Pivot Sampling in Dual-Pivot Quicksort

Sebastian Wild  Markus E. Nebel
[wild, nebel]@cs.uni-kl.de

16 June 2014

25th International Conference on Probabilistic, Combinatorial and Asymptotic Methods for the

Analysis of Algorithms
Almost identical Quicksort in all programming libraries!

2009: Java switches to Yaroslavskiy’s algorithm

- Dual-Pivot Quicksort with **new** partitioning method
- Faster running time
- No proper analysis at that time

Why not earlier?

- Other dual-pivot variants had been studied
- Could **not** beat classic Quicksort
Almost identical Quicksort in all programming libraries!

- 2009: Java switches to Yaroslavskiy’s algorithm
  - Dual-Pivot Quicksort with new partitioning method
  - Faster running time
  - No proper analysis at that time

Why not earlier?

- Other dual-pivot variants had been studied
- Could not beat classic Quicksort
Almost identical Quicksort in all programming libraries!

**2009**: Java switches to **Yaroslavskiy’s algorithm**

- **Dual-Pivot** Quicksort with **new** partitioning method
- Faster running time
- No proper analysis at that time

Why not earlier?

- Other dual-pivot variants had been studied
- Could **not** beat classic Quicksort
Almost identical Quicksort in all programming libraries!

2009: Java switches to Yaroslavskiy’s algorithm

- **Dual-Pivot** Quicksort with new partitioning method
- Faster running time
- No proper analysis at that time

Why not earlier?

- Other dual-pivot variants had been studied
- Could **not** beat classic Quicksort
## Previous Results (Average Case)

<table>
<thead>
<tr>
<th></th>
<th>Classic</th>
<th>Dual-Pivot</th>
<th>Relative</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Running Time</strong></td>
<td></td>
<td>(from various experiments)</td>
<td>±10±2%</td>
</tr>
<tr>
<td>Comparisons</td>
<td>$2n \ln n$</td>
<td>$1.9n \ln n$</td>
<td>−5%</td>
</tr>
<tr>
<td>Swaps</td>
<td>$0.3n \ln n$</td>
<td>$0.6n \ln n$</td>
<td>+80%</td>
</tr>
<tr>
<td>Bytecode Instructions</td>
<td>$18n \ln n$</td>
<td>$21.7n \ln n$</td>
<td>+20.6%</td>
</tr>
<tr>
<td>MMIX oops υ</td>
<td>$11n \ln n$</td>
<td>$13.1n \ln n$</td>
<td>+19.1%</td>
</tr>
<tr>
<td>MMIX mems μ</td>
<td>$2.6n \ln n$</td>
<td>$2.8n \ln n$</td>
<td>+5%</td>
</tr>
<tr>
<td>fresh elements (∝ cache misses)</td>
<td>$2n \ln n$</td>
<td>$1.6n \ln n$</td>
<td>−20%</td>
</tr>
</tbody>
</table>

+ $O(n)$, average case results

**Results for pivots at fixed positions.**
## Previous Results (Average Case)

<table>
<thead>
<tr>
<th></th>
<th>Classic</th>
<th>Dual-Pivot</th>
<th>Relative</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Running Time</strong></td>
<td>(from various experiments)</td>
<td></td>
<td>−10 ± 2%</td>
</tr>
<tr>
<td><strong>Comparisons</strong></td>
<td>2$n \ln n$</td>
<td>1.9$n \ln n$</td>
<td>−5%</td>
</tr>
<tr>
<td><strong>Swaps</strong></td>
<td>0.3$n \ln n$</td>
<td>0.6$n \ln n$</td>
<td>+80%</td>
</tr>
<tr>
<td><strong>Bytecode Instructions</strong></td>
<td>18$n \ln n$</td>
<td>21.7$n \ln n$</td>
<td>+20.6%</td>
</tr>
<tr>
<td>MMIX oops $\nu$</td>
<td>11$n \ln n$</td>
<td>13.1$n \ln n$</td>
<td>+19.1%</td>
</tr>
<tr>
<td>MMIX mems $\mu$</td>
<td>2.6$n \ln n$</td>
<td>2.8$n \ln n$</td>
<td>+5%</td>
</tr>
<tr>
<td><strong>fresh elements (≈ cache misses)</strong></td>
<td>2$n \ln n$</td>
<td>1.6$n \ln n$</td>
<td>−20%</td>
</tr>
</tbody>
</table>

+ $O(n)$, average case results

---

Results for pivots at fixed positions.
## Previous Results (Average Case)

<table>
<thead>
<tr>
<th></th>
<th>Classic</th>
<th>Dual-Pivot</th>
<th>Relative</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Running Time</strong></td>
<td>(from various experiments)</td>
<td></td>
<td>−10±2%</td>
</tr>
<tr>
<td><strong>Comparisons</strong></td>
<td>$2n \ln n$</td>
<td>$1.9n \ln n$</td>
<td>−5%</td>
</tr>
<tr>
<td><strong>Swaps</strong></td>
<td>$0.3n \ln n$</td>
<td>$0.6n \ln n$</td>
<td>+80%</td>
</tr>
<tr>
<td><strong>Bytecode Instructions</strong></td>
<td>$18n \ln n$</td>
<td>$21.7n \ln n$</td>
<td>+20.6%</td>
</tr>
<tr>
<td>MMIX oops $\nu$</td>
<td>$11n \ln n$</td>
<td>$13.1n \ln n$</td>
<td>+19.1%</td>
</tr>
<tr>
<td>MMIX mems $\mu$</td>
<td>$2.6n \ln n$</td>
<td>$2.8n \ln n$</td>
<td>+5%</td>
</tr>
<tr>
<td>fresh elements ($\approx$ cache misses)</td>
<td>$2n \ln n$</td>
<td>$1.6n \ln n$</td>
<td>−20%</td>
</tr>
</tbody>
</table>

+ $O(n)$, average case results

### Results for pivots at fixed positions.
## Previous Results (Average Case)

<table>
<thead>
<tr>
<th></th>
<th>Classic</th>
<th>Dual-Pivot</th>
<th>Relative</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Running Time</strong></td>
<td>(from various experiments)</td>
<td></td>
<td>$-10 \pm 2%$</td>
</tr>
<tr>
<td><strong>Comparisons</strong></td>
<td>$2n \ln n$</td>
<td>$1.9n \ln n$</td>
<td>$-5%$</td>
</tr>
<tr>
<td><strong>Swaps</strong></td>
<td>$0.3n \ln n$</td>
<td>$0.6n \ln n$</td>
<td>$+80%$</td>
</tr>
<tr>
<td><strong>Bytecode Instructions</strong></td>
<td>$18n \ln n$</td>
<td>$21.7n \ln n$</td>
<td>$+20.6%$</td>
</tr>
<tr>
<td>MMIX oops $\nu$</td>
<td>$11n \ln n$</td>
<td>$13.1n \ln n$</td>
<td>$+19.1%$</td>
</tr>
<tr>
<td>MMIX mems $\mu$</td>
<td>$2.6n \ln n$</td>
<td>$2.8n \ln n$</td>
<td>$+5%$</td>
</tr>
<tr>
<td>fresh elements ($\approx$ cache misses)</td>
<td>$2n \ln n$</td>
<td>$1.6n \ln n$</td>
<td>$-20%$</td>
</tr>
</tbody>
</table>

+ $O(n)$, average case results

### Results for pivots at fixed positions.
## Previous Results (Average Case)

<table>
<thead>
<tr>
<th></th>
<th>Classic</th>
<th>Dual-Pivot</th>
<th>Relative</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Running Time</strong></td>
<td>(from various experiments)</td>
<td></td>
<td>−10±2%</td>
</tr>
<tr>
<td><strong>Comparisons</strong></td>
<td>$2n \ln n$</td>
<td>$1.9n \ln n$</td>
<td>−5%</td>
</tr>
<tr>
<td><strong>Swaps</strong></td>
<td>$0.3n \ln n$</td>
<td>$0.6n \ln n$</td>
<td>+80%</td>
</tr>
<tr>
<td><strong>Bytecode Instructions</strong></td>
<td>$18n \ln n$</td>
<td>$21.7n \ln n$</td>
<td>+20.6%</td>
</tr>
<tr>
<td><strong>MMIX oops υ</strong></td>
<td>$11n \ln n$</td>
<td>$13.1n \ln n$</td>
<td>+19.1%</td>
</tr>
<tr>
<td><strong>MMIX mems µ</strong></td>
<td>$2.6n \ln n$</td>
<td>$2.8n \ln n$</td>
<td>+5%</td>
</tr>
<tr>
<td><strong>fresh elements (≈ cache misses)</strong></td>
<td>$2n \ln n$</td>
<td>$1.6n \ln n$</td>
<td>−20%</td>
</tr>
</tbody>
</table>

+ O(n), average case results

---

**Results for pivots at fixed positions.**

... nobody uses that!

