University of Ljubljana
Faculty of Computer and Information Science

Quicksorts

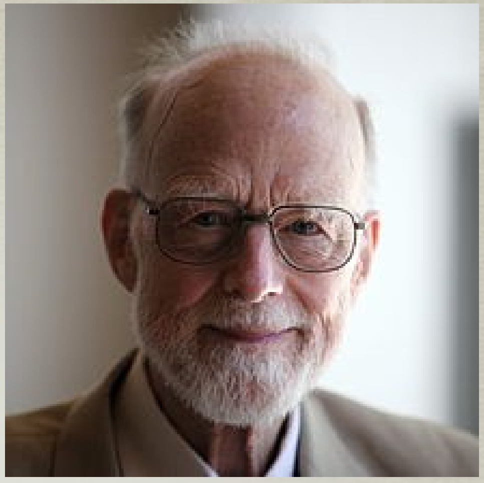
When in doubt sort

-- S. Skiena

Jurij Mihelič, UniLj, FRI

A brief history

C. A. R. Hoare, 1934



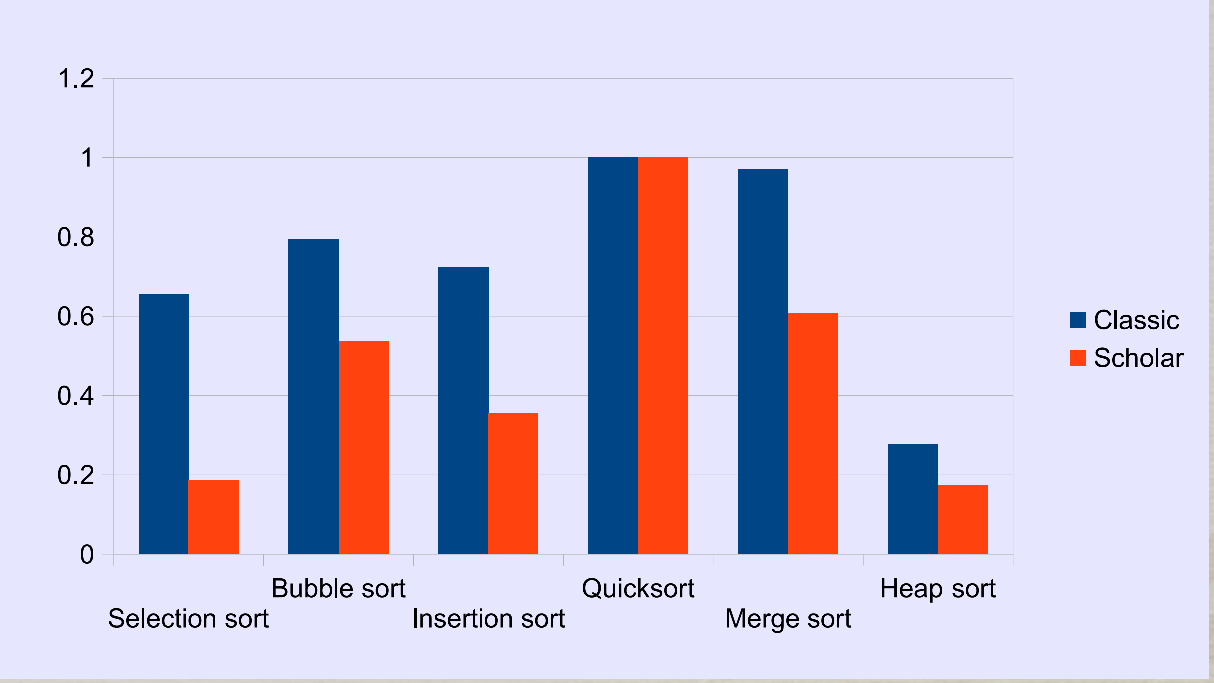
Quicksort, 1960

- Old age
  - 1959, discovered, Hoare
  - 1961, published, Hoare
  - 1975, extensive analysis, Sedgewick
  - 1993, engineering, Bentley
- New age
  - 2009, Yaroslovsky's Java7 sort
  - 2012, dual-pivot analysis, Wild, Nebel
  - 2013, triple-pivot analysis, Kushagra, Lopez-Ortiz, Munroe, Qiao

