Forty years of Quicksort and Quickselect: a personal view

Conrado Martínez

Univ. Politècnica de Catalunya, Spain

Introduction

- Quicksort and quickselect were invented in the early sixties by C.A.R. Hoare (Hoare, 1961; Hoare, 1962)

Introduction

- Quicksort and quickselect were invented in the early sixties by C.A.R. Hoare (Hoare, 1961; Hoare, 1962)
- They are simple, elegant, beatiful and practical solutions to two basic problems of Computer Science: sorting and selection

Introduction

- Quicksort and quickselect were invented in the early sixties by C.A.R. Hoare (Hoare, 1961; Hoare, 1962)
- They are simple, elegant, beatiful and practical solutions to two basic problems of Computer Science: sorting and selection
- They are primary examples of the divide-and-conquer principle

Quicksort

```
void quicksort(vector<Elem>& A, int i, int j) {
    if (i < j) {
        int p = get_pivot(A, i, j);
        swap(A[p], A[l]);
        int k;
        partition(A, i, j, k);
        //   A[i..k-1] ≤  A[k] ≤  A[k+1..j]
        quicksort(A, i, k - 1);
        quicksort(A, k + 1, j);
} }
```

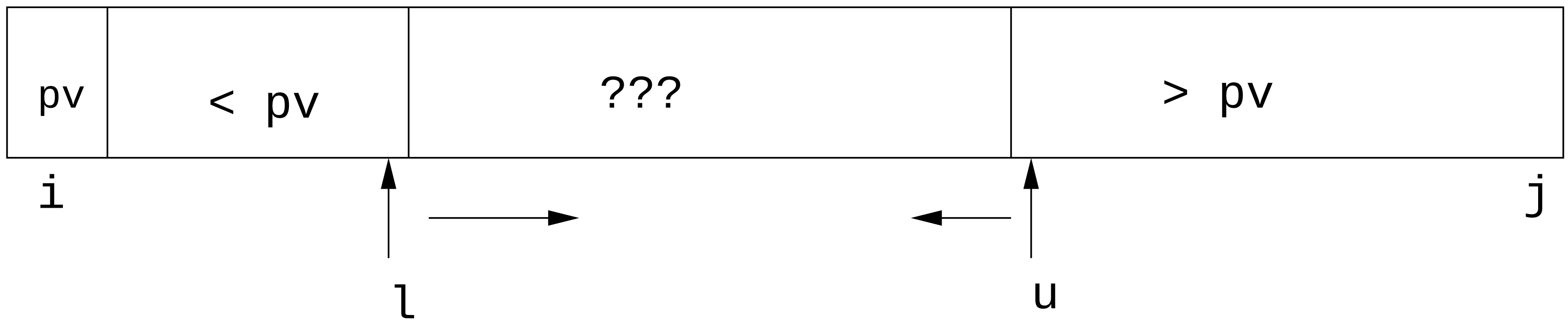
Quickselect

```
Elem quickselect(vector<Elem>& A,
                 int i, int j, int m) {
   if (i >= j) return A[i];
   int p = get_pivot(A, i, j, m);
   swap(A[p], A[l]);
   int k;
   partition(A, i, j, k);
   if (m < k)        quickselect(A, i, k - 1, m);
   else if (m > k) quickselect(A, k + 1, j, m);
   else              return A[k];
}
```

Partition

```
void partition(vector<Elem>& A,
               int i, int j, int& k) {
       int l = i; int u = j + 1; Elem pv = A[i];
       for ( ; ; ) {
          do ++l; while(A[l] < pv);
          do --u; while(A[u] > pv);
          if (l >= u) break;
          swap(A[l], A[u]);
       };
       swap(A[i], A[u]); k = u;
}
```

Partition



The Recurrences for Average Costs

- Probability that the selected pivot is the k-th of n elements: $\pi_{n,k}$

The Recurrences for Average Costs

- Probability that the selected pivot is the k-th of n elements: $\pi_{n,k}$
- Average number of comparisons Q_n to sort n elements:

$$Q_n = n - 1 + \sum_{k=1}^{n} \pi_{n,k} \cdot (Q_{k-1} + Q_{n-k})$$

The Recurrences for Average Costs

- Probability that the selected pivot is the k-th of n elements: $\pi_{n,k}$
- Average number of comparisons $C_{n,m}$ to select the m-th out of n:

$$C_{n,m} = n - 1 + \sum_{k=m+1}^{n} \pi_{n,k} \cdot C_{k-1,m} + \sum_{k=1}^{m-1} \pi_{n,k} \cdot C_{n-k,m-k}$$

Quicksort: The Average Cost

- For the standard variant, $\pi_{n,k} = 1/n$

Quicksort: The Average Cost

- For the standard variant, $\pi_{n,k} = 1/n$
- Average number of comparisons Q_n to sort n elements (Hoare, 1962):

$$\begin{aligned} Q_n &= 2(n+1)H_n - 4n \\ &= 2n\ln n + (2\gamma - 4)n + 2\ln n + \mathcal{O}(1) \end{aligned}$$

where $H_n = \sum_{1\leq k\leq n} 1/k = \ln n + \gamma + \mathcal{O}(1/n)$ is the n-th harmonic number and $\gamma = 0.577\ldots$ is Euler's gamma constant.

Quickselect: The Average Cost

- Average number of comparisons $C_{n,m}$ to select the m-th out of n elements (Knuth, 1971):

$$C_{n,m} = 2\Big(n + 3 + (n+1)H_n - (n+3-m)H_{n+1-m} - (m+2)H_m\Big)$$

Quickselect: The Average Cost

- This is $\Theta(n)$ for any m, $1 \le m \le n$. In particular,

$$m_0(\alpha) = \lim_{n\to\infty, m/n\to\alpha} \frac{C_{n,m}}{n} = 2 + 2 \cdot \mathcal{H}(\alpha),$$
$$\mathcal{H}(x) = -(x \ln x + (1-x)\ln(1-x)).$$

  with $0 \le \alpha \le 1$. The maximum is at $\alpha = 1/2$, where $m_0(1/2) = 2 + 2\ln 2 = 3.386\ldots$; the mean value is $\overline{m}_0 = 3$.

