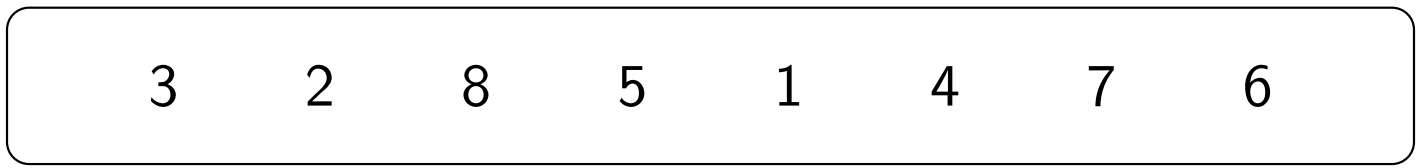
Optimal Partitioning for Dual Pivot Quicksort

Martin Aumüller, Martin Dietzfelbinger

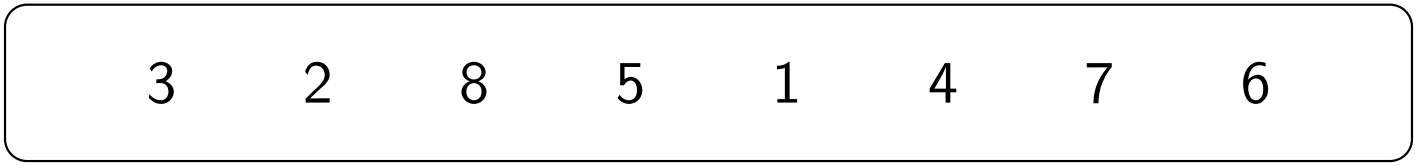
Technische Universität Ilmenau, Germany

ICALP 2013 – Riga, July 12, 2013

Dual Pivot Quicksort

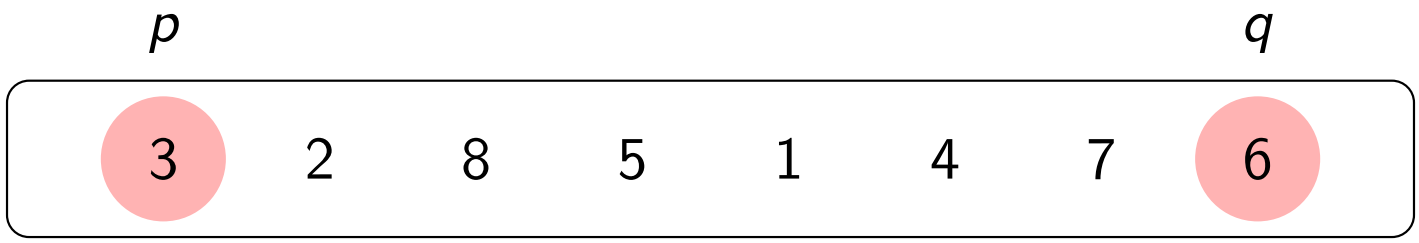


Dual Pivot Quicksort



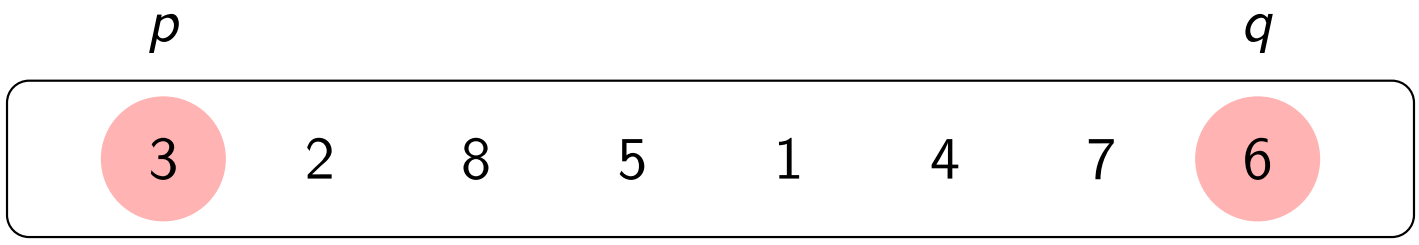
Choose **two** pivots p, q with $p < q$.

Dual Pivot Quicksort



Choose **two** pivots p, q with $p < q$.

Dual Pivot Quicksort



Partition, i.e., re-arrange elements via “swaps”.

Partition:

Dual Pivot Quicksort

Partition, i.e., re-arrange elements via “swaps”.

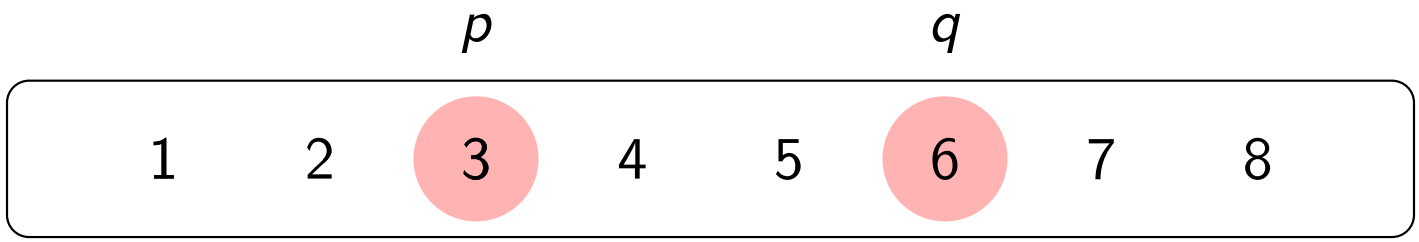
Partition:

Dual Pivot Quicksort

Use recursion to sort these **three** subarrays.

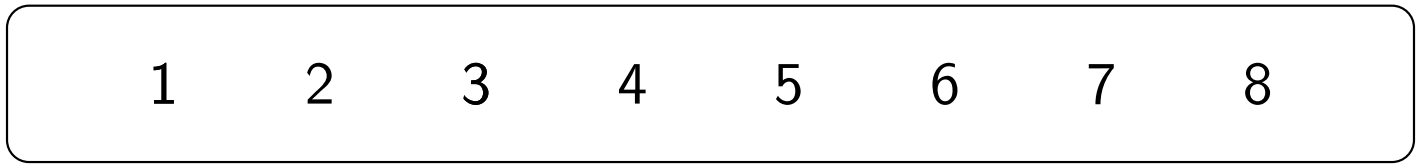
Partition:

Dual Pivot Quicksort



Use recursion to sort these **three** subarrays.

Dual Pivot Quicksort



Done.

Dual Pivot Quicksort: History

- **Robert Sedgewick (1975)**: Analyzed a dual pivot algorithm that makes much more swaps than classical quicksort on average$\rightarrow$ no further investigation.
- **Pascal Hennequin (1991)**: Thorough analysis of quicksort with $k \geq 1$ pivots.
  - for $k = 2$, no improvements found.
  - for $k \geq 3$, slight improvements, partitioning too complicated.

Dual Pivot Quicksort: History

- **Robert Sedgewick (1975)**: Analyzed a dual pivot algorithm that makes much more swaps than classical quicksort on average$\rightarrow$ no further investigation.
- **Pascal Hennequin (1991)**: Thorough analysis of quicksort with $k \geq 1$ pivots.
  - for $k = 2$, no improvements found.
  - for $k \geq 3$, slight improvements, partitioning too complicated.

So, why bother?

Dual Pivot Quicksort: History

- **Robert Sedgewick (1975)**: Analyzed a dual pivot algorithm that makes much more swaps than classical quicksort on average$\rightarrow$ no further investigation.
- **Pascal Hennequin (1991)**: Thorough analysis of quicksort with $k \geq 1$ pivots.
  - for $k = 2$, no improvements found.
  - for $k \geq 3$, slight improvements, partitioning too complicated.

So, why bother?

Java 7 (2009): Classical quicksort is replaced by a dual pivot quicksort variant due to Yaroslavskiy!

Rigorous analysis of the core of this algorithm by Wild and Nebel (2012).

Classical QS vs. Yaroslavskiy's Dual Pivot QS

	Classical QS	Yaroslavskiy
Average Comparison Count	$2n \ln n$ $(= 1.38n \log n)$	$1.9n \ln n$

- **Experiments:** Yaroslavskiy's algorithm around 10% faster!

Classical QS vs. Yaroslavskiy's Dual Pivot QS

	Classical QS	Yaroslavskiy	Optimal DP
Average Comparison Count	$2n \ln n$ $(= 1.38n \log n)$	$1.9n \ln n$	??

- **Experiments:** Yaroslavskiy's algorithm around 10% faster!
- **Question:** What is possible?

Focus of This Talk

- introduce a model that captures all dual pivot algorithms,
- give a unified analysis of the average comparison count,
- identify optimal algorithms.

