Analysis of the Expected Number of Bit Comparisons Required by Quickselect

James Allen Fill   Také Nakama

Department of Applied Mathematics and Statistics
The Johns Hopkins University

April 17, 2008

Key comparisons and bit comparisons

Two measures to quantify the performance of searching or sorting algorithms:

- Number of key comparisons
  - Algorithms compare keys pairwise irrespective of their representation.
  - Performance is analyzed in terms of the number of key comparisons required by the algorithms.
- Number of bit comparisons
  - Keys are represented as bit strings.
  - Algorithms operate on individual bits to compare keys.
  - Performance may be analyzed in terms of the number of bit comparisons required by the algorithms.

Key comparisons and bit comparisons

Two measures to quantify the performance of searching or sorting algorithms:

- Number of key comparisons
  - Algorithms compare keys pairwise irrespective of their representation.
  - Performance is analyzed in terms of the number of key comparisons required by the algorithms.
- Number of bit comparisons
  - Keys are represented as bit strings.
  - Algorithms operate on individual bits to compare keys.
  - Performance may be analyzed in terms of the number of bit comparisons required by the algorithms.

Key comparisons and bit comparisons

Two measures to quantify the performance of searching or sorting algorithms:

- Number of key comparisons
  - Algorithms compare keys pairwise irrespective of their representation.
  - Performance is analyzed in terms of the number of key comparisons required by the algorithms.
- Number of bit comparisons
  - Keys are represented as bit strings.
  - Algorithms operate on individual bits to compare keys.
  - Performance may be analyzed in terms of the number of bit comparisons required by the algorithms.

Key comparisons and bit comparisons

Two measures to quantify the performance of searching or sorting algorithms:

- Number of key comparisons
  - Algorithms compare keys pairwise irrespective of their representation.
  - Performance is analyzed in terms of the number of key comparisons required by the algorithms.
- Number of bit comparisons
  - Keys are represented as bit strings.
  - Algorithms operate on individual bits to compare keys.
  - Performance may be analyzed in terms of the number of bit comparisons required by the algorithms.

Key comparisons and bit comparisons

Two measures to quantify the performance of searching or sorting algorithms:

- Number of key comparisons
  - Algorithms compare keys pairwise irrespective of their representation.
  - Performance is analyzed in terms of the number of key comparisons required by the algorithms.
- Number of bit comparisons
  - Keys are represented as bit strings.
  - Algorithms operate on individual bits to compare keys.
  - Performance may be analyzed in terms of the number of bit comparisons required by the algorithms.

Key comparisons and bit comparisons

Two measures to quantify the performance of searching or sorting algorithms:

- Number of key comparisons
  - Algorithms compare keys pairwise irrespective of their representation.
  - Performance is analyzed in terms of the number of key comparisons required by the algorithms.
- Number of bit comparisons
  - Keys are represented as bit strings.
  - Algorithms operate on individual bits to compare keys.
  - Performance may be analyzed in terms of the number of bit comparisons required by the algorithms.

Example: Quicksort

Task: Sort keys in $\mathbb{S} := \{k_1, k_2, \ldots, k_n\}$ $(= \{k_{(1)}, k_{(2)}, \ldots, k_{(n)}\})$.

(i) Randomly select a pivot key (denote it by k_i).

(ii) Compare each of the other keys with k_i ($k_i = k_{(j)}$) and create three subsets of $\mathbb{S}$:
$\mathbb{S}_1 := \{k_{(1)}, \ldots, k_{(j-1)}\}$,
$\mathbb{S}_2 := \{k_{(j)}\}$,
$\mathbb{S}_3 := \{k_{(j+1)}, \ldots, k_{(n)}\}$.

(iii) Apply the algorithm to $\mathbb{S}_m$ if $|\mathbb{S}_m| > 1$ ($m = 1, 3$).
The algorithm accomplishes the task in a recursive and random fashion.

Example: Quicksort

Task: Sort keys in $\mathbb{S} := \{k_1, k_2, \ldots, k_n\}$ $(= \{k_{(1)}, k_{(2)}, \ldots, k_{(n)}\})$.

(i) Randomly select a pivot key (denote it by k_i).

(ii) Compare each of the other keys with k_i $(k_i = k_{(j)})$ and create three subsets of $\mathbb{S}$:
$\mathbb{S}_1 := \{k_{(1)}, \ldots, k_{(j-1)}\}$,
$\mathbb{S}_2 := \{k_{(j)}\}$,
$\mathbb{S}_3 := \{k_{(j+1)}, \ldots, k_{(n)}\}$.

(iii) Apply the algorithm to $\mathbb{S}_m$ if $|\mathbb{S}_m| > 1$ $(m = 1, 3)$.
The algorithm accomplishes the task in a recursive and random fashion.

Example: Quicksort

Task: Sort keys in $\mathbb{S} := \{k_1, k_2, \ldots, k_n\}$ $(= \{k_{(1)}, k_{(2)}, \ldots, k_{(n)}\})$.

(i) Randomly select a pivot key (denote it by k_i).

(ii) Compare each of the other keys with k_i $(k_i = k_{(j)})$ and create three subsets of $\mathbb{S}$:
$\mathbb{S}_1 := \{k_{(1)}, \ldots, k_{(j-1)}\}$,
$\mathbb{S}_2 := \{k_{(j)}\}$,
$\mathbb{S}_3 := \{k_{(j+1)}, \ldots, k_{(n)}\}$.

(iii) Apply the algorithm to $\mathbb{S}_m$ if $|\mathbb{S}_m| > 1$ $(m = 1, 3)$.
The algorithm accomplishes the task in a recursive and random fashion.

Example: Quicksort

Task: Sort keys in $\mathbb{S} := \{k_1, k_2, \ldots, k_n\}$ $(= \{k_{(1)}, k_{(2)}, \ldots, k_{(n)}\})$.

(i) Randomly select a pivot key (denote it by k_i).

(ii) Compare each of the other keys with k_i $(k_i = k_{(j)})$ and create three subsets of $\mathbb{S}$:
$\mathbb{S}_1 := \{k_{(1)}, \ldots, k_{(j-1)}\}$,
$\mathbb{S}_2 := \{k_{(j)}\}$,
$\mathbb{S}_3 := \{k_{(j+1)}, \ldots, k_{(n)}\}$.

(iii) Apply the algorithm to $\mathbb{S}_m$ if $|\mathbb{S}_m| > 1$ $(m = 1, 3)$.
The algorithm accomplishes the task in a recursive and random fashion.

Example: Quicksort

Task: Sort keys in $\mathbb{S} := \{k_1, k_2, \ldots, k_n\}$ $(= \{k_{(1)}, k_{(2)}, \ldots, k_{(n)}\})$.

(i) Randomly select a pivot key (denote it by k_i).

(ii) Compare each of the other keys with k_i $(k_i = k_{(j)})$ and create three subsets of $\mathbb{S}$:
$\mathbb{S}_1 := \{k_{(1)}, \ldots, k_{(j-1)}\}$,
$\mathbb{S}_2 := \{k_{(j)}\}$,
$\mathbb{S}_3 := \{k_{(j+1)}, \ldots, k_{(n)}\}$.

(iii) Apply the algorithm to $\mathbb{S}_m$ if $|\mathbb{S}_m| > 1$ $(m = 1, 3)$.
The algorithm accomplishes the task in a recursive and random fashion.

Key and bit comparisons required by Quicksort

$k_1 = .0010010\ldots$, $k_2 = .0110100\ldots$,
$k_3 = .0011011\ldots$, $k_4 = .0001101\ldots$.

