Иванов А.Б.

Алгоритмы и структуры данных

Сортировки

Quicksort: анализ (второй метод)

$$nC_n = (n+1)C_{n-1} + 2n$$

$$\begin{aligned}
\frac{C_n}{n+1} &= \frac{C_{n-1}}{n} + \frac{2}{n+1} = \\
&= \frac{C_{n-2}}{n-1} + \frac{2}{n} + \frac{2}{n+1} = \\
&= \frac{C_1}{2} + \frac{2}{3} + \ldots + \frac{2}{n} + \frac{2}{n+1}
\end{aligned}$$

$$C_n \sim 2(n+1)\sum_{k=1}^{n+1}\frac{1}{k} - C \cdot n \sim 2n \ln n$$

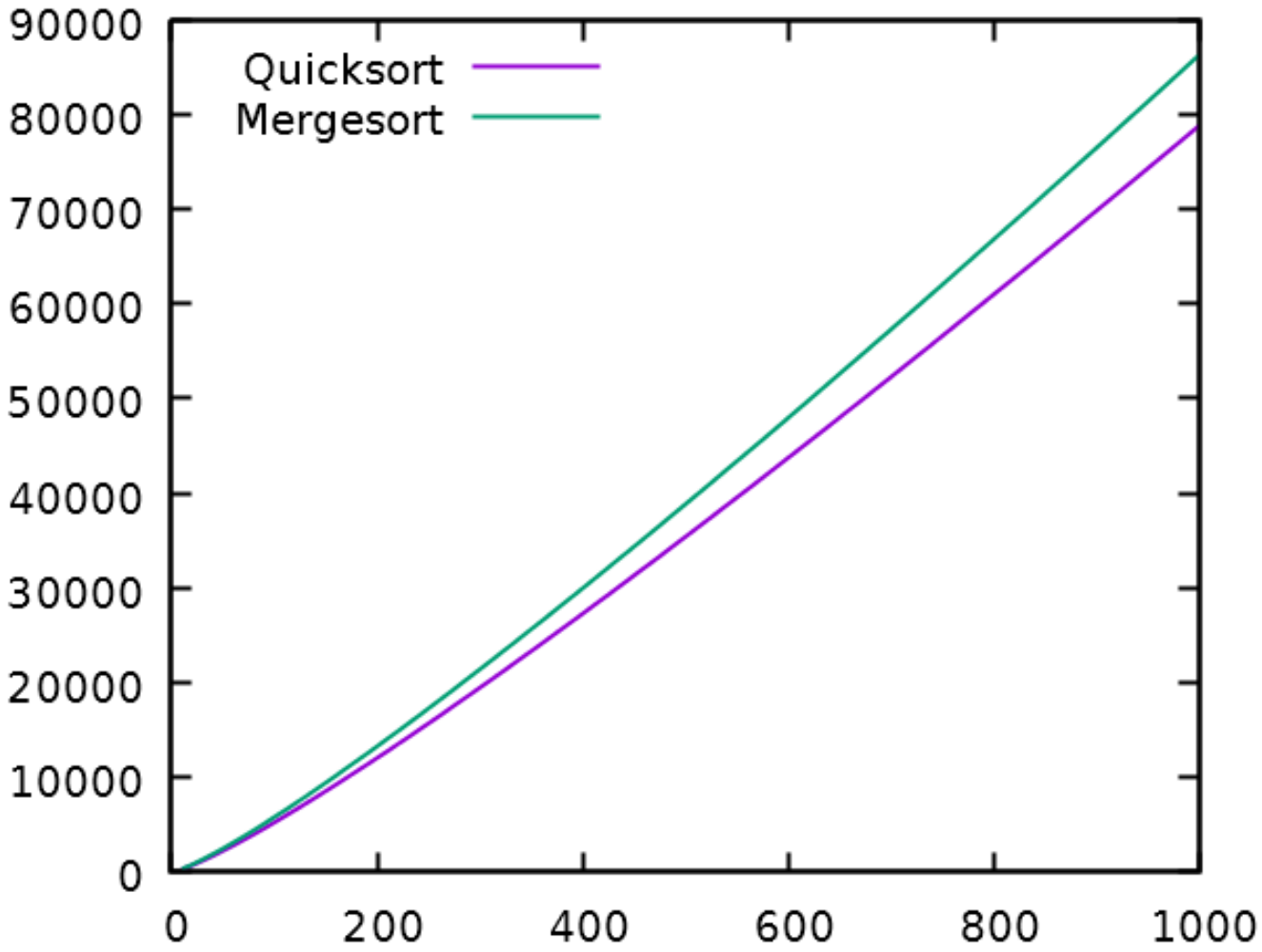
Быстрая сортировка: зачем?

