CSE 417
Algorithms and Complexity

Autumn 2023
Lecture 16
Divide and Conquer and Recurrences
Divide and Conquer

• Recurrences, Sections 5.1 and 5.2
• Algorithms
  – Median (Selection)
  – Fast Matrix Multiplication
  – Counting Inversions (5.3)
  – Multiplication (5.5)
Divide and Conquer : Merge Sort

Array MSort(Array a, int n){
    if (n <= 1) return a;
    return Merge(MSort(a[0 .. n/2], n/2), MSort(a[n/2+1 .. n-1], n/2));
}

T(n) = 2T(n/2) + n; T(1) = 1;
Unrolling the recurrence
Substitution

Prove \( T(n) \leq n (\log_2 n + 1) \) for \( n \geq 1 \)

Induction – Show \( P(1) \) and

\[
\forall k < n P(k) \implies P(n)
\]

Base Case: \( T(1) = 1 = 1 \ (\log_2 1 + 1) \)

Induction: Assume \( T(n/2) \leq n/2 (\log_2(n/2) + 1) \)

\[
T(n) = 2 \ T(n/2) + n \\
\leq 2 \ n/2 (\log_2(n/2) + 1) + n \\
= n \ (\log_2 n - 1 + 1) + n \\
= n \ (\log_2 n + 1)
\]
T(n) = aT(n/b) + n^c

Master Theorem

If $T(n) = aT(n/b) + O(n^d)$ for constants $a > 0$, $b > 1$, $d \geq 0$, then

$T(n) = O(n^d)$ if $d > \log_b a$

$T(n) = O(n^{d \log n})$ if $d = \log_b a$

$T(n) = O(n^{\log_b a})$ if $d < \log_b a$
$T(n) = T(n/2) + cn$

Where does this recurrence arise?
Solving the recurrence exactly
\[ T(n) = 4T(n/2) + n \]
T(n) = 2T(n/2) + n^2
T(n) = 2T(n/2) + n^{1/2}
Recurrences

• Three basic behaviors
  – Dominated by initial case
  – Dominated by base case
  – All cases equal – we care about the depth
Geometric Sum

\[ \sum_{i=0}^{n} x^i = \frac{x^{n+1} - 1}{x - 1} \]
What you really need to know about recurrences

• Work per level changes geometrically with the level
  
• Geometrically increasing \((x > 1)\)
  – The bottom level wins

• Geometrically decreasing \((x < 1)\)
  – The top level wins

• Balanced \((x = 1)\)
  – Equal contribution
Classify the following recurrences (Increasing, Decreasing, Balanced)

- \( T(n) = n + 5T(n/8) \)
- \( T(n) = n + 9T(n/8) \)
- \( T(n) = n^2 + 4T(n/2) \)
- \( T(n) = n^3 + 7T(n/2) \)
- \( T(n) = n^{1/2} + 3T(n/4) \)
Recursive Matrix Multiplication

Multiply 2 x 2 Matrices:
\[
\begin{pmatrix}
  r & s \\
  t & u
\end{pmatrix}
\begin{pmatrix}
  a & b \\
  c & d
\end{pmatrix}
\begin{pmatrix}
  e & g \\
  f & h
\end{pmatrix}
\]

r = ae + bf
s = ag + bh
t = ce + df
u = cg + dh

A N x N matrix can be viewed as a 2 x 2 matrix with entries that are \((N/2) \times (N/2)\) matrices.

The recursive matrix multiplication algorithm recursively multiplies the \((N/2) \times (N/2)\) matrices and combines them using the equations for multiplying 2 x 2 matrices.
Recursive Matrix Multiplication

- How many recursive calls are made at each level?
- How much work in combining the results?
- What is the recurrence?
What is the run time for the recursive Matrix Multiplication Algorithm?

• Recurrence:
Strassen’s Algorithm

Multiply 2 x 2 Matrices:
\[
\begin{array}{cc}
  r & s \\
  t & u \\
\end{array}
= \begin{array}{cc}
  a & b \\
  c & d \\
\end{array}
\begin{array}{cc}
  e & g \\
  f & h \\
\end{array}
\]

\[
\begin{align*}
  r &= p_1 + p_2 - p_4 + p_6 \\
  s &= p_4 + p_5 \\
  t &= p_6 + p_7 \\
  u &= p_2 - p_3 + p_5 - p_7
\end{align*}
\]

Where:
\[
\begin{align*}
  p_1 &= (b - d)(f + h) \\
  p_2 &= (a + d)(e + h) \\
  p_3 &= (a - c)(e + g) \\
  p_4 &= (a + b)h \\
  p_5 &= a(g - h) \\
  p_6 &= d(f - e) \\
  p_7 &= (c + d)e
\end{align*}
\]

From AHU 1974
Recurrence for Strassen’s Algorithms

• \( T(n) = 7 \ T(n/2) + cn^2 \)
• What is the runtime?

\[ \log_2 7 \approx 2.8073549221 \]