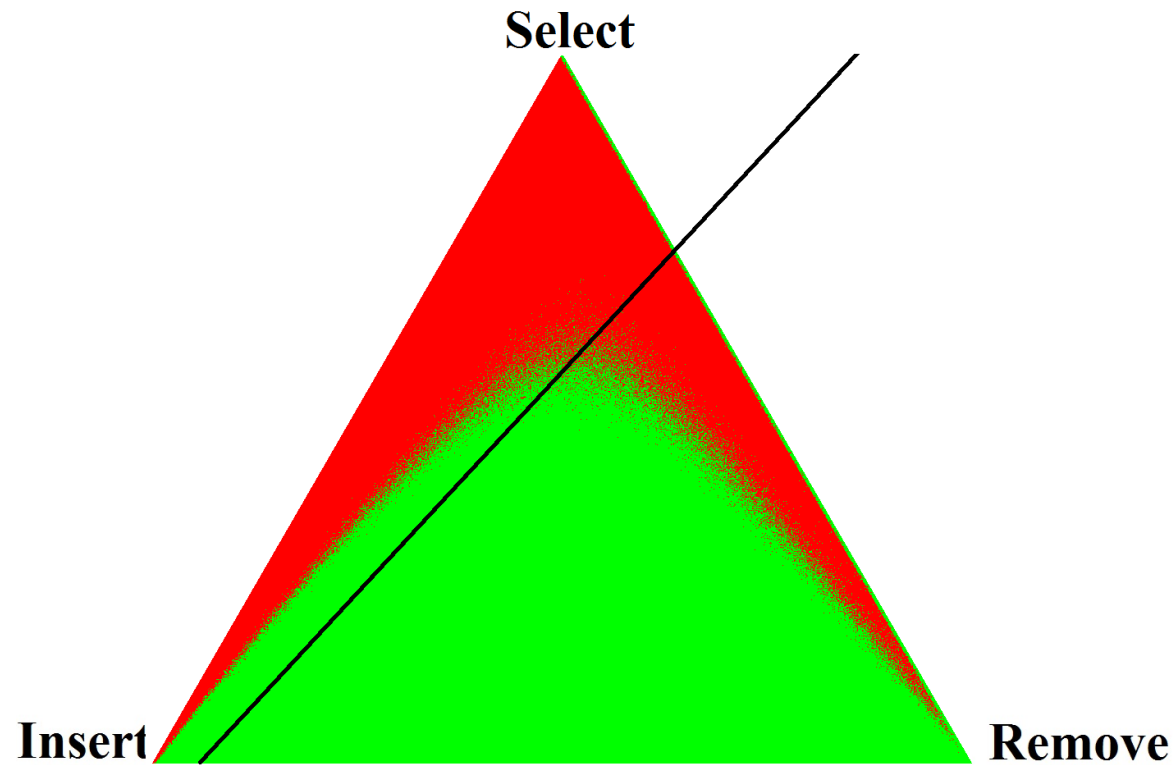
---





# Тестирование простого массива и В-дерева

---



# Результаты работы

---

Разработано программное обеспечение, в котором реализованы:

1. Визуализация нагрузки на одном контейнере.
2. Визуализация сравнения нескольких контейнеров.
3. Алгоритм получения линейного классификатора для адаптивного выбора оптимального способа хранения данных в зависимости от нагрузки.