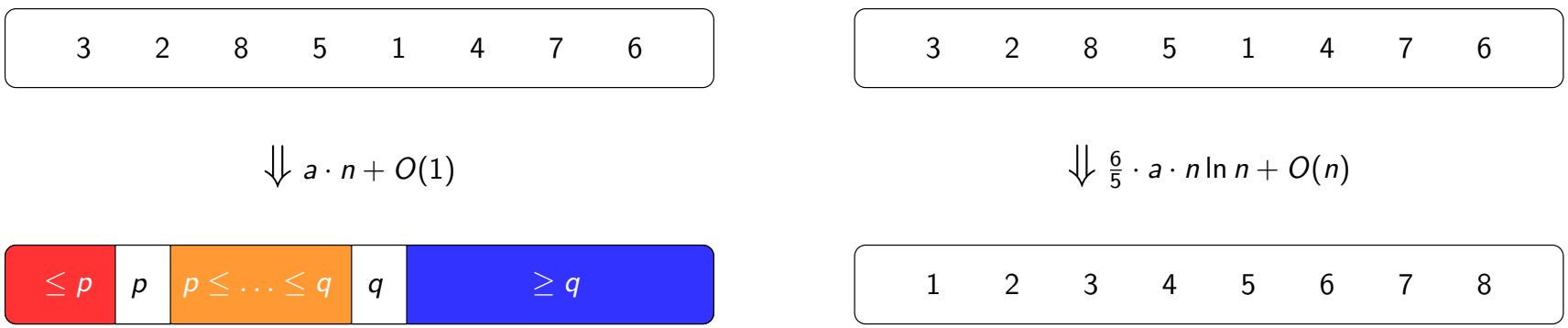
For analysis:

- input is random permutation of $\{1, \ldots, n\}$
- left-most and right-most elements are the two pivots
- analyze average sorting cost

Reduce Sorting to Partitioning

Fact (Hennequin, 1991)

For dual pivot quicksort, average partitioning cost of $\approx a \cdot n + O(1)$ leads to average sorting cost $\approx \frac{6}{5} a \cdot n \ln n + O(n)$.



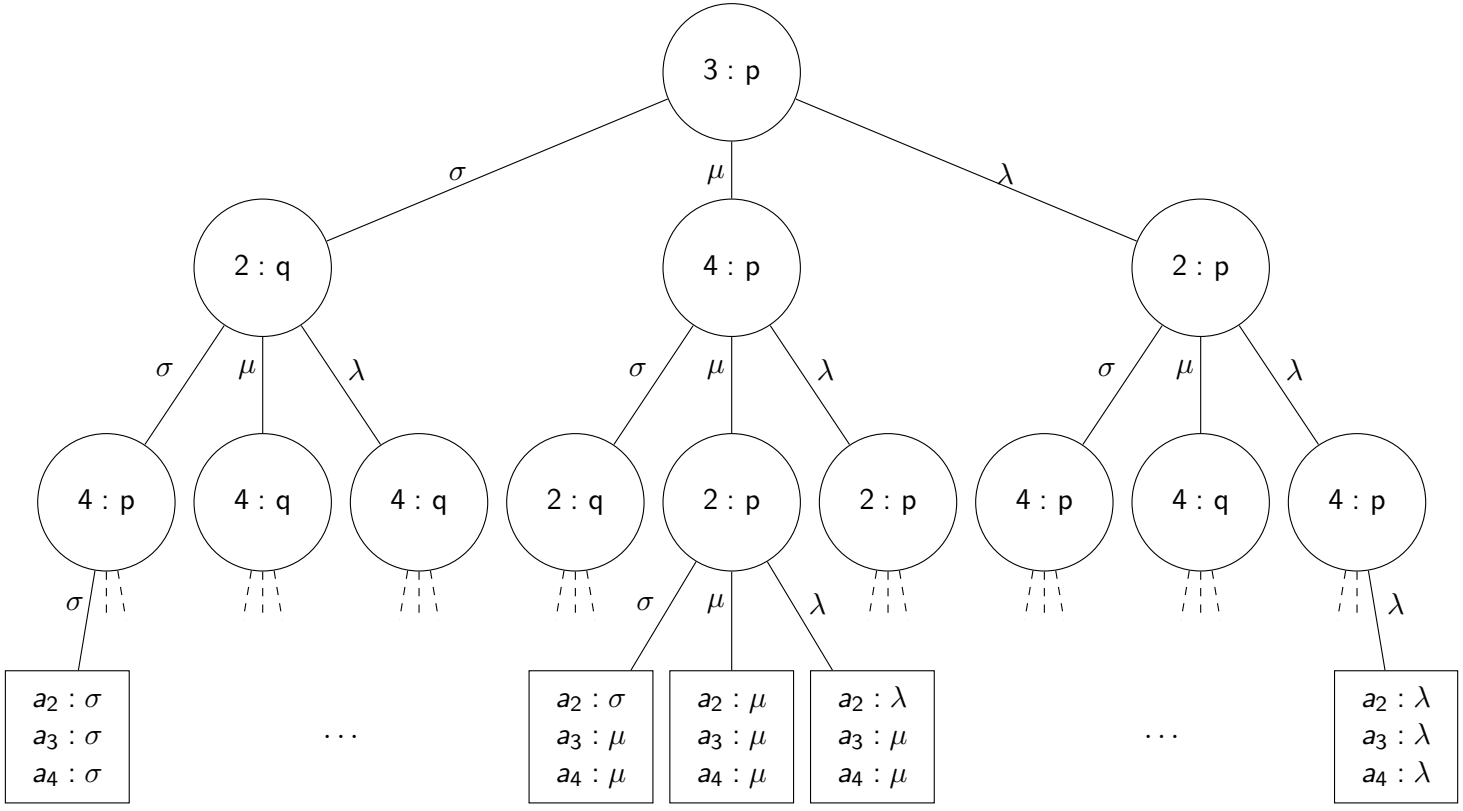
Reduce Partitioning to Classification

- simplification: focus on classifying elements as

  small, medium, or large

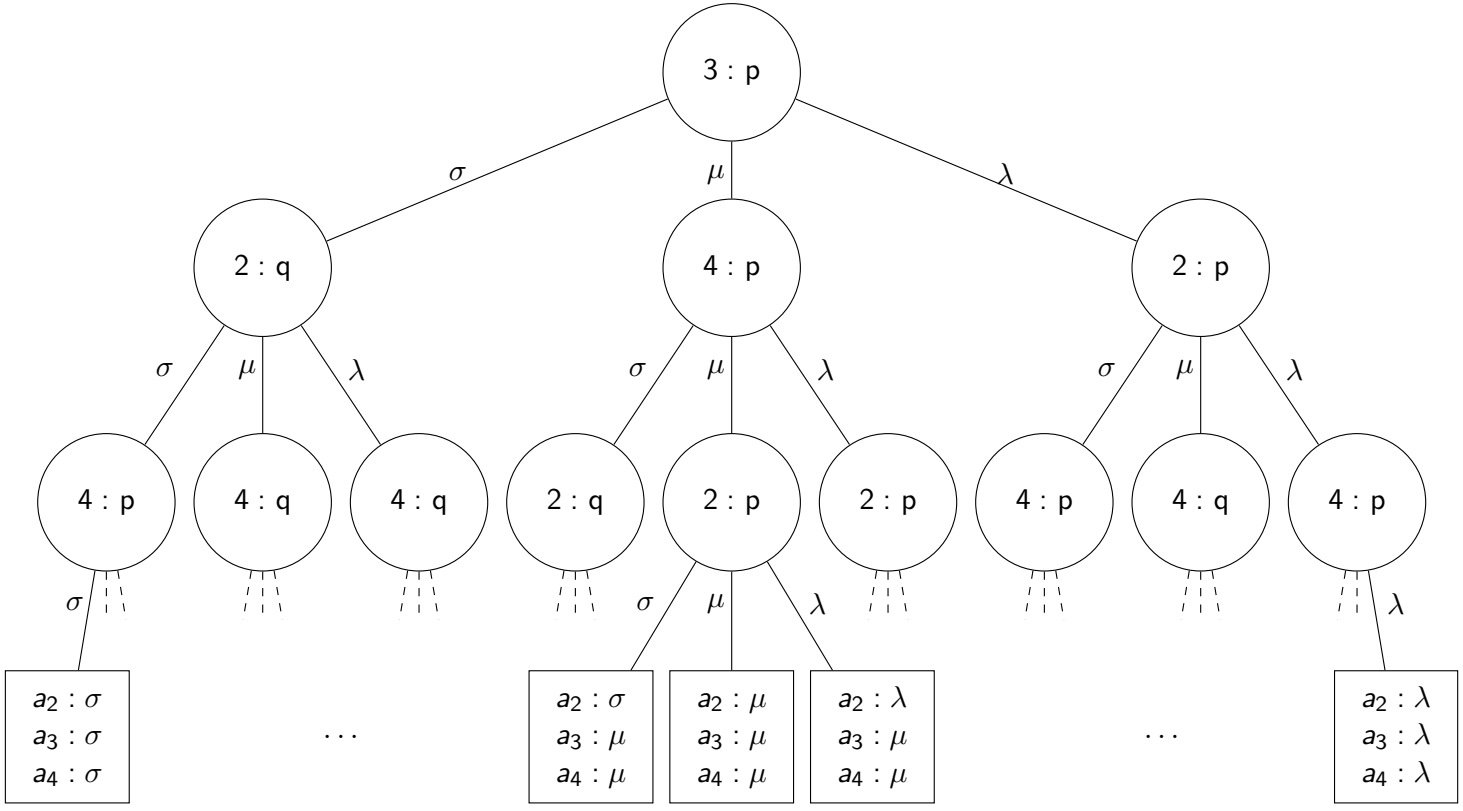
- independent of this: "clever" swap strategy to obtain a partition