(i) Suppose k_3 is selected as a pivot.

(ii) Quicksort requires:

- 4 bit comparisons to determine $k_1 < k_3$.
- 2 bit comparisons to determine $k_2 > k_3$.
- 3 bit comparisons to determine $k_4 < k_3$.

$\mathbb{S}_1 = \{k_1, k_4\}$, $\mathbb{S}_2 = \{k_3\}$, $\mathbb{S}_3 = \{k_2\}$.

(iii) Apply Quicksort to $\mathbb{S}_1$. (3 more bit comparisons to determine $k_4 < k_1$.)

In total, Quicksort requires 4 key comparisons and 12 bit comparisons to complete the task.

Key and bit comparisons required by Quicksort

$k_1 = .0010010\ldots,\ k_2 = .0110100\ldots,$
$k_3 = .0011011\ldots,\ k_4 = .0001101\ldots.$

(i) Suppose k_3 is selected as a pivot.

(ii) Quicksort requires:

- 4 bit comparisons to determine $k_1 < k_3$.
- 2 bit comparisons to determine $k_2 > k_3$.
- 3 bit comparisons to determine $k_4 < k_3$.

$\mathbb{S}_1 = \{k_1, k_4\}$, $\mathbb{S}_2 = \{k_3\}$, $\mathbb{S}_3 = \{k_2\}$.

(iii) Apply Quicksort to $\mathbb{S}_1$. (3 more bit comparisons to determine $k_4 < k_1$.)

In total, Quicksort requires 4 key comparisons and 12 bit comparisons to complete the task.

Key and bit comparisons required by Quicksort

$k_1 = .0010010\ldots,\ k_2 = .0110100\ldots,$
$k_3 = .0011011\ldots,\ k_4 = .0001101\ldots.$

(i) Suppose k_3 is selected as a pivot.

(ii) Quicksort requires:

- 4 bit comparisons to determine $k_1 < k_3$.
- 2 bit comparisons to determine $k_2 > k_3$.
- 3 bit comparisons to determine $k_4 < k_3$.

$\mathbb{S}_1 = \{k_1, k_4\}$, $\mathbb{S}_2 = \{k_3\}$, $\mathbb{S}_3 = \{k_2\}$.

(iii) Apply Quicksort to $\mathbb{S}_1$. (3 more bit comparisons to determine $k_4 < k_1$.)

In total, Quicksort requires 4 key comparisons and 12 bit comparisons to complete the task.

Key and bit comparisons required by Quicksort

$k_1 = .0010010\ldots,\ k_2 = .0110100\ldots,$
$k_3 = .0011011\ldots,\ k_4 = .0001101\ldots.$

(i) Suppose k_3 is selected as a pivot.

(ii) Quicksort requires:

- 4 bit comparisons to determine $k_1 < k_3$.
- 2 bit comparisons to determine $k_2 > k_3$.
- 3 bit comparisons to determine $k_4 < k_3$.

$\mathbb{S}_1 = \{k_1, k_4\}$, $\mathbb{S}_2 = \{k_3\}$, $\mathbb{S}_3 = \{k_2\}$.

(iii) Apply Quicksort to $\mathbb{S}_1$. (3 more bit comparisons to determine $k_4 < k_1$.)

In total, Quicksort requires 4 key comparisons and 12 bit comparisons to complete the task.

Key and bit comparisons required by Quicksort

$k_1 = .0010010\ldots,\ k_2 = .0110100\ldots,$
$k_3 = .0011011\ldots,\ k_4 = .0001101\ldots.$

(i) Suppose k_3 is selected as a pivot.

(ii) Quicksort requires:

- 4 bit comparisons to determine $k_1 < k_3$.
- 2 bit comparisons to determine $k_2 > k_3$.
- 3 bit comparisons to determine $k_4 < k_3$.

$\mathbb{S}_1 = \{k_1, k_4\}$, $\mathbb{S}_2 = \{k_3\}$, $\mathbb{S}_3 = \{k_2\}$.

(iii) Apply Quicksort to $\mathbb{S}_1$. (3 more bit comparisons to determine $k_4 < k_1$.)

In total, Quicksort requires 4 key comparisons and 12 bit comparisons to complete the task.

Key and bit comparisons required by Quicksort

$k_1 = .0010010\ldots,\ k_2 = .0110100\ldots,$
$k_3 = .0011011\ldots,\ k_4 = .0001101\ldots.$

(i) Suppose k_3 is selected as a pivot.

(ii) Quicksort requires:

- 4 bit comparisons to determine $k_1 < k_3$.
- 2 bit comparisons to determine $k_2 > k_3$.
- 3 bit comparisons to determine $k_4 < k_3$.

$\mathbb{S}_1 = \{k_1, k_4\},\ \mathbb{S}_2 = \{k_3\},\ \mathbb{S}_3 = \{k_2\}.$

(iii) Apply Quicksort to $\mathbb{S}_1$. (3 more bit comparisons to determine $k_4 < k_1$.)

In total, Quicksort requires 4 key comparisons and 12 bit comparisons to complete the task.

Key and bit comparisons required by Quicksort

$k_1 = .0010010\ldots,\ k_2 = .0110100\ldots,$
$k_3 = .0011011\ldots,\ k_4 = .0001101\ldots.$

(i) Suppose k_3 is selected as a pivot.

(ii) Quicksort requires:

- 4 bit comparisons to determine $k_1 < k_3$.
- 2 bit comparisons to determine $k_2 > k_3$.
- 3 bit comparisons to determine $k_4 < k_3$.

$\mathbb{S}_1 = \{k_1, k_4\}$, $\mathbb{S}_2 = \{k_3\}$, $\mathbb{S}_3 = \{k_2\}$.

(iii) Apply Quicksort to $\mathbb{S}_1$. (3 more bit comparisons to determine $k_4 < k_1$.)

In total, Quicksort requires 4 key comparisons and 12 bit comparisons to complete the task.

Key comparisons and bit comparisons

- It is ideal to analyze sorting or searching algorithms in terms of both key and bit comparisons. (Key-based algorithms can be compared with digital algorithms.)
- Only Quicksort has been analyzed in terms of both key and bit comparisons (Fill and Janson, 2004): Asymptotically, Quicksort requires $2n \ln n$ key comparisons and $n(\ln n)(\lg n)$ bit comparisons to sort n keys.

Key comparisons and bit comparisons

- It is ideal to analyze sorting or searching algorithms in terms of both key and bit comparisons. (Key-based algorithms can be compared with digital algorithms.)
- Only Quicksort has been analyzed in terms of both key and bit comparisons (Fill and Janson, 2004): Asymptotically, Quicksort requires $2n \ln n$ key comparisons and $n(\ln n)(\lg n)$ bit comparisons to sort n keys.

Key comparisons and bit comparisons

- It is ideal to analyze sorting or searching algorithms in terms of both key and bit comparisons. (Key-based algorithms can be compared with digital algorithms.)
- Only Quicksort has been analyzed in terms of both key and bit comparisons (Fill and Janson, 2004): Asymptotically, Quicksort requires $2n \ln n$ key comparisons and $n(\ln n)(\lg n)$ bit comparisons to sort n keys.

Our study

Objective of our study: Analyze the bit complexity of Quickselect (also known as Find)

- Quickselect finds an order statistic.
- Quickselect has been extensively analyzed with regard to the number of key comparisons required by the algorithm, but our study is the first to investigate its bit complexity.

Our study

Objective of our study: Analyze the bit complexity of Quickselect (also known as Find)

- Quickselect finds an order statistic.
- Quickselect has been extensively analyzed with regard to the number of key comparisons required by the algorithm, but our study is the first to investigate its bit complexity.

