Multiplying Numbers

Problem: Given two \( n \)-digit numbers \( x \) and \( y \), compute their product.

Grade School Multiplication

Compute "partial product" by multiplying each digit of \( y \) with \( x \) and adding the partial products.

\[
3141 \\
\times 2718 \\
25128 \\
3141 \\
21987 \\
6282 \\
8537238
\]

Time Analysis of Grade School Multiplication

- Each partial product: \( \Theta(n) \)
- Number of partial products: \( \Theta(n) \)
- Addition of partial products: \( \Theta(n^2) \)
- Total time: \( \Theta(n^2) \)
A Trick of Gauss
Carl Friedrich Gauss: 1777–1855 “Prince of Mathematicians”

Observation: Multiply two complex numbers: \((a + bi)\) and \((c + di)\)

\[(a + bi)(c + di) = ac - bd + (ad + bc)i\]

How many multiplications do we need?

Only 3! If we do extra additions and subtractions.
Compute \(ac, bd, (a + b)(c + d)\). Then \((ad + bc) = (a + b)(c + d) - ac - bd\)

Example

\[
1234 \times 5678 = (100 \times 12 + 34) \times (100 \times 56 + 78)
\]

\[
= 10000 \times 12 \times 56 + 100 \times (12 \times 78 + 34 \times 56) + 34 \times 78
\]

\[
1234 \times 5678 = (100 \times 12 + 34) \times (100 \times 56 + 78)
\]

\[
= 10000 \times 12 \times 56
\]

\[
+ 100 \times (12 \times 78 + 34 \times 56)
\]

\[
+ 34 \times 78
\]

Divide and Conquer
Assume \(n\) is a power of 2 for simplicity and numbers are in decimal.

Split each number into two numbers with equal number of digits

\(x = x_{n-1}x_{n-2} \ldots x_0\) and \(y = y_{n-1}y_{n-2} \ldots y_0\)

\(x = x_{n-1} \ldots x_{n/2}0 \ldots 0 + x_{n/2-1} \ldots x_0\)

\(x = 10^{n/2}x_L + x_R\) where \(x_L = x_{n-1} \ldots x_{n/2}\) and \(x_R = x_{n/2-1} \ldots x_0\)

Similarly \(y = 10^{n/2}y_L + y_R\) where \(y_L = y_{n-1} \ldots y_{n/2}\) and \(y_R = y_{n/2-1} \ldots y_0\)

Therefore

\[
xy = (10^{n/2}x_L + x_R)(10^{n/2}y_L + y_R)
\]

\[
= 10^n x_L y_L + 10^{n/2}(x_L y_R + x_R y_L) + x_R y_R
\]
Time Analysis

\[ xy = (10^{n/2}x_L + x_R)(10^{n/2}y_L + y_R) = 10^n x_L y_L + 10^n (x_L y_R + x_R y_L) + x_R y_R \]

4 recursive multiplications of number of size \( n/2 \) each plus 4 additions and left shifts (adding enough 0’s to the right)

\[ T(n) = 4T(n/2) + O(n) \quad T(1) = O(1) \]

\( T(n) = 2 \Theta(n^2) \). No better than grade school multiplication!

Can we invoke Gauss’s trick here?

Improving the Running Time

\[ xy = (10^{n/2}x_L + x_R)(10^{n/2}y_L + y_R) = 10^n x_L y_L + 10^n (x_L y_R + x_R y_L) + x_R y_R \]

Gauss trick: \( x_L y_R + x_R y_L = (x_L + x_R)(y_L + y_R) - x_L y_L - x_R y_R \)

Recursively compute only \( x_L y_L, x_R y_R, (x_L + x_R)(y_L + y_R) \).

Time Analysis

Running time is given by

\[ T(n) = 3T(n/2) + O(n) \quad T(1) = O(1) \]

which means \( T(n) = O(n \log 3) = O(n^{1.585}) \)

State of the Art

Schönhage-Strassen 1971: \( O(n \log n \log \log n) \) time using Fast-Fourier-Transform (FFT)

Martin Fürer 2007: \( O(n \log n 2^{O(\log^* n)}) \) time

Conjecture

There is an \( O(n \log n) \) time algorithm.