Popularity by Google

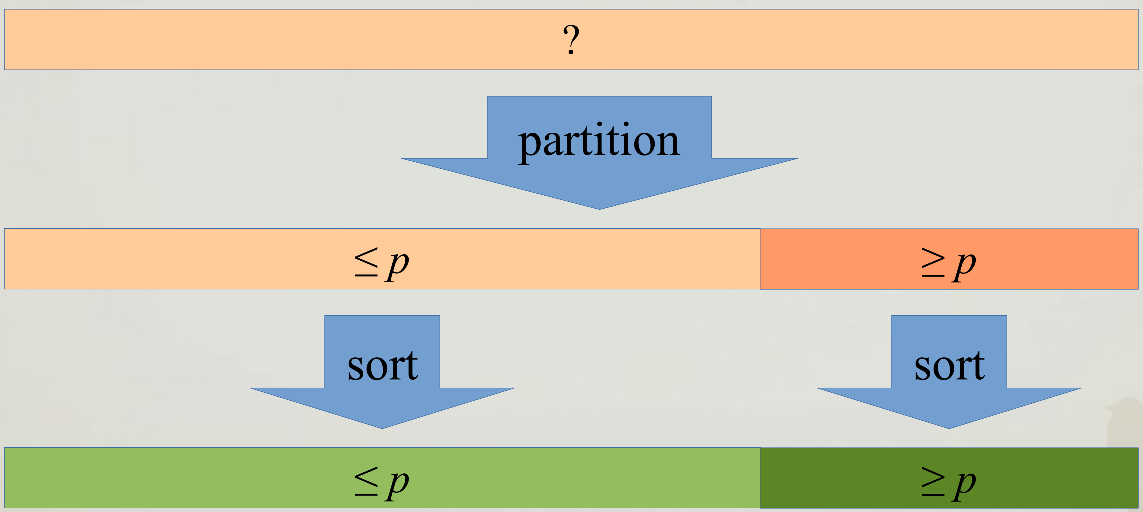
Search term	Classic	Scholar
Selection sort	358 000	3 400
Bubble sort	434 000	9 720
Insertion sort	395 000	6 440
Quicksort	**546 000**	**18 100**
Merge sort	430 000	11 000
Heap sort	152 000	3 170

Date: 30th of June, 2015 (search term in parenthesis)



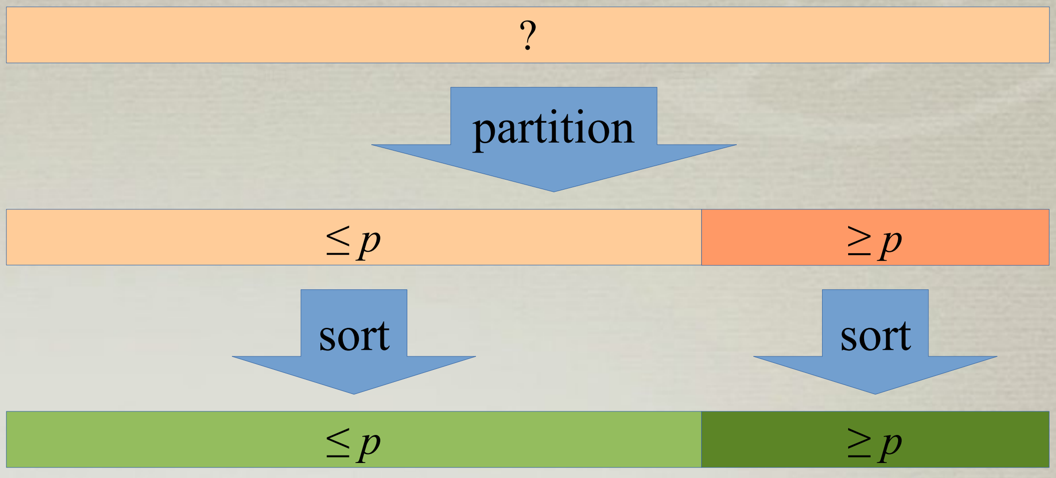
Memory refresh

- Divide & conquer
  - partition around pivot and recurse
  - in-place partitioning



Memory refresh

- Pseudocode




```
fun qs(int a[], int left, int right)
    if (left <= right) return
    p = choose_pivot(a, left, right)
    m = partition(a, p, left, right)
    qs(a, left, m - 1)
    qs(a, m, right)
```