---

**Here:** Choosing pivots from sample.
### Previous Results (Average Case)

<table>
<thead>
<tr>
<th></th>
<th>Classic</th>
<th>Dual-Pivot</th>
<th>Relative</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Running Time</strong></td>
<td>(from various experiments)</td>
<td></td>
<td>−10±2%</td>
</tr>
<tr>
<td><strong>Comparisons</strong></td>
<td>$2n \ln n$</td>
<td>$1.9n \ln n$</td>
<td>−5%</td>
</tr>
<tr>
<td><strong>Swaps</strong></td>
<td>$0.3n \ln n$</td>
<td>$0.6n \ln n$</td>
<td>+80%</td>
</tr>
<tr>
<td><strong>Bytecode Instructions</strong></td>
<td>$18n \ln n$</td>
<td>$21.7n \ln n$</td>
<td>+20.6%</td>
</tr>
<tr>
<td><strong>MMIX oops $\nu$</strong></td>
<td>$11n \ln n$</td>
<td>$13.1n \ln n$</td>
<td>+19.1%</td>
</tr>
<tr>
<td><strong>MMIX mems $\mu$</strong></td>
<td>$2.6n \ln n$</td>
<td>$2.8n \ln n$</td>
<td>+5%</td>
</tr>
<tr>
<td><strong>fresh elements (≈ cache misses)</strong></td>
<td>$2n \ln n$</td>
<td>$1.6n \ln n$</td>
<td>−20%</td>
</tr>
</tbody>
</table>

+ $O(n)$, average case results

---

**Results for pivots at fixed positions.** ... nobody uses that!

**Here:** Choosing pivots from sample.
Algorithm (Conceptual View)

1. Choose **two pivots** $P \leq Q$
2. For each element $x$, determine its **class**
   - **Small** for $x < P$
   - **Medium** for $P < x < Q$
   - **Large** for $Q < x$

by comparing $x$ to **pivots** $P$ and $Q$

3. Arrange elements according to classes:

4. Sort subarrays recursively.

*How to implement efficiently on arrays?*
Dual Pivot Quicksort

Algorithm (Conceptual View)

1. Choose two pivots \( P \leq Q \)
2. For each element \( x \), determine its class
   - **small** for \( x < P \)
   - **medium** for \( P < x < Q \)
   - **large** for \( Q < x \)
   by comparing \( x \) to pivots \( P \) and \( Q \)
3. Arrange elements according to classes:

   ![Diagram of elements arranged by classes]

4. Sort subarrays recursively.

*How to implement efficiently on arrays?*
Dual Pivot Quicksort

Algorithm (Conceptual View)

1. Choose **two pivots** $P \leq Q$
2. For each element $x$, determine its **class**
   - **small** for $x < P$
   - **medium** for $P < x < Q$
   - **large** for $Q < x$
   by comparing $x$ to pivots $P$ and $Q$
3. Arrange elements according to classes:

![Diagram of classes]

4. Sort subarrays recursively.

*How to implement 3 efficiently on arrays?*
Dual Pivot Quicksort

Algorithm (Conceptual View)

1. Choose two pivots $P \leq Q$
2. For each element $x$, determine its class
   - small for $x < P$
   - medium for $P < x < Q$
   - large for $Q < x$
   by comparing $x$ to pivots $P$ and $Q$
3. Arrange elements according to classes:

```
P       Q
```
4. Sort subarrays recursively.

How to implement efficiently on arrays?
Yaroslavskiy’s Algorithm

Invariant:

<table>
<thead>
<tr>
<th>P</th>
<th>k</th>
<th>g</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>5</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>8</td>
<td>6</td>
</tr>
</tbody>
</table>
Yaroslavskiy’s Algorithm

Invariant: P Q
Yaroslavskiy’s Algorithm

Invariant: P ___________________ Q
Yaroslavskiy’s Algorithm

Invariant: P ______ Q
Yaroslavskiy’s Algorithm

---

**Invariant:**

\[ P \leq o \leq Q \]
Yaroslavskiy’s Algorithm

Invariant: \( P \leq o \leq Q \)
Yaroslavskiy’s Algorithm

Invariant: $P \leq o \leq Q$
Yaroslavskiy’s Algorithm

Invariant: P < P \leq o \leq Q
Yaroslavskiy’s Algorithm

Invariant: $P < P \leq 0 \leq Q$
Yaroslavskiy’s Algorithm

Invariant:

Dual-Pivot Quicksort
Yaroslavskiy’s Algorithm

Invariant: \( P < P < P \leq Q \leq Q \)
Yaroslavskiy’s Algorithm

Invariant: \[ P < P \leq Q \leq Q \]

Diagram: [Diagram showing the algorithm with decision nodes and data elements]

Legend:
- \( < P \)
- \( > Q \)
- \( < Q \)
- \( > P \)
- \( \leq \leq \)
- \( \iff \)
- \( \neg \)
- \( \neg \neg \)
- \( \equiv \}

Data Elements:
- \( P \): 3, 1, 5, 7
- \( Q \): 4, 2, 8, 6

Algorithm:
- If \( < P \):
  - If \( \leq \leq \)
  - Swap \( \ell \)
  - Skip
- If \( > Q \):
  - Skip
- If \( < Q \):
  - Swap \( g \)
- If \( > P \):
  - Swap \( k \)
Yaroslavskiy’s Algorithm

![Diagram of Yaroslavskiy’s Algorithm]

**Invariant:**

- **P:** Elements less than the pivot.
- **Q:** Elements greater than or equal to the pivot.
- **P ≤ o ≤ Q:** Elements in the range between the pivot and the current pivot.

**Example:**

- Elements: 3, 1, 5, 7, 2, 8, 6
- Pivot: 4

The algorithm sorts elements based on their relationship to the pivot, ensuring the invariant is maintained throughout the sorting process.
Yaroslavkiy’s Algorithm

Invariant:

\[ P \leq \circ \leq Q \]
Yaroslavskiy’s Algorithm

Invariant:

\[
P < P \quad P \leq o \leq Q \quad Q \geq Q
\]
Yaroslavskiy’s Algorithm

- If \( < P \) then:
  - \( \ell \) to \( \ell \)
  - \( \llcorner \) to \( \llcorner \)

- If \( < Q \) then:
  - \( \llcorner \) to \( \llcorner \)

- If \( > Q \) then:
  - \( \ell \) to \( \ell \)
  - \( \llcorner \) to \( \llcorner \)

Invariant:

\[
\begin{array}{c}
P < P \quad P \leq \circ \leq Q \\
\end{array}
\]
Yaroslavskiy’s Algorithm

Invariant:

P \quad < P \quad P \leq o \leq Q \quad \geq Q \quad Q
Yaroslavskiy’s Algorithm

Invariant: \( P < P \leq \leq Q \geq Q \)
Yaroslavskiy’s Algorithm

Swap $\ell$ if $< P$?

Swap $g$ if $< Q$?

Swap $\ell$ if $\leq P$.

Swap $k$ if $\geq Q$.

Dual-Pivot Quicksort

Invariant:
Yaroslavskiy’s Algorithm

\[ < P? \]
\[ > Q? \]
\[ \text{swap } \ell \]
\[ \text{skip} \]
\[ < Q? \]
\[ \text{swap } g \]
\[ > Q? \]
\[ \text{skip} \]
\[ < P? \]
\[ \text{swap } \ell \]
\[ \text{swap } k \]

P:
3 1 2 5 4

Q:
7 8 6
Yaroslavskiy’s Algorithm

\[
\begin{array}{c}
\text{\textbf{Yaroslavskiy’s Algorithm}} \\
\end{array}
\]

\[
\begin{array}{c}
\text{Dual-Pivot Quicksort} \\
2014-06-16 5 / 16
\end{array}
\]
Yaroslavskiy’s Algorithm

- `< P?` with options: swap ℓ, skip
- `< Q?` with options: swap ℓ, swap g

- `> Q?` with options: swap ℓ, swap k, skip
- `< P?` with options: skip

---

Sebastian Wild

Dual-Pivot Quicksort

2014-06-16 5 / 16
Pivot Sampling

- Classic Quicksort: pivot as median of $k$
  (in practice: $k = 3$, pseudomedian of 9)

- We need two pivots.

- Here: parametric scheme with parameter $t = (t_1, t_2, t_3)$
  1. Sample $k = t_1 + t_2 + t_3 + 2$ elements
  2. Sort the sample
  3. Select pivots s.t. in sorted sample

  - $t_1$ smaller
  - $t_2$ between
  - $t_3$ larger than pivots

- ALENEX 2013:
  Empirical evidence that asymmetric $t = (0, 1, 2)$ beats $t = (1, 1, 1)$!
Pivot Sampling

- Classic Quicksort: pivot as **median of** $k$
  (in practice: $k = 3$, pseudomedian of 9)

- We need **two** pivots.
  