Model: Decision Tree



Example: 3 4 2 1 5

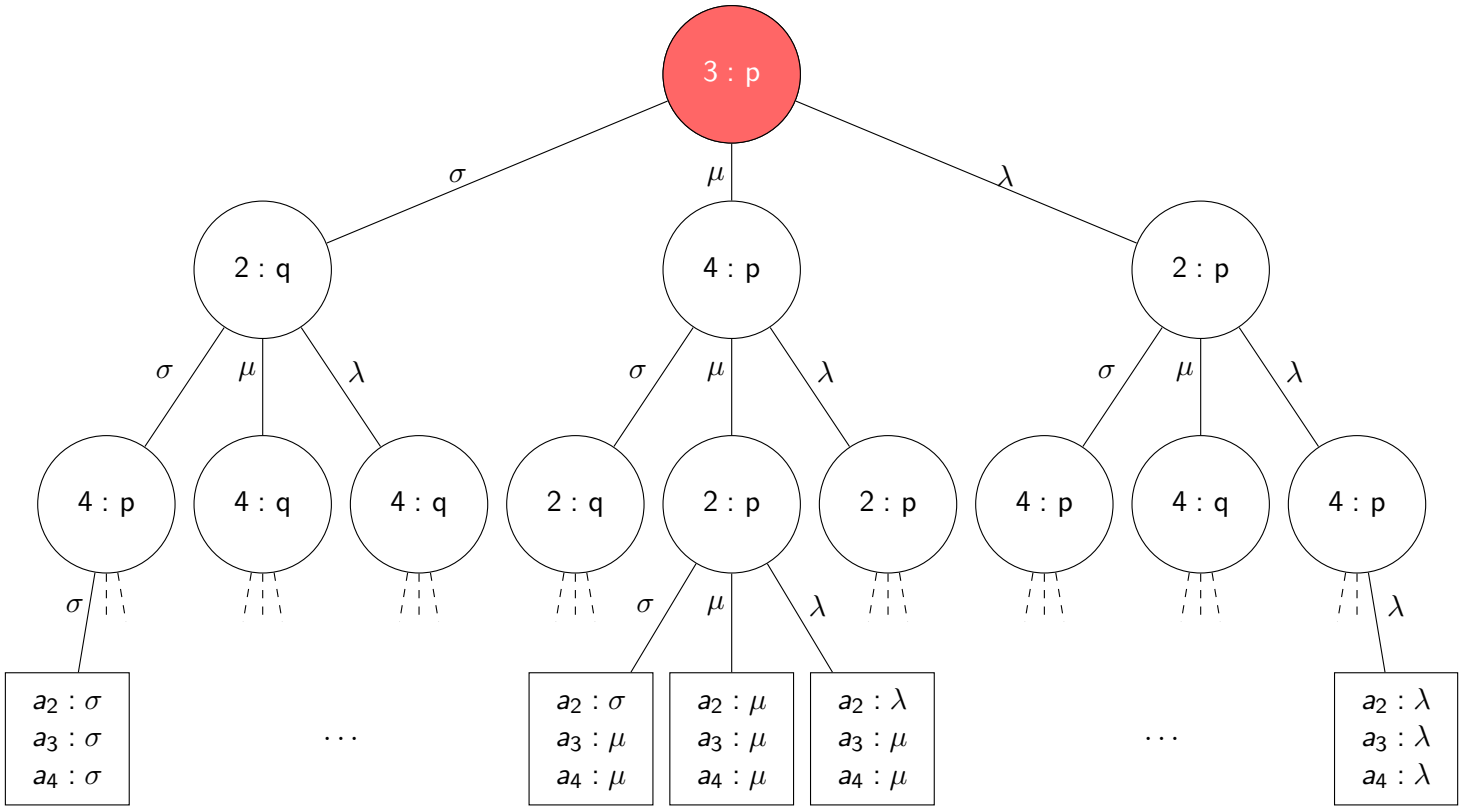
Model: Decision Tree



Example: 3 4 2 1 5

p q

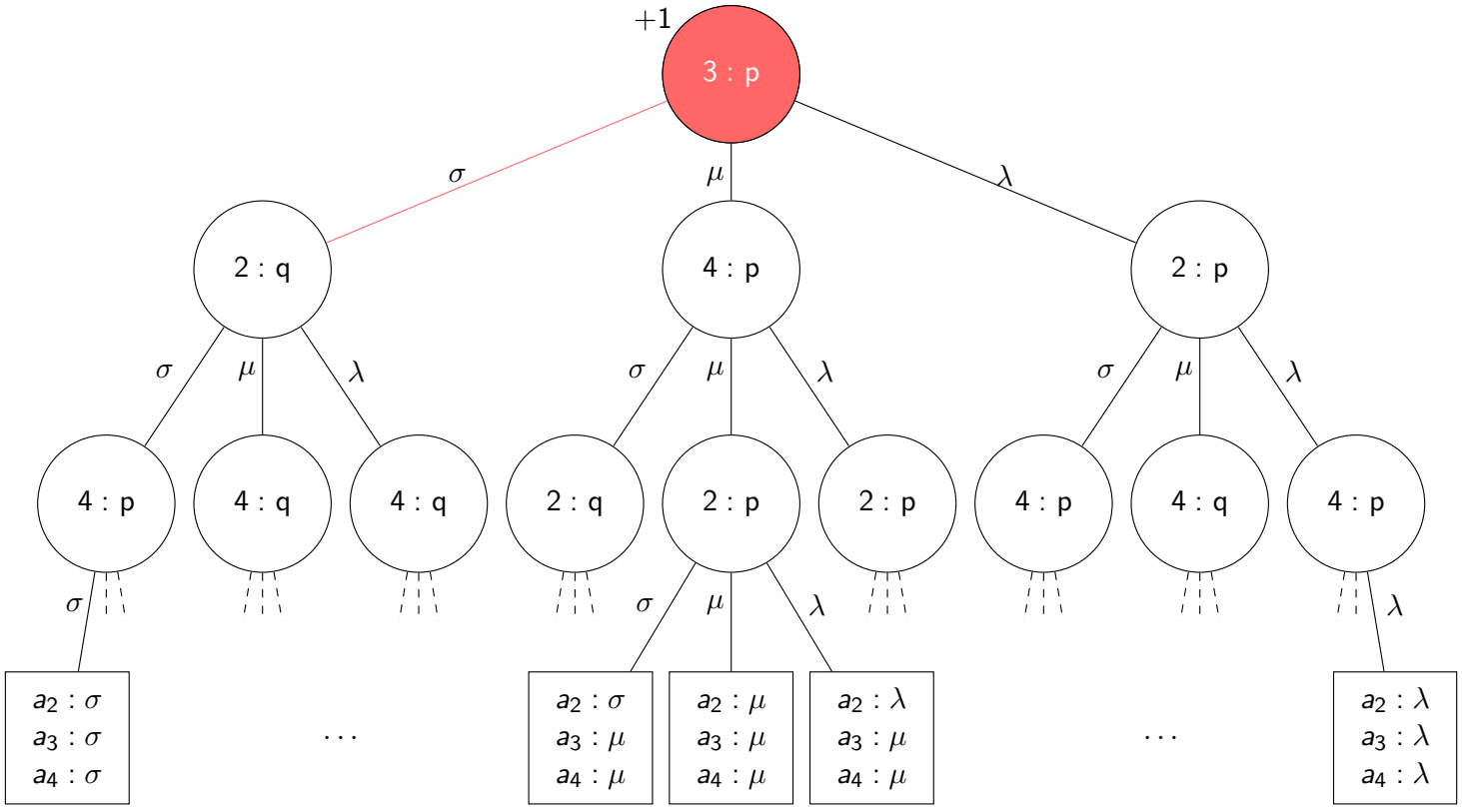
Model: Decision Tree



Example: 3 4 2 1 5

p q

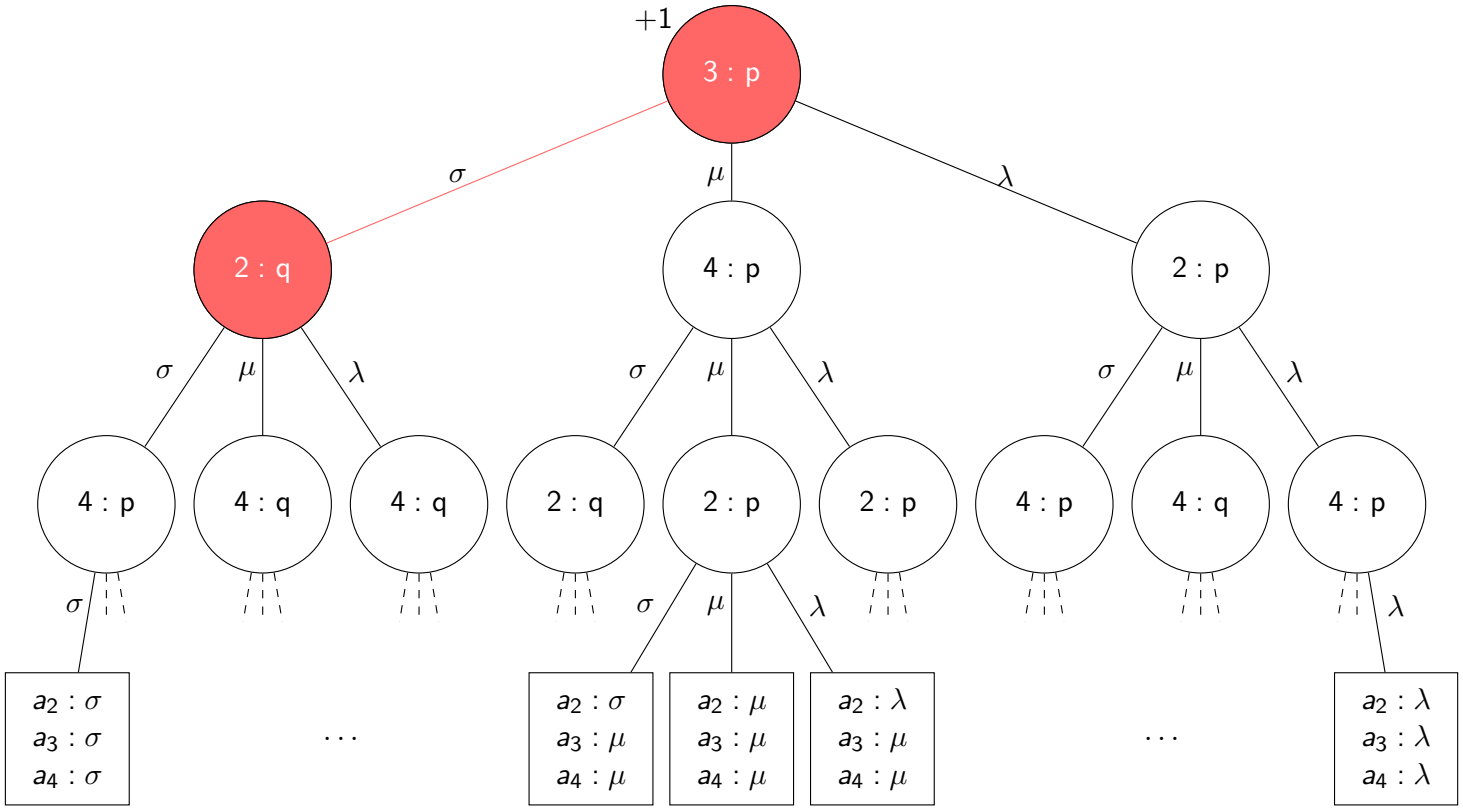
Model: Decision Tree



Example:

3	4	2	1	5
p		σ		q

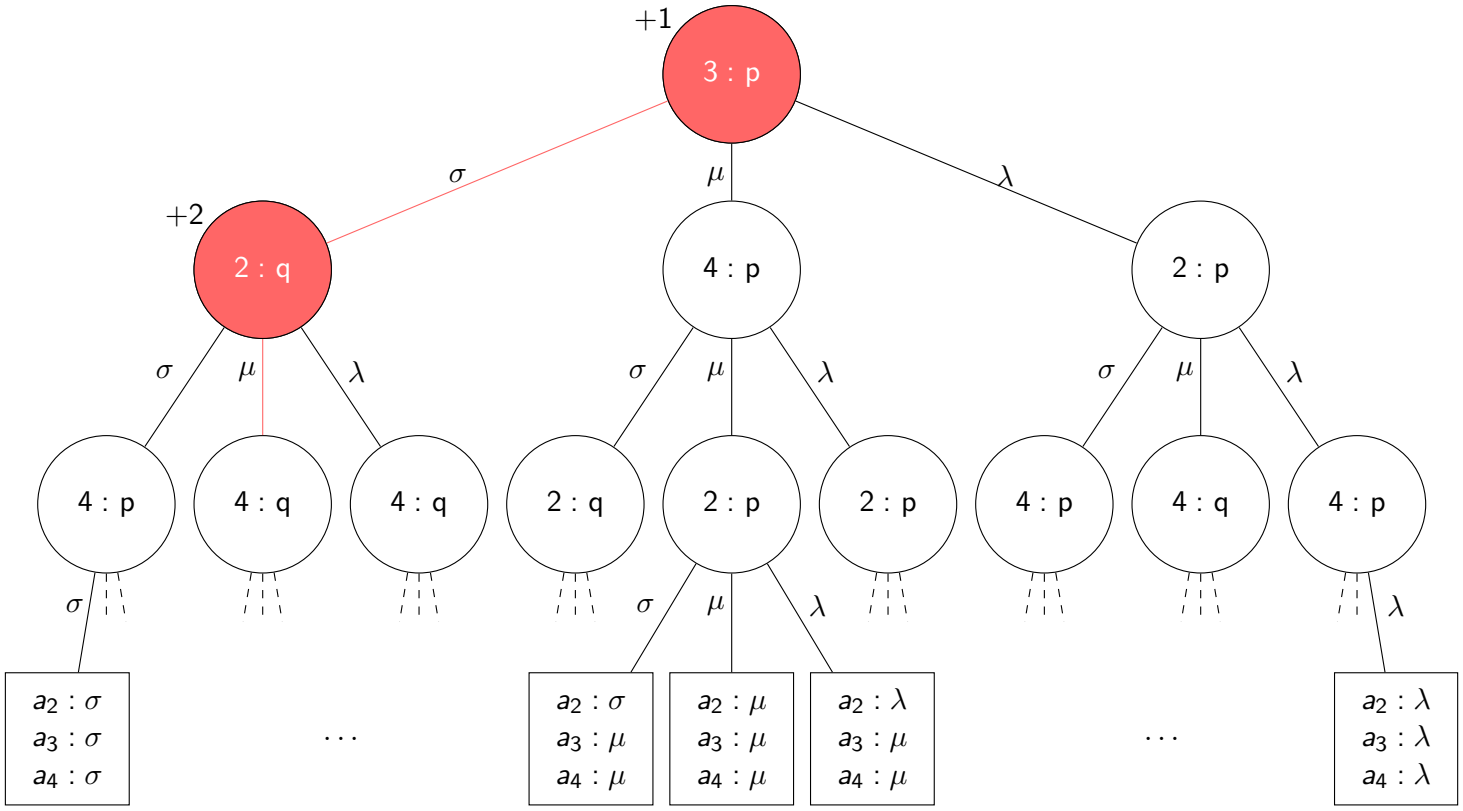
Model: Decision Tree



Example:

3	4	2	1	5
p		σ		q

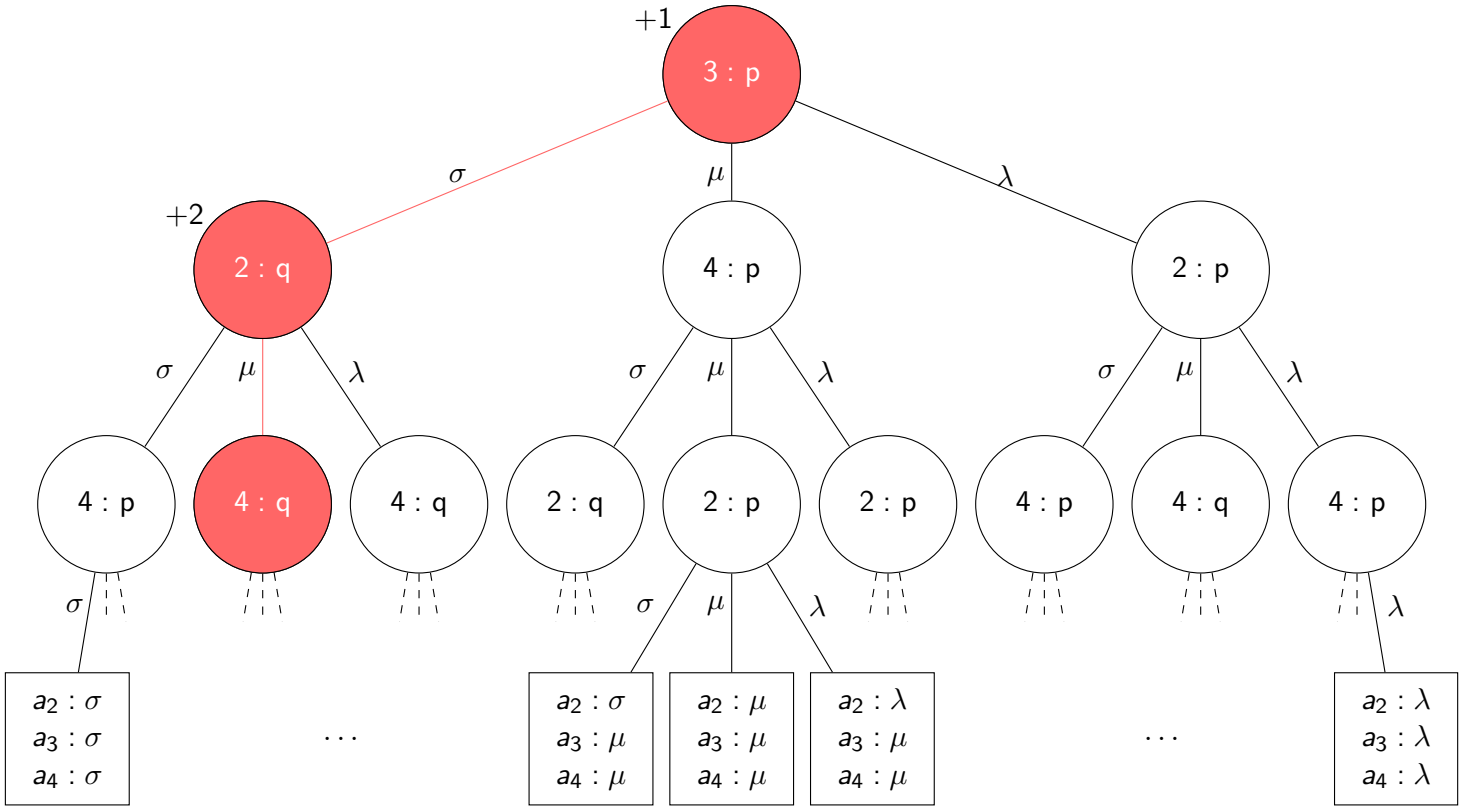
Model: Decision Tree



Example:

3	4	2	1	5
p	μ	σ		q

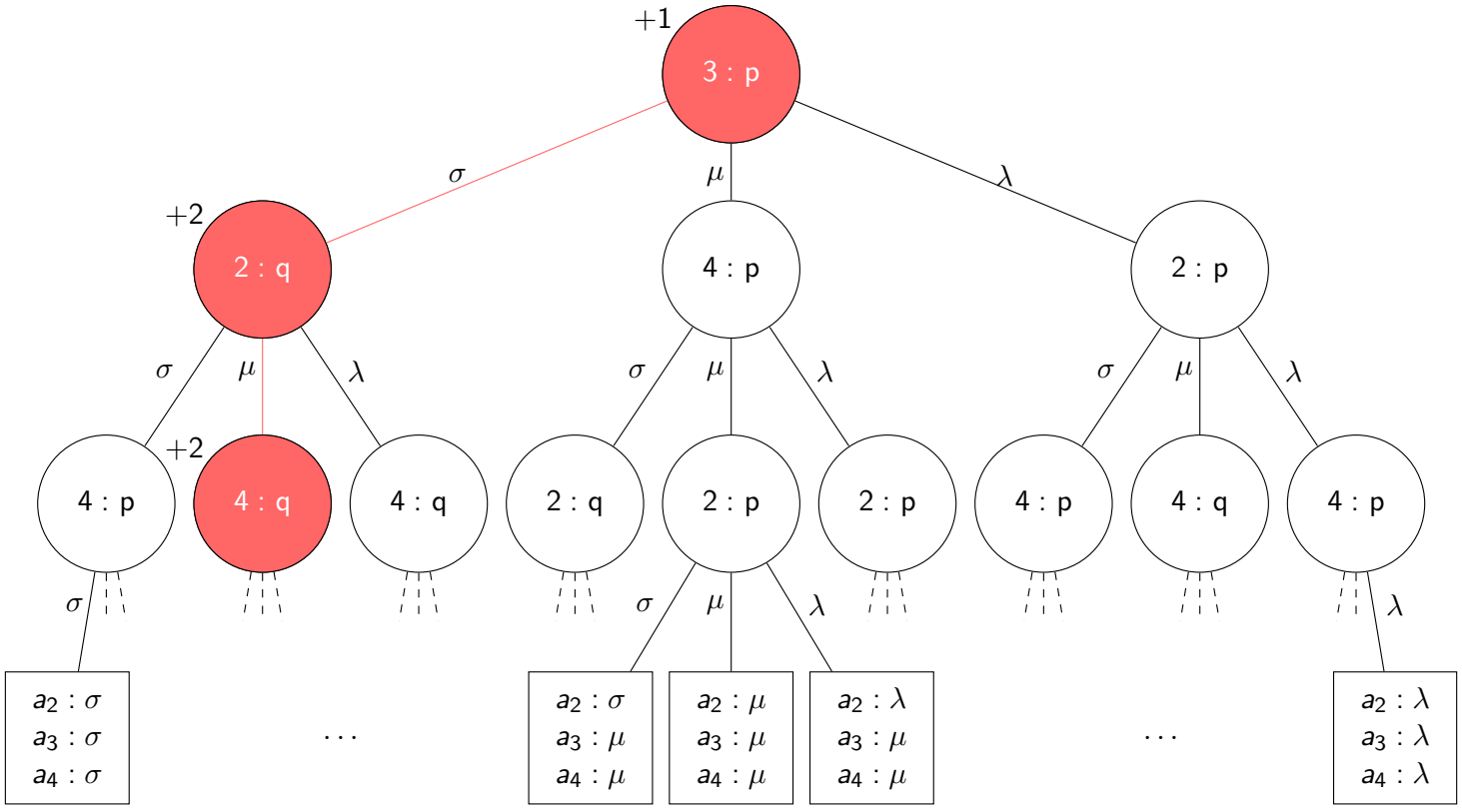
Model: Decision Tree



Example: 3 4 2 1 5

p μ σ q

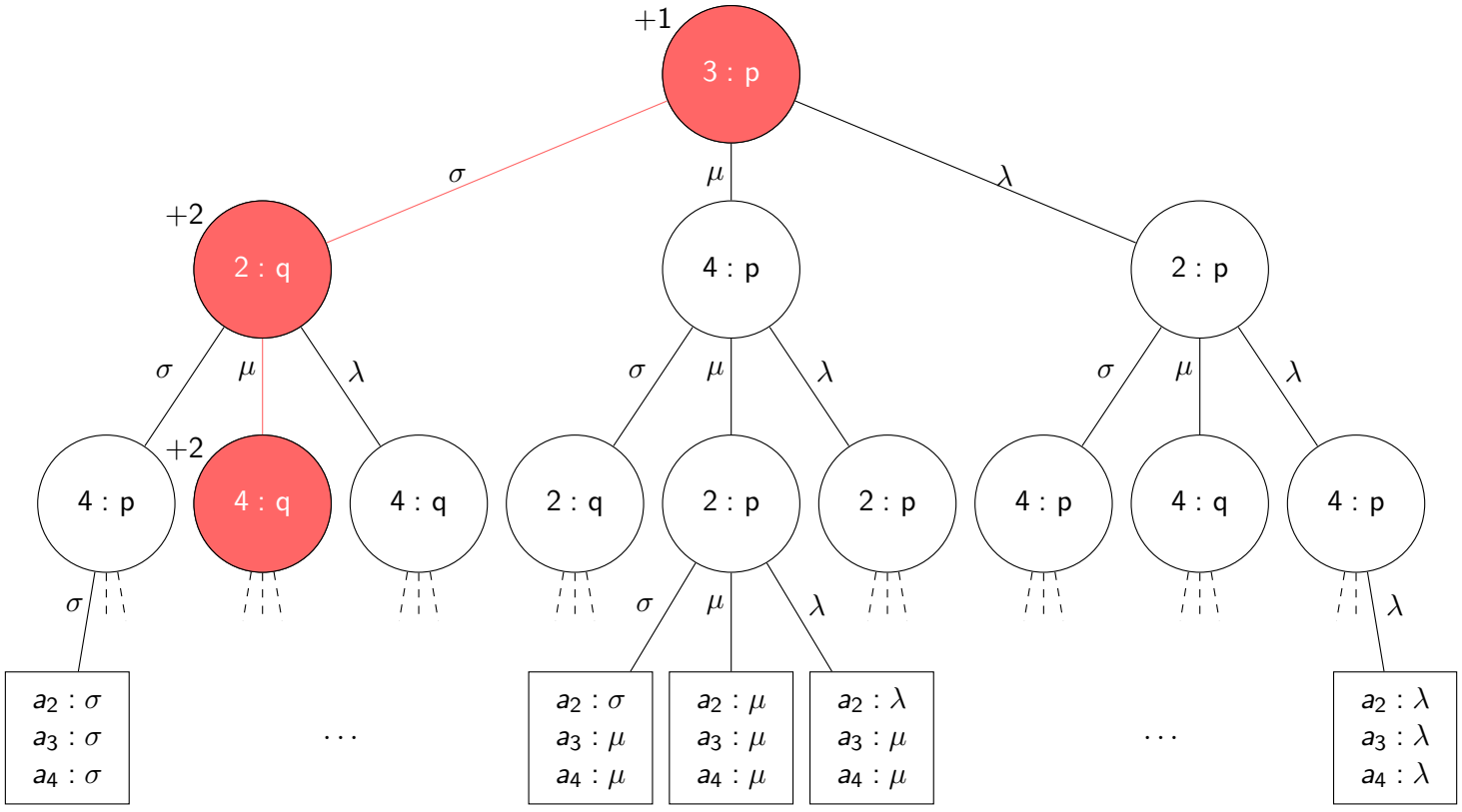
Model: Decision Tree



Example:

3	4	2	1	5
p	μ	σ	σ	q

Model: Decision Tree



Example: 3 4 2 1 5

$\Rightarrow$ 5 comparisons.

p μ σ σ q

Average Cost

For the average partitioning/classification cost p_n we get:

$$p_n \approx 4/3n + \text{“average number of additional comparisons”}$$

For all classification algorithms:

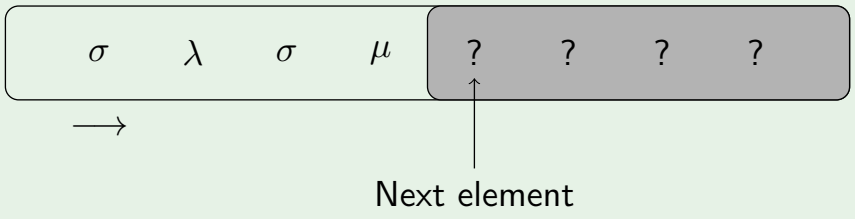
- each element has to be compared at least once against a pivot
- each medium element has to be compared to both ($\approx n/3$ on avg.)

“Additional” comparisons:

- a small element is compared with the larger pivot first
- a large element is compared with the smaller pivot first

Optimal Strategies

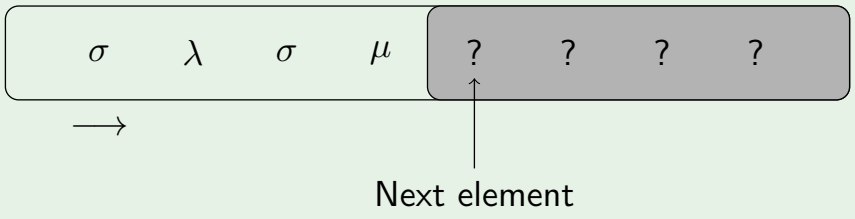
Strategy 1 (unrealistic)



Let s, ℓ denote the number of small/large elements in  area (resp.).

Optimal Strategies

Strategy 1 (unrealistic)



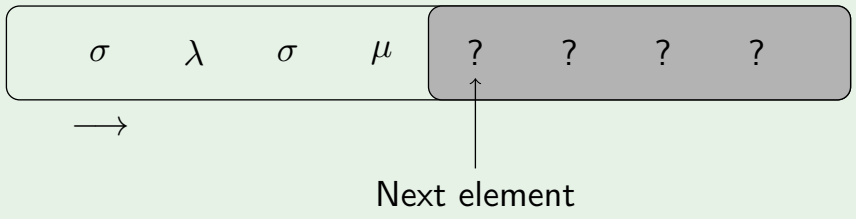
Let s, ℓ denote the number of small/large elements in  area (resp.).