Memory refresh

- Complexity analyis
  - comparison-based model of computation
  - best case: $\Theta(n \log n)$
  - worst case: $\Theta(n^2)$
    - very rare on random inputs
  - average case: $\Theta(n \log n)$
    - distinct elements
    - equiprobable inputs

$$C_n = (n+1) + \frac{1}{n} \sum_{i=0}^{n-1} (C_i + C_{n-1-i})$$

Memory refresh

- Complexity analysis
  - comparison-based model
  - best case: $O(n \log n)$
  - worst case: $O(n^2)$
    - very rare on random inputs
  - average case: $O(n \log n)$
    - distinct elements
    - equiprobable inputs
    - #comparisons
      - total: $\sim \mathbf{2\, n \ln n} + O(n)$
      - median of 3: $\sim \mathbf{1.71\, n \ln n} + O(n)$

$$C_n = (n+1) + \frac{1}{n} \sum_{i=0}^{n-1} (C_i + C_{n-1-i})$$

Pivot sampling

- Choose left, middle, or right
  - not robust
- Randomization
  - randomly perturbate the array before sorting
  - choose random element as pivot
- Median
  - of 3, 5, …, all

Small sublists (subarrays)

- How to sort a small subarray?
  - use some other sorting algorithm
  - insertion sort

- When is the subarray small?
  - Sedgewick: smaller than 5 to 15
  - Java 7: smaller than 47

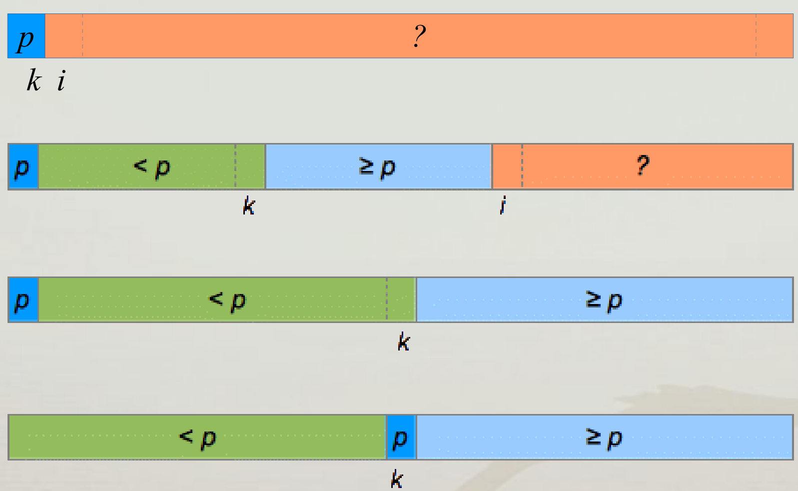
Single-pivot partitioning

- Partitioning schemes
  - inplace (vs “outplace”)
  - Lomuto's single-loop partitioning
    - popularized by Cormen et al.
  - Crossing-pointers partitioning
    - Hoare, Sedgewick, …
  - Three-way partitioning
    - classic, Bentley-McIlroy

For implementations see (on your own risk :)):
https://github.com/jurem/SortingAlgorithms

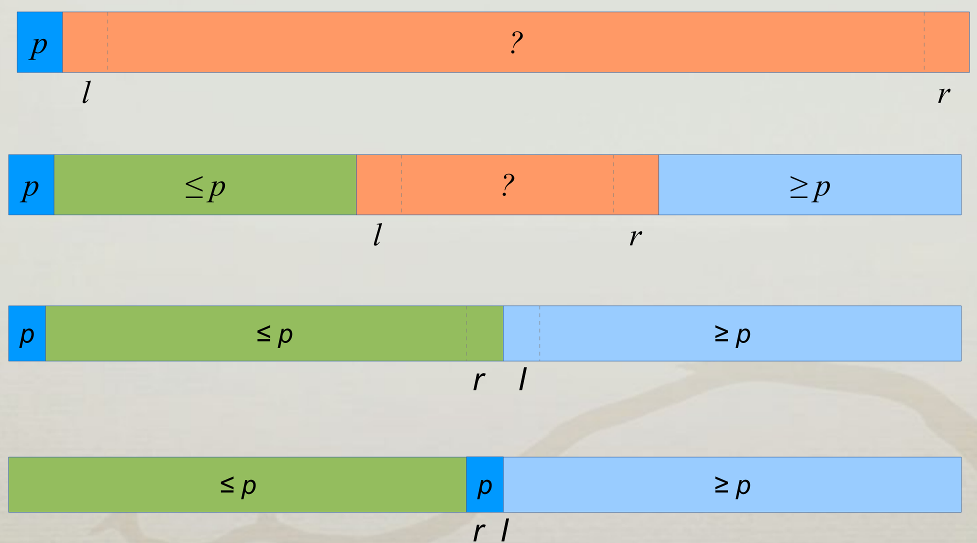
Single-pivot partitioning

- Lomuto's single-loop partitioning
  - Nico Lomuto, popularized by CLRS
  - all equal elements



