2019-04-29 Divide and Conquer

Monday, April 29, 2019 3:00 PM

General Idea

- Goal is to identify patterns of behavior in the data or process. If patterns exist in a predictable way, we can often break the problem into a series of sub problems, which themselves can be broken down
- Divide and conquer algorithms are recursive
- Candidates for divide and conquer would be:
  - If you notice yourself doing the same thing on subsets of the data
  - If the solution to a subsegment doesn't immediately impact adjacent subsegments
- Canonical divide and conquer algorithm: merge sort
  - Breaks data down into arrays of size 1 (which are implicitly sorted) and progressively merges them back together
    - From size 1, 2, 4, 8, 16….

Example #1: Matrix multiplication

- Consider the following two matrices:

$$\begin{matrix} a & b & c \\ d & e & f \\ h & i & j \end{matrix} \cdot \begin{matrix} k & l & m \\ n & o & p \\ q & r & s \end{matrix} = \begin{matrix} ak+bn+cq, & al+bo+cr, \ldots \\ dk+en+fq, & dl+eo+fr, \ldots \\ hk+in+jq, & hl+io+jr, \ldots \end{matrix}$$

$$A_{m \times n} \cdot B_{n \times p} = C_{m \times p}$$

$$\begin{matrix} A & B & C \end{matrix} \cdot \begin{matrix} D \\ E \\ F \end{matrix} = [AD+BE+CF]$$

$$1 \times 3 \qquad 3 \times 1$$

Recursive, Divide and Conquer Matrix Multiplication

- Simplest matrix is of size 1

$$[a_{11}] \cdot [b_{11}] = a_{11} b_{11}$$

- Next simplest is of size 2

size = 2x2 $\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \cdot \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21}, & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21}, & a_{21}b_{12} + a_{22}b_{22} \end{bmatrix}$

$$
\begin{matrix}
a_{ss} & a_{s,s+1} & a_{s,s+2} & \cdots & a_{s,e} \\
a_{s+1,s} & a_{s+1,s+1} & a_{s+1,s+2} & \cdots & a_{s+1,e} \\
\vdots \\
a_{e,s} & a_{e,s+1} & a_{e,s+2} & \cdots & a_{e,e}
\end{matrix}
$$

$$\text{recurse}(a_{s,e/2}; b_{s,e/2}) + \text{recurse}(a_{e/2,e}; b_{e,s})$$

Only works on matrices of size 2^i

- Each recursive step makes 8 recursive calls
- Each recursive call handles n/2 less data
- At each recursive call, we do N^2 each time

$$
\begin{matrix}
a_{11} & a_{12} & a_{13} & a_{14} \\
a_{21} & a_{22} & a_{23} & a_{24} \\
a_{31} & a_{32} & a_{33} & a_{34} \\
a_{41} & a_{42} & a_{43} & a_{44}
\end{matrix}
$$

$$T(n) = 8T\left(\frac{n}{2}\right) + n^2$$

$$n^{\log_2 8} = n^3$$

$$\Theta(n^3)$$

- Strassen's Method reduces the number of recursive calls from 8 to 7

$$T(n) = 7T\left(\frac{n}{2}\right) + n^2$$

$$\Theta(n^{\log_2 7}) \approx O(n^{2.81})$$

Maximum Subsequence Problem

- Given a series of positive and negative integers, find the subsequence that yields the maximum sum
    - e.g. 1 | 2 | 3 | 4 2 10 0 1 3 2 5 | 2 1 3

- Given a series of positive and negative integers, find the subsequence that yields the maximum sum
  - e.g. `1, 2, 3, -4, 2, -10, 0, 1, 3, -2, 5, -2, -1, 3`

3 3

5 -1

5