ALG 1.3

Deterministic Selection and Sorting:

(a) Selection Algorithms and Lower Bounds
(b) Sorting Algorithms and Lower Bounds

Problem P

size n

⇒ divide into subproblems size n₁, ..., nₖ

solve these and "glue" together solutions

\[
T(n) = \sum_{i=1}^{k} T(n_i) + g(n)
\]

time to combine solutions

Examples:

1st lecture's mult

\[
M(n) = 3 \cdot M\left(\left\lfloor \frac{n}{2} \right\rfloor \right) + \Theta(n)
\]

fast fourier transform

\[
F(n) = 2 \cdot F\left(\left\lfloor \frac{n}{2} \right\rfloor \right) + \Theta(n)
\]

binary search

\[
B(n) = B\left(\left\lfloor \frac{n}{2} \right\rfloor \right) + \Theta(1)
\]

merge sorting

\[
S(n) = 2 \cdot S\left(\left\lfloor \frac{n}{2} \right\rfloor \right) + \Theta(n)
\]

Main Reading Selections:
CLR, Chapters 7, 9, 10

Auxiliary Reading Selections:
AHU-Design, Chapters 2 and 3
AHU-Data, Chapter 8
BB, Sections 4.4, 4.6 and 10.1
Selection, and Sorting on Decision Tree Model

Input $a, b, c$

Diagram of a decision tree with the following structure:
- Root: $a=b$?
- Left: $a<b$?
- Right: $a<c$?

- $b<c$?
  - Left: $b,a,c$?
  - Right: $b,c,a$?
  - Left: $a,c,b$?
  - Right: $e,b,a$?

**Examples**

**Binary tree** with $L$ Leaves

**Facts:**
1. Has $L-1$ internal nodes
2. Max height $\geq \lceil \log L \rceil$

**Time** = (\# comparisons on longest path)
Merging

2 lists with total of n keys

input

\[ X_1 < X_2 < \ldots < X_k \]
and
\[ Y_1 < Y_2 < \ldots < Y_{n-k} \]

output

ordered merge of two key lists

Algorithm Insert

input \( (X_1 < X_2 < \ldots < X_k), (Y_1) \)

Case \( k=n-1 \)

Algorithm: Binary Search

by Divide-and-Conquer

[1] Compare \( Y_1 \) with \( X_{\lceil k/2 \rceil} \)
[2] if \( Y_1 > X_{\lceil k/2 \rceil} \) insert \( Y_1 \) into
\[ \left( X_{\lfloor k/2 \rfloor + 1} < \ldots < X_k \right) \]
otherwise \( Y_1 \leq X_{\lceil k/2 \rceil} \) and insert \( Y_1 \) into
\[ \left( X_1 < \ldots < X_{\lfloor k/2 \rfloor} \right) \]

use this Model because it

allows simple proofs of lower bounds

time = # comparisons so easy to bound time costs

goal

provably asymptotically optimal algorithm in Decision Tree Model
**Case: Merging equal length lists**

Input: 

\[
(X_1 < X_2 < ... < X_k) \\
(Y_1 < Y_2 < ... < Y_{n-k})
\]

where \( k = \frac{n}{2} \)

**Algorithm**

1. \( i \leftarrow 1, j \leftarrow 1 \)
2. \( \text{while } i \leq k \text{ and } j \leq k \) do
   - \( \text{if } X_i < Y_j \text{ then output } (X_i) \text{ and set } i \leftarrow i + 1 \)
   - \( \text{else output } (Y_j) \text{ and set } j \leftarrow j + 1 \)
3. \( \text{output remaining elements} \)

Algorithm clearly uses \( 2k - 1 = n - 1 \text{ comparisons} \)

---

**Total Comparison Cost:**

\[ \leq \lceil \log (k+1) \rceil = \lceil \log (n) \rceil \]

Since a binary tree with \( n = k + 1 \) leaves has depth \( \lceil \log(n) \rceil \), this is optimal!
**Lower bound:**

Consider case $X_1 < Y_1 < X_2 < Y_2 < ... < X_k < Y_k$

Any merge algorithm must compare

**Claim:**

1. $X_i$ with $Y_i$ for $i = 1, ..., k$
2. $Y_i$ with $X_{i+1}$ for $j = 1, ..., k-1$

(Otherwise we could flip $Y_i < X_i$ with no change)

$\Rightarrow$ so requires $\geq 2k - 1 = n - 1$ comparisons!

