CS 594: Representations in Algorithm Design **Spring 2022**

Lecture (20): March 17 2022

Lecturer: Xiaorui Sun *Scribe: Manoj Kumar Alluri*

1 Last time and today

- **Last time :** Approximate Edit Distance
- **Today :** Algebraic Representation(Matrix Multiplication), Strassen algorithm, Algebraic Definitions

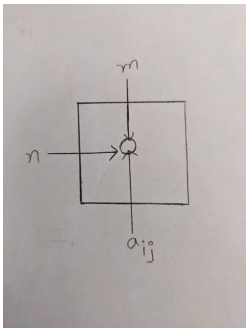
2 Algebraic Representation

In this we are going to introduce some algebraic structures.

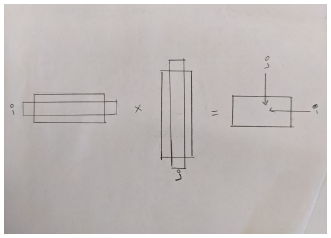
2.1 Matrix Multiplication

Assume A and B are two matrices defined over field.

- **Input :** Matrix A with dimensions of $n \times m$ and Matrix B with dimensions of $m \times p$ ($A \in \mathbb{F}^{n \times m}$ and $B \in \mathbb{F}^{m \times p}$)
- **output :** Matrix C with dimensions of $n \times p$ ($C \in \mathbb{F}^{n \times p}$)



- **Calculation Summary :** Let a_{ij} denotes the values at i^{th} row and j^{th} column. Output matrix C can be obtained by calculating value at each position by using this formula $c_{ij} = \sum_{k=1}^{m} \text{a}_{ik} \times b_{kj}$



It is computational problem as algorithm require to compute multiplication.Run time for this trivial algorithm is $O(n^3)$ where n=m=p in which there are total of n^2 entries and each entry to compute sum which is n.
The best Lower Bound time complexity is : $\Omega(n^2)$.
Generally people use omega : ω to represent best exponent in running time for matrix multiplication which is in range of $(2 \leq \omega \leq 3)$.
Now the question is that is it possible to solve matrix multiplication in $\omega(2+O(1))$ which is $\omega = 2.37...???$.For this we need to use tensor(a framework analyses long term matrix multiplications). Before starting with tensor we are going to study Strassen algorithm.

3 Strassen algorithm

- **Background :**

Strassen algorithm algorithm is a matrix multiplication algorithm in which it compute output matrix with run time of $O(n^{\log_2 7}) = O(n^{2.81...})$.
When we look at basic arithmetic operations, if these are applied to two numbers, then the running time is constant for two numbers. But if the operation is applied on matrices then situation will be different.
For example A + B (Matrix Addition) A+B : i^{th} row and j^{th} column in which a_{ij}+b_{ij}, running time is $O(n^2)$ and $\Omega(n^2)$.
Main observation is that even though it is basic with single operation it costs $O(n^2)$ and for matrices it costs $O(n^3)$.

- **Algorithm implementation :**

In generally addition and subtraction operations cost less compared to multiplication. So the idea of the algorithm is to use more addition and subtraction

operations than multiplication operations and to divide matrices into sub matrices and do addition and subtraction operations to get final output.
Let us consider a 2×2 matrices A and B :

$$A_{2\times2} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$$

$$B_{2\times2} = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$$

$$C_{2\times2} = A \times B = \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \end{bmatrix}$$

To get output matrix C we need to do 8 multiplications and 4 addition operations.
Question: Is it possible to compute matrix multiplication for 2×2 matrix with < 8 multiplications??? (Of course Yes we can do it).
Algorithm implementation is computed as:

$$\begin{aligned} P_1 &= (a_{11} + a_{22})(b_{11} + b_{22}) \\ P_2 &= (a_{21} + a_{22})b_{11} \\ P_3 &= a_{11}(b_{12} - b_{22}) \\ P_4 &= a_{22}(b_{21} - b_{11}) \\ P_5 &= (a_{11} + a_{12})b_{22} \\ P_6 &= (a_{21} - a_{11})(b_{11} + b_{12}) \\ P_7 &= (a_{12} - a_{22})(b_{21} + b_{22}) \end{aligned}$$

Now the output matrix C can be computed as follows :

$$\begin{aligned} C_{11} &= P_1 + P_4 - P_5 + P_7 \\ C_{12} &= P_3 + P_5 \\ C_{21} &= P_2 + P_4 \\ C_{22} &= P_1 - P_2 + P_3 + P_6 \end{aligned}$$

How many multiplications are used ?? - 7
How many additions or subtractions are used ?? - 18
Instead of using 8 multiplications we can use 7 multiplication operations and more additions.
To do matrix multiplication for $n \times n$ matrix where n= 2^k this can be done by using recursion. Divide the matrix into 4 sub matrices and apply the above algorithm.