Our study

Objective of our study: Analyze the bit complexity of Quickselect (also known as Find)

- Quickselect finds an order statistic.
- Quickselect has been extensively analyzed with regard to the number of key comparisons required by the algorithm, but our study is the first to investigate its bit complexity.

Quickselect

Task: Find the m-th smallest key in $\mathbb{S} := \{k_1, k_2, \ldots, k_n\}$
$(= \{k_{(1)}, k_{(2)}, \ldots, k_{(n)}\})$

(i) Randomly select a pivot key (denote it by k_i).

(ii) Compare each of the other keys with k_i ($k_i = k_{(j)}$) and create three subsets of $\mathbb{S}$:
$\mathbb{S}_1 := \{k_{(1)}, \ldots, k_{(j-1)}\}$,
$\mathbb{S}_2 := \{k_{(j)}\}$,
$\mathbb{S}_3 := \{k_{(j+1)}, \ldots, k_{(n)}\}$.

(iii)
- If $j = m$, then the algorithm returns k_i.
- If $j > m$, then the algorithm operates on $\mathbb{S}_1$ and finds the m-th smallest key in the set.
- If $j < m$, then the algorithm operates on $\mathbb{S}_3$ and finds the $(m - j)$-th smallest key in the set.

Quickselect

Task: Find the m-th smallest key in $\mathbb{S} := \{k_1, k_2, \ldots, k_n\}$ $(= \{k_{(1)}, k_{(2)}, \ldots, k_{(n)}\})$

(i) Randomly select a pivot key (denote it by k_i).

(ii) Compare each of the other keys with k_i $(k_i = k_{(j)})$ and create three subsets of $\mathbb{S}$:
$\mathbb{S}_1 := \{k_{(1)}, \ldots, k_{(j-1)}\}$,
$\mathbb{S}_2 := \{k_{(j)}\}$,
$\mathbb{S}_3 := \{k_{(j+1)}, \ldots, k_{(n)}\}$.

(iii)
- If $j = m$, then the algorithm returns k_i.
- If $j > m$, then the algorithm operates on $\mathbb{S}_1$ and finds the m-th smallest key in the set.
- If $j < m$, then the algorithm operates on $\mathbb{S}_3$ and finds the $(m - j)$-th smallest key in the set.

Quickselect

Task: Find the m-th smallest key in $\mathbb{S} := \{k_1, k_2, \ldots, k_n\}$ $(= \{k_{(1)}, k_{(2)}, \ldots, k_{(n)}\})$

(i) Randomly select a pivot key (denote it by k_i).

(ii) Compare each of the other keys with k_i $(k_i = k_{(j)})$ and create three subsets of $\mathbb{S}$:
$\mathbb{S}_1 := \{k_{(1)}, \ldots, k_{(j-1)}\}$,
$\mathbb{S}_2 := \{k_{(j)}\}$,
$\mathbb{S}_3 := \{k_{(j+1)}, \ldots, k_{(n)}\}$.

(iii)
- If $j = m$, then the algorithm returns k_i.
- If $j > m$, then the algorithm operates on $\mathbb{S}_1$ and finds the m-th smallest key in the set.
- If $j < m$, then the algorithm operates on $\mathbb{S}_3$ and finds the $(m - j)$-th smallest key in the set.

Quickselect

Task: Find the m-th smallest key in $\mathbb{S} := \{k_1, k_2, \ldots, k_n\}$ $(= \{k_{(1)}, k_{(2)}, \ldots, k_{(n)}\})$

(i) Randomly select a pivot key (denote it by k_i).

(ii) Compare each of the other keys with k_i $(k_i = k_{(j)})$ and create three subsets of $\mathbb{S}$:
$\mathbb{S}_1 := \{k_{(1)}, \ldots, k_{(j-1)}\}$,
$\mathbb{S}_2 := \{k_{(j)}\}$,
$\mathbb{S}_3 := \{k_{(j+1)}, \ldots, k_{(n)}\}$.

(iii)
- If $j = m$, then the algorithm returns k_i.
- If $j > m$, then the algorithm operates on $\mathbb{S}_1$ and finds the m-th smallest key in the set.
- If $j < m$, then the algorithm operates on $\mathbb{S}_3$ and finds the $(m - j)$-th smallest key in the set.

Quickselect

Task: Find the m-th smallest key in $\mathbb{S} := \{k_1, k_2, \ldots, k_n\}$
$(= \{k_{(1)}, k_{(2)}, \ldots, k_{(n)}\})$

(i) Randomly select a pivot key (denote it by k_i).

(ii) Compare each of the other keys with k_i $(k_i = k_{(j)})$ and create three subsets of $\mathbb{S}$:
$\mathbb{S}_1 := \{k_{(1)}, \ldots, k_{(j-1)}\}$,
$\mathbb{S}_2 := \{k_{(j)}\}$,
$\mathbb{S}_3 := \{k_{(j+1)}, \ldots, k_{(n)}\}$.

(iii)
- If $j = m$, then the algorithm returns k_i.
- If $j > m$, then the algorithm operates on $\mathbb{S}_1$ and finds the m-th smallest key in the set.
- If $j < m$, then the algorithm operates on $\mathbb{S}_3$ and finds the $(m - j)$-th smallest key in the set.

Quickselect

Task: Find the m-th smallest key in $\mathbb{S} := \{k_1, k_2, \ldots, k_n\}$ $(= \{k_{(1)}, k_{(2)}, \ldots, k_{(n)}\})$

(i) Randomly select a pivot key (denote it by k_i).

(ii) Compare each of the other keys with k_i $(k_i = k_{(j)})$ and create three subsets of $\mathbb{S}$:

$$\mathbb{S}_1 := \{k_{(1)}, \ldots, k_{(j-1)}\},$$
$$\mathbb{S}_2 := \{k_{(j)}\},$$
$$\mathbb{S}_3 := \{k_{(j+1)}, \ldots, k_{(n)}\}.$$

(iii)
- If $j = m$, then the algorithm returns k_i.
- If $j > m$, then the algorithm operates on $\mathbb{S}_1$ and finds the m-th smallest key in the set.
- If $j < m$, then the algorithm operates on $\mathbb{S}_3$ and finds the $(m - j)$-th smallest key in the set.

Quickselect

Let $\kappa(m, n)$ denote the expected number of key comparisons required by Quickselect to find the m-th order statistic in a set of n keys.

- $\kappa(m, n) = 2[n+3+(n+1)H_n-(m+2)H_m-(n+3-m)H_{n+1-m}]$ (Knuth, 1972).
- $\kappa(\bar{m}, n) := \frac{1}{n}\sum_{m=1}^{n}\kappa(m, n) = 3n - 8H_n + 13 - \frac{8H_n}{n}$ (Mahmoud, Modarres, and Smythe, 1995).

Many other results exist regarding the number of key comparisons required by Quickselect.

Quickselect

Let $\kappa(m, n)$ denote the expected number of key comparisons required by Quickselect to find the m-th order statistic in a set of n keys.

- $\kappa(m, n) = 2[n+3+(n+1)H_n-(m+2)H_m-(n+3-m)H_{n+1-m}]$ (Knuth, 1972).
- $\kappa(\bar{m}, n) := \frac{1}{n}\sum_{m=1}^{n} \kappa(m, n) = 3n - 8H_n + 13 - \frac{8H_n}{n}$ (Mahmoud, Modarres, and Smythe, 1995).

Many other results exist regarding the number of key comparisons required by Quickselect.