Improving Quicksort and Quickselect

- Apply general techniques: recursion removal, loop unwrapping, ...

Improving Quicksort and Quickselect

- Apply general techniques: recursion removal, loop unwrapping, . . .
- Reorder recursive calls to quicksort

Improving Quicksort and Quickselect

- Apply general techniques: recursion removal, loop unwrapping, . . .
- Reorder recursive calls to quicksort
- Switch to a simpler algorithm for small subfiles

Improving Quicksort and Quickselect

- Apply general techniques: recursion removal, loop unwrapping, . . .
- Reorder recursive calls to quicksort
- Switch to a simpler algorithm for small subfiles
- Use samples to select better pivots

Improving Quicksort and Quickselect

- Apply general techniques: recursion removal, loop unwrapping, . . .
- Reorder recursive calls to quicksort
- Switch to a simpler algorithm for small subfiles
- Use samples to select better pivots

Small Subfiles

- It is well known (Sedgewick, 1975) that, for quicksort, it is convenient to stop recursion for subarrays of size $\leq n_0$ and use insertion sort instead

Small Subfiles

- It is well known (Sedgewick, 1975) that, for quicksort, it is convenient to stop recursion for subarrays of size $\leq n_0$ and use insertion sort instead
- The optimal choice for n_0 is around 20 to 25 elements

Small Subfiles

- It is well known (Sedgewick, 1975) that, for quicksort, it is convenient to stop recursion for subarrays of size $\leq n_0$ and use insertion sort instead
- The optimal choice for n_0 is around 20 to 25 elements
- Alternatively, one might do nothing with small subfiles and perform a single pass of insertion sort over the whole file

Small Subfiles

- Cutting off recursion also yields benefits for quickselect

Small Subfiles

- Cutting off recursion also yields benefits for quickselect
- In (Martínez, Panario, Viola, 2002) we investigate different choices to process small subfiles and how they affect the average total cost: selection, insertion sort, optimized selection

Small Subfiles

- We have now

$$C_{n,m} = \begin{cases} t_{n,m} + \displaystyle\sum_{k=m+1}^{n} \pi_{n,k} \cdot C_{k-1,m} & \\ \qquad + \displaystyle\sum_{k=1}^{m-1} \pi_{n,k} \cdot C_{n-k,m-k}, & \text{if } n > n_0 \\ b_{n,m} & \text{if } n \le n_0 \end{cases}$$

Small Subfiles

- Let $C(z,u) = \sum_{n\geq 0} \sum_{1\leq m\leq n} C_{n,m} z^n u^m$

Small Subfiles

- Let $C(z,u) = \sum_{n\geq 0} \sum_{1\leq m\leq n} C_{n,m} z^n u^m$
- It can be shown that

$$C(z,u) = C_{n_0}(z,u) + \frac{\int_0^z (1-t)(1-ut)\frac{\partial T(t,u)}{\partial t}\, dt}{(1-z)(1-uz)}$$

where $T(z,u) = \sum_{n\geq 0} \sum_{1\leq m\leq n} t_{n,m} z^n u^m$ and $C_{n_0}(z,u)$ is the only part depending on the $b_{n,m}$'s and n_0.

Small Subfiles

- In order to determine the optimal choice for n_0 we need only to compute $[z^n u^m] C_{n_0}(z, u)$

Small Subfiles

- In order to determine the optimal choice for n_0 we need only to compute $[z^n u^m] C_{n_0}(z,u)$
- We assume $t_{n,m} = \alpha n + \beta + \gamma/(n-1)$ and

$$\begin{aligned} b_{n,m} = K_1 n^2 + K_2 n + K_3 m^2 + K_4 m + K_5 mn + K_6 \\ + K_7 g^2 + K_8 g + K_9 gn, \end{aligned}$$

where $g \equiv \min\{m, n-m+1\}$, to study the best choice for n_0, as a function of α, β, γ and the K_i's.

Small Subfiles

- Using insertion sort with $n_0 \leq 10$ reduces the average cost; the optimal choice for n_0 is 5

Small Subfiles

- Using insertion sort with $n_0 \leq 10$ reduces the average cost; the optimal choice for n_0 is 5
- Selection (we locate the minimum, then the second minimum, etc.) reduces the average cost if $n_0 \leq 11$; the optimum n_0 is 6

Small Subfiles

- Using insertion sort with $n_0 \leq 10$ reduces the average cost; the optimal choice for n_0 is 5
- Selection (we locate the minimum, then the second minimum, etc.) reduces the average cost if $n_0 \leq 11$; the optimum n_0 is 6
- Optimized selection (looks for the m-th from the minimum or the maximum, whatever is closer) yields improved average performance if $n_0 \leq 22$; the optimum n_0 is 11

Median-of-three

- In quicksort with median-of-three, the pivot of each recursive stage is selected as the median of a sample of three elements (Singleton, 1969)

Median-of-three

- In quicksort with median-of-three, the pivot of each recursive stage is selected as the median of a sample of three elements (Singleton, 1969)
- This reduces the probability of uneven partitions which lead to quadratic worst-case

Median-of-three

- We have in this case

$$\pi_{n,k} = \frac{(k-1)(n-k)}{\binom{n}{3}}$$

Median-of-three

- We have in this case

$$\pi_{n,k} = \frac{(k-1)(n-k)}{\binom{n}{3}}$$

- The average number of comparisons Q_n is (Sedgewick, 1975)

$$Q_n = \frac{12}{7} n \log n + \mathcal{O}(n),$$

roughly a 14.3% less than standard quicksort

Median-of-three

- To study quickselect with median-of-three, in (Kirschenhofer, Martínez, Prodinger, 1997), we use bivariate generating functions

$$C(z,u) = \sum_{n\geq 0} \sum_{1\leq m\leq n} C_{n,m} z^n u^m$$

Median-of-three

- To study quickselect with median-of-three, in (Kirschenhofer, Martínez, Prodinger, 1997), we use bivariate generating functions

$$C(z,u) = \sum_{n\geq 0} \sum_{1\leq m\leq n} C_{n,m} z^n u^m$$

- The recurrences translate into second-order differential equations of hypergeometric type

$$x(1-x)y'' + (c-(1+a+b)x)y' - aby = 0$$

Median-of-three

- We compute explicit solutions for comparisons and for passes; from there, one has to extract (painfully ;-)) the coefficients