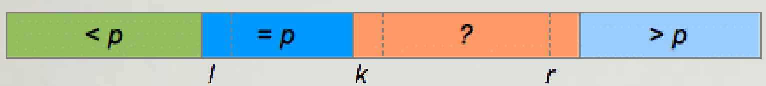
Single-pivot partitioning

- Crossing-pointers partitioning
  - C.A.R. Hoare, R. Sedgewick
  - N. Wirth



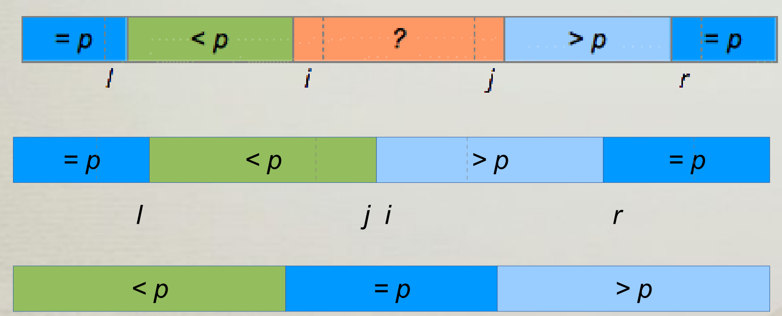
Single-pivot partitioning

- Three-way partitioning
  - if there are many equal elements
  - naive version




```
if (a[k] < p) swap(a, l++, k++);
else if (a[k] > p) swap(a, k, r--);
else k++;
```

  - Bentley-McIlroy version



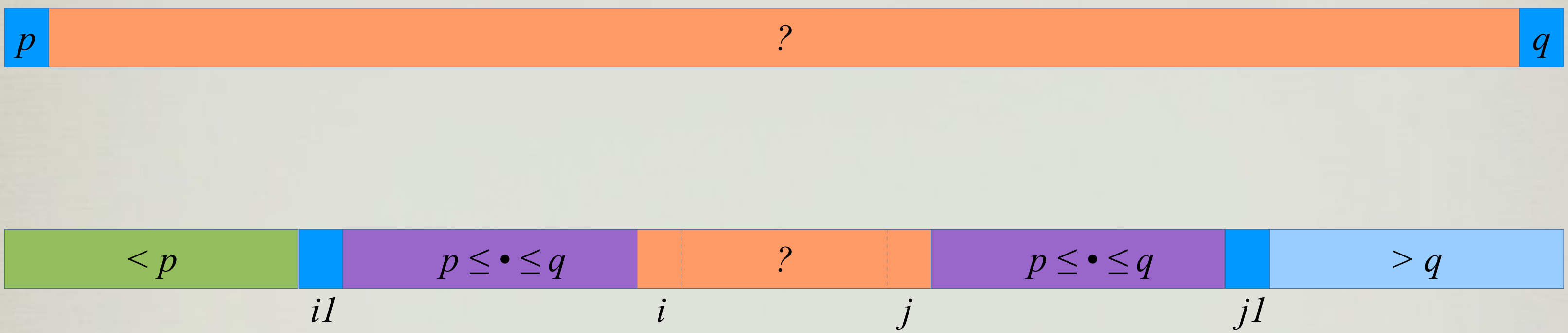

```
while (a[++i] < p) && i < right);
while (a[--j] > p);
if (i >= j) break;
swap(a, i, j);
if (a[i] == p) swap(a, ++l, i);
if (a[j] == p) swap(a, --r, j);
```

Multi-pivot partitioning

- Sedgewick, 1975
  - in-place dual-pivot QS implementation, not better than classic
- Hennequin, 1991
  - partitioning with s pivots, very small savings, complicated partitioning
- *Using multiple pivots does not pay off?*
- Yaroslavskiy, 2009
  - new implementation of dual-pivot partitioning, Java7's stdlib
  - analysis: Wild, Nebel, 2012
- Aumüller, Dietzfelbinger, 2013
- Kushagra, López-Ortiz, Munro, Qiao, 2013

