Algorithm Theory
4 Randomized Algorithms: Quicksort

Christian Schindelhauer
Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Wintersemester 2007/08
Randomized algorithms

- Classes of randomized algorithms
- Randomized Quicksort
- Randomized algorithm for Closest Pair
- Randomized primality test
- Cryptography
Classes of randomized algorithms

- **Las Vegas algorithms**
  - always correct; expected running time ("probably fast")
  - Examples:
    - randomized Quicksort,
    - randomized algorithm for closest pair

- **Monte Carlo algorithms (mostly correct):**
  - probably correct; guaranteed running time
  - Example: randomized primality test
Quicksort

Unsorted range $A[l, r]$ in array $A$

- $A[l \ldots r-1]$
- $A[l \ldots m-1]$  $\rightarrow$  $A[m+1 \ldots r]$

Quicksort  Quicksort
Quicksort

Algorithm: Quicksort

Input: unsorted range \([l, r]\) in array \(A\)

Output: sorted range \([l, r]\) in array \(A\)

1  if \(r > l\)
2       then choose pivot element \(p = A[r]\)
3       \(m = \text{divide}(A, l, r)\)
4            \(/\!*\text{Divide A according to } p:\!/\)
5                \(A[1], \ldots, A[m - 1] \leq p \leq A[m + 1], \ldots, A[r]\)
6            \(/\!*\!/\)
7       Quicksort\((A, l, m - 1)\)
8       Quicksort\((A, m + 1, r)\)
The *divide step*
The *divide* step

*divide*(A, l, r):

- returns the index of the pivot element in A
- can be done in time \( O(r – l) \)
Worst-case input

$n$ elements:

Running time: \((n-1) + (n-2) + \ldots + 2 + 1 = n\cdot(n-1)/2\)
Randomized Quicksort

Algorithm: Quicksort

Input: unsorted range \([l, r]\) in array \(A\)

Output: sorted range \([l, r]\) in array \(A\)

1. if \(r > l\)
   2. then randomly choose a pivot element \(p = A[i]\) in range \([l, r]\)
   3. swap \(A[i]\) and \(A[r]\)
   4. \(m = \text{divide}(A, l, r)\)
      /* Divide \(A\) according to \(p\):
         \(A[l], \ldots, A[m - 1] \leq p \leq A[m + 1], \ldots, A[r]\)
      */
   5. \(\text{Quicksort}(A, l, m - 1)\)
   6. \(\text{Quicksort}(A, m + 1, r)\)
Analysis 1

$n$ elements; let $S_i$ be the $i$-th smallest element

$S_1$ is chosen as pivot with probability $1/n$:
Sub-problems of sizes $0$ and $n-1$

$S_k$ is chosen as pivot with probability $1/n$:
Sub-problems of sizes $k-1$ and $n-k$

$S_n$ is chosen as pivot with probability $1/n$:
Sub-problems of sizes $n-1$ and $0$
Analysis 1

Expected running time:

\[ T(n) = \frac{1}{n} \sum_{k=1}^{n} (T(k - 1) + T(n - k)) + \Theta(n) \]

\[ = \frac{2}{n} \sum_{k=1}^{n} T(k - 1) + \Theta(n) \]

\[ = O(n \log n) \]
Analysis 2: Representation of Quicksort as a tree

\[ \pi = S_6 S_2 S_8 S_1 S_4 S_7 S_9 S_3 S_5 \]
Analysis 2

Expected number of comparisons:

\[ X_{ij} = \begin{cases} 
1 & \text{if } S_i \text{ is compared with } S_j \\
0 & \text{otherwise} 
\end{cases} \]

\[
E\left[ \sum_{i=1}^{n} \sum_{j>i} X_{ij} \right] = \sum_{i=1}^{n} \sum_{j>i} E[X_{ij}] 
\]

\[ p_{ij} = \text{probability that } S_i \text{ is compared with } S_j \]

\[
E[X_{ij}] = 1 \times p_{ij} + 0 \times (1 - p_{ij}) = p_{ij} 
\]
Calculation of $p_{ij}$

- $S_i$ is compared with $S_j$ iff $S_i$ or $S_j$ is chosen as pivot element in $\pi$ before any other $S_l$, $i<l<j$.
  $$\{S_i \ldots S_l \ldots S_j\}$$

- Each of the elements $S_i, \ldots, S_j$ is chosen first as the pivot with the same probability.

$$p_{ij} = \frac{2}{j-i+1}$$
Analysis 2

Expected number of comparisons:

\[
\sum_{i=1}^{n} \sum_{j>i} p_{ij} = \sum_{i=1}^{n} \sum_{j>i} \frac{2}{j-i+1}
\]

\[
= \sum_{i=1}^{n} \sum_{k=2}^{n-i+1} \frac{2}{k}
\]

\[
\leq 2 \sum_{i=1}^{n} \sum_{k=1}^{n} \frac{1}{k}
\]

\[
= 2n \sum_{k=1}^{n} \frac{1}{k}
\]

\[
H_n = \sum_{k=1}^{n} 1/k \approx \ln n
\]
Algorithm Theory
4 Randomized Algorithms: Quicksort

Christian Schindelhauer

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Wintersemester 2007/08