Divide and Conquer Algorithms

6CCS3PAL-7CCSMPDA
Contents

- The divide and conquer method
- Some examples of algorithms

- In-order tree traversal
- Implementing 'parallel for' in multi-threading
- Horner's method

- In later notes
- Various approaches to sorting
- Other topics as time permits
Divide and conquer is an algorithm design paradigm based on multi-branched recursion.
A divide and conquer algorithm works by recursively breaking down a problem into two or more sub-problems of the same type, until these become simple enough to be solved directly. The solutions to the sub-problems are then combined to give a solution to the original problem. (wikipedia)
Recursive Fibonacci: FIB1(6)

Figure: Cormen (CLRS) Introduction to Algorithms Chapter 27
Divide and conquer strategy

1. Divide input into partitions of almost equal size
2. Recursively solve the subproblems defined by the partition
3. Combine the solutions to the subproblems into a single answer and pass it back as the answer to the recursion call

It must be possible to perform steps 1, 3 efficiently for the D+C approach to work well.

Subproblems must be independent for step 2 to be possible.

Adding up the entries in an array satisfies these conditions.

Can implement D+C in the Multithreading model `spawn`, `sync`.

Also in the PRAM model if we assume the existence of an instruction `in parallel do ...`
Divide and conquer strategy

1. Divide input into partitions of almost equal size
2. Recursively solve the subproblems defined by the partition
3. Combine the solutions to the subproblems into a single answer and pass it back as the answer to the recursion call

It must be possible to perform steps 1,3 efficiently for the D+C approach to work well. Subproblems must be independent for step 2 to be possible. Adding up the entries in an array satisfies these conditions.

Can implement D+C in the Multi-treading model spawn, sync.

Also in the PRAM model if we assume the existence of an instruction in parallel do ...
Generalized Divide and Conquer

See Berman and Paul. Algorithms: Sequential, Parallel and Distributed. Chapter 8

Input: $I$ (an input to the given problem)
Output: $J$ (a solution to problem for input $I$)

Algorithm **Divide-and-Conquer**($I$, $J$)

if $I \in$ Known-Answers then
    return $J$ (a solution to $I$)
else
    Divide $I$ into $m \geq 2$ disjoint sub-problems ($I_1, \cdots, I_m$)
    for $k = 1 \ldots m$ do
        **Divide-and-Conquer**($I_k$, $J_k$)
    end
    Combine solutions ($J_1, \cdots, J_m$) into solution $J$
    return $J$
end
In-order traversal

In computer science, tree traversal is the process of visiting (checking and/or updating) each node in a tree data structure, exactly once.

In-order: [A, B, C, D, E, F, G, H, I]

https://en.wikipedia.org/wiki/Tree_traversal
Binary tree $T$ with data held at vertices. Assume existence of following functions:
$Left(T)$, $Right(T)$ to obtain left and right sub-trees of $T$,
$Data(T)$ to obtain the value of the data in the root of $T$,
$IsEmpty(T)$ to test if $T$ is an empty tree

Algorithm $\textbf{P-In-Order} (T)$
   If $IsEmpty(T)$ then
      return
   else
      $x = \text{spawn } \textbf{P-In-Order}(Left(T))$
      $y = Data(T)$
      $z = \textbf{P-In-Order}(Right(T))$
      sync
      return $x, y, z$
end
Exercises

- What was the corresponding sequential algorithm?
- Write a parallel algorithm in the multi-threading model which uses divide and conquer to count the number of occurrences of an item $X$ held as data in the vertices of a binary tree. Use the functions given in the previous slide.
- Write a parallel divide and conquer algorithm in multi-threading model to add up a numeric array $A[1..n]$.
- Write a parallel D+C algorithm in the multi-threading model to count the number of occurrences of an item $X$ in an array $A[1..n]$.
- Write an explicit (non D+C) array based CREW PRAM algorithm to count the number of occurrences of an item $X$ in an array $A[1..n]$. 