Multi-pivot partitioning

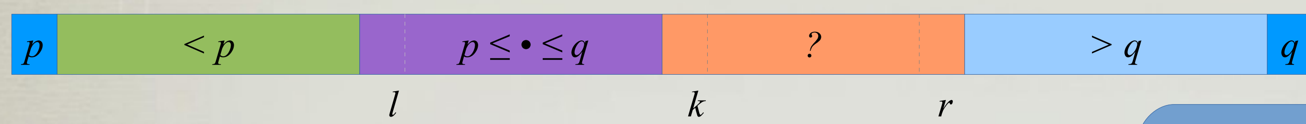
- Sedgewick's dual-pivot QS, 1975
  - based on crossing-pointers technique



  - #comparisons
    - total: $\sim \mathbf{2.13}\ \boldsymbol{n}\ \mathbf{ln}\ \boldsymbol{n} + O(n)$

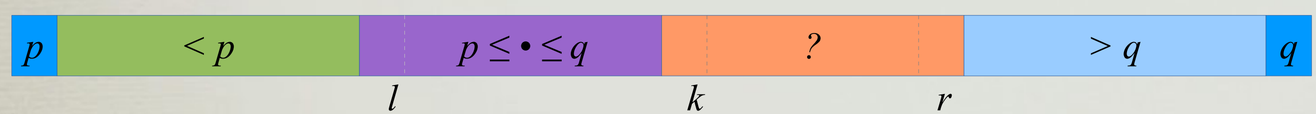
Multi-pivot partitioning

- Yaroslavskiy's dual-pivot QS, 2009
  - simple scheme




```
if (a[k] < p) swap(a, l++, k++);
else if (a[k] > q) swap(a, k, r--);
else k++;
```

  - full scheme




```
if (a[k] < p) swap(a, l++, k);
else if (a[k] > q) {
        while (a[r] > q && k < r) r--;
        swap(a, k, r--);
        if (a[k] < p) swap(a, l++, k);
}
k++;
```

Multi-pivot partitioning

- Yaroslavskiy's dual-pivot QS, 2009
  - analysis, Wild, Nebel, 2012
  - #comparisons
    - simplified partitioning scheme
      - per element: $1/3 \cdot 1 + 2/3 \cdot 2 = 5/3$
    - full partitioning scheme
      - lower than simple: $19/12$
    - total: $\sim 1.9\, n \ln n + O(n)$

Hoare:
- **2 n ln n** – 3 n - 3

Median of 3:
- $\mathbf{1.71\, n \ln n} + O(n)$

Sedgewick 2-pivot:
- **2.13 n ln n** – 2.57 n + O(ln n)

Yaroslavskiy – Wild, Nebel:
- **1.9 n ln n** – 2.46 n + O(ln n)

Multi-pivot partitioning

- Aumüller, Dietzfelbinger, 2013
  - What is the best possible #comparisons?
    - in any dual-pivot partitioning
    - mimimizes expected number of comparisons among all algorithms
  - #comparisons
    - total: $\sim \mathbf{1.8}\, \boldsymbol{n} \,\mathbf{log}\, \boldsymbol{n} + O(n)$
  - experiments
    - integers: Yaroslavskiy wins (3%)
    - strings: A&D wins (2%)

Multi-pivot partitioning

- Kushagra, López-Ortiz, Munro, Qiao, 2013
  - triple-pivot partitioning
  - #comparisons
    - total: $\sim \mathbf{1.846}\ \boldsymbol{n}\ \mathbf{ln}\ \boldsymbol{n} + O(n)$
  - experiments
    - the fastest QS

For implementations see (on your own risk :)):
https://github.com/jurem/SortingAlgorithms

