GitHub решение: https://github.com/kurokolover/algorithms_hm/tree/main/SET5_A3

HyperLogLog

реализованы: генератор потока строк, генератор 32-битной хеш-функции, стандартный HyperLogLog для оценки числа различных элементов N_t на префиксах потока, а также улучшенная версия с упаковкой регистров в 6 бит. Эксперименты выполняются на сгенерированных потоках, сравнение проводится с точным числом уникальных строк.

1. Генерация данных

Поток состоит из строк длиной до 30 символов. Алфавит: латиница (A–Z, a–z), цифры 0–9, дефис -. Генерация управляется `seed`. Для имитации момента времени t поток разбивается на равные доли с шагом `step` (например, 0.05 = 5%).

Чтобы поток был реалистичным (не все элементы уникальны), используется параметр `repeat` — доля элементов, которая с заданной вероятностью берётся из ранее появлявшихся строк (контролируемые повторы).

Код: `code/src/hll.hpp` (класс `RandomStreamGen`).

2. Хеш-функция $h : U \to \{0..2^{\{32\}} - 1\}$

Используется MurmurHash3 (вариант x86 32-bit) со случайным `seed` внутри `HashFuncGen`. Цель — близкое к равномерному распределение значений по 32-битному пространству.

Код: `code/src/hll.hpp` (класс `HashFuncGen`, функция `murmur3_32`).

3. Стандартный HyperLogLog

Пусть B — число старших бит для индекса регистра, тогда число регистров $m = 2^B$. Для каждого элемента вычисляем $x = h(s)$.

- индекс регистра: $j = x[0..B-1]$ (старшие B бит),
- оставшиеся биты используются для вычисления ρ — позиции первого единичного бита (число ведущих нулей + 1) в хвосте.

Обновление: $M[j] \leftarrow \max(M[j], \rho)$.

Оценка мощности множества:

$Z = \left(\sum_{j=0}^{m-1} 2^{-M[j]}\right)^{-1}$,

$E = \alpha_m \cdot m^2 \cdot Z$.

Константа:

$\alpha_m = 0.673$ при $m = 16$; $\alpha_m = 0.697$ при $m = 32$; $\alpha_m = 0.709$ при $m = 64$; иначе $\alpha_m = \frac{0.7213}{1+\frac{1.079}{m}}$.

Исправления:

- малые мощности: если $E \le 2.5m$ и число нулевых регистров $V > 0$, то $E \leftarrow m \cdot \ln\left(\frac{m}{V}\right)$ (linear counting);
- очень большие: для 32-битного хеша, если $E > \left(\frac{1}{30}\right) \cdot 2^{\{32\}}$, то $E \leftarrow -2^{\{32\}} \cdot \ln\left(1 - \frac{E}{2^{\{32\}}}\right)$.

Выбор B: в экспериментах использовано $B = 14$ (то есть $m = 16384$) как компромисс память/точность и для хорошего распределения по регистрам.

4. Эксперименты и графики

Для каждого потока и каждого шага времени вычисляются:

- точное число уникальных строк F_0^t (через `unordered_set`),
- оценка N_t (HyperLogLog).

по всем потокам для каждого шага считаются среднее $E(N_t)$ и стандартное отклонение σ_{N_t}.