Classification

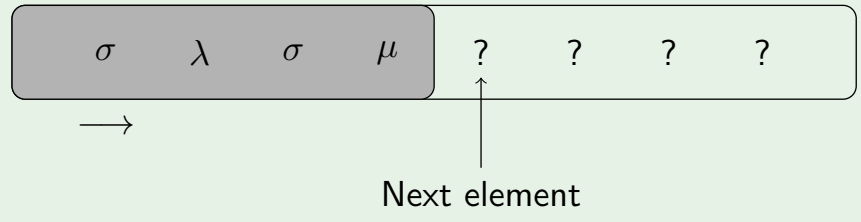
- $\ell > s$: compare with larger pivot first.
- $\ell \leq s$: compare with smaller pivot first.

Optimal Strategies

Strategy 1 (unrealistic)



Strategy 2



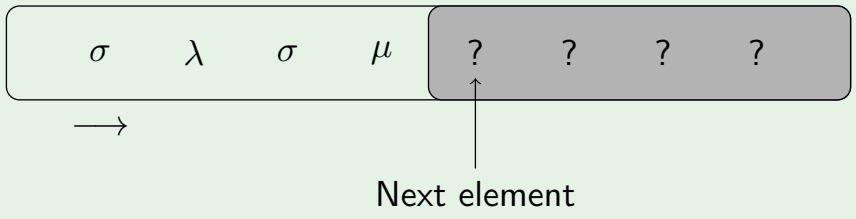
Let s, ℓ denote the number of small/large elements in  area (resp.).

Classification

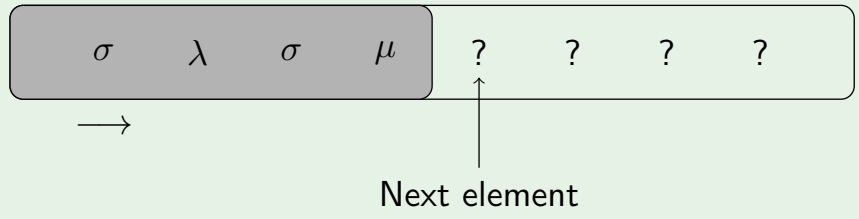
- $\ell > s$: compare with larger pivot first.
- $\ell \leq s$: compare with smaller pivot first.

Optimal Strategies

Strategy 1 (unrealistic)



Strategy 2



Let s, ℓ denote the number of small/large elements in  area (resp.).

Classification

- $\ell > s$: compare with larger pivot first.
- $\ell \leq s$: compare with smaller pivot first.

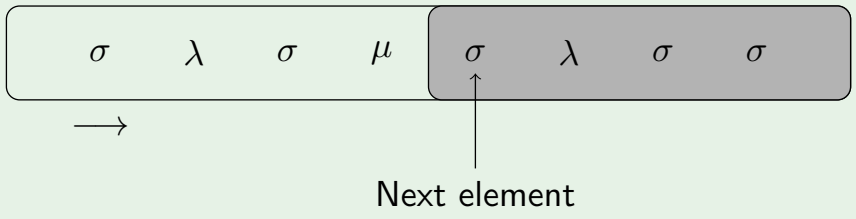
Theorem

These strategies (turned into dp algorithms) have the minimum possible average sorting cost:

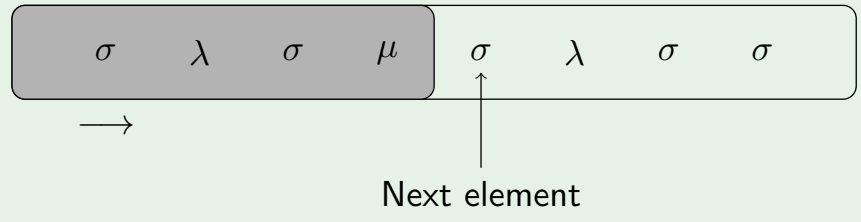
$$1.8n \ln n + O(n).$$

Optimal Strategies

Strategy 1 (unrealistic)



Strategy 2



Let s, ℓ denote the number of small/large elements in  area (resp.).

Classification

- $\ell > s$: compare with larger pivot first.
- $\ell \leq s$: compare with smaller pivot first.

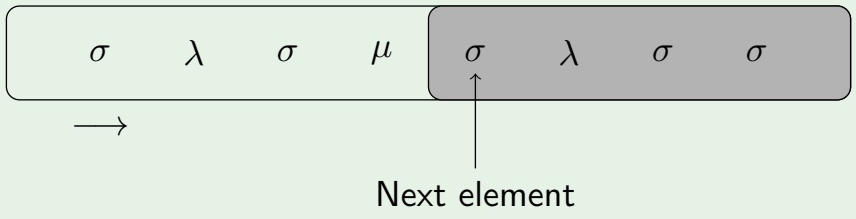
Theorem

These strategies (turned into dp algorithms) have the minimum possible average sorting cost:

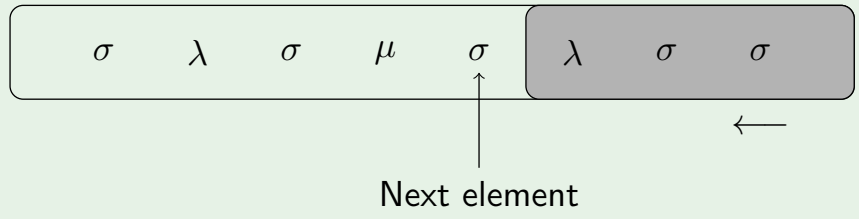
$$1.8n \ln n + O(n).$$

Optimal Strategies

Strategy 1 (unrealistic)



Strategy 2



Let s, ℓ denote the number of small/large elements in  area (resp.).

Classification

- $\ell > s$: compare with larger pivot first.
- $\ell \leq s$: compare with smaller pivot first.

Theorem

These strategies (turned into dp algorithms) have the minimum possible average sorting cost:

$$1.8n \ln n + O(n).$$

Cost of an Arbitrary Decision Tree

$$p_n \approx 4/3n + \text{“average number of additional comparisons”}$$

Cost of an Arbitrary Decision Tree

$$p_n \approx 4/3n + \text{“average number of additional comparisons”}$$

Central: $\boxed{f^{\mathrm{q}}_{s,\ell}}$ – average number of comparisons to the larger pivot first.

Lemma

For the average partition cost p_n of a decision tree T we have:

$$p_n = \frac{4}{3}n + \frac{1}{\binom{n}{2} \cdot (n-2)} \sum_{s+\ell \le n-2} \left(f^{q}_{s,\ell} \cdot s \; + \; (n-2-f^{q}_{s,\ell}) \cdot \ell \right) + o(n).$$

$\Rightarrow$ dependence introduces an error term of $o(n)$.

Cost of an Arbitrary Decision Tree

$$p_n \approx 4/3n + \text{“average number of additional comparisons”}$$

Central: $\boxed{f^{\mathrm{q}}_{s,\ell}}$ – average number of comparisons to the larger pivot first.

Lemma

For the average partition cost p_n of a decision tree T we have:

$$p_n = \frac{4}{3}n + \frac{1}{\binom{n}{2} \cdot (n-2)} \sum_{s+\ell \le n-2} \left(f^{q}_{s,\ell} \cdot s + (n-2-f^{q}_{s,\ell}) \cdot \ell \right) + o(n).$$

$\Rightarrow$ dependence introduces an error term of $o(n)$.

Cost of an Arbitrary Decision Tree

$$p_n \approx 4/3n + \text{“average number of additional comparisons”}$$

Central: $\boxed{f^{\mathrm{q}}_{s,\ell}}$ – average number of comparisons to the larger pivot first.

Lemma

For the average partition cost p_n of a decision tree T we have:

$$p_n = \frac{4}{3}n + \frac{1}{\binom{n}{2} \cdot (n-2)} \sum_{s+\ell \le n-2} \left(f^{q}_{s,\ell} \cdot s \; + (n-2-f^{q}_{s,\ell}) \cdot \ell \right) + o(n).$$

$\Rightarrow$ dependence introduces an error term of $o(n)$.

Cost of an Arbitrary Decision Tree

$$p_n \approx 4/3n + \text{“average number of additional comparisons”}$$

Central: $\boxed{f^{\mathrm{q}}_{s,\ell}}$ – average number of comparisons to the larger pivot first.

Lemma

For the average partition cost p_n of a decision tree T we have:

$$p_n = \frac{4}{3}n + \frac{1}{\binom{n}{2} \cdot (n-2)} \sum_{s+\ell \le n-2} \left(f^{q}_{s,\ell} \cdot s \; + \; (n-2-f^{q}_{s,\ell}) \cdot \ell \right) + o(n).$$

$\Rightarrow$ dependence introduces an error term of $o(n)$.

Cost of an Arbitrary Decision Tree

$$p_n \approx 4/3n + \text{“average number of additional comparisons”}$$

Central: $\boxed{f^{\mathrm{q}}_{s,\ell}}$ – average number of comparisons to the larger pivot first.

Lemma

For the average partition cost p_n of a decision tree T we have:

$$p_n = \frac{4}{3}n + \frac{1}{\binom{n}{2} \cdot (n-2)} \sum_{s+\ell \le n-2} \left(f^{q}_{s,\ell} \cdot s \; + \; (n-2-f^{q}_{s,\ell}) \cdot \ell \right) + o(n).$$

$\Rightarrow$ dependence introduces an error term of $o(n)$.

Message: $f^{\mathrm{q}}_{s,\ell}$ fully describes the average cost (up to lower order terms.)