---

**Sorting by Divide-and-Conquer**

**Algorithm**

Merge Sort

**input**

Set $S$ of $n$ keys

---

[1] **partition** $S$ into set $X$ of $\left\lceil \frac{n}{2} \right\rceil$ keys

and set $Y$ of $\left\lfloor \frac{n}{2} \right\rfloor$ keys

[2] **Recursively compute**

Merge Sort $(X) = (X_1, X_2, ..., X_{\left\lfloor \frac{n}{2} \right\rfloor})$

Merge Sort $(Y) = (Y_1, Y_2, ..., Y_{\left\lfloor \frac{n}{2} \right\rfloor})$

[3] **merge above sequences**

using $n - 1$ comparisons

[4] **output**

Merged sequence
Time Analysis

\[ T(n) = T \left( \left\lfloor \frac{n}{2} \right\rfloor \right) + T \left( \frac{n}{2} \right) + n - 1 \]

\[ T(1) = 0 \]

\[ \Rightarrow T(n) = n \left\lfloor \log n \right\rfloor - 2 \left\lceil \log n \right\rceil + 1 \]

\[ = \Theta(n \log n) \]

Lower Bounds on Sorting (on decision tree model)

Depth \( \geq \left\lceil \log (n!) \right\rceil \)

n! distinct leaves
**Easy Approximation** (via Integration)

\[
\log(n!) = \log(n) + \log(n-1) + \ldots + \log(2) + \log(1)
\]

\[
\geq \int_{n-1}^{n} \log x \, dx + \ldots + \int_{1}^{2} \log x \, dx
\]

\[
\geq \int_{1}^{n} \log x \, dx \quad \text{(Since } \log k \geq \int_{k-1}^{k} \log x \, dx) \quad \geq n \log n - n \log e + \log e
\]

**Better bound** using **Sterling Approximation**

\[
n! \geq \sqrt{2\pi n} \left( \frac{n}{e} \right)^n \left( 1 + \frac{1}{12n} \right)
\]

\[
\Rightarrow \log(n!) \geq n \log n - n \log e + \frac{1}{2} \log (2\pi n)
\]

**Selection Problems**

**input** \( X_1, X_2, \ldots, X_n \) and index \( k \in \{1, \ldots, n\} \)

**output** \( x(k) = \text{the k'th best} \)
History:

Rev C.L. Dodge (Lewis Carol) wrote an article on lawn tennis tournament in James Gazett, 1883.

felt *prizes unjust* because:
- although winner $X^{(1)}$ always gets 1st prize
- second $X^{(2)}$ may *not* get 2nd prize

Note: $X$ not declared 2nd best left branch.

Carol proposed his own (nonoptimal) tournament....

Selection of the champion $X^{(1)}$

- $X^{(1)}$ is easily determined in $n-1$ comparisons

Proof:

everyone except the champion $X^{(1)}$ must lose at least once!
Selection of the second best \( X_{(2)} \) using \( n-2 + \lceil \log n \rceil \) comparisons

Algorithm

[1] form a balanced binary tree for tournament to find \( X_{(1)} \) using \( n-1 \) comparisons

[2] Let \( Y \) be the set of players knocked out by champion \( X_{(1)} \)

\[ |Y| \leq \lceil \log n \rceil \]

