Algorithm Analysis

Zoe Siegelnickel and Palak Yadav
1. Introduction - Zoe
2. Time Complexity - Palak
3. Search Algorithms - brief
   a. Linear vs Binary – graph
4. Recursive Algorithms - brief -- Palak
   a. Intro
   b. Fibonacci
   c. Merge Sort
5. Karatsuba - Palak
   a. Naive
   b. Algorithm Proof + Run Time
6. Akra Bazzi - Zoe
   a. Matrix Multiplication -- maybe
      i. Strassen
Intro to Algorithm Analysis

**Algorithm:** A set of instructions that a computer follows and applies on input
- **Input:** data entered into an algorithm
- **Output:** results produced

**Algorithm Analysis:** How long it takes for the computer to follow these instructions
- Best measured in terms of large input sizes
Measuring Efficiency

- Can’t measure time it takes to run ~ machine dependent

**Need:**
- Machine Independence
- How algorithm behaves as input size increase

**Run Time:**
# of steps or operations executed
Depends on input size (#elements)

**Input Size:**
#elements inserted in algorithm
Represented by \( n \)

**Cases to consider:**
- Worst-case
- Best-case
- Average case
Big O notation $O(n)$

- describe the upper-bound
- worst-case scenario

Big-Omega $\Omega$

- lower bound
- best-case scenario

Theta $\Theta$

- describes best and worst case scenario.
- gives the exact bound.
Search Algorithms

Linear Search

**Linear Search:** Sort through until desired element is found

Binary Search:
- Divides data set in half
- Compares target value with middle term
- Eliminates half set that does not contain T
- Repeats until T found

**O(n)**

**Binary Search**

Worst-case binary search (8-element array)

```
1 2 5 6 7 9 11 13
```

Step 1: 9 > 7 (choose right)

```
7 9 11 13
```

Step 2: 9 < 11 (choose left)

```
7 9
```

Step 3: 9 = 9 (key located)

Key located in 3 operations

$\log(8) = 3$

**O(log_2 n)**
Recursive Algorithms

- Divide the larger problem into subproblems by calling itself
- Divides until the base case is reached and performs the algorithm's objective on easily solvable inputs
Fibonacci Sequence

\[ f(n) = \begin{cases} 
0 & \text{if } n = 0 \\
1 & \text{if } n = 1 \\
F(n-1) + F(n-2) & \text{if } n > 1 
\end{cases} \]

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987 ...

Each number is the sum of the previous two numbers.
Karatsuba Algorithm

- Fast Multiplication Algorithm
- Reduces time it takes to multiple two even \( n \)-digit numbers
  - If odd, add zeros
Naive Method of Multiplying

- Each digit in $x$ multiplied to each digit in $y$
- Total: 4 single-digit computations to find $x \times y$
- $n^2$ operations ($n \rightarrow \# \text{ digits in each number}$)

$$\begin{align*}
45 & \times 32 \\
45 & \times 32 \\
45 & \times 32 \\
45 & \times 32
\end{align*}$$

$n^2$ operations $\rightarrow$ Time Complexity $O(n^2)$
Deriving Karatsuba Algorithm

1. Split number in half

\[ x = 45 = 40 + 5 \]  \hspace{1cm}  \[ y = 32 = 30 + 2 \]

High bit: 4 = \( a \)  \hspace{1cm}  High bit: 3 = \( c \)

Low bit: 5 = \( b \)  \hspace{1cm}  Low bit: 2 = \( d \)

2. Another way to express multiplication

\[ xy = (40 + 5) \times (30 + 2) \]

3. Distributive property

\[ xy = (40 \times 30) + (40 \times 2) + (5 \times 30) + (5 \times 2) \]

Even this way, still 4 computations (\( n^2 \))

\( n \rightarrow \# \text{digits in each number} \)
**Karatsuba Algorithm**

\[ x = 45 = 40 + 5 \]

High bit: 4 = \( a \)
Low bit: 5 = \( b \)

\[ y = 32 = 30 + 2 \]

High bit: 3 = \( c \)
Low bit: 2 = \( d \)

\[ xy = (40 \times 30) + (40 \times 2) + (5 \times 30) + (5 \times 2) \]