Analyzing the Recurrences

- Basic divide and conquer: \( T(n) = 4T(n/2) + O(n) \), \( T(1) = 1 \). Claim: \( T(n) = \Theta(n^2) \).
- Saving a multiplication: \( T(n) = 3T(n/2) + O(n) \), \( T(1) = 1 \). Claim: \( T(n) = \Theta(n^{1+\log 1.5}) \)

Use recursion tree method:
- In both cases, depth of recursion \( L = \log n \).
- Work at depth \( i \) is \( 4^i n/2^i \) and \( 3^i n/2^i \) respectively: number of children at depth \( i \) times the work at each child
- Total work is therefore \( n \sum_{i=0}^{L} 2^i \) and \( n \sum_{i=0}^{L} (3/2)^i \) respectively.
Recursion tree analysis

Part II
Selecting in Unsorted Lists

Rank of element in an array

**A**: an unsorted array of **n** integers

**Definition**
For 1 ≤ j ≤ n, element of rank j is the j’th smallest element in A.

<table>
<thead>
<tr>
<th>Unsorted array</th>
<th>16</th>
<th>14</th>
<th>34</th>
<th>20</th>
<th>12</th>
<th>5</th>
<th>3</th>
<th>19</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ranks</td>
<td>6</td>
<td>5</td>
<td>9</td>
<td>8</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>Sort of array</td>
<td>3</td>
<td>5</td>
<td>11</td>
<td>12</td>
<td>14</td>
<td>16</td>
<td>19</td>
<td>20</td>
<td>34</td>
</tr>
</tbody>
</table>

Problem - Selection

**Input** Unsorted array **A** of **n** integers and integer **j**

**Goal** Find the **j**th smallest number in **A** (rank **j** number)

**Median**: \(j = \lceil (n + 1)/2 \rceil\)

Simplifying assumption for sake of notation: elements of **A** are distinct
Algorithm I

1. Sort the elements in $A$
2. Pick $j$th element in sorted order
   Time taken = $O(n \log n)$

Do we need to sort? Is there an $O(n)$ time algorithm?

Algorithm II

If $j$ is small or $n - j$ is small then
1. Find $j$ smallest/largest elements in $A$ in $O(jn)$ time. (How?)
2. Time to find median is $O(n^2)$.

Divide and Conquer Approach

1. Pick a pivot element $a$ from $A$
2. Partition $A$ based on $a$.
   $A_{\text{less}} = \{x \in A \mid x \leq a\}$ and $A_{\text{greater}} = \{x \in A \mid x > a\}$
3. $|A_{\text{less}}| = j$: return $a$
4. $|A_{\text{less}}| > j$: recursively find $j$th smallest element in $A_{\text{less}}$
5. $|A_{\text{less}}| < j$: recursively find $k$th smallest element in $A_{\text{greater}}$
   where $k = j - |A_{\text{less}}|$.

Example

$\begin{bmatrix}
16 & 14 & 34 & 20 & 12 & 5 & 3 & 19 & 11
\end{bmatrix}$
Partitioning step: $O(n)$ time to scan $A$
- How do we choose pivot? Recursive running time?

Suppose we always choose pivot to be $A[1]$.

Say $A$ is sorted in increasing order and $j = n$.
Exercise: show that algorithm takes $\Omega(n^2)$ time.

Intuition: The median
Find the median
Break input
Find median
Use

A game of medians

Divide and Conquer Approach

Idea
- Break input $A$ into many subarrays: $L_1, \ldots, L_k$.
- Find median $m_i$ in each subarray $L_i$.
- Find the median $x$ of the medians $m_1, \ldots, m_k$.
- Intuition: The median $x$ should be close to being a good median of all the numbers in $A$.
- Use $x$ as pivot in previous algorithm.