Parallel loops: multi-threading

CLRS Chapter 27 pages 785-787 (also 793- D+C)

parallel for $i = 1, ..., n$ statement $S(i)$

We can use divide and conquer to implement parallel for

Execute parallel for $i = 1..n/2$ and parallel for $i = n/2 + 1..n$ in parallel

**ParallelFor**($i, j, S$)

  if $i = j$ (end condition) evaluate $S(i)$
  else
    
    $m = \lfloor (i + j)/2 \rfloor$
    
    spawn **ParallelFor**($i, m, S$)
    **ParallelFor**($m + 1, j, S$)

  sync

end
Span of ParallelFor

\( T(n) \): Run time of ParallelFor(1, n, S)

Assume \( S(i) \) is a simple statement so that ParallelFor(1, 1, S) has runtime \( T(1) = 1 \)
Assume \( n = 2^m \)

Each D+C step is parallel (unit time)

\[
T(n) = 1 + T(n/2) = k + T(n/2^k) = m + T(1) = \log_2 n + 1 = \Theta(\log n)
\]

The span is \( \Theta(\log n) \)
Horner’s Method

A technique as old as the hills. Typical applications include

- Evaluating polynomials.
  What is \( y = 3x^7 + 2x^2 + 1 \) when \( x = 4 \)?

- Finding roots of polynomials.
  When is \( x^7 + 2x^2 + 1 = 0 \)?

- Changing number bases. Convert hexadecimal to decimal.

- Horner’s method is a fast, code-efficient method for multiplication and division of binary numbers on a micro-controller with no hardware multiplier. Uses register shift followed by add.

See https://en.wikipedia.org/wiki/Horner’s_method

...
Many error correcting codes are based on polynomial algebra.
Polynomials on a computer?

Many error correcting codes are based on polynomial algebra.

How to represent \( y = 3x^7 + 2x^2 + 1 \) on a computer?

Use an array \( Y[0..7] \). First entry is 1, second 0, third 2 etc.

\[ Y = [1, 0, 2, 0, 0, 0, 0, 3] \]

Wait a minute! How can we evaluate \( Y(x) \) at e.g. \( x = 104 \)?
Polynomials on a computer?

Many error correcting codes are based on polynomial algebra

How to represent $y = 3x^7 + 2x^2 + 1$ on a computer?

How to represent $y = 1 + 2x^2 + 3x^7$ on a computer?
Polynomials on a computer?

Many error correcting codes are based on polynomial algebra

How to represent \( y = 3x^7 + 2x^2 + 1 \) on a computer?

How to represent \( y = 1 + 2x^2 + 3x^7 \) on a computer?

How to represent \( y = 1x^0 + 0x^1 + 2x^2 + 0x^3 + 0x^4 + 0x^5 + 0x^6 + 3x^7 \) on a computer?

Use an array \( Y[0..7] \). First entry is 1, second 0, third 2 etc.

\[ Y = [1, 0, 2, 0, 0, 0, 0, 3] \]

Wait a minute! How can we evaluate \( Y(x) \) at e.g. \( x = 104 \)
Many error correcting codes are based on polynomial algebra.

How to represent $y = 3x^7 + 2x^2 + 1$ on a computer?

How to represent $y = 1 + 2x^2 + 3x^7$ on a computer?

How to represent $y = 1x^0 + 0x^1 + 2x^2 + 0x^3 + 0x^4 + 0x^5 + 0x^6 + 3x^7$ on a computer?

Use an array $Y[0..7]$. First entry is 1, second 0, third 2 etc.
Polynomials on a computer?

Many error correcting codes are based on polynomial algebra

How to represent \( y = 3x^7 + 2x^2 + 1 \) on a computer?

How to represent \( y = 1 + 2x^2 + 3x^7 \) on a computer?

How to represent
\[
y = 1x^0 + 0x^1 + 2x^2 + 0x^3 + 0x^4 + 0x^5 + 0x^6 + 3x^7
\]
on a computer?

Use an array \( Y[0..7] \). First entry is 1, second 0, third 2 etc

\( Y = [1, 0, 2, 0, 0, 0, 0, 3] \)
Many error correcting codes are based on polynomial algebra.

How to represent \( y = 3x^7 + 2x^2 + 1 \) on a computer?

How to represent \( y = 1 + 2x^2 + 3x^7 \) on a computer?

How to represent \( y = 1x^0 + 0x^1 + 2x^2 + 0x^3 + 0x^4 + 0x^5 + 0x^6 + 3x^7 \) on a computer?

Use an array \( Y[0..7] \). First entry is 1, second 0, third 2 etc.

\[ Y = [1, 0, 2, 0, 0, 0, 0, 3] \]

Wait a minute! How can we evaluate \( Y(x) \) at e.g. \( x = 104 \)?
Sequential Horner

Evaluate polynomial held as an array $A[0..n]$
Note: array index $i$ is coefficient of $x^i$

**Sequential-Horner**$(A, n, x)$

- $y = 0$ (subtotal)
- $i = n$ (descending index)

while $i \geq 0$ do
  
  $y = A[i] + x \cdot y$
  
  $i = i - 1$

end
Horner's method is $\Theta(n)$ for array size $n$.