- Сложность
- Необходимая память
- Устойчивость

- $O(n \log n)$
- $\approx \log(n)$
- не является устойчивым

Быстрая сортировка: зачем?

Сортировка слиянием: $12.5\,x\log(x)$
Быстрая сортировка: $11.67\,x\log(x) - 1.74\,x$



Быстрая сортировка с трехчастным разбиением (3-way quicksort)

Какое время работы алгоритма на массиве, состоящем из одинаковых элементов?

Быстрая сортировка с трехчастным разбиением (3-way quicksort)

Какое время работы алгоритма на массиве, состоящем из одинаковых элементов?

Стандартный quicksort: $\Theta(n^2)$.

Решение: 3-way quicksort.

Разбиваем массив на 3 части:

- $a[i] < p$ для $0 \le i \le r$
- $a[i] = p$ для $r + 1 \le i \le s$
- $a[i] > p$ для $s + 1 \le i \le n - 1$

3-way quicksort

Задача о голландском национальном флаге (Edsger Wybe Dijkstra)

Отсортировать массив, состоящий из 0, 1 и 2

3-WAY QUICKSORT

Инвариант цикла:

$a[j] < p$
при $j \in [low+1, lt)$

$a[j] = p$
при $j \in [lt, i)$

$a[j] > p$
при $j \in (gt, high]$

```
procedure QuickSort3Way(a[]):
  // a[] - массив от 0 до n-1
 перемешать (shuffle) массив a
  QuickSort3Way(a, 0, n-1)
end

procedure QuickSort3Way(a[], low, high):
  p = a[low]
  i = low + 1
  lt = low + 1
  gt = high
  while i <= gt:
    if a[i] < p:
      Exch(a, lt++, i++)
    else if a[i] > p:
      Exch(a, i, gt--)
    else
      i++
  end
  Exch(a, low, --lt)

  QuickSort3Way(a, low, lt - 1)
  QuickSort3Way(a, gt + 1, high)
end
```

QuickSort: переполнение стека

В самом худшем случае будет n рекурсивных вызовов, что может привести к переполнению стека.

QuickSort: переполнение стека

В самом худшем случае будет n рекурсивных вызовов, что может привести к переполнению стека.

Решение: вместо двух рекурсивных вызовов делать только один (для части меньшего размера). Вторую (большую) часть обрабатывать, не используя рекурсивный вызов.

Практические улучшения

На массивах небольшого размера выгоднее использовать сортировку вставками.

Выбор опорного элемента: медиана из трех (median-of-3). Выбираем 3 случайных элемента и в качестве опорного берем второй по порядку (медиану).

Tukey's ninther: выбираем 3 тройки случайных элементов, в каждой тройке находим медиану и в качестве опорного берем второй медиану медиан.