  - **Here:** parametric scheme with parameter $t = (t_1, t_2, t_3)$
    1. Sample $k = t_1 + t_2 + t_3 + 2$ elements
    2. Sort the sample
    3. Select pivots s. t. in sorted sample

    | $t_1$ smaller,          | Example with $k = 8$ and $t = (3, 2, 1)$: |
    | $t_2$ between and       | P | Q |
    | $t_3$ larger than pivots| $t_1$ | $t_2$ | $t_3$ |

- ALENEX 2013:
  Empirical evidence that **asymmetric** $t = (0, 1, 2)$ beats $t = (1, 1, 1)$!
Pivot Sampling

- Classic Quicksort: pivot as median of \( k \)
  (in practice: \( k = 3 \), pseudomedian of 9)

- We need two pivots.

- Here: parametric scheme with parameter \( t = (t_1, t_2, t_3) \)
  1. Sample \( k = t_1 + t_2 + t_3 + 2 \) elements
  2. Sort the sample
  3. Select pivots s.t. in sorted sample

\[ t_1 \text{ smaller}, \quad t_2 \text{ between}, \quad t_3 \text{ larger} \quad \text{than pivots} \]

Example with \( k = 8 \) and \( t = (3, 2, 1) \):

ALENEX 2013:

Empirical evidence that asymmetric \( t = (0, 1, 2) \) beats \( t = (1, 1, 1) \)!
Pivot Sampling

- Classic Quicksort: pivot as median of \( k \)
  (in practice: \( k = 3 \), pseudomedian of 9)
- We need **two** pivots.
- **Here:** parametric scheme with parameter \( t = (t_1, t_2, t_3) \)
  1. Sample \( k = t_1 + t_2 + t_3 + 2 \) elements
  2. **Sort** the sample
  3. Select pivots s. t. in sorted sample

\[ t_1 \text{ smaller}, \quad t_2 \text{ between} \quad \text{and} \quad t_3 \text{ larger than pivots} \]

**Example** with \( k = 8 \) and \( t = (3, 2, 1) \):

- ALENEX 2013:
  Empirical evidence that **asymmetric** \( t = (0, 1, 2) \) beats \( t = (1, 1, 1) \)!
**Goal:** Count *comparisons* in Dual-Pivot Quicksort with Pivot Sampling

$C_n$: (random!) #cmps to sort $n$ i.i.d. $Uniform(0, 1)$ elements

$\leadsto$ Distributional Recurrence:

$$C_n \overset{D}{=} T_n + C_{J_1} + C_{J_2} + C_{J_3}$$

- $J = (J_1, J_2, J_3)$: (random) subproblem sizes
- $T_n$: (random) #cmps of first partitioning step

$\leadsto$ Taking expectations and conditioning on $J$:

$$E[C_n] = E[T_n] + \sum_{j=0}^{n-2} w_{n,j} E[C_j]$$

with $w_{n,j} = \Pr[J_1 = j] + \Pr[J_2 = j] + \Pr[J_3 = j]$
Expected Number of Comparisons

**Goal:** Count *comparisons* in Dual-Pivot Quicksort with Pivot Sampling

$C_n$: (random!) #cmps to sort $n$ i.i.d. *Uniform*(0, 1) elements

 Arnab \[ \text{Distributional Recurrence:} \]

$$C_n \overset{D}{=} T_n + C_{J_1} + C_{J_2} + C_{J_3}$$

- $J = (J_1, J_2, J_3)$: (random) subproblem sizes
- $T_n$: (random) #cmps of first partitioning step

Arnab \[ \text{Taking expectations and conditioning on } J: \]

$$E[C_n] = E[T_n] + \sum_{j=0}^{n-2} w_{n,j} E[C_j]$$

with $w_{n,j} = Pr[J_1 = j] + Pr[J_2 = j] + Pr[J_3 = j]$
Goal: Count comparisons in Dual-Pivot Quicksort with Pivot Sampling

\( \mathbb{E}[C_n] = \mathbb{E}[T_n] + \sum_{j=0}^{n-2} w_{n,j} \mathbb{E}[C_j] \)

with \( w_{n,j} = \Pr[J_1 = j] + \Pr[J_2 = j] + \Pr[J_3 = j] \)
Solution of Recurrence

**Recurrence:** \( \mathbb{E}[C_n] = \mathbb{E}[T_n] + \sum_{j=0}^{n-2} w_{n,j} \mathbb{E}[C_j] \) with \( \mathbb{E}[T_n] \sim \gamma n \)

Apply *Continuous Master Theorem* [Roura 2001]

Idea: Approximate sum by integral over relative subproblem size \( \alpha = j/n \)

\[
\sum_{j=0}^{n-2} w_{n,j} \mathbb{E}[C_j] \approx \int_{0}^{n} w_{n,j} \mathbb{E}[C_j] \, dj \approx \int_{0}^{1} n \cdot w_{n,\alpha n} \mathbb{E}[C_{\alpha n}] \, d\alpha
\]

\[
\Rightarrow \text{Continuous Recurrence: } \mathbb{E}[C_n] \approx \mathbb{E}[T_n] + \int_{0}^{1} w(\alpha) \mathbb{E}[C_{\alpha n}] \, d\alpha
\]

Ansatz: \( \mathbb{E}[C_n] = \gamma n \ln n \)

\[
\Rightarrow \text{fulfills recurrence for } T_n = \sum_{k=1}^{n} \frac{n+1}{k+1} (2k+1 - 2(n+1))
\]
Solution of Recurrence

Recurrence: $\mathbb{E}[C_n] = \mathbb{E}[T_n] + \sum_{j=0}^{n-2} w_{n,j} \mathbb{E}[C_j]$ with $\mathbb{E}[T_n] \sim \gamma n$

Apply *Continuous Master Theorem* [Roura 2001]

Idea: Approximate sum by integral over relative subproblem size $\alpha = j/n$

$$\sum_{j=0}^{n-2} w_{n,j} \mathbb{E}[C_j] \approx \int_0^n w_{n,j} \mathbb{E}[C_j] \, dj \approx \int_0^1 n \cdot w_{n,\alpha n} \mathbb{E}[C_{\alpha n}] \, d\alpha$$

$\Rightarrow$ Continuous Recurrence: $\mathbb{E}[C_n] \approx \mathbb{E}[T_n] + \int_0^1 w(\alpha) \mathbb{E}[C_{\alpha n}] \, d\alpha$

Ansatz: $\mathbb{E}[C_n] = \frac{\gamma}{H} n \ln n$

$\Rightarrow$ fulfills recurrence for $H = \sum_{l=1}^3 \frac{t_l + 1}{k + 1} (\mathcal{H}_{k+1} - \mathcal{H}_{t_l+1})$

$\Rightarrow$ $\mathbb{E}[C_n] \sim \frac{\gamma}{H} n \ln n$

Next: What is $\gamma$?
Solution of Recurrence

**Recurrence:** \( \mathbb{E}[C_n] = \mathbb{E}[T_n] + \sum_{j=0}^{n-2} w_{n,j} \mathbb{E}[C_j] \) with \( \mathbb{E}[T_n] \sim \gamma n \)

Apply *Continuous Master Theorem* [Roura 2001]

**Idea:** Approximate sum by integral over relative subproblem size \( \alpha = j/n \)

\[
\sum_{j=0}^{n-2} w_{n,j} \mathbb{E}[C_j] \approx \int_0^n w_{n,j} \mathbb{E}[C_j] \, dj \approx \int_0^1 n \cdot w_{n,\alpha n} \mathbb{E}[C_{\alpha n}] \, d\alpha
\]

\( \rightsquigarrow \) **Continuous Recurrence:** \( \mathbb{E}[C_n] \approx \mathbb{E}[T_n] + \int_0^1 w(\alpha) \mathbb{E}[C_{\alpha n}] \, d\alpha \)

**Ansatz:** \( \mathbb{E}[C_n] = \frac{\gamma}{H} n \ln n \)

\( \rightsquigarrow \) fulfills recurrence for \( H = \sum_{l=1}^{3} \frac{t_l + 1}{k + 1} (\mathcal{H}_{k+1} - \mathcal{H}_{t_l+1}) \)

\( \rightsquigarrow \mathbb{E}[C_n] \sim \frac{\gamma}{H} n \ln n \)

Next: What is \( \gamma \)?
Solution of Recurrence

**Recurrence:** $E[C_n] = E[T_n] + \sum_{j=0}^{n-2} w_{n,j} E[C_j]$ with $E[T_n] \sim \gamma n$

Apply *Continuous Master Theorem* [Roura 2001]

**Idea:** Approximate sum by integral over relative subproblem size $\alpha = j/n$

$$\sum_{j=0}^{n-2} w_{n,j} E[C_j] \approx \int_0^n w_{n,j} E[C_j] \, dj \approx \int_0^1 n \cdot w_{n,\alpha n} E[C_{\alpha n}] \, d\alpha$$