Quickselect

Let $\kappa(m, n)$ denote the expected number of key comparisons required by Quickselect to find the m-th order statistic in a set of n keys.

- $\kappa(m, n) = 2[n+3+(n+1)H_n-(m+2)H_m-(n+3-m)H_{n+1-m}]$ (Knuth, 1972).
- $\kappa(\bar{m}, n) := \frac{1}{n}\sum_{m=1}^{n} \kappa(m, n) = 3n - 8H_n + 13 - \frac{8H_n}{n}$ (Mahmoud, Modarres, and Smythe, 1995).

Many other results exist regarding the number of key comparisons required by Quickselect.

Quickselect

Let $\kappa(m, n)$ denote the expected number of key comparisons required by Quickselect to find the m-th order statistic in a set of n keys.

- $\kappa(m, n) = 2[n+3+(n+1)H_n-(m+2)H_m-(n+3-m)H_{n+1-m}]$ (Knuth, 1972).
- $\kappa(\bar{m}, n) := \frac{1}{n}\sum_{m=1}^{n}\kappa(m, n) = 3n - 8H_n + 13 - \frac{8H_n}{n}$ (Mahmoud, Modarres, and Smythe, 1995).

Many other results exist regarding the number of key comparisons required by Quickselect.

Framework of our study

- Quickselect is applied to a set of n distinct keys uniformly and independently distributed in (0,1).
- Each key is represented as a bit string, and Quickselect operates on individual bits in order to find a target key.
- We derive exact and asymptotic formulae for the expected numbers of bit comparisons required by Quickselect.

Framework of our study

- Quickselect is applied to a set of n distinct keys uniformly and independently distributed in (0,1).
- Each key is represented as a bit string, and Quickselect operates on individual bits in order to find a target key.
- We derive exact and asymptotic formulae for the expected numbers of bit comparisons required by Quickselect.

Framework of our study

- Quickselect is applied to a set of n distinct keys uniformly and independently distributed in (0,1).
- Each key is represented as a bit string, and Quickselect operates on individual bits in order to find a target key.
- We derive exact and asymptotic formulae for the expected numbers of bit comparisons required by Quickselect.

Preliminaries

Quickselect finds the m-th smallest key in a set of n keys $U_1, \ldots, U_n$. Let $U_{(i)}$ denote the i-th smallest key.

- $P\{U_{(i)} \text{ and } U_{(j)} \text{ are compared}\} = \begin{cases} \frac{2}{j-m+1} & \text{if } m \le i \\ \frac{2}{j-i+1} & \text{if } i < m < j \\ \frac{2}{m-i+1} & \text{if } j \le m. \end{cases}$
- $f_{U_{(i)},U_{(j)}}(s,t) := \binom{n}{i-1,1,j-i-1,1,n-j} s^{i-1}(t-s)^{j-i-1}(1-t)^{n-j}$.
- The event that $U_{(i)}$ and $U_{(j)}$ are compared is independent of the random variables $U_{(i)}$ and $U_{(j)}$.

Preliminaries

Quickselect finds the m-th smallest key in a set of n keys $U_1, \ldots, U_n$. Let $U_{(i)}$ denote the i-th smallest key.

- $P\{U_{(i)} \text{ and } U_{(j)} \text{ are compared}\} = \begin{cases} \frac{2}{j-m+1} & \text{if } m \leq i \\ \frac{2}{j-i+1} & \text{if } i < m < j \\ \frac{2}{m-i+1} & \text{if } j \leq m. \end{cases}$
- $f_{U_{(i)},U_{(j)}}(s,t) := \binom{n}{i-1,1,j-i-1,1,n-j} s^{i-1}(t-s)^{j-i-1}(1-t)^{n-j}.$
- The event that $U_{(i)}$ and $U_{(j)}$ are compared is independent of the random variables $U_{(i)}$ and $U_{(j)}$.

Preliminaries

Quickselect finds the m-th smallest key in a set of n keys $U_1, \ldots, U_n$. Let $U_{(i)}$ denote the i-th smallest key.

- $P\{U_{(i)} \text{ and } U_{(j)} \text{ are compared}\} = \begin{cases} \frac{2}{j-m+1} & \text{if } m \leq i \\ \frac{2}{j-i+1} & \text{if } i < m < j \\ \frac{2}{m-i+1} & \text{if } j \leq m. \end{cases}$

- $f_{U_{(i)},U_{(j)}}(s,t) := \binom{n}{i-1,1,j-i-1,1,n-j} \, s^{i-1}(t-s)^{j-i-1}(1-t)^{n-j}.$

- The event that $U_{(i)}$ and $U_{(j)}$ are compared is independent of the random variables $U_{(i)}$ and $U_{(j)}$.

Preliminaries

Quickselect finds the m-th smallest key in a set of n keys $U_1, \ldots, U_n$. Let $U_{(i)}$ denote the i-th smallest key.

- $$P\{U_{(i)} \text{ and } U_{(j)} \text{ are compared}\} = \begin{cases} \frac{2}{j-m+1} & \text{if } m \le i \\ \frac{2}{j-i+1} & \text{if } i < m < j \\ \frac{2}{m-i+1} & \text{if } j \le m. \end{cases}$$
- $$f_{U_{(i)},U_{(j)}}(s,t) := \binom{n}{i-1,1,j-i-1,1,n-j} s^{i-1}(t-s)^{j-i-1}(1-t)^{n-j}.$$
- The event that $U_{(i)}$ and $U_{(j)}$ are compared is independent of the random variables $U_{(i)}$ and $U_{(j)}$.

Preliminaries

- Define $P_1(s,t,m,n) := \sum_{m \le i < j \le n} \frac{2}{j-m+1} f_{U_{(i)},U_{(j)}}(s,t)$,
  $P_2(s,t,m,n) := \sum_{1 \le i < m < j \le n} \frac{2}{j-i+1} f_{U_{(i)},U_{(j)}}(s,t)$,
  $P_3(s,t,m,n) := \sum_{1 \le i < j \le m} \frac{2}{m-i+1} f_{U_{(i)},U_{(j)}}(s,t)$,
  $P(s,t,m,n) := P_1(s,t,m,n) + P_2(s,t,m,n) + P_3(s,t,m,n)$.
- we can write the expectation $\mu(m,n)$ of the number of bit comparisons required to find the rank-m key in a set of n keys as

$$\mu(m,n) = \int_0^1 \int_s^1 \beta(s,t) P(s,t,m,n)\, dt\, ds,$$

  where $\beta(s,t)$ denotes the first bit at which the keys s and t differ.

Preliminaries

- Define $P_1(s,t,m,n) := \sum_{m \le i < j \le n} \frac{2}{j-m+1} f_{U_{(i)},U_{(j)}}(s,t)$,
  $P_2(s,t,m,n) := \sum_{1 \le i < m < j \le n} \frac{2}{j-i+1} f_{U_{(i)},U_{(j)}}(s,t)$,
  $P_3(s,t,m,n) := \sum_{1 \le i < j \le m} \frac{2}{m-i+1} f_{U_{(i)},U_{(j)}}(s,t)$,
  $P(s,t,m,n) := P_1(s,t,m,n) + P_2(s,t,m,n) + P_3(s,t,m,n)$.
- we can write the expectation $\mu(m,n)$ of the number of bit comparisons required to find the rank-m key in a set of n keys as

$$\mu(m,n) = \int_0^1 \int_s^1 \beta(s,t) P(s,t,m,n)\, dt\, ds,$$

where $\beta(s,t)$ denotes the first bit at which the keys s and t differ.

