Lecture 4: February 14, 2002

Today:

- *Randomized* algorithm for sorting: Quicksort
Randomized algorithms

- Can make decisions based on coin flips
- Helps to “fool” adversary
- This is \textit{not} average case!
More on randomized algorithms

- We can choose an element $x$ uniformly at random from a specified range
- The time could be a random variable $\Rightarrow$ need to bound
  - the expected time
  - the probability the time exceeds certain value
- The correctness could be a random variable as well (does not include homeworks)
Quicksort

Yet another sorting algorithm, but:

- Sorts “in place” (doesn’t require additional array)
  - like insertion sort
  - unlike merge sort
- Very practical (with tuning)
- $\Theta(n \log n)$ time with “high” probability
Merge sort

1. Divide: Partition array into 2 subarrays of equal size.
2. Conquer: Recursively sort subarrays.
3. Combine: Merge two sorted subarrays.
\[ \Theta(n \lg n) \]
Quicksort

1. **Divide:** Partition array into 2 subarrays such that elements in lower part \( \leq \) elements in higher part.

| \( \leq x \) | \( \geq x \) |

2. **Conquer:** Recursively sort 2 subarrays.

3. **Combine:** Trivial (because in place).
**Partition procedure**

**Partition** \((A, p, r)\)

( Partition \(A[p..r]\) around random element \(x = A[k]\) )

\(k \leftarrow \text{Random}(p \ldots r)\)

\(x \leftarrow A[k]\)

\(i \leftarrow p - 1\)

\(j \leftarrow r + 1\)

**while** TRUE **do**

**repeat** \(j \leftarrow j - 1\) **until** \(A[j] \leq x\)

**repeat** \(i \leftarrow i + 1\) **until** \(A[i] \geq x\)

**if** \(i < j\) **then** exchange \(A[i] \leftrightarrow A[j]\)

**else** quit (and return \(j\))
Correctness proof idea

Loop invariant:

<table>
<thead>
<tr>
<th></th>
<th>( \leq x )</th>
<th>?</th>
<th>( \geq x )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p )</td>
<td>( i )</td>
<td>( j )</td>
<td>( r )</td>
</tr>
</tbody>
</table>

Time = \( \Theta(n) \) for \( n \)-element subarray
**Quicksort — Recursive algorithm**

\[\text{QUICKSORT}(A, p, r)\]

\[\text{if } p < r \]

\[\text{then } q \leftarrow \text{PARTITION}(A, p, r)\]

\[\text{QUICKSORT}(A, p, q - 1)\]

\[\text{QUICKSORT}(A, q + 1, r)\]

Initial call: \[\text{QUICKSORT}(A, 1, length[A])\]
Quicksort analysis

- Correctness - done
- Running time ?
Analysis of Quicksort (best case)

If we’re lucky, Partition always splits array evenly

\[ T(n) = 2T(n/2) + \Theta(n) \]

\[ \lg n \]

\[ n \]

\[ n/2 \]

\[ n/4 \]

\[ n/8 \]

\[ \Theta(n \lg n) \]
Analysis of \textbf{QUICKSORT} (OK case)

Suppose the split is $\frac{1}{10} : \frac{9}{10}$

\begin{align*}
T(n) &= T(n/10) + T(9n/10) + \Theta(n) \\
&= \Theta(n \lg n) \quad \text{Still lucky!}
\end{align*}
Analysis of Quicksort (worst case)

How might we be unlucky?

- one side of partition has 1 element

\[
T(n) = T(1) + T(n-1) + \Theta(n)
\]

\[
= T(n-1) + \Theta(n) \quad \text{because } T(1) = \Theta(1)
\]

\[
= \sum_{k=1}^{n} \Theta(k) = \Theta(\sum_{k=1}^{n} k)
\]

\[
= \Theta(n^2) \quad [\text{arithmetic series}]
\]
Analysis of Quicksort

- Optimistic case
- Semi-optimistic case
- Worst case
- ...
- Need more scientific approach
- Will show the time is at most $Cn \log n$ with high probability
Key ideas

- The running time bounded by $O(n)$ times the depth of the recursion tree (as seen on earlier pictures)
- “Nice splits” happen with “nice probability”
- If we have large enough number of trials, and each trial has “nice probability” of success, we have many successes with high probability
- For technical reasons, assume all input elements are distinct. Picking random index of element is equivalent to picking random element.
Lucky partitions

Let $a_i$ be the $i$-th smallest element in $A[\cdot]$ (i.e., with the rank $i$).

Let $D_i$ denote the depth of $a_i$ in the recursion tree. The total tree depth is $D = \max_i D_i$.

We say a split (say, of $A[p \ldots r]$) is “lucky” for $a_i$, if the part where $a_i$ ends up in is of size $\leq 3/4(r - p + 1)$, i.e., if we reduce the array size by a factor of $3/4$.

<table>
<thead>
<tr>
<th>$a_i$</th>
<th>$x$</th>
</tr>
</thead>
</table>

After $\log_{4/3} n$ lucky splits, $a_i$ is a leaf!
“Lucky” lemma

**Lemma:** At any time, \( a_i \) is lucky with probability at least 1/2.

**Proof:** To get a lucky split, it is sufficient that \( x \) has rank in \( \{n/4 \ldots 3n/4\} \). This happens with probability \( \geq 1/2 \).
The analysis

Let \( l = C \log_{4/3} n \), where \( C \) is a “large” constant. What is the probability that \( D_i > l \)?

It is at most the probability that in \( l \) trials, each having success probability \( \geq 1/2 \), we were \textit{not} successful \( \geq l - \log_{4/3} n \) times.

The latter probability is \textit{at most}

\[
\binom{l}{l-t} \left( \frac{1}{2} \right)^{l-t}
\]

for \( t = \log_{4/3} n \).
The analysis ctd.

\[
\left( \binom{l}{l-t} \right)^{1/2} \leq \frac{e^l}{t} \leq \frac{\binom{e^l}{t}}{t^{1/2}} \leq \left( \frac{eC \log_4/3 n}{\log_4/3 n} \right)^t 1/2(C-1) \log_4/3 n = \left( \frac{eC}{2C-1} \right)^{\log_4/3 n}
\]

When \( C \) is large enough (e.g., 20), the probability is smaller than \( 1/n^2 \).
Finishing the analysis

We proved that \( \Pr[D_i > l] \leq 1/n^2 \).

Therefore, the probability that there exists \( i \) such that \( D_i > l \) is at most

\[
\Pr[D_1 > l] + \Pr[D_2 > l] \ldots + \Pr[D_n > l] \leq n \cdot 1/n^2 = 1/n
\]

Therefore probability that \( \max_i D_i \leq l \) is at least

\( 1 - 1/n \).
Appendix

Why is the probability that “in \( l \) trials, each having success probability \( \geq 1/2 \), we were not successful \( \geq l - t \) times” at most

\[
\left( \frac{l}{l - t} \right) \left( \frac{1}{2} \right)^{l-t}
\]

?

- if we were not successful \( \geq l - t \) times, we were not successful during some set of \( l - t \) trials
- the probability we were not successful during a fixed set of exactly \( l - t \) trials is at most \( 1/2^{l-t} \)
- there are \( \binom{l}{l-t} \) subsets of trials of size \( l - t \)