$T^*(n) = \Theta(n)$

Can't improve on this.
Need to inspect $n + 1$ array entries to evaluate
Example: \( P(x) = 3 + 5x + x^2 + 7x^3 \)
Example: $P(x) = 3 + 5x + x^2 + 7x^3$

Evaluate at $x = 2$. $P(2) = ???
Example: \( P(x) = 3 + 5x + x^2 + 7x^3 \)

Evaluate at \( x = 2 \). \( P(2) =??? \)

Answer should be \( P(2) = 73 \)
Example: $P(x) = 3 + 5x + x^2 + 7x^3$

Evaluate at $x = 2$. $P(2) = ???$

Answer should be $P(2) = 73$

Example: \( P(x) = 3 + 5x + x^2 + 7x^3 \)

Evaluate at \( x = 2 \). \( P(2) =??? \)

Answer should be \( P(2) = 73 \)

Array \( A[0..n] \). \( A = [3, 5, 1, 7] \)
\( A[0..3] \) so \( n = 3 \)

Method is \( P(x) = 3 + x(5 + x(1 + x(7))) \))
Example: \( P(x) = 3 + 5x + x^2 + 7x^3 \)

Evaluate at \( x = 2 \). \( P(2) = ??? \)

Answer should be \( P(2) = 73 \)

Array \( A[0..n] \). \( A = [3, 5, 1, 7] \)
\( A[0..3] \) so \( n = 3 \)

Method is \( P(x) = 3 + x(5 + x(1 + x(7))) \)

<table>
<thead>
<tr>
<th>( i )</th>
<th>( A[i] )</th>
<th>( y ) update</th>
<th>( y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initialize</td>
<td></td>
<td>( y = 0 )</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>( y = 7 + 2 \times 0 )</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>( y = 1 + 2 \times 7 )</td>
<td>15</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>( y = 5 + 2 \times 15 )</td>
<td>35</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>( y = 3 + 2 \times 35 )</td>
<td>73</td>
</tr>
</tbody>
</table>

Exercise: Evaluate \( P(3) \) in this way
Parallel Horner to evaluate polynomial

Most D+C algorithms are about splitting something in the middle (at the median data point)


Split array (and hence polynomial) at median \( s = n/2 \)

If \( x^s \) factored from \( A[s..t] \) then resulting polynomial is

\[ Q(x) = A[s] + A[s+1]x + \cdots + A[t]x^{t-s} \]

Array indexes and coefficient powers get out of step by \( s \)

\[ P(x) = A[0] + A[1]x + \cdots + A[s-1]x^{s-1} + x^s Q(x) \]
Parallel Horner \((A, 0, n, x)\)

\[
\text{Parallel-Horner}(A, p, r, x) \\
\text{if } p == r \text{ then return } A[p] \\
\text{else} \\
\quad q = \lfloor (p + r)/2 \rfloor \\
\quad \text{in parallel do} \\
\quad \quad L = \text{Parallel-Horner}(A, p, q, x) \\
\quad \quad M = \text{Parallel-Horner}(A, q + 1, r, x) \\
\quad \text{return } L + x^{q+1-p} \cdot M
\]

We need to pass the array indices \(p, \ldots, r\) to the method to retrieve the appropriate array entries. Horner method treats \(A[p]\) as \(A[0]\). If the median of \(p, r\) is \(s = q + 1\), then we shift the power of \(x\) down to \(q + 1 - p\).
We can evaluate $x^s$ in parallel (by D+C)

\[ \text{PPower}(x, k) \]

if $k == 1$ return $x$
else

\[ j = \lfloor k/2 \rfloor \]

in parallel do

\[ a = \text{PPower}(x, j) \]
\[ b = \text{PPower}(x, k - j) \]

return $a \times b$
end
Parallel Horner is $O(\log^2 n)$
Recursion depth of Parallel-Horner is $O(\log n)$
Takes $O(\log n)$ parallel steps to evaluate $x^n$
\[ P(x) = 3 + 5x + x^2 + 7x^3 \text{ at } x = 2 \]

Build recursive tree structure
Instantiate $x = 2$
Evaluate

\[ 13 + 60 = 73 \]

- \[ 3 + 10 = 13 \]
- \[ 5 \times 2 = 10 \]
- \[ 4 \times 15 = 60 \]
- \[ 1 + 14 = 15 \]
- \[ 7 \times 2 = 14 \]

\[ \text{Ans} = 73 \]