Методы сортировок: характеристики

Алгоритм	stable	in-place	Время	Доп. память
Selection	−	+	N^2	1
Insertion	+	+	между N и N^2	1
Shell	−	+	$N \log N$? $N^{6/5}$?	1
Quicksort	−	+	$N \log N$	$\log N$
3-way quicksort	−	+	между N и $N \log N$	$\log N$
Mergesort	+	−	$N \log N$	N
Timsort	+	−	между N и $N \log N$	между 1 и N
Heapsort	−	+	$N \log N$	1
???	+	+	$N \log N$	1

QuickSelect

Дано: массив размера n, $1 \leq k \leq n$
Найти: k-ю порядковую статистику

QuickSelect

Дано: массив размера n, $1 \le k \le n$
Найти: k-ю порядковую статистику

Пусть $C(n, k)$ – матожидание количества сравнений

$$C(n, \cdot) \le n - 1 + \frac{1}{n} \sum_{i=1}^{n} C(\max\{i - 1, n - i\}, \cdot))$$

$$C(n, \cdot) \le n - 1 + \frac{2}{n} \sum_{i=1}^{n/2} C(n - i, \cdot))$$

Индукция: $C(m, \cdot) \le cm$ для $m < n \Rightarrow$

$$C(n, \cdot) \le \left(1 + \frac{3}{4}c\right) n \le cn$$

Детерминированный Select[1]

Blum
Floyd: Алгоритм Флойда
Pratt: Алгоритм Кнута — Морриса — Пратта
Rivest: RSA (Rivest, Shamir, Adleman)
Tarjan: Алгоритм Тарьяна

`Select(массив размера n, k)`

- разбиваем на пятерки ($m = \lceil n/5 \rceil$), сортируем каждую, выбираем медиану в каждой пятерке
- находим медиану p: `Select(массив размера m, m/2)`
- выполняем `Partition` с опорным элементом p
- определяем в какой из частей находится k-я статистика, вызываем `Select` рекурсивно

---

[1] Blum, M.; Floyd, R. W.; Pratt, V. R.; Rivest, R. L.; Tarjan, R. E. (August 1973). "Time bounds for selection". Journal of Computer and System Sciences. 7 (4): 448–461

Детерминированный Select (медиана медиан)