Preliminaries

- Hence

$$\mu(m,n) = \int_0^1 \int_s^1 \beta(s,t) P(s,t,m,n)\, dt\, ds$$
$$= \sum_{k=0}^{\infty} \sum_{l=1}^{2^k} \int_{(l-1)2^{-k}}^{(l-\frac{1}{2})2^{-k}} \int_{(l-\frac{1}{2})2^{-k}}^{l2^{-k}} (k+1) P(s,t,m,n)\, dt\, ds,$$

  where k represents the last bit at which s and t agree.
- We analyze this expression in order to quantify the bit complexity of Quickselect.

Preliminaries

- Hence

$$\mu(m,n) = \int_0^1 \int_s^1 \beta(s,t) P(s,t,m,n)\,dt\,ds$$
$$= \sum_{k=0}^{\infty} \sum_{l=1}^{2^k} \int_{(l-1)2^{-k}}^{(l-\frac{1}{2})2^{-k}} \int_{(l-\frac{1}{2})2^{-k}}^{l2^{-k}} (k+1) P(s,t,m,n)\,dt\,ds,$$

  where k represents the last bit at which s and t agree.
- We analyze this expression in order to quantify the bit complexity of Quickselect.

Results: Exact computation of $\mu(1, n)$

- The expected number $\mu(1, n)$ of bit comparisons required by Quickselect to find the smallest key in a set of n keys satisfies

$$\mu(1, n) = 2n(H_n - 1) + 2\sum_{j=2}^{n-1} B_j \frac{n - j + 1 - \binom{n}{j}}{j(j-1)(1 - 2^{-j})},$$

  where B_j denotes the j-th Bernoulli number. (Note that $\mu(1, n) = \mu(n, n)$ by symmetry.)

- We analyzed this expression (in particular, $t_n := \sum_{j=2}^{n-1} B_j \frac{n-j+1-\binom{n}{j}}{j(j-1)(1-2^{-j})}$) to obtain an asymptotic expression for $\mu(1, n)$.

Results: Exact computation of $\mu(1, n)$

- The expected number $\mu(1, n)$ of bit comparisons required by Quickselect to find the smallest key in a set of n keys satisfies

$$\mu(1, n) = 2n(H_n - 1) + 2\sum_{j=2}^{n-1} B_j \frac{n - j + 1 - \binom{n}{j}}{j(j-1)(1-2^{-j})},$$

  where B_j denotes the j-th Bernoulli number. (Note that $\mu(1, n) = \mu(n, n)$ by symmetry.)

- We analyzed this expression (in particular, $t_n := \sum_{j=2}^{n-1} B_j \frac{n-j+1-\binom{n}{j}}{j(j-1)(1-2^{-j})}$) to obtain an asymptotic expression for $\mu(1, n)$.

Results: Asymptotic analysis of $\mu(1, n)$

Lemma. For $n \geq 2$, let $u_n := t_{n+1} - t_n$ (with $t_2 = 0$) and $v_n := v_{n+1} - v_n$. Let γ denote Euler's constant ($\doteq 0.57722$), and define $\chi_k := \frac{2\pi ik}{\ln 2}$. Then

(*i*) $v_n = \frac{1}{n+1} + \frac{\frac{H_{n+2}}{\ln 2} - (\frac{\gamma}{\ln 2} - \frac{1}{2})}{(n+1)(n+2)} - \Sigma_n$,
where
$\Sigma_n := \sum_{k \in \mathbb{Z}\setminus\{0\}} \frac{\zeta(1-\chi_k)\Gamma(n+1)\Gamma(1-\chi_k)}{(\ln 2)\Gamma(n+3-\chi_k)}$;

(*ii*) $u_n = -H_n + a - \frac{H_{n+1}}{(\ln 2)(n+1)} + \left(\frac{\gamma-1}{\ln 2} - \frac{1}{2}\right)\frac{1}{n+1} + \tilde{\Sigma}_n$,
where
$a := \frac{14}{9} + \frac{17-6\gamma}{18\ln 2} - \frac{2}{\ln 2}\sum_{k \in \mathbb{Z}\setminus\{0\}} \frac{\zeta(1-\chi_k)\Gamma(1-\chi_k)}{\Gamma(4-\chi_k)(1-\chi_k)}$,
$\tilde{\Sigma}_n := \sum_{k \in Z\setminus\{0\}} \frac{\zeta(1-\chi_k)\Gamma(1-\chi_k)}{(\ln 2)(1-\chi_k)} \frac{\Gamma(n+1)}{\Gamma(n+2-\chi_k)}$;

Results: Asymptotic analysis of $\mu(1, n)$

Lemma. For $n \geq 2$, let $u_n := t_{n+1} - t_n$ (with $t_2 = 0$) and $v_n := v_{n+1} - v_n$. Let γ denote Euler's constant ($\doteq 0.57722$), and define $\chi_k := \frac{2\pi ik}{\ln 2}$. Then

(*i*) $v_n = \frac{1}{n+1} + \frac{\frac{H_{n+2}}{\ln 2} - (\frac{\gamma}{\ln 2} - \frac{1}{2})}{(n+1)(n+2)} - \Sigma_n$,
where
$\Sigma_n := \sum_{k \in \mathbb{Z} \setminus \{0\}} \frac{\zeta(1-\chi_k)\Gamma(n+1)\Gamma(1-\chi_k)}{(\ln 2)\Gamma(n+3-\chi_k)}$;

(*ii*) $u_n = -H_n + a - \frac{H_{n+1}}{(\ln 2)(n+1)} + \left(\frac{\gamma - 1}{\ln 2} - \frac{1}{2}\right)\frac{1}{n+1} + \tilde{\Sigma}_n$,
where
$a := \frac{14}{9} + \frac{17 - 6\gamma}{18 \ln 2} - \frac{2}{\ln 2} \sum_{k \in \mathbb{Z} \setminus \{0\}} \frac{\zeta(1-\chi_k)\Gamma(1-\chi_k)}{\Gamma(4-\chi_k)(1-\chi_k)}$,
$\tilde{\Sigma}_n := \sum_{k \in Z \setminus \{0\}} \frac{\zeta(1-\chi_k)\Gamma(1-\chi_k)}{(\ln 2)(1-\chi_k)} \frac{\Gamma(n+1)}{\Gamma(n+2-\chi_k)}$;

Results: Asymptotic analysis of $\mu(1, n)$

Lemma. For $n \geq 2$, let $u_n := t_{n+1} - t_n$ (with $t_2 = 0$) and $v_n := v_{n+1} - v_n$. Let γ denote Euler's constant ($\doteq 0.57722$), and define $\chi_k := \frac{2\pi ik}{\ln 2}$. Then

(*i*) $v_n = \frac{1}{n+1} + \frac{\frac{H_{n+2}}{\ln 2} - (\frac{\gamma}{\ln 2} - \frac{1}{2})}{(n+1)(n+2)} - \Sigma_n$,
where
$\Sigma_n := \sum_{k \in \mathbb{Z} \setminus \{0\}} \frac{\zeta(1-\chi_k)\Gamma(n+1)\Gamma(1-\chi_k)}{(\ln 2)\Gamma(n+3-\chi_k)}$;

(*ii*) $u_n = -H_n + a - \frac{H_{n+1}}{(\ln 2)(n+1)} + \left(\frac{\gamma - 1}{\ln 2} - \frac{1}{2}\right) \frac{1}{n+1} + \tilde{\Sigma}_n$,
where
$a := \frac{14}{9} + \frac{17 - 6\gamma}{18 \ln 2} - \frac{2}{\ln 2} \sum_{k \in \mathbb{Z} \setminus \{0\}} \frac{\zeta(1-\chi_k)\Gamma(1-\chi_k)}{\Gamma(4-\chi_k)(1-\chi_k)}$,
$\tilde{\Sigma}_n := \sum_{k \in Z \setminus \{0\}} \frac{\zeta(1-\chi_k)\Gamma(1-\chi_k)}{(\ln 2)(1-\chi_k)} \frac{\Gamma(n+1)}{\Gamma(n+2-\chi_k)}$;

Results: Asymptotic analysis of $\mu(1, n)$

Lemma.