Cost of an Arbitrary Decision Tree

$$p_n \approx 4/3n + \text{“average number of additional comparisons”}$$

Central: $\boxed{f^{\mathrm{q}}_{s,\ell}}$ – average number of comparisons to the larger pivot first.

Lemma

For the average partition cost p_n of a decision tree T we have:

$$p_n = \frac{4}{3}n + \frac{1}{\binom{n}{2} \cdot (n-2)} \sum_{s+\ell \le n-2} \left(f^{q}_{s,\ell} \cdot s \; + \; (n-2-f^{q}_{s,\ell}) \cdot \ell \right) + o(n).$$

$\Rightarrow$ dependence introduces an error term of $o(n)$.

Message: $f^{\mathrm{q}}_{s,\ell}$ fully describes the average cost (up to lower order terms.)

Example: Yaroslavskiy's algorithm: $f^{\mathrm{q}}_{s,\ell} = \ell$. (Leads to: $1.9n \ln n$.)

An Optimal Strategy Using An Oracle

Assume: An oracle tells us whether or not $\ell > s$ for an input:

Classification

- $\ell > s$: Compare all elements to the larger pivot first
- $\ell \leq s$: Compare all elements to the smaller pivot first

An Optimal Strategy Using An Oracle

Assume: An oracle tells us whether or not $\ell > s$ for an input:

Classification

- $\ell > s$: Compare all elements to the larger pivot first
- $\ell \leq s$: Compare all elements to the smaller pivot first

Why is it optimal?

It minimizes

$$p_n = \frac{4}{3}n + \frac{1}{\binom{n}{2} \cdot (n-2)} \sum_{s+\ell \leq n-2} \left(f^{\mathrm{q}}_{s,\ell} \cdot s + (n-2-f^{\mathrm{q}}_{s,\ell}) \cdot \ell \right) + o(n).$$

An Optimal Strategy Using An Oracle

Assume: An oracle tells us whether or not $\ell > s$ for an input:

Classification

- $\ell > s$: Compare all elements to the larger pivot first ($f^{\mathrm{q}}_{s,\ell} = n - 2$).
- $\ell \leq s$: Compare all elements to the smaller pivot first ($f^{\mathrm{q}}_{s,\ell} = 0$).

Why is it optimal?

It minimizes

$$p_n = \frac{4}{3}n + \frac{1}{\binom{n}{2} \cdot (n-2)} \sum_{s+\ell \leq n-2} \left(f^{\mathrm{q}}_{s,\ell} \cdot s + (n - 2 - f^{\mathrm{q}}_{s,\ell}) \cdot \ell \right) + o(n).$$

An Optimal Strategy Using An Oracle

Assume: An oracle tells us whether or not $\ell > s$ for an input:

Classification

- $\ell > s$: Compare all elements to the larger pivot first ($f^{\mathrm{q}}_{s,\ell} = n - 2$).
- $\ell \leq s$: Compare all elements to the smaller pivot first ($f^{\mathrm{q}}_{s,\ell} = 0$).

Why is it optimal?

It minimizes

$$p_n = \frac{4}{3}n + \frac{1}{\binom{n}{2} \cdot (n-2)} \sum_{s+\ell \leq n-2} \left(f^{\mathrm{q}}_{s,\ell} \cdot s + (n - 2 - f^{\mathrm{q}}_{s,\ell}) \cdot \ell \right) + o(n).$$

Average sorting cost: $\boxed{1.8 n \ln n + o(n \ln n)}$

Random Sampling

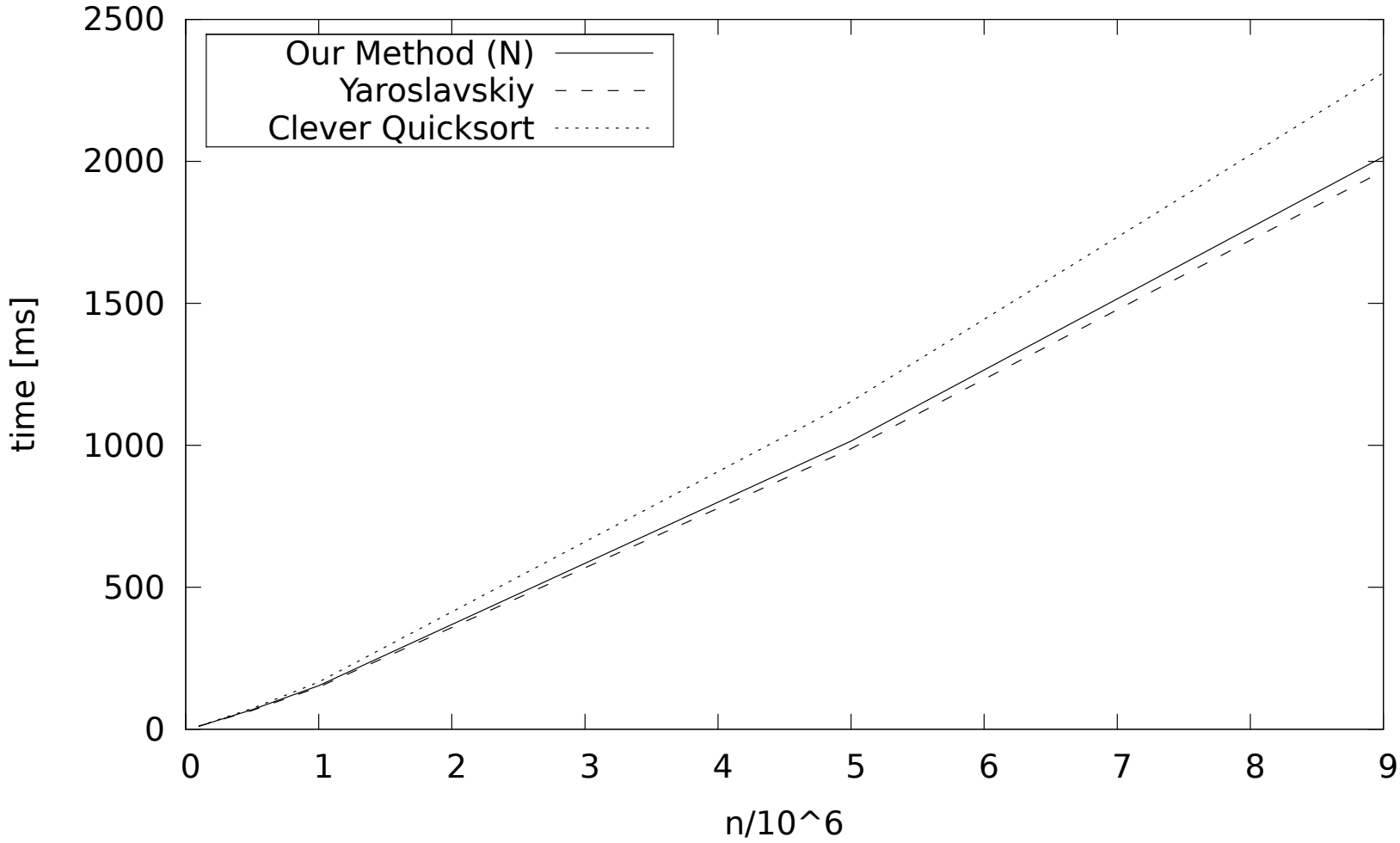
Just look at, e.g., $n^{2/3}$ elements.

Classification

Seen more large elements than small elements? Compare all elements with the larger pivot first, otherwise with the smaller pivot first.

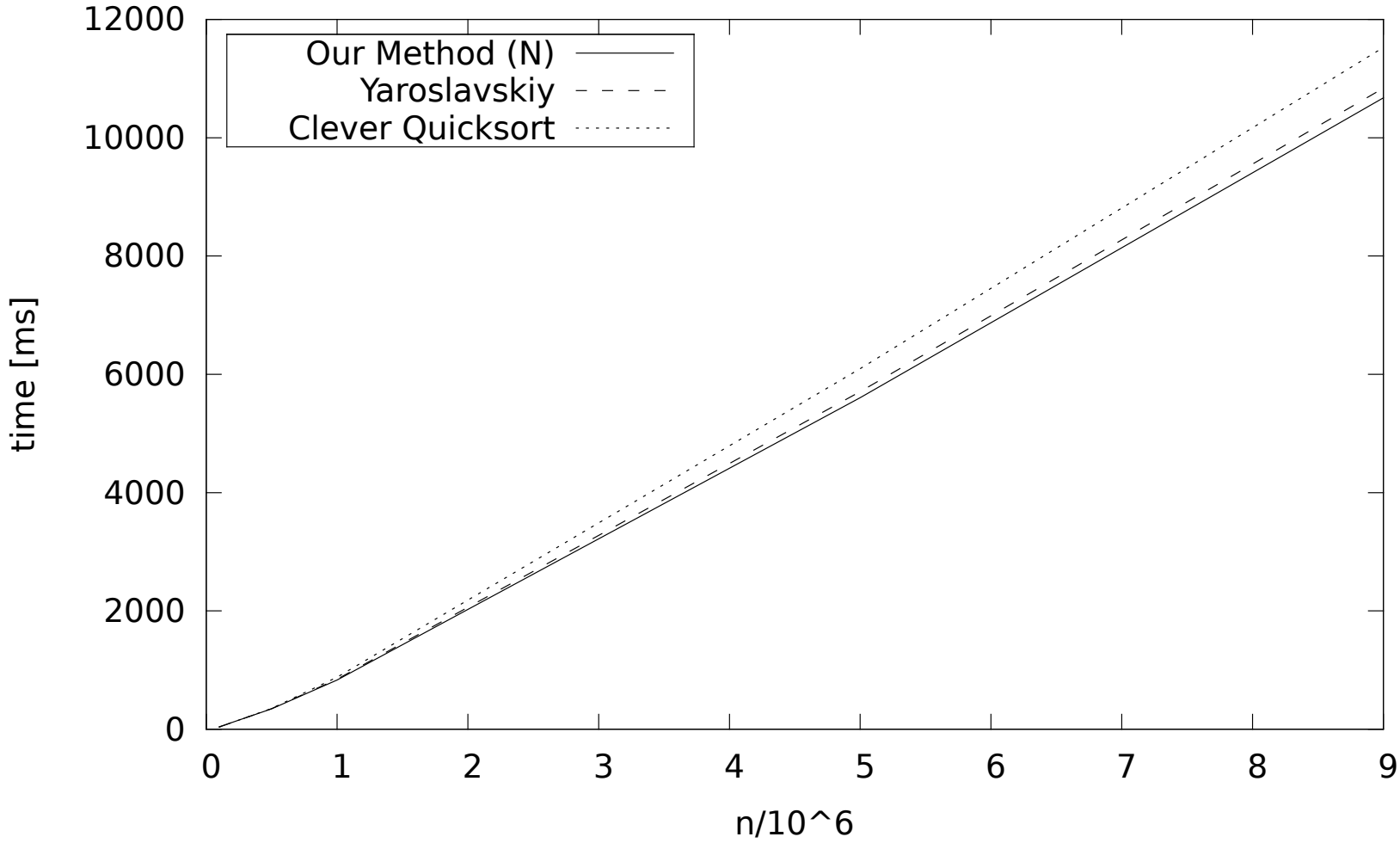
- proof that this works: standard machinery (tail estimates).
- difference to oracle estimation is $o(n)$.
- Average sorting cost: $1.8n \ln n + o(n \ln n)$.

