1 Сортировки основанные на внутреннем виде данных

Имеем n чисел $[0, U-1], U=2^w$, числа укладываются в RAM-модель

Сортировка подсчетом Заводим массив cnt[U], $cnt[x] = |\{i: a_i = x\}|$. Дальше переводим $count \to pref$, pref[x] = pref[x-1] + count[x] O(n+U)

Поразрядная сортировка b_{ij} —j-й бит i-го числа. (Сортируем бинарные строки длины w)

Поочередно сортируем строки, разбивая их на классы эквивалентности по iter последним символам. После чего мы стабильно сортируем по (iter + 1)-му символу.

 $O(n \log U)$

Bucket sort Разбиваем множество на корзины, каждой корзине соответствует отрезок. В каждой корзине запускаемся рекурсивно.

 $O(n \log U)$

В продакшне используют первую пару итераций, чтобы сильно снизить размерность на реальных данных.

Пусть мы хотим отсортить равновероятные числа из [0,1]. В каждом бакете отсортируем за квадрат. Получим O(n).

$$t(n) \le \sum_{i=1}^{n} c \cdot (1 + E(cnt_i)^2) \le c \cdot n + c \cdot \sum_{i=1}^{n} E(cnt_i^2) =$$

$$= c \cdot n + c \cdot \sum_{i=1}^{n} \sum_{j=1}^{n} EI_{A_{ij}} = c \cdot n + c \cdot n^2 \cdot \frac{1}{n} \le 2 \cdot c \cdot n$$

2 Иерархия памяти

Нас интересует задержка (latency), пропускная способность (throughput). Подгрузка x данных занимает $l+\frac{x}{t}$

От долгой к быстрой

- 1. external machine / internet
- 2. HDD
- 3. SSD
- 4. RAM
- 5. L3
- 6. L2
- 7. L1
- 8. registers

3 Алгоритмы во внешней памяти

```
n — размер задачи M — размер RAM B — блок данных B \ll M \ll n \log n \ll B B < \sqrt{M} (но это неявно и не факт)
```

Mergesort во внешней памяти Обычный mergesort, но три типа событий в merge:

- 1. Кончился первый буфер подгружаем новый
- 2. Кончился второй аналогично
- 3. Кончился буфер для слияния выписываем обратно в RAM и сбрасываем

$$O(\frac{n}{B}\log n) o O(\frac{n}{B}\log \frac{n}{B}) o O(\frac{n}{B}\log \frac{m}{B}\frac{n}{B}) + 2$$
идеи:

- 1. Дошли до размера M явно посортим в RAM
- 2. Можем сливать сразу $\frac{M}{B}$ массивов