(*iii*) $$t_n = -(nH_n - n - 1) + a(n-2) - \frac{1}{2\ln 2}\left[H_n^2 + H_n^{(2)} - \frac{7}{2}\right] + \left(\frac{\gamma-1}{\ln 2} - \frac{1}{2}\right)\left(H_n - \frac{3}{2}\right) + b - \tilde{\tilde{\Sigma}}_n,$$

where

$$b := \sum_{k\in\mathbb{Z}\setminus\{0\}} \frac{2\zeta(1-\chi_k)\Gamma(-\chi_k)}{(\ln 2)(1-\chi_k)\Gamma(3-\chi_k)},$$

$$\tilde{\tilde{\Sigma}}_n := \sum_{k\in\mathbb{Z}\setminus\{0\}} \frac{\zeta(1-\chi_k)\Gamma(-\chi_k)\Gamma(n+1)}{(\ln 2)(1-\chi_k)\Gamma(n+1-\chi_k)},$$

and $H_n^{(2)}$ denotes the n-th Harmonic number of order 2, i.e., $H_n^{(2)} := \sum_{i=1}^{n} \frac{1}{i^2}$.

Results: Asymptotic analysis of $\mu(1, n)$

Asymptotic expression for $\mu(1, n)$:

$$\mu(1, n) = cn - \frac{1}{\ln 2}(\ln n)^2 - \left(\frac{2}{\ln 2} + 1\right) \ln n + O(1),$$

where $c \doteq 5.27938$.

Cf. the expectation for *key* comparisons is asymptotically $2n$.

Results: Asymptotic analysis of $\mu(1, n)$

Asymptotic expression for $\mu(1, n)$:

$$\mu(1, n) = cn - \frac{1}{\ln 2}(\ln n)^2 - \left(\frac{2}{\ln 2} + 1\right)\ln n + O(1),$$

where $c \doteq 5.27938$.

Cf. the expectation for *key* comparisons is asymptotically $2n$.

Results: Exact computation for average case: $\mu(\bar{m}, n)$

$$\begin{aligned}\mu(\bar{m}, n) &:= \tfrac{1}{n}\textstyle\sum_{m=1}^{n} \mu(m, n)\\ &= 2(n-1) - \tfrac{8}{n}F_1(n) + \tfrac{4}{n}F_2(n) + \tfrac{4}{9}F_3(n) - 4F_4(n) + \tfrac{8}{n}F_5(n),\end{aligned}$$

where

$$F_1(n) := \sum_{j=3}^{n} \frac{(-1)^j\binom{n}{j}}{(j-1)(j-2)}, \qquad F_2(n) := \sum_{j=2}^{n-1} \frac{B_j}{j(1-2^{-j})}\left[\frac{n-\binom{n}{j}}{j-1} - 1\right],$$

$$F_3(n) := \sum_{j=2}^{n-1} \frac{(-1)^j\binom{n-1}{j}}{j-1},$$

$$F_4(n) := \sum_{j=3}^{n-1} \frac{B_j}{j(j-1)(1-2^{-j})}\left[\frac{n-1-\binom{n-1}{j-1}}{j-2} - 1\right],$$

$$F_5(n) := \sum_{j=3}^{n} \frac{(-1)^j\binom{n}{j}}{j(j-1)(j-2)[1-2^{-(j-1)}]}.$$

Results: Asymptotic analysis of $\mu(\bar{m}, n)$

Asymptotic expression for $\mu(\bar{m}, n)$:

$$\mu(\bar{m}, n) = \tilde{c}n - \frac{4}{\ln 2}(\ln n)^2 + 4\left(\frac{2}{\ln 2} - 1\right)\ln n + O(1),$$

where $\tilde{c} \doteq 8.20731$.

Cf. the expectation for *key* comparisons is asymptotically $3n$.

Results: Asymptotic analysis of $\mu(m, n)$

Asymptotic analysis of $\mu(m, n)$ for fixed m has yet to be completed.

Results: Closed formula for $\mu(m, n)$

$$\mu(m, n) = \sum_{k=0}^{\infty} \sum_{l=1}^{2^k} \int_{(l-1)2^{-k}}^{(l-\frac{1}{2})2^{-k}} \int_{(l-\frac{1}{2})2^{-k}}^{l2^{-k}} (k+1) P(s, t, m, n)\, dt\, ds$$

$$= \sum_{b=1}^{n-1} (1-2b)^{-2} \sum_{f=m-1}^{n-2} \sum_{h=\alpha}^{n-f-2} \sum_{j=\beta}^{f+h+1} a_{j,b+j-(f+h+2)}$$

$$\times \frac{1}{(n+1)(f+1)} \sum_{i=m}^{f+1} \sum_{j=f+2}^{f+h+2} \binom{j-i-1}{f-i+1} \binom{n-j}{n-j+f+2} (-1)^{n-i-j+1}$$

$$\times \frac{2}{j-m+1} \binom{n}{i-1, 1, j-i-1, 1, n-j} (-1)^{f+h-j+1} \left(\frac{1}{2}\right)^{n-j+2}$$

$$\times \sum_{j'=0 \vee (j-1-h)}^{(j-1) \wedge f} \binom{f+1}{j'} \binom{h+1}{j-1-j'} \left[\left(\frac{1}{2}\right)^{j'} - \left(\frac{1}{2}\right)^{f+1} \right], \text{ where}$$

$$a_{j,r} := \frac{B_r}{r} \binom{j-1}{r-1} \ \text{ if } r \geq 2; \ := \frac{1}{j}, \frac{1}{2} \ \text{ if } r = 0, 1.$$

The running time for the computation is of order n^7.

Results: Closed formula for $\mu(m, n)$

$$\mu(m,n) = \sum_{k=0}^{\infty} \sum_{l=1}^{2^k} \int_{(l-1)2^{-k}}^{(l-\frac{1}{2})2^{-k}} \int_{(l-\frac{1}{2})2^{-k}}^{l2^{-k}} (k+1) P(s,t,m,n)\, dt\, ds$$

$$= \sum_{b=1}^{n-1} (1-2b)^{-2} \sum_{f=m-1}^{n-2} \sum_{h=\alpha}^{n-f-2} \sum_{j=\beta}^{f+h+1} a_{j,b+j-(f+h+2)}$$

$$\times \frac{1}{(n+1)(f+1)} \sum_{i=m}^{f+1} \sum_{j=f+2}^{f+h+2} \binom{j-i-1}{f-i+1} \binom{n-j}{n-j+f+2} (-1)^{n-i-j+1}$$

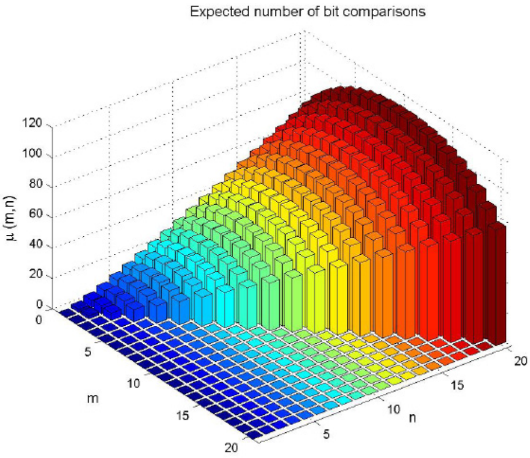
$$\times \frac{2}{j-m+1} \binom{n}{i-1,1,j-i-1,1,n-j} (-1)^{f+h-j+1} \left(\tfrac{1}{2}\right)^{n-j+2}$$

$$\times \sum_{j'=0 \vee (j-1-h)}^{(j-1) \wedge f} \binom{f+1}{j'} \binom{h+1}{j-1-j'} \left[\left(\tfrac{1}{2}\right)^{j'} - \left(\tfrac{1}{2}\right)^{f+1} \right], \text{ where}$$

$$a_{j,r} := \frac{B_r}{r} \binom{j-1}{r-1} \text{ if } r \geq 2; \; := \frac{1}{j}, \frac{1}{2} \text{ if } r = 0, 1.$$

The running time for the computation is of order n^7.