Median-of-three

- We compute explicit solutions for comparisons and for passes; from there, one has to extract (painfully ;-)) the coefficients
- For instance, for the average number of passes we get

$$P_{n,m} = \frac{24}{35} H_n + \frac{18}{35} H_m + \frac{18}{35} H_{n+1-m} + \mathcal{O}(1)$$

Median-of-three

- We compute explicit solutions for comparisons and for passes; from there, one has to extract (painfully ;-)) the coefficients
- And for the average number of comparisons

$$C_{n,m} = 2n + \frac{72}{35}H_n - \frac{156}{35}H_m - \frac{156}{35}H_{n+1-m} + 3m - \frac{(m-1)(m-2)}{n} + \mathcal{O}(1)$$

Median-of-three

- An important particular case is $m = \lceil n/2 \rceil$ (the median) were the average number of comparisons is

$$\frac{11}{4}n + o(n)$$

Compare to $(2 + 2\ln 2)n + o(n)$ for standard quickselect.

Median-of-three

- In general,

$$m_1(\alpha) = \lim_{n\to\infty, m/n\to\alpha} \frac{C_{n,m}}{n} = 2 + 3 \cdot \alpha \cdot (1-\alpha)$$

with $0 \le \alpha \le 1$. The mean value is $\overline{m}_1 = 5/2$; compare to $3n + o(n)$ comparisons for standard quickselect on random ranks.

Optimal Sampling

- In (Martínez, Roura, 2001) we study what happens if we use samples of size $s = 2t + 1$ to pick the pivots, but $t = t(n)$

Optimal Sampling

- In (Martínez, Roura, 2001) we study what happens if we use samples of size $s = 2t + 1$ to pick the pivots, but $t = t(n)$
- The comparisons needed to pick the pivots have to be taken into account:

$$Q_n = n - 1 + \Theta(s) + \sum_{k=1}^{n} \pi_{n,k} \cdot (Q_{k-1} + Q_{n-k})$$

Optimal Sampling

- Traditional techniques to solve recurrences cannot be used here

Optimal Sampling

- Traditional techniques to solve recurrences cannot be used here
- We make extensive use of the continuous master theorem (Roura, 1997)

Optimal Sampling

- Traditional techniques to solve recurrences cannot be used here
- We make extensive use of the continuous master theorem (Roura, 1997)
- We also study the cost of quickselect when the rank of the sought element is random

Optimal Sampling

- Traditional techniques to solve recurrences cannot be used here
- We make extensive use of the continuous master theorem (Roura, 1997)
- We also study the cost of quickselect when the rank of the sought element is random
- Total cost: # of comparisons $+\ \xi \cdot$ # of exchanges

Optimal Sampling

Theorem 1. *If we use samples of size* s*, with* $s = o(n)$ *and* $s = \omega(1)$ *then the average total cost* Q_n *of quicksort is*

$$Q_n = (1 + \xi/4) n \log_2 n + o(n \log n)$$

and the average total cost C_n *of quickselect to find an element of given random rank is*

$$C_n = 2(1 + \xi/4)n + o(n)$$

Optimal Sampling

Theorem 2. *Let* $s^* = 2t^* + 1$ *denote the optimal sample size that minimizes the average total cost of quickselect; assume the average total cost of the algorithm to pick the medians from the samples is* $\beta s + o(s)$*. Then*

$$t^* = \frac{1}{2\sqrt{\beta}} \cdot \sqrt{n} + o\left(\sqrt{n}\right)$$

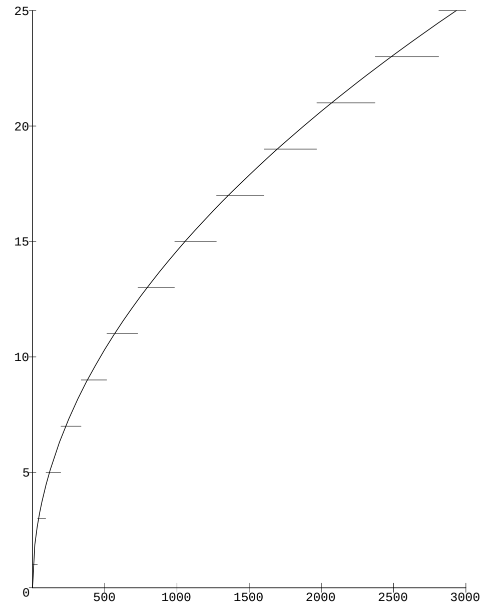
Optimal Sampling

Theorem 3. *Let* $s^* = 2t^* + 1$ *denote the optimal sample size that minimizes the average number of comparisons made by quicksort. Then*

$$t^* = \sqrt{\frac{1}{\beta}\left(\frac{4 - \xi(2\ln 2 - 1)}{8\ln 2}\right)} \cdot \sqrt{n} + o\left(\sqrt{n}\right)$$

if $\xi < \tau = 4/(2\ln 2 - 1) \approx 10.3548$

Optimal Sampling



Optimal sample size (Theorem 3) vs. exact values

Optimal Sampling

- If exchanges are expensive ($\xi \geq \tau$) we have to use fixed-size samples and pick the median (not optimal) or pick the $(\psi \cdot s)$-th element of a sample of size $\Theta(\sqrt{n})$

Optimal Sampling

- If exchanges are expensive ($\xi \geq \tau$) we have to use fixed-size samples and pick the median (not optimal) or pick the $(\psi \cdot s)$-th element of a sample of size $\Theta(\sqrt{n})$
- If the position of the pivot is close to either end of the array, then few exchanges are necessary on that stage, but a poor partition leads to more recursive steps. This trade-off is relevant if exchanges are very expensive