[3] Play a tournament among the \( Y \)'s

[4] output \( X_{(2)} = \) champion of the \( Y \)'s using \( \lceil \log n \rceil - 1 \) more comparisons
**Lower Bounds on finding $X_{(2)}$**

requires $\geq n-2 + \log n$ comparisons

---

**proof**

#comparison $\geq m_1 + m_2 + ...$

where $m_i = \#$players who lost $i$ or more matches

---

**Claim** $m_1 \geq n-1$, since at end we must know $X_{(1)}$ as well as $X_{(2)}$

**Claim** $m_2 \geq (\#who lost to X_{(1)}) - 1$

since everyone (except $X_{(2)}$) who lost to $X_{(1)}$ must also have lost one more time.
lemma

\[
(\text{#who lost to } X_{(1)}) \geq \lceil \log n \rceil
\]

in worst case

proof

Use oracle who "fixes" results of games so that champion \(X_{(1)}\) plays \(\geq \lceil \log n \rceil\) matches

\[
\Rightarrow \text{forces path from } X_{(1)} \text{ to root to have length } \geq \lceil \log n \rceil
\]

\[X_i > X_j\text{ if}\]

(a) \(X_i\) previously undefeated and \(X_j\) lost at least once

(b) both undefeated but \(X_i\) played more matches

(c) otherwise, decide consistently with previous decisions
Selection by Divide-and-Conquer

Algorithm \text{Select}_{k}(X)

\text{input} \quad \text{set } X \text{ of } n \text{ keys and index } k

\textbf{1} \quad \text{if } n < c_0 \text{ then output } \text{X}_{(k)} \text{ by sorting } X \text{ and halt}

\textbf{2} \quad \text{divide } X \text{ into } \frac{n}{d} \text{ sequences of } d \text{ elements each (with } < d \text{ leftover),}
\quad \text{and sort each sequence}

\textbf{3} \quad \text{let } M \text{ be the } \text{medians} \text{ of each of these sequences}

\textbf{4} \quad m \leftarrow \text{Select}_{\lceil\frac{|M|}{2}\rceil}(M)

\textbf{5} \quad \text{let } X^- = \{x \in X \mid x < m\}
\quad \text{let } X^+ = \{x \in X \mid x > m\}

\textbf{6} \quad \text{if } |X^-| \geq k \text{ then output Select}_{k}(X^-)
\quad \text{else if } n - |X^+| = k \text{ then output } m
\quad \text{else output Select}_{k-(n-|X^+|)}(X^+)
Proposition \[ |X^-|, |X^+| \text{ each } \leq n \cdot \left( \frac{d+1}{2} \right)^j \cdot \frac{n}{2d} \leq \frac{3}{4} n \]

\[
T(n) \leq \begin{cases} 
  c_1 & \text{if } n < c_0 \\
  T \left( \left\lfloor \frac{n}{d} \right\rfloor \right) + T \left( \frac{3}{4} n \right) + c_1 n & \text{otherwise}
\end{cases}
\]

for a sufficiently large constant \( c_1 \)

(assuming \( d \) is constant)

If say \( d=5 \), \( T(n) \leq 20c_1 n = O(n) \)
Lower Bounds for Selecting $X(k)$

**input**

$X = \{x_1, \ldots, x_n\}$, index $k$

**Theorem**

Every leaf of Decision Tree has depth $\geq n-1$

**proof**

Fix a path $p$ from root to leaf.

The comparisons done on $p$ define a relation $R_p$

Let $R_p^+ = \text{transitive closure of } R_p$

**Lemma**

If path $p$ determines $X_m = X_{(k)}$ then for all $i \neq m$ either $x_i R_p^+ x_m$ or $x_m R_p^+ x_i$.

**proof**

Suppose $x_i$ is un related to $x_m$ by $R_p^+$.

Then can replace $x_i$ in linear order either before or after $x_m$ to violate $x_m = x_{(k)}$

Let the "key" comparison for $x_i$ be when $x_i$ is compared with $x_j$ where either

1. $j=m$
2. $x_i R_p x_j$ and $x_j R_p^+ x_m$, or
3. $x_j R_p x_i$ and $x_m R_p^+ x_j$

**Fact**

$x_i$ has unique "key" comparison determining either $x_i R_p^+ x_m$ or $x_m R_p^+ x_i$.