Let A and B be the matrices with 4 sub-matrices as elements in 2×2 matrix.

$$A_{2\times2} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$$

$$B_{2\times2} = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

$$C_{2\times2} = A \times B = \begin{bmatrix} A_{11}B_{11} + A_{12}B_{21} & A_{11}B_{12} + A_{12}B_{22} \\ A_{21}B_{11} + A_{22}B_{21} & A_{21}B_{12} + A_{22}B_{22} \end{bmatrix}$$

Repeat the process recursively to get final output matrix.

- **Run time Calculation :** Let T(n) be the running time to calculate matrix multiplication for $n \times n$ matrices

$$T(n) = \begin{cases} C & \text{if } n = 1 \\ 7T(n/2) + 18\mathrm{O}(\mathrm{n}^2) & \text{if } n > 1 \end{cases}$$

By solving the above run time equation we get $T(n) = O(n^{\log_2 7})$.
For trivial algorithm the run time complexity is represented as :

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

Which on reduction equals to : $T(n) = O(n^3)$

4 Algebraic definitions

- **Quadratic Problem :**

Let variables $x_1, x_2, x_3, - - - -, x_n \in \mathbb{F}$ (Real numbers)

F = $\{f_1, f_2, - - -, f_k\}$ a set of quadratic function.

$$f_k = \sum_{i,j=1}^{N} \mathrm{t}_{\ i,j,k} \times x_i \times x_j$$

F = $(f_1, f_2, - - -, f_k)$ Goal is to compute F

- **Bi-linear Problem :**

  In this case we have two set of variables $x_1 - - - - x_n, y_1 - - - - y_m$ then our function becomes monomial such that

$$f_k = \sum_{i=1}^{N} \sum_{j=1}^{M} \text{t}_{\ i,j,k} \times x_i \times y_j$$

$$\{t_{i,j,k}\} i, j, k = 1 (3tensor)$$

  Goal is to treat Matrix Multiplication problem as a bi-linear problem. Construct a tensor such that all variables X together corresponds to entries of A and variables Y together corresponds to entries of B.

$$\text{F}=(\text{f}_1, f_2, - - - -, f_n) = \text{A} \times B$$

  Let variables A and B are:

$$A = \begin{pmatrix} x_{11} & x_{12} & - & - & x_{1n} \\ - & - & - & - & - \\ - & - & - & - & - \\ - & - & - & - & x_{nn} \end{pmatrix}$$

$$B = \begin{pmatrix} y_{11} & y_{12} & - & - & y_{1n} \\ - & - & - & - & - \\ - & - & - & - & - \\ - & - & - & - & y_{nn} \end{pmatrix}$$

$$Output = A \times B = \begin{pmatrix} z_{11} & z_{12} & - & - & z_{1n} \\ - & - & - & - & - \\ - & - & - & - & - \\ - & - & - & - & z_{nn} \end{pmatrix}$$

  Let $\text{f}_{(i,j)} = z_{ij}$ *and* $\text{f}_{(i,k)} = \sum_{t_{(i_0,j_0)(j_1,k_1)(i,k)}} \text{x}_{\ i_0,j_0} \times y_{j_1,k_1}$

  Question : Assume we have this variables at the end we want result of the function, how do we carefully set $f_{(i,k)} = \sum_{j=1}^{n} \text{x}_{\ i,j} \times y_{j,k}$

$$t_{(i_0,j_0)(j_1,k_1)(i,k)} = \begin{cases} 1 & \text{if } i_0 = i, j_0 = j, k_0 = k \\ 0 & \text{otherwise} \end{cases}$$

  Enumerate through all the steps we get a time complexity of $O(n^6)$.
  Rank of tensor => cost of solving Bi-linear problem.