Optimal Sampling

- The variance of quickselect when $s = s(n) \to \infty$ is

$$V_n = \Theta\left(\max\left\{\frac{n^2}{s}, n \cdot s\right\}\right)$$

Optimal Sampling

- The variance of quickselect when $s = s(n) \to \infty$ is

$$V_n = \Theta\left(\max\left\{\frac{n^2}{s}, n \cdot s\right\}\right)$$

- The best choice is $s = \Theta(\sqrt{n})$; then $V_n = \Theta(n^{3/2})$ and there is concentration in probability

Optimal Sampling

- The variance of quickselect when $s = s(n) \to \infty$ is

$$V_n = \Theta\left(\max\left\{\frac{n^2}{s}, n \cdot s\right\}\right)$$

- The best choice is $s = \Theta(\sqrt{n})$; then $V_n = \Theta(n^{3/2})$ and there is concentration in probability
- We conjecture this type of result holds for quicksort too

Adaptive Sampling

- In (Martínez, Panario, Viola, 2004) we study choosing pivots with relative rank in the sample close to $\alpha = m/n$

Adaptive Sampling

- In (Martínez, Panario, Viola, 2004) we study choosing pivots with relative rank in the sample close to $\alpha = m/n$
- In general: $r(\alpha)$ = rank of the pivot within the sample, when selecting the m-th out of n elements and $\alpha = m/n$

Adaptive Sampling

- In (Martínez, Panario, Viola, 2004) we study choosing pivots with relative rank in the sample close to $\alpha = m/n$
- In general: $r(\alpha)$ = rank of the pivot within the sample, when selecting the m-th out of n elements and $\alpha = m/n$
- Divide $[0, 1]$ into ℓ intervals with endpoints $0 = a_0 < a_1 < a_2 < \cdots < a_\ell = 1$ and let r_k denote the value of $r(\alpha)$ for α in the k-th interval

Adaptive Sampling

- For median-of- $(2t+1)$: $\ell = 1$ and $r_1 = t+1$

Adaptive Sampling

- For median-of- $(2t+1)$: $\ell = 1$ and $r_1 = t+1$
- For proportion-from- s: $\ell = s$, $a_k = k/s$ and $r_k = k$

Adaptive Sampling

- For median-of- $(2t+1)$: $\ell = 1$ and $r_1 = t+1$
- For proportion-from- s: $\ell = s$, $a_k = k/s$ and $r_k = k$
- "Proportion-from"-like strategies: $\ell = s$ and $r_k = k$, but the endpoints of the intervals $a_k \neq k/s$

Adaptive Sampling

- For median-of- $(2t+1)$: $\ell = 1$ and $r_1 = t+1$
- For proportion-from- s: $\ell = s$, $a_k = k/s$ and $r_k = k$
- "Proportion-from"-like strategies: $\ell = s$ and $r_k = k$, but the endpoints of the intervals $a_k \neq k/s$
- A sampling strategy is symmetric if

$$r(\alpha) = s + 1 - r(1-\alpha)$$

Adaptive Sampling

Theorem 4. *Let* $f(\alpha) = \lim_{n\to\infty, m/n\to\alpha} \frac{C_{n,m}}{n}$. *Then*

$$f(\alpha) = 1 + \frac{s!}{(r(\alpha)-1)!(s-r(\alpha))!} \times$$
$$\left[\int_\alpha^1 f\left(\frac{\alpha}{x}\right) x^{r(\alpha)} (1-x)^{s-r(\alpha)}\, dx\right.$$
$$\left. + \int_0^\alpha f\left(\frac{\alpha - x}{1-x}\right) x^{r(\alpha)-1} (1-x)^{s+1-r(\alpha)}\, dx\right].$$

Adaptive Sampling: Proportion-from-2

- Here $f(\alpha)$ is composed of two “pieces” f_1 and f_2 for the intervals $[0, 1/2]$ and $(1/2, 1]$

Adaptive Sampling: Proportion-from-2

- Here $f(\alpha)$ is composed of two "pieces" f_1 and f_2 for the intervals $[0, 1/2]$ and $(1/2, 1]$
- Because of symmetry we need only to solve for f_1

$$f_1(x) = a\left((x-1)\ln(1-x) + \frac{x^3}{6} + \frac{x^2}{2} - x\right) - b(1 + \mathcal{H}(x)) + cx + d.$$

Adaptive Sampling: Proportion-from-2

- The maximum is at $\alpha = 1/2$. There $f(1/2) = 3.112\ldots$

Adaptive Sampling: Proportion-from-2

- The maximum is at $\alpha = 1/2$. There $f(1/2) = 3.112\ldots$
- Proportion-from-2 beats standard quickselect: $f(\alpha) \leq m_0(\alpha)$