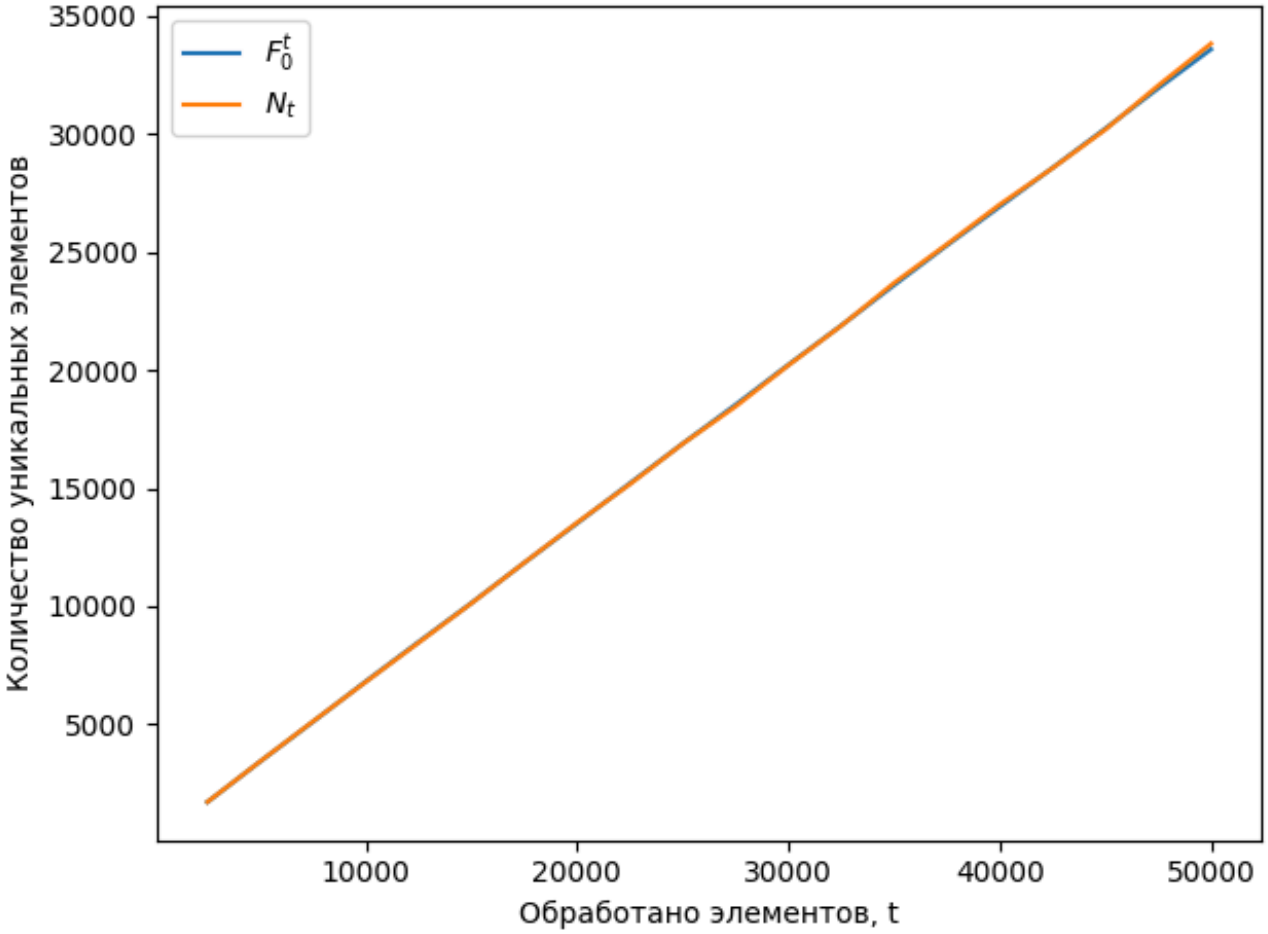

Рис. 1. График №1: F_0^t и N_t (standard).

