Carlos Mc Gregor c.mcgregormuro@mail.utoronto.ca

CSC373S: Algorithm Design, Analysis & Complexity

Tutorial 01

Monday January 16, 2017
based on notes by our TA [RW 110]

Divide & Conquer

Matrix Multiplication

Input: $n \times n$ matrices A, B, made of values $A = (i, j), B = (i, j)$ respectively, for $i, j = 1, 2, \ldots, n$.
Output: Matrix $C = A \times B$, such that $c_{i,j} = \sum_{k=1}^{n} a_{i,k} \cdot b_{k,j}$
Visually:

$$\begin{bmatrix} a_{1,1} & a_{1,2} & \ldots & a_{1,n} \\ & & & \\ & & & \\ & & & \end{bmatrix} \cdot \begin{bmatrix} b_{1,1} & & \\ b_{2,1} & & \\ \vdots & & \\ b_{n,1} & & \end{bmatrix} = \begin{bmatrix} a_{1,1} \cdot b_{1,1} + a_{1,2} \cdot b_{2,1} + \cdots + a_{1,n} \cdot b_{n,1} & & \\ & & \\ & & \\ & & \end{bmatrix}$$

<u>Algorithm 1 (Nave):</u>

```
def NaiveMatrixMult(A, B):
  n = A.rows
  Let C be a new n x n matrix

  for i = 1 to n:
    for j = 1 to n
      C[i,j] = 0
      for k = 1 to n
        # happens in O(1)
        C[i,j] = C[i,j] + A[i,k] * B[k,j]
```

<u>Runtime:</u> $\Theta(n^3)$

Using *Divide & Conquer*, we can divide each matrix into four submatrices:

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} C = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}$$

Thus, $C_{n \times n} = A_{n \times n} \cdot B_{n \times n}$, such that:

$$\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \cdot \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$$

This means that we get the following sub-operations:

$$C_{11} = A_{11} \cdot B_{11} + A_{12} \cdot B_{21} \; C_{12} = A_{11} \cdot B_{12} + A_{12} \cdot B_{22} \; C_{21} = A_{21} \cdot B_{11} + A_{22} \cdot B_{21}$$
$$C_{22} = A_{21} \cdot B_{12} + A_{22} \cdot B_{22}$$

Algorithm 2 (Divide & Conquer):

```
def BetterMatrixMult(A, B):
  n = A.rows
  let C be a new n * n matrix
  if n == 1:
    C[1,1] = A[1,1] * B[1,1]

  else:
    partition A, B and C into four submatrices each
    C_11 = BetterMatrixMult(A_11,B_11) + BetterMatrixMult(A_12,B_21)
    .
    .
    .
    C_22 = BetterMatrixMult(A_21,B_12) + BetterMatrixMult(A_22,B_22)
    build C again from submatrices C_11, C_12, C_21, C_22

  return C
```

Time Analysis:

$$T(n) = \begin{cases} \Theta(1), & \text{if } n = 1 \\ 8T(n/2) + \Theta(n^2), & \text{if } n > 1 \end{cases}$$

By the *Master Theorem*, this evaluates to $T(n) = \Theta(n^{\log_2 8}) = \Theta(n^3)$.

Strassen's Method

Input: $A_{n\times n}, B_{n\times n}, n = 2^m$
Output: $C_{n\times n}$

Approach:

1. Divide A, B, C into $n/2 \times n/2$ submatrices. $\Theta(1)$ by index calculations.

2. Create 10 matrices $S_1, S_2, \ldots, S_10$, each of which is $n/2 \times n/2$, with $\Theta(n^2)$.

$$
\begin{gathered}
S_1 = B_{12} - B_{22} \\
S_2 = A_{11} + A_{12} \\
S_3 = A_{21} + A_{22} \\
S_4 = B_{21} - B_{11} \\
S_5 = A_{11} + A_{22} \\
S_6 = B_{11} + B_{22} \\
S_7 = A_{12} - A_{22} \\
S_8 = B_{21} + B_{22} \\
S_9 = A_{11} - A_{21} \\
S_{10} = B_{11} + B_{12}
\end{gathered}
$$

3. Using the submatrices in Step 1 and 2, recursively compute 7 matrix products $p_1, \ldots, p_7$:

$$
\begin{gathered}
p_1 = A_{11} \cdot S_1 \\
p_2 = S_2 \cdot B_{22} \\
p_3 = S_3 \cdot B_{11} \\
p_4 = A_{22} \cdot S_4 \\
p_5 = S_5 \cdot S_6 \\
p_6 = S_7 \cdot S_8 \\
p_7 = S_9 \cdot S_{10}
\end{gathered}
$$