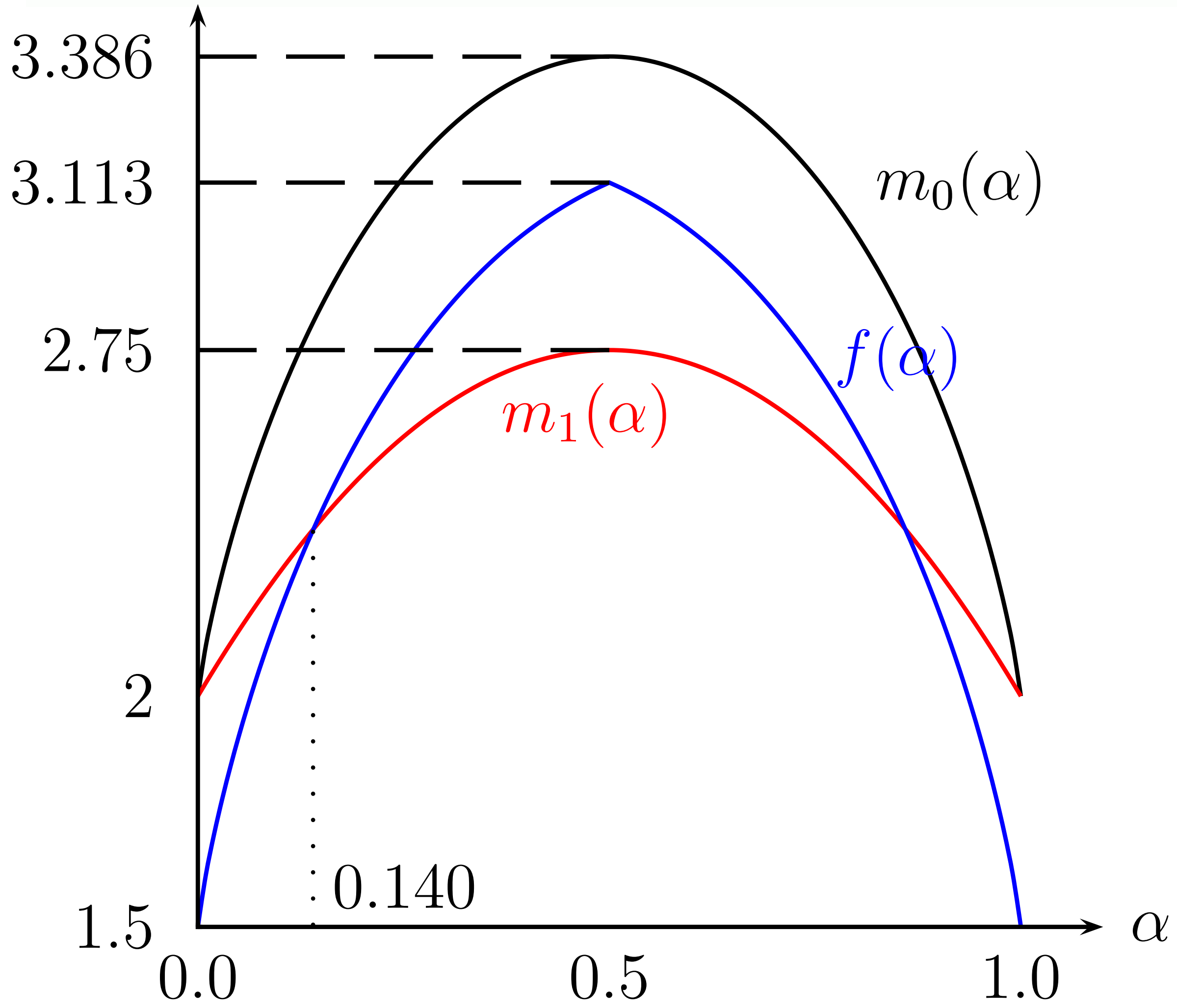
Adaptive Sampling: Proportion-from-2

- The maximum is at $\alpha = 1/2$. There $f(1/2) = 3.112\ldots$
- Proportion-from-2 beats standard quickselect: $f(\alpha) \leq m_0(\alpha)$
- Proportion-from-2 beats median-of-three in some regions: $f(\alpha) \leq m_1(\alpha)$ if $\alpha \leq 0.140\ldots$ or $\alpha \geq 0.860\ldots$

Adaptive Sampling: Proportion-from-2

- The maximum is at $\alpha = 1/2$. There $f(1/2) = 3.112\ldots$
- Proportion-from-2 beats standard quickselect: $f(\alpha) \leq m_0(\alpha)$
- Proportion-from-2 beats median-of-three in some regions: $f(\alpha) \leq m_1(\alpha)$ if $\alpha \leq 0.140\ldots$ or $\alpha \geq 0.860\ldots$
- The grand-average: $C_n = 2.598 \cdot n + o(n)$

Adaptive Sampling: Proportion-from-2



Adaptive Sampling: Proportion-from-3

For proportion-from-3,

$$
\begin{aligned}
f_1(x) &= -C_0(1+\mathcal{H}(x)) + C_1 + C_2 x \\
&\quad + C_3 K_1(x) + C_4 K_2(x), \\
f_2(x) &= -C_5(1+\mathcal{H}(x)) + C_6 x(1-x) + C_7,
\end{aligned}
$$

with

$$
\begin{aligned}
K_1(x) &= \cos(\sqrt{2}\ln x)\cdot\sum_{n\geq 0} A_n x^{n+4} + \sin(\sqrt{2}\ln x)\cdot\sum_{n\geq 0} B_n x^{n+4}, \\
K_2(x) &= \sin(\sqrt{2}\ln x)\cdot\sum_{n\geq 0} A_n x^{n+4} - \cos(\sqrt{2}\ln x)\cdot\sum_{n\geq 0} B_n x^{n+4}.
\end{aligned}
$$

Adaptive Sampling: Proportion-from-3

- Two maxima at $\alpha = 1/3$ and $\alpha = 2/3$. There $f(1/3) = f(2/3) = 2.883\ldots$

Adaptive Sampling: Proportion-from-3

- Two maxima at $\alpha = 1/3$ and $\alpha = 2/3$. There $f(1/3) = f(2/3) = 2.883\ldots$
- The median is not the most difficult rank: $f(1/2) = 2.723\ldots$