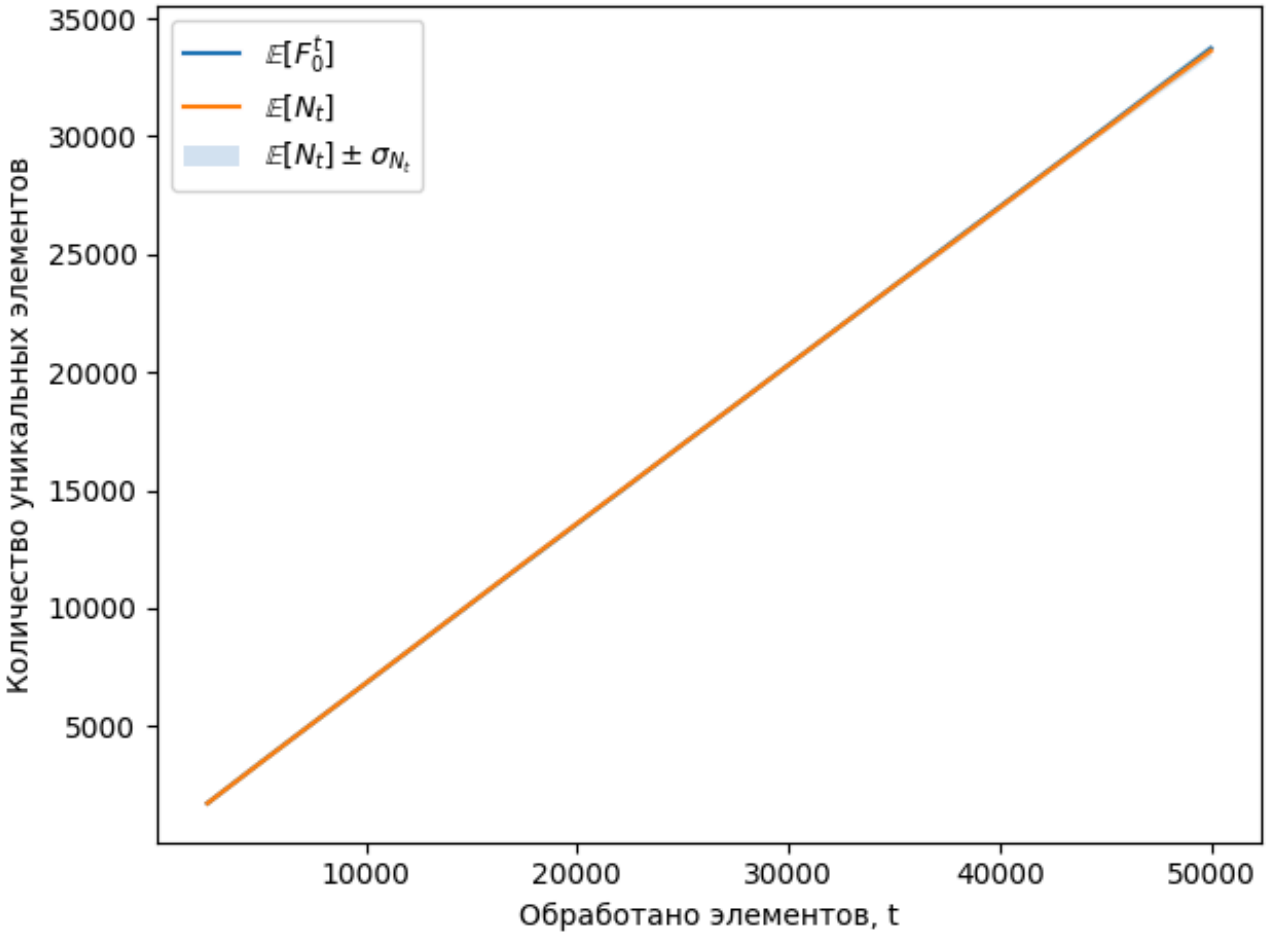

Рис. 2. График №2: $E(N_t)$ и $E(N_t) \pm \sigma_{N_t}$ (standard).

5. Сравнение с теорией

Теоретическая относительная стандартная ошибка (RSE) для HLL приближённо: $\text{RSE} \approx \frac{1.04}{\sqrt{m}}$.

При $B = 14$ имеем $m = 16384$ и $\text{RSE}_{\text{theory}} \approx 0.008125$.

По данным эксперимента (последний шаг, $t = 1$) получено:

- $F_0^1 \approx 33723.45$
- $E(N_1) \approx 33618.74$
- $\sigma_{N_1} \approx 265.60$
- относительное смещение: −0.311%
- $\text{RSE}_{\text{emp}} = \frac{\sigma}{E} \approx 0.007900$

6. Улучшения (этап 4)

Упаковка регистров в 6 бит: регистры хранятся в битовом массиве (в 64-битных словах), чтобы снизить память при сохранении точности.

Память:

- standard: m байт,
- packed: $\left\lceil 6\frac{m}{8} \right\rceil$ байт (≈ 0.75m).