$\rightarrow w(\alpha)$

$\Rightarrow$ Continuous Recurrence: $E[C_n] \approx E[T_n] + \int_0^1 w(\alpha) E[C_{\alpha n}] \, d\alpha$

**Ansatz:** $E[C_n] = \frac{\gamma}{H} n \ln n$

$\Rightarrow$ fulfills recurrence for $H = \sum_{l=1}^{3} \frac{t_l + 1}{k + 1} (H_{k+1} - H_{t_l + 1})$

$\Rightarrow E[C_n] \sim \frac{\gamma}{H} n \ln n$

Next: What is $\gamma$?
Solution of Recurrence

Recurrence: $\mathbb{E}[C_n] = \mathbb{E}[T_n] + \sum_{j=0}^{n-2} w_{n,j} \mathbb{E}[C_j]$ with $\mathbb{E}[T_n] \sim \gamma n$

Apply **Continuous Master Theorem** [Roura 2001]

**Idea:** Approximate sum by integral over relative subproblem size $\alpha = j/n$

$$\sum_{j=0}^{n-2} w_{n,j} \mathbb{E}[C_j] \approx \int_0^n w_{n,j} \mathbb{E}[C_j] \, dj \approx \int_0^1 n \cdot w_{n,\alpha n} \mathbb{E}[C_{\alpha n}] \, d\alpha \to w(\alpha)$$

$\sim$ **Continuous** Recurrence: $\mathbb{E}[C_n] \approx \mathbb{E}[T_n] + \int_0^1 w(\alpha) \mathbb{E}[C_{\alpha n}] \, d\alpha$

**Ansatz:** $\mathbb{E}[C_n] = \frac{\gamma}{H} n \ln n$

$\sim$ fulfills recurrence for $H = \sum_{l=1}^3 \frac{t_l + 1}{k + 1} (H_{k+1} - H_{t_l+1})$

$\Rightarrow \mathbb{E}[C_n] \sim \frac{\gamma}{H} n \ln n$

Next: What is $\gamma$?
Solution of Recurrence

**Recurrence:** \( \mathbb{E}[C_n] = \mathbb{E}[T_n] + \sum_{j=0}^{n-2} w_{n,j} \mathbb{E}[C_j] \) with \( \mathbb{E}[T_n] \sim \gamma n \)

Apply *Continuous Master Theorem* [Roura 2001]

**Idea:** Approximate sum by integral over relative subproblem size \( \alpha = j/n \)

\[
\sum_{j=0}^{n-2} w_{n,j} \mathbb{E}[C_j] \approx \int_0^n w_{n,j} \mathbb{E}[C_j] \, dj \approx \int_0^1 n \cdot w_{n,\alpha n} \mathbb{E}[C_{\alpha n}] \, d\alpha \to w(\alpha)
\]

\( \sim \) Continuous Recurrence: \( \mathbb{E}[C_n] \approx \mathbb{E}[T_n] + \int_0^1 w(\alpha) \mathbb{E}[C_{\alpha n}] \, d\alpha \)

**Ansatz:** \( \mathbb{E}[C_n] = \frac{\gamma}{H} n \ln n \)

\( \sim \) fulfills recurrence for \( H = \sum_{l=1}^3 \frac{t_l + 1}{k + 1} (\mathcal{H}_{k+1} - \mathcal{H}_{t_l+1}) \)

\( \sim \) \( \mathbb{E}[C_n] \sim \frac{\gamma}{H} n \ln n \)

Next: What is \( \gamma \)?
Solution of Recurrence

Recurrence: \( \mathbb{E}[C_n] = \mathbb{E}[T_n] + \sum_{j=0}^{n-2} w_{n,j} \mathbb{E}[C_j] \) with \( \mathbb{E}[T_n] \sim \gamma n \)

Apply \textit{Continuous Master Theorem} [Roura 2001]

Idea: Approximate sum by \textbf{integral} over \textbf{relative} subproblem size \( \alpha = j/n \)

\[
\sum_{j=0}^{n-2} w_{n,j} \mathbb{E}[C_j] \approx \int_0^n w_{n,j} \mathbb{E}[C_j] \, dj \approx \int_0^1 \underbrace{n \cdot w_{n,\alpha n}} \mathbb{E}[C_{\alpha n}] \, d\alpha \rightarrow w(\alpha)
\]

\( \xrightarrow{\sim} \) \textbf{Continuous} Recurrence: \( \mathbb{E}[C_n] \approx \mathbb{E}[T_n] + \int_0^1 w(\alpha) \mathbb{E}[C_{\alpha n}] \, d\alpha \)

Ansatz: \( \mathbb{E}[C_n] = \frac{\gamma}{H} n \ln n \)

\( \xrightarrow{\sim} \) fulfills recurrence for \( H = \sum_{l=1}^{3} \frac{t_l + 1}{k + 1} (\mathcal{H}_{k+1} - \mathcal{H}_{t_l+1}) \)

\( \xrightarrow{\sim} \) \( \mathbb{E}[C_n] \sim \frac{\gamma}{H} n \ln n \)

Next: What is \( \gamma \)?
Solution of Recurrence

**Recurrence:** \( \mathbb{E}[C_n] = \mathbb{E}[T_n] + \sum_{j=0}^{n-2} w_{n,j} \mathbb{E}[C_j] \) with \( \mathbb{E}[T_n] \sim \gamma n \)

Apply **Continuous Master Theorem** [Roura 2001]

**Idea:** Approximate sum by integral over relative subproblem size \( \alpha = j/n \)

\[
\sum_{j=0}^{n-2} w_{n,j} \mathbb{E}[C_j] \approx \int_0^n w_{n,j} \mathbb{E}[C_j] \, dj \approx \int_0^1 n \cdot w_{n,\alpha n} \mathbb{E}[C_{\alpha n}] \, d\alpha
\]

\( \rightarrow w(\alpha) \)

\( \sim \) **Continuous** Recurrence: \( \mathbb{E}[C_n] \approx \mathbb{E}[T_n] + \int_0^1 w(\alpha) \mathbb{E}[C_{\alpha n}] \, d\alpha \)

**Ansatz:** \( \mathbb{E}[C_n] = \frac{\gamma}{H} n \ln n \)

\( \sim \) fulfills recurrence for \( H = \sum_{l=1}^{3} \frac{t_l + 1}{k + 1} (\mathcal{H}_{k+1} - \mathcal{H}_{t_l+1}) \)

\( \rightarrow \mathbb{E}[C_n] \sim \frac{\gamma}{H} n \ln n \)

**Next:** What is \( \gamma \)?
Solution of Recurrence

**Recurrence:** \( \mathbb{E}[C_n] = \mathbb{E}[T_n] + \sum_{j=0}^{n-2} w_{n,j} \mathbb{E}[C_j] \) with \( \mathbb{E}[T_n] \sim \gamma n \)

Apply **Continuous Master Theorem** [Roura 2001]

**Idea:** Approximate sum by integral over relative subproblem size \( \alpha = j/n \)

\[
\sum_{j=0}^{n-2} w_{n,j} \mathbb{E}[C_j] \approx \int_0^n w_{n,j} \mathbb{E}[C_j] \, dj \approx \int_0^1 n \cdot w_{n,\alpha n} \mathbb{E}[C_{\alpha n}] \, d\alpha \\
\rightarrow w(\alpha)
\]

\( \sim \) **Continuous** Recurrence: \( \mathbb{E}[C_n] \approx \mathbb{E}[T_n] + \int_0^1 w(\alpha) \mathbb{E}[C_{\alpha n}] \, d\alpha \)

**Ansatz:** \( \mathbb{E}[C_n] = \frac{\gamma}{H} n \ln n \)

\( \sim \) fulfills recurrence for \( H = \sum_{l=1}^{3} \frac{t_l + 1}{k + 1} (\mathcal{H}_{k+1} - \mathcal{H}_{t_l+1}) \)

\( \sim \) \( \mathbb{E}[C_n] \sim \frac{\gamma}{H} n \ln n \)

**Next:** What is \( \gamma \)?
How many comparisons for one element? It depends! Either 2 or 1.

~ T_n \sim 2n \text{ – “cheap elements”:
- } s @ K: \text{ small in } k’s \text{ range}
- } l @ G: \text{ large in } g’s \text{ range}

~ need ranges K and G!

Recall Invariant:

|K| \sim I_1 + I_2
|G| \sim I_3

with I = (I_1, I_2, I_3) sizes of partitions
How many comparisons for one element? It depends! Either 2 or 1.

Recall Invariant:

\[ P \quad < P \quad P \leq o \leq Q \quad \geq Q \quad Q \]

- \(|K| \sim I_1 + I_2\)
- \(|G| \sim I_3\)

\(\Rightarrow\) need ranges \(K\) and \(G\)!

\(\Rightarrow\) sizes of partitions \(I = (I_1, I_2, I_3)\)
How many comparisons for one element? It depends! Either 2 or 1.

龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃龃日讯

ساء

Recall Invariant: trans

_sizes of partitions

\begin{align*}
|K| & \sim I_1 + I_2 \\
|G| & \sim I_3
\end{align*}
How many comparisons for one element? It depends! Either 2 or 1.

- $< P \ ?$
  - $\checkmark$
  - $\times$
  - swap $\ell$
  - skip
  - $< Q \ ?$
    - $\checkmark$
    - $\times$
    - swap $g$

- $> Q \ ?$
  - $\times$
  - $\checkmark$
  - skip
  - $< P \ ?$
    - $\checkmark$
    - $\times$
    - swap $\ell$
    - swap $k$

$\Rightarrow T_n \sim 2n$ — "cheap elements":
- $s @ K$: small in $k$'s range
- $l @ G$: large in $g$'s range

$\Rightarrow$ need ranges $K$ and $G$!

Recall Invariant:

\[
\begin{array}{c|c|c|c|c|c}
P & < P & P \leq o \leq Q & \geq Q & Q \\
\end{array}
\]

- $|K| \sim I_1 + I_2$
- $|G| \sim I_3$

with $I = (I_1, I_2, I_3)$ sizes of partitions
#Comparisons per Partitioning

How many comparisons for one element? It depends! Either 2 or 1.

$$\begin{align*}
\text{if } < \ P & \text{ then swap } \ell \\
\text{if } \geq \ Q & \text{ then swap } g
\end{align*}$$

$$\begin{align*}
\text{if } < \ P & \text{ then skip} \\
\text{if } > \ Q & \text{ then swap } k
\end{align*}$$

$$\Rightarrow T_n \sim 2n \ - \ "cheap\ elements":$$

- $s @ K$: small in $k$’s range
- $l @ G$: large in $g$’s range

$$\Rightarrow \text{need ranges } K \text{ and } G!$$

Recall Invariant:

$$\begin{align*}
P \ &< \ P \ &P \leq \ o \leq \ Q \ &\geq \ Q \ &Q
\end{align*}$$

- $|K| \sim I_1 + I_2$
- $|G| \sim I_3$

with $I = (I_1, I_2, I_3)$ sizes of partitions
How many comparisons for one element? It depends! Either 2 or 1.

\[ \begin{align*}
\text{if } P &< Q, \\
\text{then use } &\text{swap } \ell \\
\quad \text{or } &\text{swap } g
\end{align*} \]

\[ \begin{align*}
\text{if } P &> Q, \\
\text{then use } &\text{swap } k
\end{align*} \]

\[ \begin{align*}
\text{if } P &< Q, \\
\text{then use } &\text{swap } \ell \\
\quad \text{or } &\text{swap } k
\end{align*} \]

\[ \begin{align*}
\text{if } P &> Q, \\
\text{then use } &\text{swap } k
\end{align*} \]

\[ T_n \sim 2n \sim \text{“cheap elements”:\n\begin{align*}
\text{if } s &\in \mathcal{K}, \\
\text{then } &\text{small in } k\text{’s range}
\end{align*} \]

\[ \begin{align*}
\text{if } l &\in \mathcal{G}, \\
\text{then } &\text{large in } g\text{’s range}
\end{align*} \]

\[ \sim \text{need ranges } \mathcal{K} \text{ and } \mathcal{G}! \]

Recall Invariant:

\[ \begin{array}{c}
\begin{array}{c}
\text{P} \\
\text{< P} \\
\text{P \leq \circ \leq Q} \\
\text{\geq Q} \\
\text{Q}
\end{array}
\end{array} \]

\[ \begin{align*}
|\mathcal{K}| &\sim I_1 + I_2 \\
|\mathcal{G}| &\sim I_3
\end{align*} \]

with \( I = (I_1, I_2, I_3) \) sizes of partitions
# Comparisons per Partitioning

How many comparisons for one element? It depends! Either 2 or 1.

唆 Tₙ ～ 2n — ”cheap elements“:
- s @ K: small in k’s range
- l @ G: large in g’s range

唆 need ranges K and G!

Recall Invariant:

| K | ~ I₁ + I₂  
| G | ~ I₃  

with I = (I₁, I₂, I₃) sizes of partitions
Distribution of $s@K$ and $l@G$

Consider $I$ fixed.

- $|K| \sim I_1 + I_2$
- $|G| \sim I_3$
- $\#_{small} = I_1$
- $\#_{large} = I_3$

Draw positions of small elements:
1. All array indices in urn
2. Draw without replacement
3. Count green balls

$I = (3, 2, 3)$
Distribution of $s@K$ and $l@G$

Consider $I$ fixed.

- $|K| \sim I_1 + I_2$
- $|G| \sim I_3$
- $\text{#small} = I_1$
- $\text{#large} = I_3$

Draw positions of small elements:

1. All array indices in urn
2. Draw without replacement
3. Count green balls

$I = (3, 2, 3)$

$\Rightarrow s@K = 2$
Consider \( I \) fixed.

\[
\begin{align*}
|K| & \sim I_1 + I_2 \\
|G| & \sim I_3
\end{align*}
\]

\( \#\text{small} = I_1 \)

\( \#\text{large} = I_3 \)

Draw \textit{positions} of small elements:
1. All array \textit{indices} in urn
2. Draw without replacement
3. Count green balls

\( I = (3, 2, 3) \)

\( \Rightarrow s @ K = 2 \)
Distribution of $s_{@K}$ and $l_{@G}$

Consider $I$ fixed.

$|K| \sim I_1 + I_2$

$|G| \sim I_3$

$\#small = I_1$

$\#large = I_3$

Draw positions of small elements:

1. All array indices in urn
2. Draw without replacement
3. Count green balls

$I = (3, 2, 3)$

$\Rightarrow s_{@K} = 2$
Distribution of $s_{\mathcal{K}}$ and $l_{\mathcal{G}}$

Consider $I$ fixed.

\[ |\mathcal{K}| \sim I_1 + I_2 \quad \text{and} \quad |\mathcal{G}| \sim I_3 \]

\[ \#_{\text{small}} = I_1 \quad \text{and} \quad \#_{\text{large}} = I_3 \]

Draw positions of small elements:

1. All array indices in urn
2. Draw without replacement
3. Count green balls

$I = (3, 2, 3)$

\[ \sim \to s_{\mathcal{K}} = 2 \]
Distribution of $s@K$ and $l@G$

Consider $I$ fixed.

$$|K| \sim I_1 + I_2$$  
$$|G| \sim I_3$$  

$$\#\text{small} = I_1$$  
$$\#\text{large} = I_3$$

Draw positions of small elements:
1. All array indices in urn
2. Draw without replacement
3. Count green balls

$I = (3, 2, 3)$

$$\Rightarrow s@K = 2$$
Consider I fixed.

\[ |K| \sim I_1 + I_2 \quad \text{and} \quad |G| \sim I_3 \]

\[ \#\text{small} = I_1 \quad \text{and} \quad \#\text{large} = I_3 \]

Draw **positions** of small elements:
1. All array **indices** in urn
2. Draw without replacement
3. Count green balls

\[ I = (3, 2, 3) \]

\[ \implies s@K = 2 \]
Distribution of $s@\mathcal{K}$ and $l@\mathcal{G}$

Consider $I$ fixed.

$|\mathcal{K}| \sim I_1 + I_2$

$|\mathcal{G}| \sim I_3$

$\#\text{small} = I_1$

$\#\text{large} = I_3$

Draw positions of small elements:

1. All array indices in urn
2. Draw without replacement
3. Count green balls

$I = (3, 2, 3)$

$\sim \ s@\mathcal{K} = 2$
Distribution of \( s@K \) and \( l@G \)

Consider \( I \) fixed.

\[
\begin{align*}
|K| & \sim I_1 + I_2 \\
|G| & \sim I_3 \\
\text{#small} & = I_1 \\
\text{#large} & = I_3
\end{align*}
\]

\( s@K \overset{D}{=} \text{Hypergeometric}(\text{#small}, |K|, n - k) \)

Draw positions of small elements:
1. All array indices in urn
2. Draw without replacement
3. Count green balls

\( I = (3, 2, 3) \)

\[
\begin{array}{cccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
\end{array}
\]

\( \sim \) \( s@K = 2 \)

Similarly: \( l@G \overset{D}{=} \text{Hypergeometric}(\text{#large}, |G|, n - k) \)
Consider $I$ fixed.

\[ |K| \sim I_1 + I_2 \quad \#\text{small} = I_1 \]
\[ |G| \sim I_3 \quad \#\text{large} = I_3 \]

\[ s@K \overset{D}{=} \text{Hypergeometric}(\#\text{small}, |K|, n - k) \]

Draw positions of small elements:
1. All array indices in urn
2. Draw without replacement
3. Count green balls

\[ I = (3, 2, 3) \]

Similarly:
\[ l@G \overset{D}{=} \text{Hypergeometric}(\#\text{large}, |G|, n - k) \]
Recall: Input $n$ i.i.d. $\text{Uniform}(0, 1)$ variables.

