**КТ 2 Хеширование Колонин**

**Задание 1**

**Пусть х – четырехзначное число.** Являются ли следующие примеры функций Hash(x) хорошими хеш-функциями? И почему?

**i)** Hash (x) = random (0, 3000) (возвращает случайное число от 0 до 3000)

Данная Хеш-функция **не является хорошей**, т.к. она не детерминированная. Подавая одно и тоже значение хеш будет определяться каждый раз случайно.

Данная функция также обладает большой коллизией, что плохо.

**ii)** Hash (x) = x (mod 1301) (возвращает остаток от деления на 1301)

Деление с остатком от простого числа – это **хорошая хеш функция.**

Хеш легко вычислить для любого входящего сообщения

Минусом будет являться то, что хеши не будут распределяться равномерно, если на вход подаются числа х, которые не сильно отличаются друг от друга.

**iii)** Hash (x) = len (x) (возвращает длину строки)

**Плохая хеш функция.**

Данная функция имеет огромное число коллизий, что плохо.

Разные сообщения одной длины будут генерировать одинаковый хеш.

**iv)** Hash (x) = x[0] + x[1] + x[2] +x[3] (возвращает сумму цифр)

Небольшое изменение во входящем сообщении приведёт к небольшому изменению хеш кода. Поэтому **данная хеш функция плохая**, т.к. её можно будет взломать, если выявить эту закономерность.

**v)** Hash (x) = x[0] + x[1]\*5 + x[2]\*25 +x[3]\*125

Распределение хешей у этой функции будет намного лучше, т.к. числа с небольшими изменениями на выходе будут давать достаточно разные хеши.

Разгадать входящее сообщение будет сложно. **Поэтому функция хороша.**

**Задание 2**

Как будет выглядеть хеш-таблица после того, как в нее поместили последовательно элементы с ключами 5, 28, 19, 15, 20, 33, 12, 17, 10, число позиций в таблице = 9,

хеш-функция имеет вид Hash (k) = k (mod 9**)**

**Расчёты:**

Размер таблицы = Hashsize = 9

Hash(5) = 5 mod 9 = 5

Hash(28) = 28 mod 9 = 1

Hash(19) = 19 mod 9 = 1

Hash(15) = 15 mod 9 = 6

Hash(20) = 20 mod 9 = 2

Hash(33) = 33 mod 9 = 6

Hash(12) = 12 mod 9 = 3

Hash(17) = 17 mod 9 = 8

Hash(10) = 10 mod 9 = 1

ВАЖЕН ПОРДОК В КОТОРОМ ДОБАВЛЯЮТСЯ ЭЛЕМЕНТЫ В ТАБЛИЦУ

**Значение хеша от входных данных = индексу в массиве, что представляет хеш таблицу.**

Индексы в таблице начинаются с 0 (во-первых, т.к. массив. Во-вторых, т.к. остаток от деления на 9 не может быть = 9)

Хорошая хеш таблица должна заполняться равномерно.

**Значение хеша от входных данных = индексу в массиве, от которого будет производиться вставка данных.**

Индексов в таблице будет много. Заполнение будет происходить вместе с изменением вида массива.

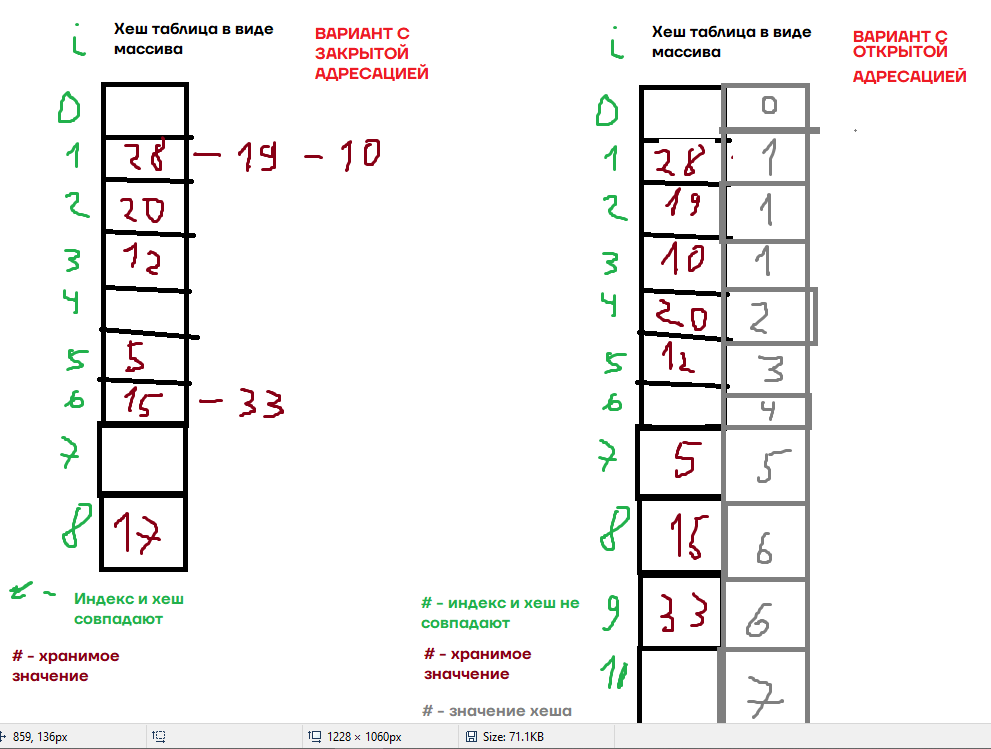
Размышления:

В таблицу на 9 позиций помещают 9 разных ключей.

**i) С закрытой адресацией** => будет создаваться список конфликтующих элементов

Итоговый вид таблицы:

**ii) С открытой адресацией =>** конфликтующие элементы будут вставляться друг за другом.



**Задание 3**

Борисов предполагает, что хеш-таблица с закрытой адресацией будет эффективнее, если вторичная структура хранения будет упорядочена. Прав ли он - как упорядоченность повлияет на эффективность поиска, вставки, удаления?

**РЕШЕНИЕ:**

Пусть упорядоченная структура – это массив.

В упорядоченном массиве можно осуществлять бинарный поиск. Бинарный поиск – это очень быстро. В такой структуре поиск элементов будет происходить быстрее. (log).

На эффективности вставки это скажется плохо, если мы решим поддерживать упорядоченность, т.к. перед каждой вставкой будет нужно найти правильное место для элемента.

Удаление элементов из упорядоченной структуры происходит также, как и из неупорядоченной. Поэтому эффективность этой операции останется прежней.

Правдивость высказывания Борисова зависит от того, какие операции чаще будут производиться с этой структурой данных (для чего она разрабатывается).

Если поиск будет осуществляться чаще, чем вставка, то да, хеш таблица будет эффективнее.

**С Днём Рождения!**