Multi-pivot partitioning

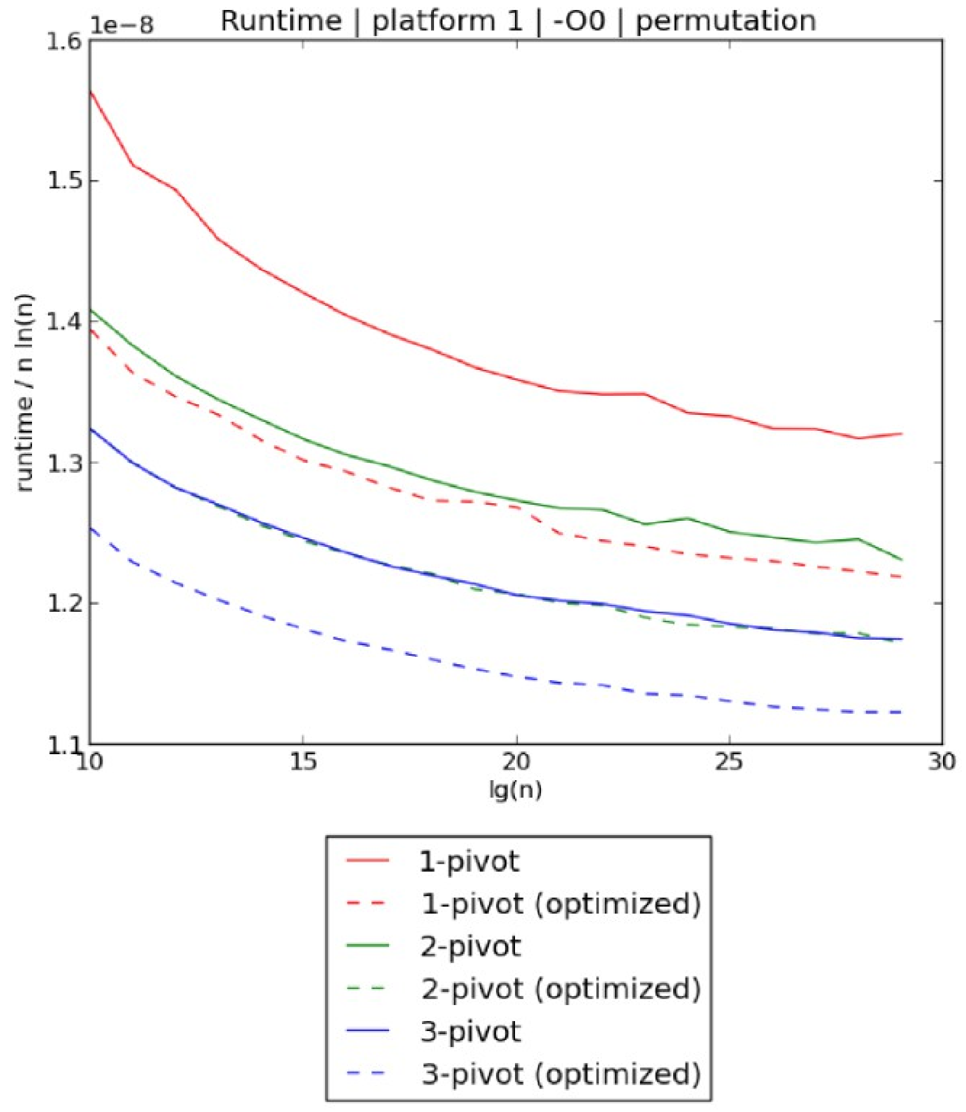
- Average case analysis
  - comparisons

Algorithm	Comparisons $+O(n)$
1-pivot	$\mathbf{2}\, n \ln n$
1-pivot (median of 3)	$\mathbf{1.71}\, n \ln n$
Sedgewick's 2-pivot	$\mathbf{2.13}\, n \ln n$
Yaroslavskiy's 2-pivot	$\mathbf{1.9}\, n \ln n$
Aumüller et al. 2-pivot	$\mathbf{1.8}\, n \ln n$
KLMQ 3-pivot	$\mathbf{1.846}\, n \ln n$

Multi-pivot partitioning

- Experiments
  - switch from java to C
  - confirm previous results
- Results
  - 3P > 2P > 1P
    - 1.85, 1.9, 2
  - 3P > 1P-m3
    - 1.85, 1.71