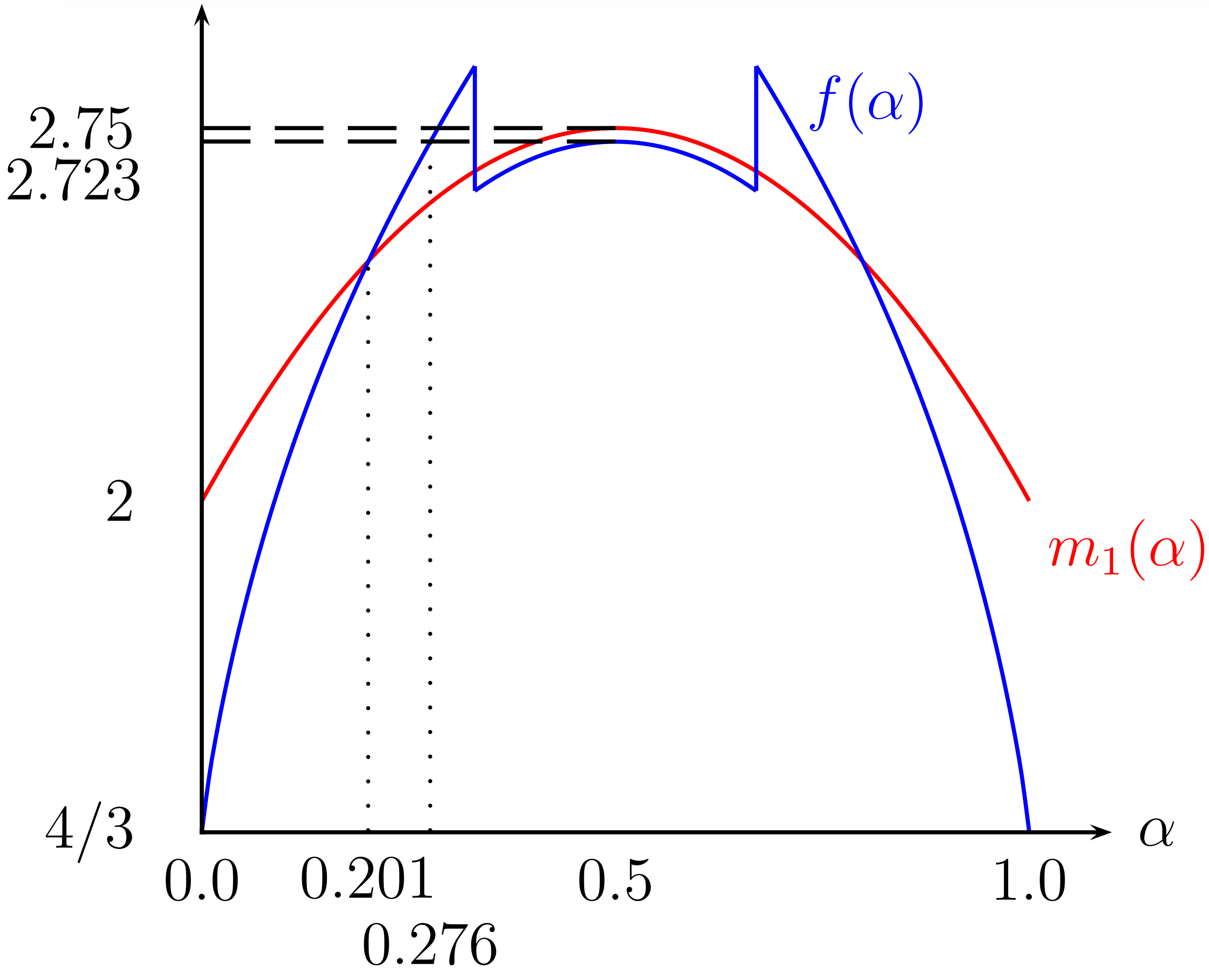
Adaptive Sampling: Proportion-from-3

- Two maxima at $\alpha = 1/3$ and $\alpha = 2/3$. There $f(1/3) = f(2/3) = 2.883\ldots$
- The median is not the most difficult rank: $f(1/2) = 2.723\ldots$
- Proportion-from-3 beats median-of-three in some regions: $f(\alpha) \leq m_1(\alpha)$ if $\alpha \leq 0.201\ldots$, $\alpha \geq 0.798\ldots$ or $1/3 < \alpha < 2/3$