- Assume fixed pivot values $P$ and $Q$
- Classes of elements are i.i.d. with

\[
\begin{align*}
\Pr[x \text{ small}] &= P \\
\Pr[x \text{ medium}] &= Q - P \\
\Pr[x \text{ large}] &= 1 - Q
\end{align*}
\]

- Nicer: In terms of Spacings $D = (D_1, D_2, D_3)$

Partition sizes $I$ = cumulative result of independent repetitions

\[
I \sim \text{Multinomial}(n-1, D)
\]
Recall: Input n i.i.d. Uniform(0,1) variables.

- Assume fixed pivot values P and Q
- Classes of elements are i.i.d. with

\[ \Pr[x \text{ small}] = P =: D_1 \]
\[ \Pr[x \text{ medium}] = Q - P =: D_2 \]
\[ \Pr[x \text{ large}] = 1 - Q =: D_3 \]

Nicer: In terms of Spacings \( D = (D_1, D_2, D_3) \)

\[ \Rightarrow \] Partition sizes \( I = \) cumulative result of independent repetitions

\[ I = \text{Multinomial}(n; D_1, D_2, D_3) \]
Recall: Input \( n \) \emph{i.i.d.} \( \text{Uniform}(0, 1) \) variables.

- Assume \textbf{fixed pivot values} \( P \) and \( Q \)
- \textbf{Classes} of elements are \emph{i.i.d.} with
  \[
  \Pr[x \text{ small}] = P =: D_1, \quad \Pr[x \text{ medium}] = Q - P =: D_2, \quad \Pr[x \text{ large}] = 1 - Q =: D_3
  \]

\textbf{Nicer:} In terms of \textbf{Spacings} \( D = (D_1, D_2, D_3) \)

\( \sim \) Partition sizes \( I = \) cumulative result of independent repetitions

\[
I \overset{D}{=} \text{Multinomial}(n - k, D)
\]
Distribution of Partition Sizes I

**Recall:** Input \( n \) i.i.d. Uniform(0, 1) variables.

- Assume **fixed pivot values** \( P \) and \( Q \)
- **Classes** of elements are i.i.d. with
  
  \[
  \begin{align*}
  \Pr[x \text{ small}] &= P \quad \equiv \quad D_1 \\
  \Pr[x \text{ medium}] &= Q - P \quad \equiv \quad D_2 \\
  \Pr[x \text{ large}] &= 1 - Q \quad \equiv \quad D_3
  \end{align*}
  \]

- **Nicer:** In terms of **Spacings** \( D = (D_1, D_2, D_3) \)

  \[
  0 \quad \overset{D_1}{\longrightarrow} \quad P \quad \overset{D_2}{\longrightarrow} \quad Q \quad \overset{D_3}{\longrightarrow} \quad 1
  \]

- Partition sizes \( I = \) cumulative result of independent repetitions

\[
I \overset{D}{=} \text{Multinomial}(n - k, D)
\]
Recall: Input \( n \) i.i.d. Uniform(0, 1) variables.

- Assume fixed pivot values \( P \) and \( Q \)

\( \Rightarrow \) Classes of elements are i.i.d. with

\[
\begin{align*}
\Pr[x & \text{ small}] = P =: D_1 \\
\Pr[x & \text{ medium}] = Q - P =: D_2 \\
\Pr[x & \text{ large}] = 1 - Q =: D_3
\end{align*}
\]

\( \Rightarrow \) Nicer: In terms of Spacings \( D = (D_1, D_2, D_3) \)

\( \Rightarrow \) Partition sizes \( I = \) cumulative result of independent repetitions

\[
I \overset{D}{=} \text{Multinomial}(n - k, D)
\]
Distribution of Partition Sizes I

**Recall:** Input \( n \ i. i. d. \ Uniform(0, 1) \) variables.

- Assume **fixed pivot values** \( P \) and \( Q \)
- **Classes** of elements are \( i. i. d. \) with
  
  \[
  \begin{align*}
  \Pr[x \text{ small}] &= P =: D_1 \\
  \Pr[x \text{ medium}] &= Q - P =: D_2 \\
  \Pr[x \text{ large}] &= 1 - Q =: D_3
  \end{align*}
  \]

- **Nicer:** In terms of **Spacings** \( \mathbf{D} = (D_1, D_2, D_3) \)

  0 \( \overset{I_1}{\bullet} \) \( P \overset{I_2}{\bullet} \) \( Q \overset{I_3}{\bullet} \) 1

  \[\begin{array}{c}
  D_1 \\
  \hline
  D_2 \\
  \hline
  D_3
  \end{array}\]

- **Partition sizes** \( \mathbf{I} = \) cumulative result of independent repetitions

  \[
  \mathbf{I} \overset{D}{=} \text{Multinomial}(n - k, \mathbf{D})
  \]
Distribution of Partition Sizes I

**Recall:** Input \( n \) i.i.d. Uniform(0, 1) variables.

- Assume **fixed pivot values** \( P \) and \( Q \)

\[ \sim \text{Classes of elements are i.i.d. with} \]

\[
\begin{align*}
\Pr[x \text{ small}] &= P =: D_1 \\
\Pr[x \text{ medium}] &= Q - P =: D_2 \\
\Pr[x \text{ large}] &= 1 - Q =: D_3
\end{align*}
\]

- **Nicer:** In terms of **Spacings** \( D = (D_1, D_2, D_3) \)

\[
D_1 D_2 D_3
\]

\[ \sim \text{Partition sizes } I = \text{cumulative result of independent repetitions} \]

\[ I \overset{D}{=} \text{Multinomial}(n - k, D) \]
Recall: Input \( n \) i.i.d. \( \text{Uniform}(0, 1) \) variables.

- Assume fixed pivot values \( P \) and \( Q \)

\[ \rightsquigarrow \text{Classes of elements are i.i.d. with} \]

\[
\begin{align*}
\Pr[x \text{ small}] &= P &=: D_1 \\
\Pr[x \text{ medium}] &= Q - P &=: D_2 \\
\Pr[x \text{ large}] &= 1 - Q &=: D_3
\end{align*}
\]

- Nicer: In terms of Spacings \( D = (D_1, D_2, D_3) \)

\[ \rightsquigarrow \text{Partition sizes } I = \text{cumulative result of independent repetitions} \]

\[ I \overset{D}{=} \text{Multinomial}(n - k, D) \]
P and Q are **order statistics** of \( k \) i.i.d. \( \text{Uniform}(0, 1) \) variables.

\[
\begin{array}{c}
0 \\
\hline \\
1 \\
\end{array}
\]

\( t = (1, 2, 3) \)

\[
\leadsto \text{Density for spacings: } f_D(d_1, d_2, d_3) \propto d_1^{t_1} \cdot d_2^{t_2} \cdot d_3^{t_3}
\]

\[\leadsto D \overset{D}{\sim} \text{Dirichlet}(t + 1) = \text{Dirichlet}(t_1 + 1, t_2 + 1, t_3 + 1)\]
P and Q are order statistics of k i.i.d. Uniform(0, 1) variables

\[ 0 \quad u_i \quad 1 \]

\[ t = (1, 2, 3) \]

\[ \sim \text{Density for spacings: } f_D(d_1, d_2, d_3) \propto d_1^{t_1} \cdot d_2^{t_2} \cdot d_3^{t_3} \]

\[ \sim D \overset{D}{=} \text{Dirichlet}(t + 1) = \text{Dirichlet}(t_1 + 1, t_2 + 1, t_3 + 1) \]
P and Q are order statistics of \( k \) i. i. d. Uniform(0, 1) variables.

\[ t = (1, 2, 3) \]

Density for spacings: \( f_D(d_1, d_2, d_3) \propto d_1^{t_1} \cdot d_2^{t_2} \cdot d_3^{t_3} \)

\[ \sim D \xrightarrow{D} \text{Dirichlet}(t + 1) = \text{Dirichlet}(t_1 + 1, t_2 + 1, t_3 + 1) \]
P and Q are order statistics of k i.i.d. Uniform(0, 1) variables

\[ U_1, U_2, U_3 \]

\[ t = (1, 2, 3) \]

\[ f_D(d_1, d_2, d_3) \propto d_1^{t_1} \cdot d_2^{t_2} \cdot d_3^{t_3} \]

\[ D \sim \text{Dirichlet}(t + 1) = \text{Dirichlet}(t_1 + 1, t_2 + 1, t_3 + 1) \]
P and Q are order statistics of \( k \) i.i.d. \( \text{Uniform}(0, 1) \) variables.