Для $m = 16384$:

- standard: 16384 байт,

- packed: 12288 байт (экономия 25%).

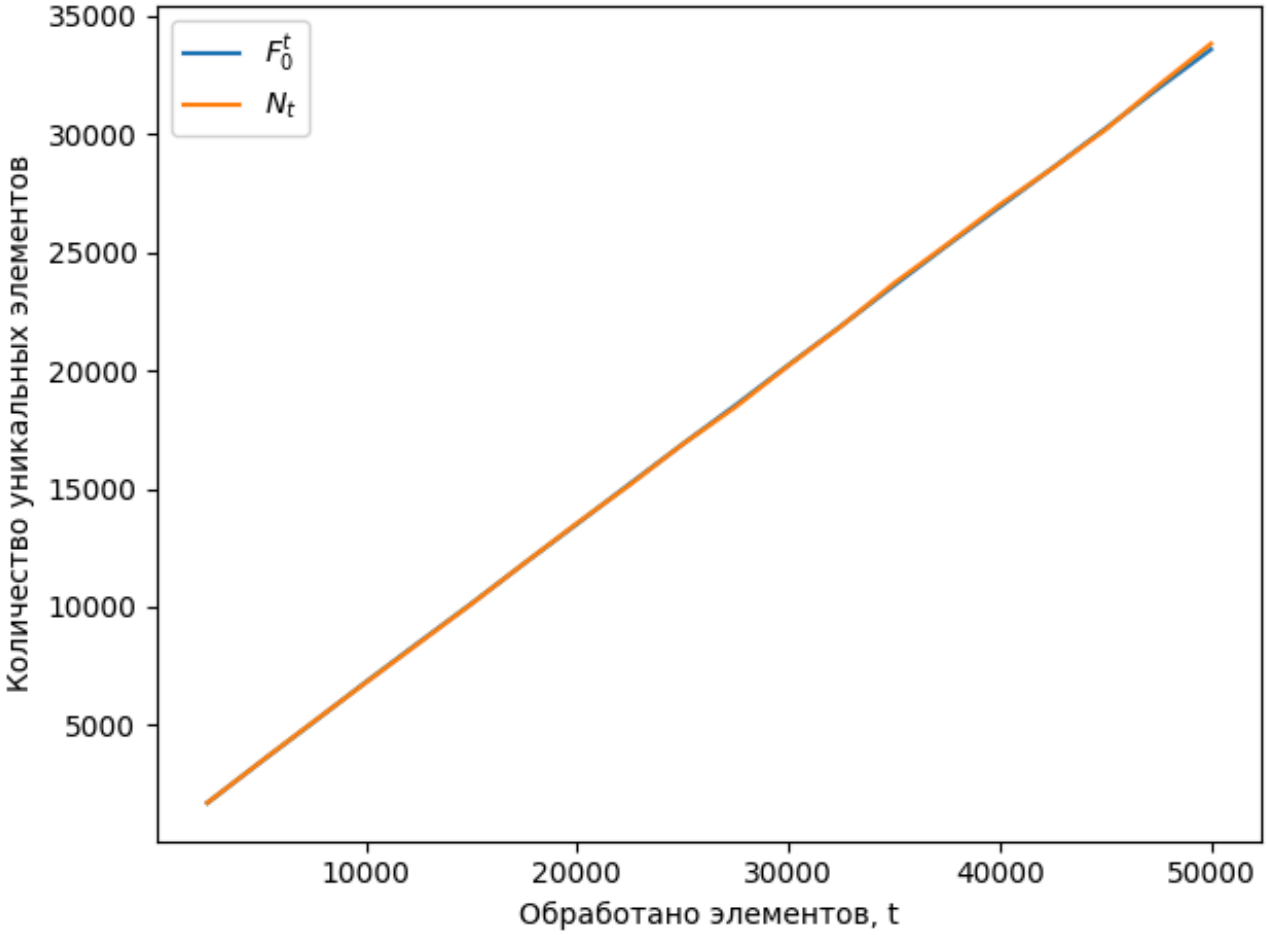

Рис. 3. График №1: F_0^t и N_t (packed).