Time Analysis

- Tail recursive call: Select element of rank $50$ out of $56$ elements.
**Algorithm for Selection**

A storm of medians

```java
select(A, j):
    Form lists \(L_1, L_2, \ldots, L_{\lceil n/5 \rceil}\) where \(L_i = \{A[5i - 4], \ldots, A[5i]\}\)
    Find median \(b_i\) of each \(L_i\) using brute-force
    Find median \(b\) of \(B = \{b_1, b_2, \ldots, b_{\lceil n/5 \rceil}\}\)
    Partition \(A\) into \(A_{\text{less}}\) and \(A_{\text{greater}}\) using \(b\) as pivot
    if \(|A_{\text{less}}| = j\) return \(b\)
    else if \(|A_{\text{less}}| > j\)
        return \(select(A_{\text{less}}, j)\)
    else
        return \(select(A_{\text{greater}}, j - |A_{\text{less}}|)\)
```

How do we find median of \(B\)? Recursively!

**Choosing the pivot**

A clash of medians

1. Partition array \(A\) into \(\lceil n/5 \rceil\) lists of 5 items each.
   \(L_1 = \{A[1], A[2], \ldots, A[5]\}, L_2 = \{A[6], \ldots, A[10]\}, \ldots, L_{\lceil n/5 \rceil} = \{A[5\lceil n/5 \rceil - 4], \ldots, A[n]\}\)
2. For each \(L_i\) find median \(b_i\) of \(L_i\) using brute-force in \(O(1)\) time.
   Total \(O(n)\) time
3. Let \(B = \{b_1, b_2, \ldots, b_{\lceil n/5 \rceil}\}\)
4. Find median \(b\) of \(B\)

**Lemma**

Median of \(B\) is an approximate median of \(A\). That is, if \(b\) is used a pivot to partition \(A\), then

\[|A_{\text{less}}| \leq 7n/10 + 6\] and

\[|A_{\text{greater}}| \leq 7n/10 + 6.\]
Running time of deterministic median selection
A dance with recurrences

\[ T(n) \leq T(\lceil n/5 \rceil) + \max\{ T(|A_{less}|), T(|A_{greater}|)\} + O(n) \]

From Lemma,

\[ T(n) \leq T(\lceil n/5 \rceil) + T(\lceil 7n/10 + 6 \rceil) + O(n) \]

and

\[ T(n) = O(1) \quad n < 10 \]

Exercise: show that \( T(n) = O(n) \)

---

Median of Medians: Proof of Lemma

**Proposition**

There are at least \( 3n/10 - 6 \) elements smaller than the median of medians \( b \).

**Proof.**

At least half of the \( \lfloor n/5 \rfloor \) groups have at least 3 elements smaller than \( b \), except for the group containing \( b \) which has 2 elements smaller than \( b \). Hence number of elements smaller than \( b \) is:

\[ 3\lfloor \frac{n/5}{2} \rfloor + 1 - 1 \geq \frac{3n}{10} - 6 \]
### Median of Medians: Proof of Lemma

**Proposition**
There are at least \( \frac{3n}{10} - 6 \) elements smaller than the median of medians \( b \).

**Corollary**
\( |A_{\text{greater}}| \leq \frac{7n}{10} + 6 \).

Via symmetric argument,

**Corollary**
\( |A_{\text{less}}| \leq \frac{7n}{10} + 6 \).

### Summary: Selection in linear time

**Theorem**
The algorithm \( \text{select}(A[1 \ldots n], k) \) computes in \( O(n) \) deterministic time the \( k \)-th smallest element in \( A \).

On the other hand, we have:

**Lemma**
The algorithm \( \text{QuickSelect}(A[1 \ldots n], k) \) computes the \( k \)-th smallest element in \( A \). The running time of \( \text{QuickSelect} \) is \( \Theta(n^2) \) in the worst case.

### Questions to ponder

- Why did we choose lists of size 5? Will lists of size 3 work?
- Write a recurrence to analyze the algorithm’s running time if we choose a list of size \( k \).

### Median of Medians Algorithm

Due to:
"Time bounds for selection".

How many Turing Award winners in the author list?
All except Vaughn Pratt!
Takeaway Points

- Recursion tree method and guess and verify are the most reliable methods to analyze recursions in algorithms.
- Recursive algorithms naturally lead to recurrences.
- Sometimes one can look for certain type of recursive algorithms (reverse engineering) by understanding recurrences and their behavior.