Results: Closed formula for $\mu(m, n)$



Summary

- At least for finding the smallest (or largest) key and in the average case, the expected number of bit comparisons required by Quickselect is asymptotically different from that of key comparisons only by a constant factor.
- Asymptotic analysis of $\mu(m, n)$ for fixed m has yet to be completed.
- Exact computation of $\mu(m, n)$ for fixed m can be achieved by $O(n^7)$ elementary operations.
- Ongoing work: Generalize the bit-string input model, for example to Bernoulli trials with success probability p.

Summary

- At least for finding the smallest (or largest) key and in the average case, the expected number of bit comparisons required by Quickselect is asymptotically different from that of key comparisons only by a constant factor.
- Asymptotic analysis of $\mu(m, n)$ for fixed m has yet to be completed.
- Exact computation of $\mu(m, n)$ for fixed m can be achieved by $O(n^7)$ elementary operations.
- Ongoing work: Generalize the bit-string input model, for example to Bernoulli trials with success probability p.

Summary

- At least for finding the smallest (or largest) key and in the average case, the expected number of bit comparisons required by Quickselect is asymptotically different from that of key comparisons only by a constant factor.
- Asymptotic analysis of $\mu(m, n)$ for fixed m has yet to be completed.
- Exact computation of $\mu(m, n)$ for fixed m can be achieved by $O(n^7)$ elementary operations.
- Ongoing work: Generalize the bit-string input model, for example to Bernoulli trials with success probability p.

Summary

- At least for finding the smallest (or largest) key and in the average case, the expected number of bit comparisons required by Quickselect is asymptotically different from that of key comparisons only by a constant factor.
- Asymptotic analysis of $\mu(m, n)$ for fixed m has yet to be completed.
- Exact computation of $\mu(m, n)$ for fixed m can be achieved by $O(n^7)$ elementary operations.
- Ongoing work: Generalize the bit-string input model, for example to Bernoulli trials with success probability p.

Ongoing work: More general bit-string input models

- This was not on a previous slide, but we recall

$$\mu(1,n) = 2\int_0^1\int_0^t \beta(s,t)F(t)^{-2}[(1-F(t))^n - 1 + nF(t)]\,dF(s)\,dF(t)$$

with input (key) distribution function $F(t) \equiv t$.

- By the same argument, this is true for general continuous F on $[0,1]$.
- Since

$$0 \le (1-F(t))^n - 1 + nF(t) \le (n-1)F(t),$$

it follows by the dominated convergence theorem that if

$$c \equiv c_F := 2\int_0^1\int_0^t \beta(s,t)F(t)^{-1}\,dF(s)\,dF(t) < \infty$$

then $\mu(1,n) \sim c\,n$ as $n \to \infty$.

Ongoing work: More general bit-string input models

- This was not on a previous slide, but we recall

$$\mu(1,n) = 2\int_0^1 \int_0^t \beta(s,t) F(t)^{-2} [(1-F(t))^n - 1 + nF(t)]\, dF(s)\, dF(t)$$

with input (key) distribution function $F(t) \equiv t$.

- By the same argument, this is true for general continuous F on $[0,1]$.
- Since

$$0 \le (1-F(t))^n - 1 + nF(t) \le (n-1)F(t),$$

it follows by the dominated convergence theorem that if

$$c \equiv c_F := 2\int_0^1 \int_0^t \beta(s,t) F(t)^{-1}\, dF(s)\, dF(t) < \infty$$

then $\mu(1,n) \sim c\, n$ as $n \to \infty$.

Ongoing work: More general bit-string input models

- This was not on a previous slide, but we recall

$$\mu(1,n) = 2\int_0^1 \int_0^t \beta(s,t) F(t)^{-2} [(1-F(t))^n - 1 + nF(t)]\, dF(s)\, dF(t)$$

with input (key) distribution function $F(t) \equiv t$.

- By the same argument, this is true for general continuous F on $[0,1]$.
- Since

$$0 \le (1-F(t))^n - 1 + nF(t) \le (n-1)F(t),$$

it follows by the dominated convergence theorem that if

$$c \equiv c_F := 2\int_0^1 \int_0^t \beta(s,t) F(t)^{-1}\, dF(s)\, dF(t) < \infty$$

then $\mu(1,n) \sim c\, n$ as $n \to \infty$.

Asymptotic slope c

- The asymptotic slope constant

$$c \equiv c_F := 2 \int_0^1 \int_0^t \beta(s,t) F(t)^{-1} \, dF(s) \, dF(t)$$

is not always finite; a necessary condition is that $\int_0^1 \log(1/t) \, dF(t) < \infty$.

- In the Bernoulli(p)-strings case, one can show $c = 2 \sum_{k=0}^{\infty} \gamma_k$ converges geometrically quickly, where

$$\gamma_k = 1 + \sum_{j=1}^{2^k} \left[F\left(\tfrac{j}{2^k}\right) - F\left(\tfrac{j-1}{2^k}\right) \right] \ln F\left(\tfrac{j}{2^k}\right) \quad \text{and}$$

$$F(.b_1 b_2 \ldots b_k) = q \sum_{m=1}^{k} b_m \prod_{i=1}^{m-1} q^{1-b_i} p^{b_i}.$$

Asymptotic slope c

- The asymptotic slope constant

$$c \equiv c_F := 2 \int_0^1 \int_0^t \beta(s,t) F(t)^{-1} \, dF(s) \, dF(t)$$

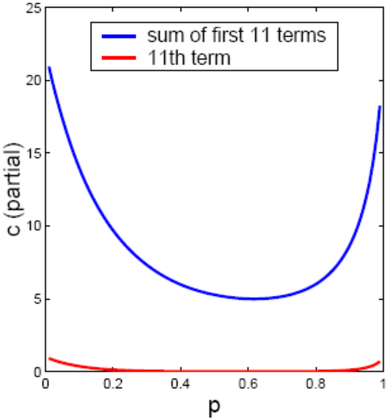
is not always finite; a necessary condition is that $\int_0^1 \log(1/t) \, dF(t) < \infty$.