Density for spacings: 
\[
f_D(d_1, d_2, d_3) \propto d_1^{t_1} \cdot d_2^{t_2} \cdot d_3^{t_3}
\]

\( D \sim \text{Dirichlet}(t + 1) = \text{Dirichlet}(t_1 + 1, t_2 + 1, t_3 + 1) \)
P and Q are order statistics of k i.i.d. Uniform(0, 1) variables

\[ t = (1, 2, 3) \]

\[ \sim \] Density for spacings: 
\[ f_D(d_1, d_2, d_3) \propto d_1^{t_1} \cdot d_2^{t_2} \cdot d_3^{t_3} \]

\[ \sim \] \( D \overset{D}{\sim} \text{Dirichlet}(t + 1) = \text{Dirichlet}(t_1 + 1, t_2 + 1, t_3 + 1) \]
P and Q are \textit{order statistics} of \(k\) \textit{i.i.d. Uniform}(0, 1) variables

\[t = (1, 2, 3)\]

\(\Rightarrow\) Density for spacings: 
\[f_D(d_1, d_2, d_3) \propto d_1^{t_1} \cdot d_2^{t_2} \cdot d_3^{t_3}\]

\(\Rightarrow D \overset{D}{=} \text{Dirichlet}(t + 1) = \text{Dirichlet}(t_1 + 1, t_2 + 1, t_3 + 1)\)
P and Q are **order statistics** of $k$ i.i.d. *Uniform*(0, 1) variables

$$t = (1, 2, 3)$$

Density for spacings: $f_D(d_1, d_2, d_3) \propto d_1^{t_1} \cdot d_2^{t_2} \cdot d_3^{t_3}$

$D \overset{D}{=} \text{Dirichlet}(t + 1) = \text{Dirichlet}(t_1 + 1, t_2 + 1, t_3 + 1)$
P and Q are order statistics of \( k \) i.i.d. Uniform\((0, 1)\) variables

\[
\begin{align*}
0 & \quad u_1 \quad u_3 \quad u_5 \quad u_6 \quad u_8 \quad u_2 \quad u_7 \quad u_4 \quad 1 \\
\end{align*}
\]

\( t = (1, 2, 3) \)

\[
\leadsto \text{Density for spacings: } f_D(d_1, d_2, d_3) \propto d_1^{t_1} \cdot d_2^{t_2} \cdot d_3^{t_3}
\]

\[
\leadsto D \overset{D}{=} \text{Dirichlet}(t + 1) = \text{Dirichlet}(t_1 + 1, t_2 + 1, t_3 + 1)
\]
P and Q are order statistics of k i.i.d. Uniform(0, 1) variables

\[
t = (1, 2, 3)
\]

Density for spacings: 

\[
f_D(d_1, d_2, d_3) \propto d_1^{t_1} \cdot d_2^{t_2} \cdot d_3^{t_3}
\]

\[
D \overset{D}{\sim} Dirichlet(t + 1) = Dirichlet(t_1 + 1, t_2 + 1, t_3 + 1)
\]
P and Q are order statistics of \( k \) i.i.d. Uniform\((0, 1)\) variables

\[
\begin{align*}
\text{Density for spacings: } f_D(d_1, d_2, d_3) & \propto d_{t_1}^{t_1} \cdot d_{t_2}^{t_2} \cdot d_{t_3}^{t_3} \\
\text{D ~ Dirichlet}(t+1) & = \text{Dirichlet}(t_1 + 1, t_2 + 1, t_3 + 1)
\end{align*}
\]
P and Q are order statistics of \( k \) i.i.d. Uniform(0, 1) variables.

\[ t = (1, 2, 3) \]

Density for spacings: \( f_D(d_1, d_2, d_3) \propto d_1^{t_1} \cdot d_2^{t_2} \cdot d_3^{t_3} \)

\[ D \overset{D}{\sim} Dirichlet(t + 1) = Dirichlet(t_1 + 1, t_2 + 1, t_3 + 1) \]
P and Q are order statistics of \( k \) i.i.d. Uniform(0, 1) variables.

\[
\begin{align*}
D_1 & \quad D_2 \quad D_3 \\
\mathbf{t} & = (1, 2, 3)
\end{align*}
\]

\[\leadsto \text{Density for spacings: } f_D(d_1, d_2, d_3) \propto d_1^{t_1} \cdot d_2^{t_2} \cdot d_3^{t_3}\]

\[\leadsto \mathbf{D} \overset{\mathcal{D}}{=} \text{Dirichlet}(\mathbf{t} + 1) = \text{Dirichlet}(t_1 + 1, t_2 + 1, t_3 + 1)\]
Expected #Comparisons per Partitioning

Summary of distributional analysis

- $s \vDash K \equiv \text{HypG}(I_1, I_1 + I_2, n) + O(1)$
- $I \equiv \text{Mult}(n, D) + O(1)$
- $D \equiv \text{Dir}(t + 1)$

$\leadsto$ expected value by successive unconditioning:
Summary of distributional analysis

- \( s \oplus K \overset{D}{=} \text{HypG}(I_1, I_1 + I_2, n) + O(1) \)
- \( I \overset{D}{=} \text{Mult}(n, D) + O(1) \)
- \( D \overset{D}{=} \text{Dir}(t + 1) \)

\(~\rightarrow\) expected value by *successive unconditioning*:

\[ E[s \oplus K] \]
Summary of distributional analysis

- $s \circledast K \overset{D}{=} \text{HypG}(I_1, I_1 + I_2, n) + O(1)$
- $I \overset{D}{=} \text{Mult}(n, D) + O(1)$
- $D \overset{D}{=} \text{Dir}(t + 1)$

$\sim$ expected value by \textit{successive unconditioning}: 

$$\mathbb{E}[s \circledast K \mid I]$$
Expected #Comparisons per Partitioning

Summary of distributional analysis

\( s \, @ \, K \overset{D}{=} HypG( I_1, I_1 + I_2, n ) + O(1) \)

\( I \overset{D}{=} Mult(n, D) + O(1) \)

\( D \overset{D}{=} Dir(t + 1) \)

\( \leadsto \) expected value by **successive unconditioning:**

\[ \mathbb{E}_I [ \mathbb{E}_I [ s \, @ \, K \mid I ] \mid D ] \]
Expected #Comparisons per Partitioning

Summary of distributional analysis

- $s \circ \mathcal{K} \overset{d}{=} \text{HypG}(I_1, I_1 + I_2, n) + O(1)$
- $I \overset{d}{=} \text{Mult}(n, D) + O(1)$
- $D \overset{d}{=} \text{Dir}(t + 1)$

\( \sim \) expected value by successive unconditioning:

$$\mathbb{E}_D \left[ \mathbb{E}_I \left[ \mathbb{E}[s \circ \mathcal{K} \mid I] \mid D \right] \right]$$
Summary of distributional analysis

- $s \oplus_K \overset{D}{=} HypG(I_1, I_1 + I_2, n) + O(1)$
- $I \overset{D}{=} Mult(n, D) + O(1)$
- $D \overset{D}{=} Dir(t + 1)$

Expected value by successive unconditioning:

$$E_D \left[ E_I \left[ E[s @ K | I] | D \right] \right] = \frac{(t_1 + 1)(t_1 + t_2 + 3)}{(k + 2)(k + 1)} \cdot n + O(1)$$
Expected #Comparisons per Partitioning

Summary of distributional analysis

- \( s @ K \overset{\mathcal{D}}{=} HypG( I_1, I_1 + I_2, n ) + O(1) \)
- \( I \overset{\mathcal{D}}{=} Mult(n, D) + O(1) \)
- \( D \overset{\mathcal{D}}{=} Dir(t + 1) \)

\( \rightsquigarrow \) expected value by **successive unconditioning**:

\[
\mathbb{E}_D \left[ \mathbb{E}_I \left[ \mathbb{E} [ s @ K | I ] | D \right] \right] = \frac{(t_1 + 1)(t_1 + t_2 + 3)}{(k + 2)(k + 1)} \cdot n + O(1)
\]

\[
\mathbb{E}_D \left[ \mathbb{E}_I \left[ \mathbb{E} [ l @ G | I ] | D \right] \right] = \frac{(t_3 + 2)(t_3 + 1)}{(k + 2)(k + 1)} \cdot n + O(1)
\]
Expected #Comparisons

Final Result:

$$\mathbb{E}[C_n] \sim \frac{\gamma}{H} \cdot n \ln n$$
Final Result:

\[ \mathbb{E}[C_n] \sim \frac{\gamma}{\sum_{l=1}^{3} \frac{t_l + 1}{k + 1} (H_{k+1} - H_{t_l+1})} \cdot n \ln n \]
Expected #Comparisons

Final Result:

\[ \mathbb{E}[C_n] \sim 2 - \frac{s@K}{n} - \frac{l@G}{n} \cdot n \ln n \]

\[ \sum_{l=1}^{3} \frac{t_l + 1}{k + 1} (H_{k+1} - H_{t_l+1}) \]
Expected #Comparisons

Final Result: \[ E[C_n] \sim 2 - \frac{(t_1 + 1)(t_1 + t_2 + 3)}{(k + 2)(k + 1)} - \frac{(t_3 + 2)(t_3 + 1)}{(k + 2)(k + 1)} \cdot \sum_{l=1}^{3} \frac{t_l + 1}{k + 1} (H_{k+1} - H_{t_l+1}) \cdot n \ln n \]
Expected #Comparisons

Final Result:

\[
\mathbb{E}[C_n] \sim 2 - \frac{(t_1 + 1)(t_1 + t_2 + 3)}{(k + 2)(k + 1)} - \frac{(t_3 + 2)(t_3 + 1)}{(k + 2)(k + 1)} \cdot n \ln n
\]

\[
\sum_{l=1}^{3} \frac{t_l + 1}{k + 1} (\mathcal{H}_{k+1} - \mathcal{H}_{t_l+1})
\]

... a little hard to interpret.