Kushagra, López-Ortiz, Munro, Qiao, 2013



Multi-pivot partitioning

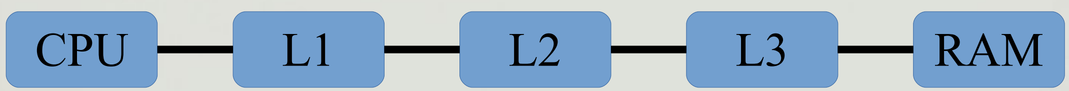
- Average case analysis
  - swaps

Algorithm	Comparisons $+O(n)$	Swaps $+O(n)$
1-pivot	$\mathbf{2}\ n \ln n$	$\mathbf{0.33}\ n \ln n$
1-pivot (median of 3)	$\mathbf{1.71}\ n \ln n$	$\mathbf{0.34}\ n \ln n$
Sedgewick's 2-pivot	$\mathbf{2.13}\ n \ln n$	$\mathbf{0.8}\ n \ln n$
Yaroslavskiy's 2-pivot	$\mathbf{1.9}\ n \ln n$	$\mathbf{0.6}\ n \ln n$
Aumüller et al. 2-pivot	$\mathbf{1.8}\ n \ln n$	
KLMQ 3-pivot	$\mathbf{1.846}\ n \ln n$	$\mathbf{0.615}\ n \ln n$

Multi-pivot partitioning - cache

- Hennesy, Patterson, 1996
  - performance increase per year
    - cpu: 60%, memory: 10%
  - performance difference

    □ increase of levels of cache



- It's all about the cache
  - multi-pivot quicksorts are driven by cache effects
    - more computation, less cache misses

Multi-pivot partitioning - cache

- Memory hierarchy

Level	Access		Size		Price	
	Time	Factor	B	Factor	€/GiB	Factor
Registers	0,5 ns	1	64 B	1	10000	
Cache.	5 ns	10	16 MiB	250 000		
DRAM	100 ns	200	16 GiB	250 000 000	10	
NVMe	10 μs	20 000	1 TiB	17 000 000 000		
SSD	100 μs	200 000	1 TiB	17 000 000 000	0,5	
HDD	5 ms	10 000 000	4 TiB	70 000 000 000	0,05	
Net	100 ms	200 000 000				

Multi-pivot partitioning - cache

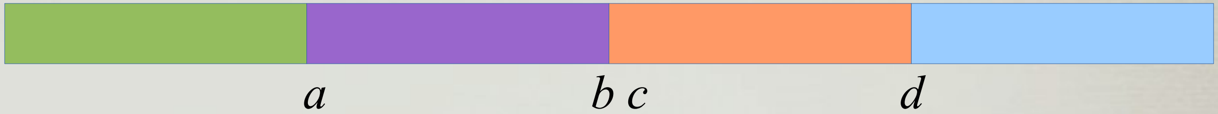
- Cache performance analysis
  - M … size of cache, B … size of cache line

$$CM(n) \leq \alpha \frac{n+1}{B} \ln\left(\frac{n+1}{M+2}\right) + O\left(\frac{n}{B}\right)$$

Algorithm	**α**
1-pivot	**2**
1-pivot (median of 3)	12/7 ~ **1.715**
Yaroslavskiy's 2-pivot	8/5 = **1.6**
KLMQ 3-pivot	18/13 ~ **1.385**

Multi-pivot partitioning - cache

- Cache performance analysis
  - Why triple pivot is better?
  - triple pivot



    - 2 pointers b and c go each over half of the array
    - 2 pointers a and d go each over quarter of the array
    - total: $2 \cdot \frac{1}{2} + 2 \cdot \frac{1}{4} = 1.5$
  - single pivot
    - 2 pointers go each over half of the array
    - need 2 partitionings for the same effect
    - total: $2 \cdot 2 \cdot \frac{1}{2} = 2$

Sorting

- Putting it all together

Algorithm	Comparisons	Swaps	Cache misses
1-pivot	**2**	**0.33**	**2**
1-pivot (median of 3)	**1.71**	**0.34**	**1.715**
Sedgewick's 2-pivot	2.13	0.8	
Yaroslavskiy's 2-pivot	**1.9**	**0.6**	**1.6**
Aumüller et al. 2-pivot	1.8		
KLMQ 3-pivot	**1.846**	**0.615**	**1.385**

Sorting

- Other engineering tricks
  - short sublists: insertion sort
  - almost sorted (#runs): merge sort
  - pivot sampling: median of 3, terciles of 5, …
  - equal pivots: fallback to single pivot
  - pivot presampling (2%)
    - sample $\sqrt{n}$ of elements, sort, use as pivots
    - run out of pivots → fallback to standard algorithm
  - multiple threads:
  - …

Sorting

- State-of-the-art in practice
  - 2002, TimSort: MS+IS
    - python / java for objects (stable sort)
  - 2009, Jaroslavskiy dual-pivot
    - for primitive type
  - 2014, 5-pivotno hitro urejanje, predpomnilnik
    - Kushagra, Lopez-Ortiz, Munro, Qiao