- **High bits**: \( ac \)
- **Middle bits**: \( ad + bc \)
- **Low bits**: \( bd \)

KA divides problem into 3 sub-problems (instead of 4)
Karatsuba Algorithm Cont.

\[
\begin{align*}
x &= 45 = 40 + 5 \\
y &= 32 = 30 + 2
\end{align*}
\]

High bit: 4 = \(a\)  
Low bit: 5 = \(b\)  
High bit: 3 = \(c\)  
Low bit: 2 = \(d\)

\[
xy = (40 \times 30) + (40 \times 2) + (5 \times 30) + (5 \times 2)
\]

\[
\begin{align*}
(40 \times 30) + & \quad (40 \times 2 + 5 \times 30) + (5 \times 2)
\end{align*}
\]

High bits  +  Middle bits  +  Low bits

\[
\begin{align*}
ac + (ad + bc) + bd
\end{align*}
\]

\[
\text{Middle Term}
\]

\[
(ad + bc) = (a + b)(c + d) - ac - bd
\]

\[
\text{Proof } \rightarrow \text{ Gauss’ Trick:}
\]

\[
(ac + ad + bc + bd) - ac - bd
\]

Subtract high & low bits from total to find middle

\[
\text{NOTE: Bases of ten (zeros) can be ignored for now and added on at the end}
\]
Generalized Karatsuba Algorithm

\[ x.y = (10^n ac + 10^{n/2} (ad + bc) + bd \]

H                      M                   L

1. Recursively compute ac  \( \text{High bits} \)
2. Recursively compute bd \( \text{Low bits} \)
3. Recursively compute \((a+b)(c+d) = ac+b+ad+bc\) \( \text{Middle bits} \)

Gauss’ Trick : \((3) - (1) - (2) = ad + bc\)

\textbf{NOTE:} Bases of ten (zeros) can be ignored for now and added on at the end
Karatsuba Run Time

**High bits**

\[ ac \]

\[ n/2 \times n/2 \]

1 single-digit computation

**Middle bits**

\[ (a + b)(c + d) - ac - bd \]

\[ (n/2) \times (n/2) \]

1 single-digit computation

**Low bits**

\[ bd \]

\[ n/2 \times n/2 \]

1 single-digit computation

**Total: 3 computations** instead of 4
Time Complexity

\[ T(n) = 3T\left(\frac{n}{2}\right) + O(n). \]

\[ \Theta\left(n^{\log_2 3}\right) \approx \Theta(n^{1.585}). \]

\( T \rightarrow \) run time for multiplication
\( O(n) \rightarrow \) standard time for arithmetic
Akra - Bazzi
Akra Bazzi Method

Recurrence relation: expression of a term as a function of the terms before it.

\[ T(x) = g(x) + \sum_{i=1}^{k} a_i T(b_i x + h_i(x)) \]

Takes recurrence relation as input: outputs asymptotic time complexity.

\[ \sum_{i=1}^{k} a_i b_i^p = 1 \]

\[ T(x) = \Theta \left( x^p \left( 1 + \int_1^x \frac{g(u)}{u^{(p+1)}} du \right) \right) \]
Strassen Algorithm
Strassen Algorithm

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \times \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} ae + bg & af + bh \\ ce + dg & cf + dh \end{bmatrix}$$
\[ M_1 = (A_{11} + A_{22})(B_{11} + B_{22}); \]
\[ M_2 = (A_{21} + A_{22})B_{11}; \]
\[ M_3 = A_{11}(B_{12} - B_{22}); \]
\[ M_4 = A_{22}(B_{21} - B_{11}); \]
\[ M_5 = (A_{11} + A_{12})B_{22}; \]
\[ M_6 = (A_{21} - A_{11})(B_{11} + B_{12}); \]
\[ M_7 = (A_{12} - A_{22})(B_{21} + B_{22}), \]

\[ T(x) = 7T(x/2) + \Theta(n^3) \]
Proof: Akra Bazzi

\[ T(x) = 7T(x/2) + \Theta(n^3) \]

\[ a = 7 \]
\[ b = \frac{1}{2} \]
\[ p = \log 7 \]
Works Cited


Acknowledgement

We would like to express our gratitude to our mentor John Shackleton for his guidance. Thank you for all your help!
Thank you!