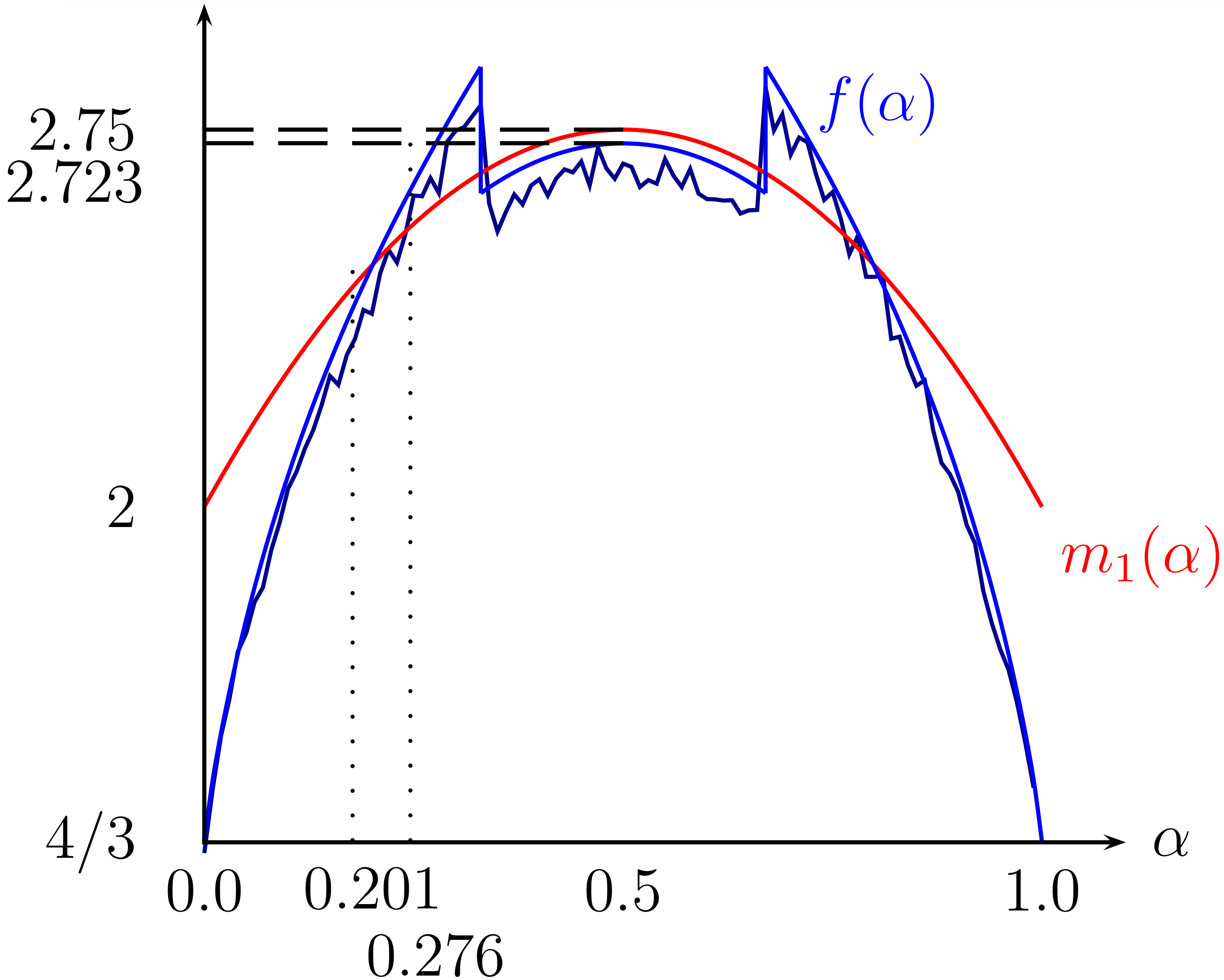
Adaptive Sampling: Proportion-from-3

- Two maxima at $\alpha = 1/3$ and $\alpha = 2/3$. There $f(1/3) = f(2/3) = 2.883\ldots$
- The median is not the most difficult rank: $f(1/2) = 2.723\ldots$
- Proportion-from-3 beats median-of-three in some regions: $f(\alpha) \leq m_1(\alpha)$ if $\alpha \leq 0.201\ldots$, $\alpha \geq 0.798\ldots$ or $1/3 < \alpha < 2/3$
- The grand-average: $C_n = 2.421 \cdot n + o(n)$

Adaptive Sampling: Batfind



Adaptive Sampling: Batfind



Adaptive Sampling: ν-find

- Like proportion-from-3, but $a_1 = \nu$ and $a_2 = 1 - \nu$

Adaptive Sampling: ν-find

- Like proportion-from-3, but $a_1 = \nu$ and $a_2 = 1 - \nu$
- Same differential equation, same f_i's, with $C_i = C_i(\nu)$

Adaptive Sampling: ν-find

- Like proportion-from-3, but $a_1 = \nu$ and $a_2 = 1 - \nu$
- Same differential equation, same f_i's, with $C_i = C_i(\nu)$
- If $\nu \to 0$ then $f_\nu \to m_1$ (median-of-three)

Adaptive Sampling: ν-find

- Like proportion-from-3, but $a_1 = \nu$ and $a_2 = 1 - \nu$
- Same differential equation, same f_i's, with $C_i = C_i(\nu)$
- If $\nu \to 0$ then $f_\nu \to m_1$ (median-of-three)
- If $\nu \to 1/2$ then f_ν is similar to proportion-from-2, but it is not the same

Adaptive Sampling: ν-find

Theorem 5. *There exists a value* ν^**, namely,* $\nu^* = 0.182\ldots$*, such that for any* ν*,* $0 < \nu < 1/2$*, and any* α*,*

$$f_{\nu^*}(\alpha) \leq f_\nu(\alpha).$$

Furthermore, ν^* *is the unique value of* ν *such that* f_ν *is continuous,i.e.,*

$$f_{\nu^*,1}(\nu^*) = f_{\nu^*,2}(\nu^*).$$

Adaptive Sampling: ν-find

- Obviously, the value ν^* minimizes the maximum

$$f_{\nu^*}(1/2) = 2.659\ldots$$

and the mean

$$\overline{f}_{\nu^*} = 2.342\ldots$$

Adaptive Sampling: ν-find

- Obviously, the value ν^* minimizes the maximum

$$f_{\nu^*}(1/2) = 2.659\ldots$$

and the mean

$$\overline{f}_{\nu^*} = 2.342\ldots$$

- If $\nu > \tilde{\nu} = 0.268\ldots$ then f_ν has two absolute maxima at $\alpha = \nu$ and $\alpha = 1 - \nu$; otherwise there is one absolute maximum at $\alpha = 1/2$

Adaptive Sampling: ν-find

- If $\nu \le \overline{\nu}' = 0.404\ldots$ then ν-find beats median-of-3 on average ranks: $\overline{f}_\nu \le 5/2$