$\Rightarrow$ So there are $n-1$ "key" comparisons, each distinct!
A hard to analyze sort:

**SHELLSORT**

input keys $X_1, \ldots, X_n$

(1) $\Delta \leftarrow \left\lfloor \frac{n}{2} \right\rfloor$

(2) \textbf{while} $\Delta > 0$ \textbf{do}

\hspace{1em} for $i = \Delta + 1$ \textbf{to} $n$ \textbf{do}

\hspace{2em} \begin{align*}
\text{increment} & \hspace{1em} j \leftarrow i - \Delta \\
\text{sort} & \hspace{1em} \textbf{while} j > 0 \textbf{ do}
\hspace{2em} \textbf{if} \quad x_j > x_{j+\Delta} \textbf{ then}
\hspace{3em} \textbf{begin}
\hspace{4em} \text{SWAP} (x_j, x_{j+\Delta}) \\
\hspace{4em} j \leftarrow j - \Delta \\
\hspace{3em} \text{end}
\hspace{2em} \textbf{else} \quad j \leftarrow 0
\end{align*}

\hspace{1em} \textbf{end}

\hspace{1em} $\Delta \leftarrow \left\lfloor \Delta / 2 \right\rfloor$

AHU Data Structures & Alg., pp. 290-291
passes of SHELLSORT:

1. increment sort \( (X_k, X_{\frac{n}{2^k}+k}) \) for \( k=1, \ldots, \frac{n}{2} \)

2. increment sort \( (X_k, X_{\frac{n}{4^k}+k}, X_{\frac{n}{2^k}+k}, X_{\frac{3n}{4^k}+k}) \)
   for \( k=1, \ldots, \frac{n}{4} \)

procedure

increment sort \( (Y_1, \ldots, Y_l) \)

\[
\begin{align*}
&\text{for } i = z \text{ by } 1 \text{ until } i > n \text{ or } X_{i-1} < X_i \\
&\quad \text{do for } j = 1 \text{ by } -1 \text{ until } 1 \\
&\quad \quad \text{if } X_{j-1} > X_j \text{ then } \text{swap} \ (X_{j-1}, X_j)
\end{align*}
\]

facts

1. if \( X_1, X_{\frac{n}{p^2}+1} \) sorted in pass \( p \) then they remain sorted in later passes

2. distance between comparisons diminish as \( \frac{n}{2}, \frac{n}{4}, \ldots, \frac{n}{p^2}, \ldots \)

3. The best known time bound is \( O\left(n^{1.5}\right)\)
procedure \textsc{Radixsort}

\textbf{input} \quad \(X_1, \ldots, X_n \in \{1, \ldots, n\}\)

\[1\] \quad \textbf{for} \quad j=1, \ldots, n \quad \textbf{do} \\
\quad \text{initialize } B[j] \text{ to be the empty list}

\[2\] \quad \textbf{for} \quad i=1, \ldots, n \quad \textbf{do} \\
\quad \text{add } i \text{ to } B[X_i]

\[3\] \quad \text{let } L = (i_1, i_2, \ldots, i_n) \text{ be the} \\
\quad \text{concatenation of } B[1], \ldots, B[n]

\[4\] \quad \textbf{output} \quad X_{i_1} \leq X_{i_2} \leq \ldots \leq X_{i_n}

\textbf{--} \textit{Costs }O(n)\textit{ time} \quad \text{on unit cost RAM}

\textbf{--} \textit{avoids }\Omega(n \log n)\textit{ lower bound on }\textsc{sort} \quad \text{by avoiding comparisons}
\quad \text{instead uses indexing of RAM}

\textbf{--} \textit{generalizes (in }c\textit{ passes) to key} \\
\quad \textit{domains }\{1, \ldots, n^c\}
open problems in sorting

(1) Complexity of SHELLSORT
   -- very good in practice
      claims Sedgewick
   -- Is it $\theta(n^{1.5})$?

(2) Complexity of variable length
   -- sort on multitape TM or RAM
   -- Is it $\Omega(n \log n)$?