**Алгоритми та складність**

**Завдання № 1**

**Ідеальне хешування**

**Звіт**

**Виконала:**

**Гур’янова Дар’я група к-28**

**Київ-2018**

**Ідеальне хешування**

**•** Умова завдання. Реалізувати ідеальне хешування.

• **Основна ідея**

Ідеальне хешування використовується в задачах із статичним безліччю ключів (тобто після того, як всі ключі збережені в таблиці, їх безліч ніколи не змінюється) для забезпечення гарної асимптотики навіть в гіршому випадку. При цьому ми можемо додатково хотіти, щоб розмір таблиці залежав від кількості ключів лінійно.

В такому хешуванні для доступу до даних потрібно лише обчислення хеш-функцій (однієї або декількох), що робить даний підхід наібистрейшего для доступу до статичних даних. Дана технологія застосовується в різних словниках і базах даних, в алгоритмах із статичною (відомої заздалегідь) інформацією.

**• Опис алгоритму.**

1)**перший етап**

n ключів хешіруются в m комірок з використанням хеш-функції h (k) = ((a⋅k + b) modp) modm

випадково обраної з сімейства універсальних хеш-функцій Hp, m

Hp,m, Де p- просте число, що перевищує m

2)**другий етап**

На даному рівні замість створення списку ключів будемо використовувати вторинну хеш-таблицю Sj

що зберігає всі ключі, хешованих функцією h в комірку j, зі своєю функцією hj (k) = ((aj⋅k + bj) modp) , вибраних з безлічі Hp, mj. Шляхом точного вибору хеш-функції hj ми можемо гарантувати відсутність колізій на цьому рівні. Для цього потрібно, щоб розмір mj хеш-таблиці Sj дорівнює квадрату числа nj ключів, хешованих функцією h в комірку j.

**• Аналіз алгоритму.**

Множина ключів статична – не змінюється після збереження в таблицю

Ο(1) звертань до пам’яті в найгіршому випадку.

Очікувана загальна пам’ять під таку структуру O(n).

**• Реалізація алгоритму.**

Реалізовано на С++.

**• Висновки.**

Даний алгоритм дозволяє виконувати пошук за О(1) звернень до памяті.

**• Використані джерела.**

Т. Кормен «Алгоритмы, построение и анализ».