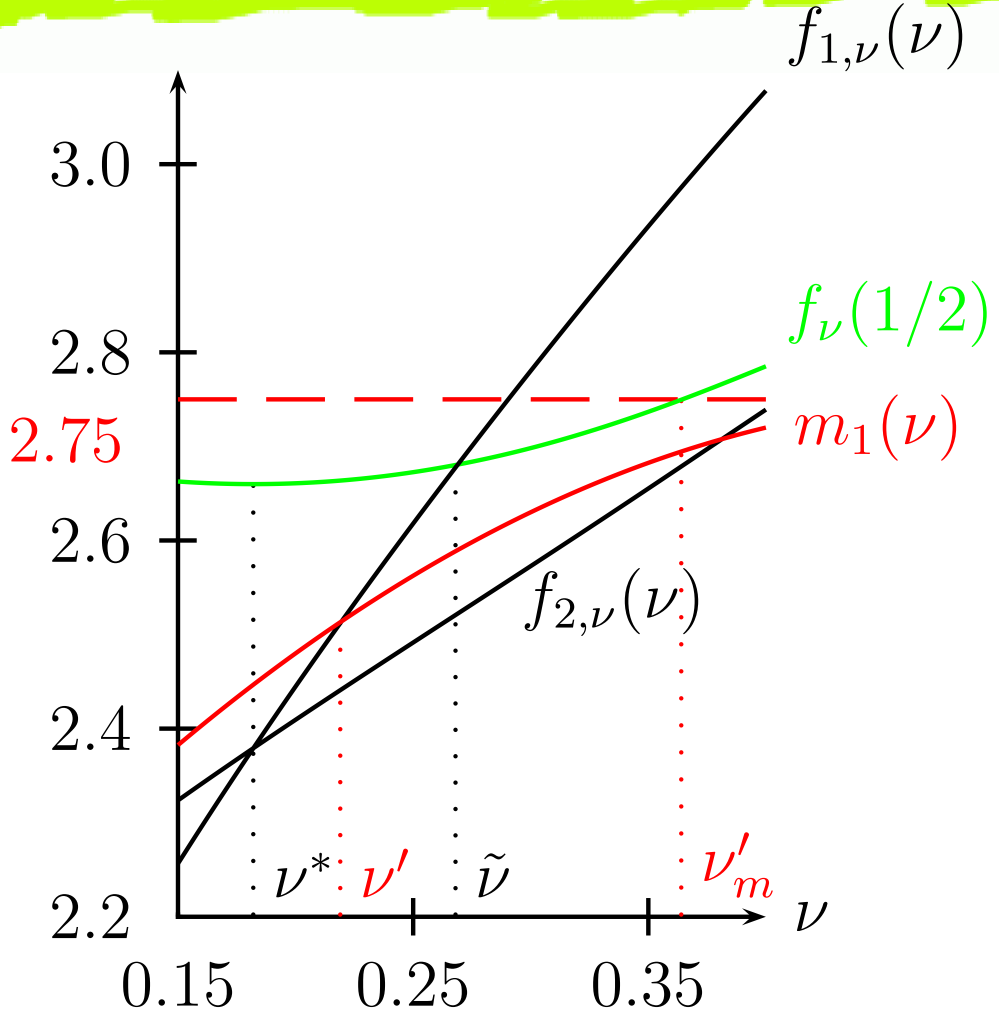
Adaptive Sampling: ν-find

- If $\nu \le \overline{\nu}' = 0.404\ldots$ then ν-find beats median-of-3 on average ranks: $\overline{f}_\nu \le 5/2$
- If $\nu \le \nu'_m = 0.364\ldots$ then ν-find beats median-of-3 to find the median: $f_\nu(1/2) \le 11/4$

Adaptive Sampling: ν-find

- If $\nu \le \overline{\nu}' = 0.404\ldots$ then ν-find beats median-of-3 on average ranks: $\overline{f}_\nu \le 5/2$
- If $\nu \le \nu'_m = 0.364\ldots$ then ν-find beats median-of-3 to find the median: $f_\nu(1/2) \le 11/4$
- If $\nu \le \nu' = 0.219\ldots$ then ν-find beats median-of-3 for all ranks: $f_\nu(\alpha) \le m_1(\alpha)$

Adaptive Sampling: ν-find



Theorem 6. *Let* $f^{(s)}(\alpha) = \lim_{n\to\infty, m/n\to\alpha} \frac{C_{n,m}}{n}$ *when using samples of size* s. *Then for any adaptive sampling strategy such that* $\lim_{s\to\infty} r(\alpha)/s = \alpha$

$$f^{(\infty)}(\alpha) = \lim_{s\to\infty} f^{(s)}(\alpha) = 1 + \min(\alpha, 1-\alpha).$$

Partial Sort

- Partial sort: Given an array A of n elements, return the m smallest elements in A in ascending order

Partial Sort

- Partial sort: Given an array A of n elements, return the m smallest elements in A in ascending order
- Heapsort-based partial sort: Build a heap, extract m times the minimum; the cost is $\Theta(n + m \log n)$

Partial Sort

- Partial sort: Given an array A of n elements, return the m smallest elements in A in ascending order
- Heapsort-based partial sort: Build a heap, extract m times the minimum; the cost is $\Theta(n + m \log n)$
- “Quickselsort”: find the m-th with quickselect, then quicksort $m-1$ elements to its left; the cost is $\Theta(n + m \log m)$

Partial Quicksort

```
void partial_quicksort(vector<Elem>& A,
                       int i, int j, int m)
{
    if (i < j) {
        int p = get_pivot(A, i, j);
        swap(A[p], A[l]);
        int k;
        partition(A, i, j, k);
        partial_quicksort(A, i, k - 1, m);
        if (k < m-1)
            partial_quicksort(A, k + 1, j, m);
} }
```

Partial Quicksort

- Average number of comparisons $P_{n,m}$ to sort the m smallest elements:

$$P_{n,m} = n - 1 + \sum_{k=m+1}^{n} \pi_{n,k} \cdot P_{k-1,m} + \sum_{k=1}^{m} \pi_{n,k} \cdot \left(P_{k-1,k-1} + P_{n-k,m-k}\right)$$

Partial Quicksort

- Average number of comparisons $P_{n,m}$ to sort the m smallest elements:

$$P_{n,m} = n - 1 + \sum_{k=m+1}^{n} \pi_{n,k} \cdot P_{k-1,m} + \sum_{k=1}^{m} \pi_{n,k} \cdot (P_{k-1,k-1} + P_{n-k,m-k})$$

- But $P_{n,n} = Q_n = 2(n+1)H_n - 4n$!

Partial Quicksort

- The recurrence for $P_{n,m}$ is the same as for quickselect but the toll function is

$$n - 1 + \sum_{0 \le k < m} \pi_{n,k} Q_k$$

Partial Quicksort

- The recurrence for $P_{n,m}$ is the same as for quickselect but the toll function is

$$n - 1 + \sum_{0 \le k < m} \pi_{n,k} Q_k$$

- For $\pi_{n,k} = 1/n$, the solution is

$$P_{n,m} = 2n + 2(n+1)H_n \\ - 2(n+3-m)H_{n+1-m} - 6m + 6$$

Partial Quicksort

- Partial quicksort makes

$$2m - 4H_m + 2$$

comparisons less than “quickselsort”

Partial Quicksort

- Partial quicksort makes

$$2m - 4H_m + 2$$

comparisons less than “quickselsort”

- It makes $m/3 - 5H_m/6 + 1/2$ exchanges less than “quickselsort”

Partial Quicksort

- Partial quicksort makes

$$2m - 4H_m + 2$$

  comparisons less than “quickselsort”

- It makes $m/3 - 5H_m/6 + 1/2$ exchanges less than “quickselsort”

- “Quickselsort” forgets the position of the pivots used for the selection of the m-th to the left of m; partial quicksort leaves these at their correct positions and does not compare them against other elements afterwards