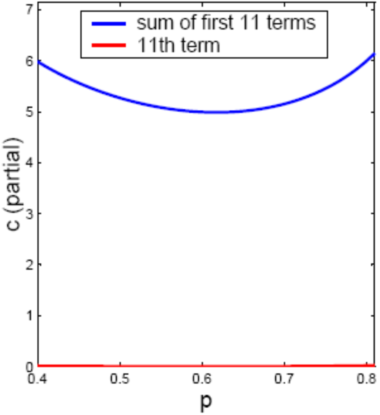
- In the Bernoulli(p)-strings case, one can show $c = 2 \sum_{k=0}^{\infty} \gamma_k$ converges geometrically quickly, where

$$\gamma_k = 1 + \sum_{j=1}^{2^k} \left[F\left(\tfrac{j}{2^k}\right) - F\left(\tfrac{j-1}{2^k}\right) \right] \ln F\left(\tfrac{j}{2^k}\right) \quad \text{and}$$

$$F(.b_1 b_2 \ldots b_k) = q \sum_{m=1}^{k} b_m \prod_{i=1}^{m-1} q^{1-b_i} p^{b_i}.$$

Asymptotic slope c: Bernoulli(p) strings





Asymptotic slope c: uniform case

- To be investigated: How does c behave as a function of the success probability p?
- In the uniform case $F(t) \equiv t$ (i.e., $p = 1/2$), the series-formula for $c = 2\sum_{k=0}^{\infty} \gamma_k = 5.27937\,82410\,80958+$ reduces:

$$\gamma_k = 1 + 2^{-k} \sum_{j=1}^{2^k} \ln\left(\frac{j}{2^k}\right).$$

Earlier, complex analysis gave, with $\chi_k := 2\pi i k / \ln 2$,

$$c = \frac{28}{9} + \frac{17 - 6\gamma}{9 \ln 2} - \frac{4}{\ln 2} \sum_{k \neq 0} \frac{\zeta(1-\chi_k)\Gamma(1-\chi_k)}{\Gamma(4-\chi_k)(1-\chi_k)}.$$

- We, and independently Grabner and Prodinger (2007), first found the real series for c_{unif} by "reverse engineering".

Asymptotic slope c: uniform case

- To be investigated: How does c behave as a function of the success probability p?
- In the uniform case $F(t) \equiv t$ (i.e., $p = 1/2$), the series-formula for $c = 2\sum_{k=0}^{\infty} \gamma_k = 5.27937\,82410\,80958+$ reduces:

$$\gamma_k = 1 + 2^{-k} \sum_{j=1}^{2^k} \ln\left(\frac{j}{2^k}\right).$$

Earlier, complex analysis gave, with $\chi_k := 2\pi i k / \ln 2$,

$$c = \frac{28}{9} + \frac{17 - 6\gamma}{9 \ln 2} - \frac{4}{\ln 2} \sum_{k \neq 0} \frac{\zeta(1 - \chi_k)\Gamma(1 - \chi_k)}{\Gamma(4 - \chi_k)(1 - \chi_k)}.$$

- We, and independently Grabner and Prodinger (2007), first found the real series for c_{unif} by "reverse engineering".

Asymptotic slope c: uniform case

- To be investigated: How does c behave as a function of the success probability p?
- In the uniform case $F(t) \equiv t$ (i.e., $p = 1/2$), the series-formula for $c = 2\sum_{k=0}^{\infty} \gamma_k = 5.27937\,82410\,80958+$ reduces:

$$\gamma_k = 1 + 2^{-k} \sum_{j=1}^{2^k} \ln\left(\frac{j}{2^k}\right).$$

  Earlier, complex analysis gave, with $\chi_k := 2\pi i k / \ln 2$,

$$c = \frac{28}{9} + \frac{17 - 6\gamma}{9 \ln 2} - \frac{4}{\ln 2} \sum_{k \neq 0} \frac{\zeta(1-\chi_k)\Gamma(1-\chi_k)}{\Gamma(4-\chi_k)(1-\chi_k)}.$$

- We, and independently Grabner and Prodinger (2007), first found the real series for c_{unif} by "reverse engineering".

Still to do (or at least to try)

- Higher moments? (or at least concentration)
- Get beyond lead term for $p \neq 1/2$ and other F with $c_F < \infty$?
- What if $c_F = \infty$? We can even have $\mu(1,2) = \infty$.
- Handle Quicksort similarly. This is actually easier, at least for Bernoulli(p) strings: With

$$\mathcal{E}(p) = \text{entropy} = -[p \ln p + (1-p)\ln(1-p)], \quad \text{we have}$$

$$\mu_n = 2 \sum_{j=2}^{n} \frac{(-1)^j \binom{n}{j}}{j(j-1)(1-p^j-q^j)} \sim \frac{n(\ln n)^2}{\mathcal{E}(p)},$$

and periodic fluctuations are no longer involved. Among distributions F with a density* f, lead-order asymptotics are not affected by choice of f [Fill and Janson, 2004].

Still to do (or at least to try)

- Higher moments? (or at least concentration)
- Get beyond lead term for $p \neq 1/2$ and other F with $c_F < \infty$?
- What if $c_F = \infty$? We can even have $\mu(1,2) = \infty$.
- Handle Quicksort similarly. This is actually easier, at least for Bernoulli(p) strings: With

$$\mathcal{E}(p) = \text{entropy} = -[p \ln p + (1-p)\ln(1-p)], \quad \text{we have}$$

$$\mu_n = 2\sum_{j=2}^{n} \frac{(-1)^j \binom{n}{j}}{j(j-1)(1-p^j-q^j)} \sim \frac{n(\ln n)^2}{\mathcal{E}(p)},$$

and periodic fluctuations are no longer involved. Among distributions F with a density* f, lead-order asymptotics are not affected by choice of f [Fill and Janson, 2004].

Still to do (or at least to try)

- Higher moments? (or at least concentration)
- Get beyond lead term for $p \neq 1/2$ and other F with $c_F < \infty$?
- What if $c_F = \infty$? We can even have $\mu(1,2) = \infty$.
- Handle Quicksort similarly. This is actually easier, at least for Bernoulli(p) strings: With

$$\mathcal{E}(p) = \text{entropy} = -[p \ln p + (1-p)\ln(1-p)], \quad \text{we have}$$

$$\mu_n = 2\sum_{j=2}^{n} \frac{(-1)^j \binom{n}{j}}{j(j-1)(1-p^j-q^j)} \sim \frac{n(\ln n)^2}{\mathcal{E}(p)},$$

and periodic fluctuations are no longer involved. Among distributions F with a density* f, lead-order asymptotics are not affected by choice of f [Fill and Janson, 2004].

Still to do (or at least to try)

- Higher moments? (or at least concentration)
- Get beyond lead term for $p \neq 1/2$ and other F with $c_F < \infty$?
- What if $c_F = \infty$? We can even have $\mu(1,2) = \infty$.
- Handle Quicksort similarly. This is actually easier, at least for Bernoulli(p) strings: With

$$\mathcal{E}(p) = \text{entropy} = -[p \ln p + (1-p)\ln(1-p)], \quad \text{we have}$$

$$\mu_n = 2\sum_{j=2}^{n} \frac{(-1)^j \binom{n}{j}}{j(j-1)(1-p^j-q^j)} \sim \frac{n(\ln n)^2}{\mathcal{E}(p)},$$

and periodic fluctuations are no longer involved. Among distributions F with a density* f, lead-order asymptotics are not affected by choice of f [Fill and Janson, 2004].

Still to do (or at least to try)

- Back to Quickselect, how much is saved if compared bits are remembered? (For Quicksort, Fill and Janson [2004] showed that this can remove extra log-factor from lead order of asymptotics.)
- What happens if we work in higher order bases? In particular, how do results for Bernoulli trials generalize to multinomial trials?

Still to do (or at least to try)

- Back to Quickselect, how much is saved if compared bits are remembered? (For Quicksort, Fill and Janson [2004] showed that this can remove extra log-factor from lead order of asymptotics.)
- What happens if we work in higher order bases? In particular, how do results for Bernoulli trials generalize to multinomial trials?