Let’s have some pictures.
**Expected #Comparisons**

**Final Result:**

\[
\mathbb{E}[C_n] \sim 2 - \frac{(t_1 + 1)(t_1 + t_2 + 3)}{(k + 2)(k + 1)} - \frac{(t_3 + 2)(t_3 + 1)}{(k + 2)(k + 1)} \cdot n \ln n \\
\sum_{l=1}^{3} \frac{t_l + 1}{k + 1} (\mathcal{H}_{k+1} - \mathcal{H}_{t_l+1})
\]
Expected #Comparisons

Final Result:

$$\mathbb{E}[C_n] \sim 2 - \frac{(t_1 + 1)(t_1 + t_2 + 3)}{(k + 2)(k + 1)} - \frac{(t_3 + 2)(t_3 + 1)}{(k + 2)(k + 1)} \sum_{l=1}^{3} \frac{t_1 + 1}{k + 1} (H_{k+1} - H_{t_1+1}) \cdot n \ln n$$
Expected #Comparisons

Final Result:

\[
E[C_n] \sim 2 - \frac{(t_1 + 1)(t_1 + t_2 + 3)}{(k + 2)(k + 1)} - \frac{(t_3 + 2)(t_3 + 1)}{(k + 2)(k + 1)} \\
\sum_{l=1}^{3} \frac{t_l + 1}{k + 1} (\mathcal{H}_{k+1} - \mathcal{H}_{t_l+1}) \cdot n \ln n
\]
Expected #Comparisons

Final Result:

\[
\mathbb{E}[C_n] \sim \frac{2 - \frac{(t_1 + 1)(t_1 + t_2 + 3)}{(k + 2)(k + 1)} - \frac{(t_3 + 2)(t_3 + 1)}{(k + 2)(k + 1)}}{\sum_{l=1}^{3} \frac{t_l + 1}{k + 1}(H_{k+1} - H_{t_l+1})} \cdot n \ln n
\]
Expected #Comparisons

Final Result:

\[ \mathbb{E}[C_n] \sim 2 - \frac{(t_1 + 1)(t_1 + t_2 + 3)}{(k + 2)(k + 1)} - \frac{(t_3 + 2)(t_3 + 1)}{(k + 2)(k + 1)} \sum_{l=1}^{3} \frac{t_l + 1}{k + 1} (H_{k+1} - H_{t_l+1}) \cdot n \ln n \]
Final Result:
\[
E[C_n] \sim 2 - \frac{(t_1 + 1)(t_1 + t_2 + 3)}{(k + 2)(k + 1)} - \frac{(t_3 + 2)(t_3 + 1)}{(k + 2)(k + 1)} \sum_{l=1}^{3} \frac{t_l + 1}{k + 1}(H_{k+1} - H_{t_l+1}) \cdot n \ln n
\]
Similarly: full (basic block) frequency analysis

⇝ #Bytecode instructions, MMIX costs etc.

Recall: Optimum for comparisons

\[ \tau^* \approx (0.4288, 0.2688, 0.3024) \]

⇝ Contrary skew of pivots needed!

⇝ Optimal skew of pivots heavily depends on cost measure!
Similarly: full (basic block) frequency analysis

⇝ #Bytecode instructions, MMIX costs etc.

Optimum for Bytecodes
\[ \tau^*_{BC} \approx (0.2068, 0.3485, 0.4447) \]

Recall:
Optimum for comparisons
\[ \tau^* \approx (0.4288, 0.2688, 0.3024) \]

⇝ Contrary skew of pivots needed!

⇝ Optimal skew of pivots heavily depends on cost measure!
**Similarly:** full (basic block) frequency analysis

\[ \Rightarrow \text{#Bytecode instructions, MMIX costs etc.} \]

**Optimum for Bytecodes**

\[ \tau^*_\text{BC} \approx (0.2068, 0.3485, 0.4447) \]

**Recall:**

**Optimum for comparisons**

\[ \tau^* \approx (0.4288, 0.2688, 0.3024) \]

\[ \Rightarrow \text{Contrary skew of pivots needed!} \]

\[ \Rightarrow \text{Optimal skew of pivots heavily depends on cost measure!} \]
**Beyond Comparisons**

*Similarly*: full *(basic block) frequency analysis

\[ \cdots \rightarrow \text{#Bytecode instructions, MMIX costs etc.} \]

![#Bytecodes graph](image)

- Optimum for *Bytecodes*
  \[ \tau_{BC}^* \approx (0.2068, 0.3485, 0.4447) \]

- Recall:
  Optimum for comparisons
  \[ \tau^* \approx (0.4288, 0.2688, 0.3024) \]

\[ \cdots \rightarrow \text{Contrary skew of pivots needed!} \]

\[ \cdots \rightarrow \text{Optimal skew of pivots heavily depends on cost measure!} \]
**Similarly:** full (basic block) frequency analysis

\[ \text{#Bytecode instructions, MMIX costs etc.} \]

- Optimum for **Bytecodes**
  \[ \tau^*_BC \approx (0.2068, 0.3485, 0.4447) \]

- Recall:
  Optimum for **comparisons**
  \[ \tau^* \approx (0.4288, 0.2688, 0.3024) \]

\[ \text{Contrary skew of pivots needed!} \]

\[ \text{Optimal skew of pivots heavily depends on cost measure!} \]
Similarly: full (basic block) frequency analysis

⇝ #Bytecode instructions, MMIX costs etc.

Optimum for Bytecodes
\[ \tau_{BC}^* \approx (0.2068, 0.3485, 0.4447) \]

Recall:
Optimum for comparisons
\[ \tau^* \approx (0.4288, 0.2688, 0.3024) \]

⇝ Contrary skew of pivots needed!

⇝ Optimal skew of pivots heavily depends on cost measure!
**Similarly:** full (basic block) frequency analysis

\[ \Rightarrow \] #Bytecode instructions, MMIX costs etc.

- **Optimum for Bytecodes**
  \[ \tau_{BC}^* \approx (0.2068, 0.3485, 0.4447) \]

- **Recall:**
  Optimum for comparisons
  \[ \tau^* \approx (0.4288, 0.2688, 0.3024) \]

\[ \Rightarrow \text{Contrary skew of pivots needed!} \]

\[ \Rightarrow \text{Optimal skew of pivots heavily depends on cost measure!} \]
Similarly: full (basic block) frequency analysis

⇝ #Bytecode instructions, MMIX costs etc.

- Optimum for Bytecodes
  \( \tau^*_BC \approx (0.2068, 0.3485, 0.4447) \)

- Recall:
  Optimum for comparisons
  \( \tau^* \approx (0.4288, 0.2688, 0.3024) \)

⇝ Contrary skew of pivots needed!

⇝ Optimal skew of pivots heavily depends on cost measure!
Similarly: full (basic block) frequency analysis

\[ \Rightarrow \] #Bytecode instructions, MMIX costs etc.

- Optimum for **Bytecodes**
  \[ \tau^*_B \approx (0.2068, 0.3485, 0.4447) \]

- Recall:
  Optimum for **comparisons**
  \[ \tau^* \approx (0.4288, 0.2688, 0.3024) \]

\[ \Rightarrow \] Contrary skew of pivots needed!

\[ \Rightarrow \] Optimal skew of pivots heavily depends on **cost measure**!
We’ve seen:

- **Asymmetric** partitioning calls for **skewed** pivots
- **Optimal** skew **sensitive** to employed cost measure

→ Realistic cost measure and detailed analysis needed!

Open Questions:

- What exactly makes Yaroslavskiy’s algorithm fast in **practice**?
  - Memory Hierarchy Effects?
- Is Yaroslavskiy’s algorithm **special**?
  - Are there even faster methods?
- Will they be asymmetric?
We’ve seen:

- **Asymmetric** partitioning calls for **skewed** pivots
- **Optimal** skew **sensitive** to employed cost measure

⇒ Realistic cost measure and detailed analysis needed!

Open Questions:

- What **exactly** makes Yaroslavskiy’s algorithm fast in **practice**?
  - Memory Hierarchy Effects?
- Is Yaroslavskiy’s algorithm **special**?
  - Are there even faster methods?
- Will they be asymmetric?