$$T(n) \leq T\left(\frac{n}{5}\right) + T\left(\frac{7n}{10}\right) + O(n)$$

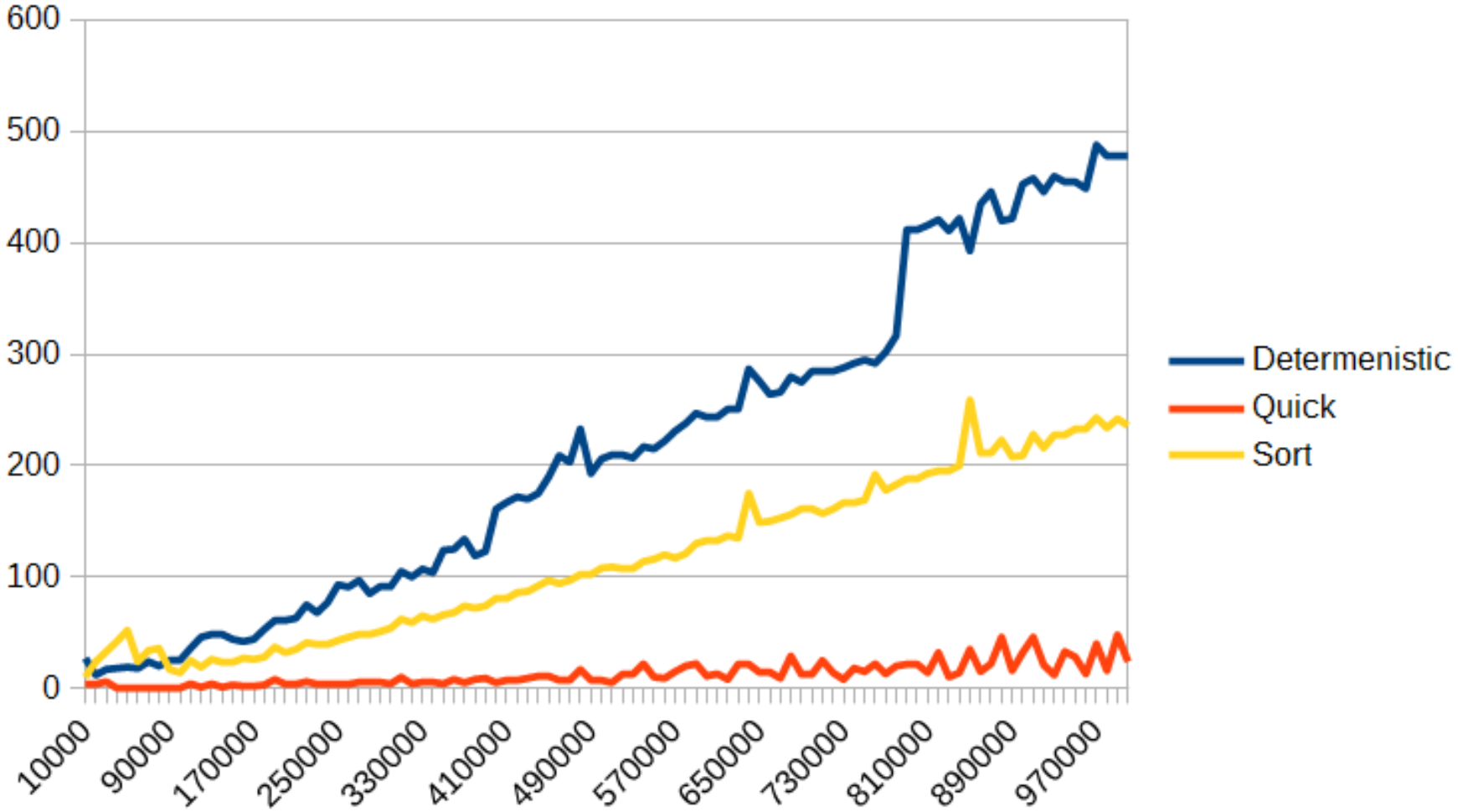
или

$$T(n) \leq T\left(\frac{n}{5}\right) + T\left(\frac{7n}{10}\right) + an$$

угадаем и докажем по индукции:

$$T(n) \leq Cn, \textbf{ где } C = 10a$$

Сравнение реализаций Select



Сортировка подсчетом

Задача: отсортировать массив из N чисел от 0 до $R - 1$.

Наивная сортировка подсчетом:
Посчитать количество 0, количество 1, количество 2...
Расставить в массиве нужное количество 0, нужное количество 1...

Не годится. На практике надо уметь сортировать объекты по некоторому ключу (например: людей по росту). Уточненная формулировка:

Задача: U – пространство объектов, $f : U \to [0..R - 1]$.
Отсортировать массив из N объектов по возрастанию $f(x)$.

Сортировка подсчетом

Задача: U – пространство объектов, $f : U \to [0..R-1]$.
Отсортировать массив из N объектов по возрастанию $f(x)$.

- Посчитаем количество объектов для каждого значения ключа: $c_i = \{x : f(x) = i\}$.
- На каких позициях в отсортированном должны стоять элементы с ключом 0? С ключом 1? С ключом i?
- Посчитаем нарастающим итогом (running total)...
  0, c_0, $c_0 + c_1$, $c_0 + c_1 + c_2$, ...
- Пройдем по массиву: для каждого элемента знаем позицию, куда его надо поставить.

Сортировка подсчетом

Задача: U – пространство объектов, $f : U \to [0..R-1]$.
Отсортировать массив из N объектов по возрастанию $f(x)$.

`count[i]` – счетчик объектов/на какой позиции нужно поставить очередной объект. `aux` – вспомогательный массив

```
int N = a.length;
int[] count = new int[R+1];

for (int i = 0; i < N; i++)
    count[a[i]+1]++;

for (int r = 0; r < R; r++)
    count[r+1] += count[r];

for (int i = 0; i < N; i++)
    aux[count[a[i]]++] = a[i];

for (int i = 0; i < N; i++)
    a[i] = aux[i];
```

Сортировка подсчетом

- Сложность $O(N + R)$
- Дополнительная память $O(N + R)$
- Устойчивая

Дальше: поразрядные сортировки (radix sorts).