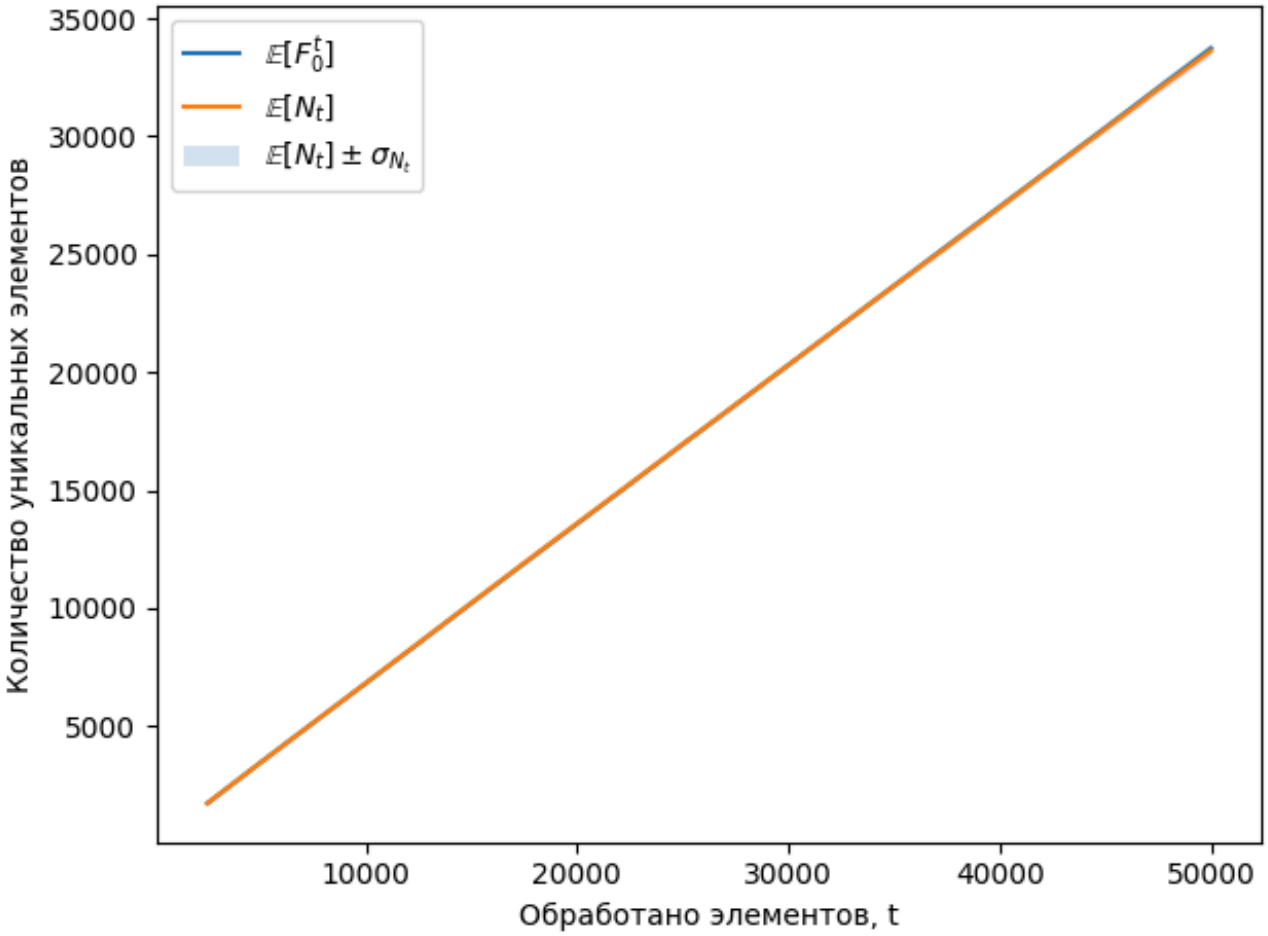

Рис. 4. График №2: $E(N_t)$ и $E(N_t) \pm \sigma_{N_t}$ (packed).