Experiments



Sorting integers: About 3% slower than Yaroslavskiy.

Experiments (2)



Sorting strings: About 2% faster than Yaroslavskiy.

Conclusion and Open Questions

So far:

- gave a unified model for dual pivot quicksort algorithms,
- showed how to calculate the average cost for general algorithms,
- identified three (but only two "algorithmic") optimal dual pivot quicksort algorithms.
  Optimum: $1.8n \ln n$. (QS: $2n \ln n$, Yaroslavskiy: $1.9n \ln n$.)

Open questions:

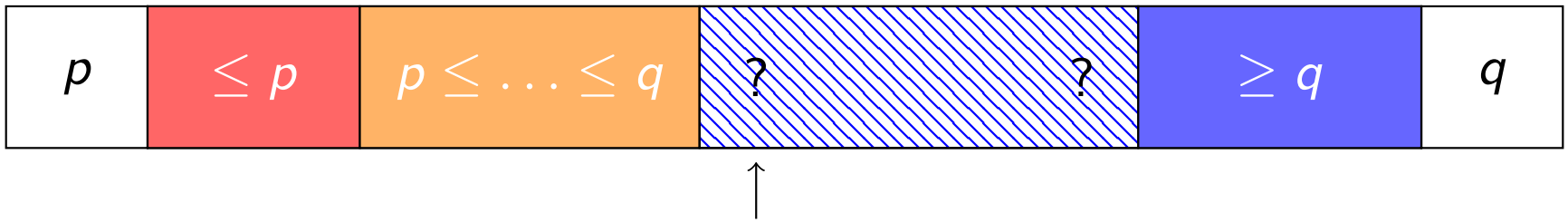
- Cost measure that describes running time behavior accurately?
- Optimal algorithms for multi-pivot quicksort with $k \geq 3$ pivots?
  (Best (known) 3-pivot algorithm: $1.846n \ln n$ comparisons on average)

Thanks!

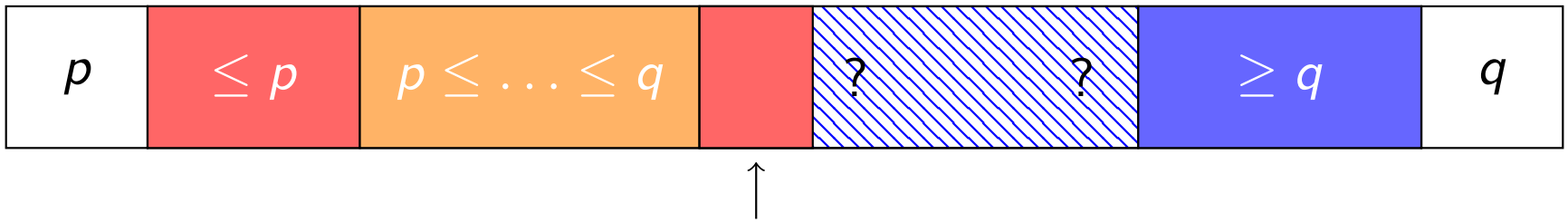
Any questions?

Theoretical Result

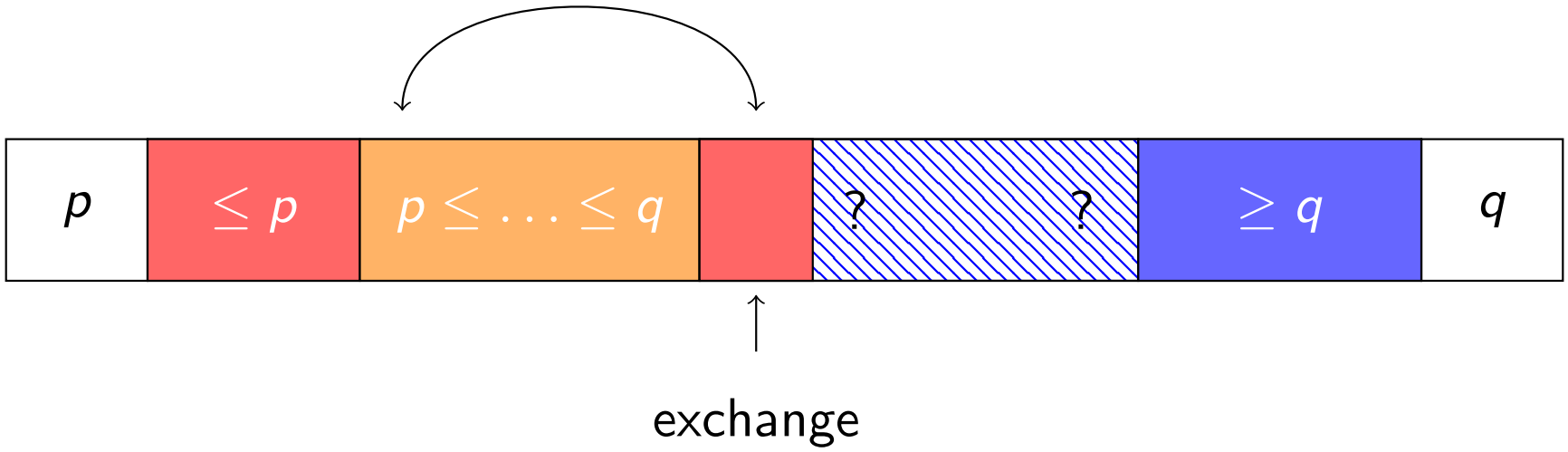
	Classical QS	Yaroslavskiy	Our algorithm
Average Comparison Count	$2n \ln n$	$1.9n \ln n$	$1.8n \ln n$
Average Swap Count	$0.33..n \ln n$	$0.6n \ln n$	$0.33..n \ln n$ LB: $0.3n \ln n$

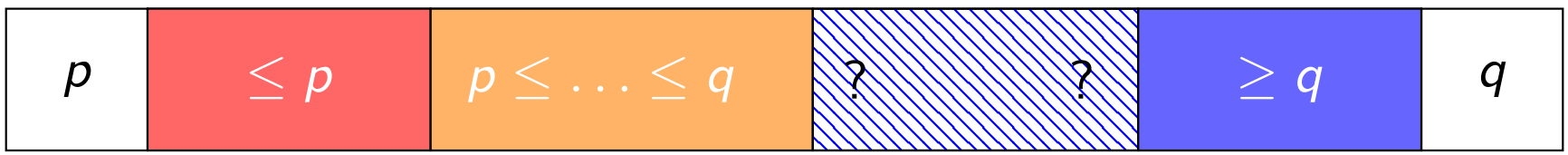


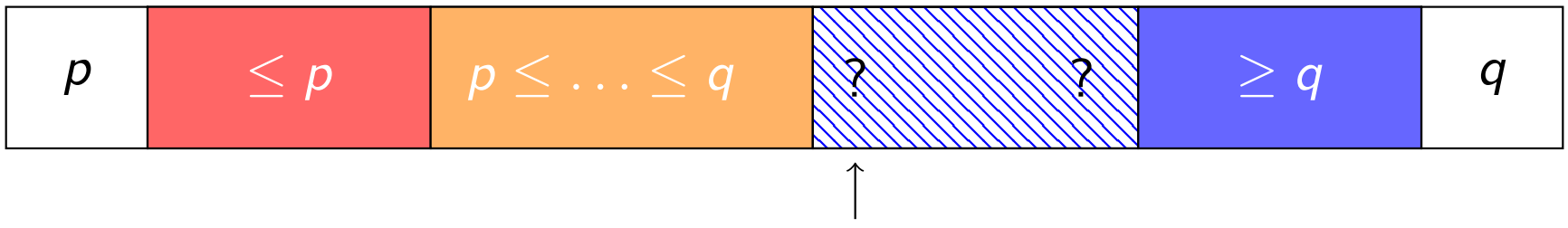
current element



small element


p
≤ p
p ≤ . . . ≤ q
?
?
≥ q
q
exchange


p
$\leq p$
$p \leq \ldots \leq q$
?
?
$\geq q$
q



inspect next element