Computational Complexity

1. We learnt about automaton

- DFA
  - Regular language
  - Regular Expressions
- PDA
  - CFLs
  - CFG
- Turing Machines

- Decidable problem
- Undecidable problems.

We will focus on

Decidable problems

Even when a problem is decidable if the problem requires an inordinate amount of time and memory, it may not be solvable in practice.

computational complexity is an investigation of time, memory or other resources required for solving computational problems.

Definition 7.1

Let M be a deterministic Turing Machine that halts on all inputs. Thus, the running time or time complexity of M is the function $f: \mathbb{N} \longrightarrow \mathbb{N}$, where $f(n)$ is the maximum number of steps that M uses on any input of length n.

If $f(n)$ is the running time of M, we say that M runs in time $f(n)$ and M is an $f(n)$ time Turing Machine.

$$f(n) = n + 8n + 3$$

$$f_1(n) = 2^n + 212n \log n + 5$$

Definition

Let f and g be functions, $f, g : \mathbb{N} \longrightarrow \mathbb{R}^+$. We say that $f(n) = O(g(n))$

$$\exists c > 0, n_0 \geq 0 : \forall n \geq n_0 : f(n) \leq c \cdot g(n)$$

When $f(n) = O(g(n))$, we say that $g(n)$ is an upper bound for $f(n)$

or

$g(n)$ is an asymptotic upper bound on $f(n)$

or

f is less than or equal to g if we disregard the differences up to constant factor

$f(n) = 2n^2 + n + 100$

$g(n) = n^2$

$f(n) = O(n^2)$

$f(n) = 5n^3 + 2n^2 + 11n + 5$

$g(n) = n^3$

$f(n) = O(n^3)$

Pick $c = 23$

$$23n^3 \geq 5n^3 + 2n^2 + 11n + 5$$

for $n \geq 1$

Little-o-notation

Let f and g be functions, $f, g : \mathbb{N} \longrightarrow \mathbb{R}^+$, say $f(n) = o(g(n))$ if

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0$$

In other words, $f(n) = o(g(n))$ means for any constant $c > 0$ a number n_0 exists s.t $f(n) < g(n) \quad \forall n \geq n_0$

we can classify languages based on their time requirements.

Definition

$$TIME(t(n)) = \{ L \mid L \text{ is a language decided by an } O(t(n)) \text{ time TM} \}$$

Class of P

$$P = \bigcup_k TIME(n^k)$$

The class of languages P contains all decision problems that can be solved in polynomial time on a deterministic single-tape turing machine.

All languages that can be decided in polynomial time is contained in this class.

Note that problems that can be solved in $O(n^{100})$ are in P, even though in practice, this is a practically difficult problem to solve.

$$PATH = \{\langle G, s, t\rangle \mid G \text{ is a directed graph with nodes } s \text{ and } t \text{ and there is a path from } s \text{ to } t\}$$

$PATH \in P$

Dynamic Programming

An efficient technique that can be used to solve problems that can be divided into smaller problems.

Matrix Multiplication

Given $M_1, M_2, \ldots, M_n$, compute the product $M_1 \cdot M_2 \cdot M_3 \cdots M_n$, where M_i has dimensions $d_{i-1} \times d_i$

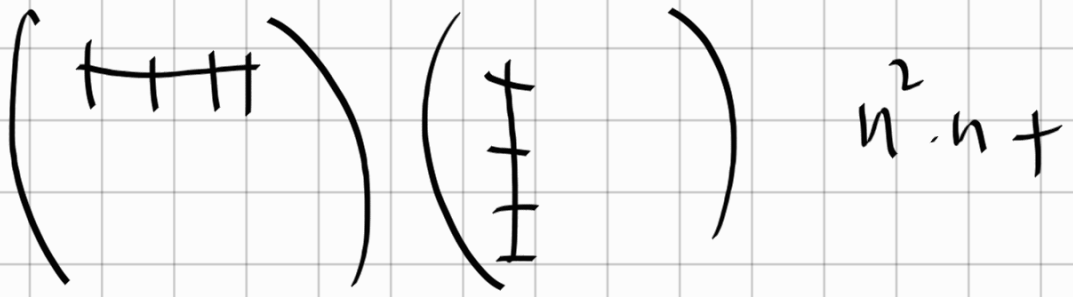
objective: Minimize the total scalar multiplications.

$$\underset{2\times 2}{\begin{pmatrix} a & b \\ c & d \end{pmatrix}} \begin{pmatrix} e \\ f \end{pmatrix} = \underset{2\times 1}{\begin{pmatrix} ae+bf \\ ce+df \end{pmatrix}}$$

Ex: $A_{n\times n} \cdot B_{n\times n} \cdot X_{n\times 1}$

$(A_{n\times n} \cdot B_{n\times n})_{n\times n} \cdot X_{n\times 1} \Rightarrow$

$A_{n\times n} \cdot (B_{n\times n} \cdot X_{n\times 1}) =$



$n^2 \cdot n +$

Rephrase problem!

Parenthesize the product $M_1 \cdot M_2 \cdots M_n$ in a way to minimize the # of scalar multiplications.

$$M_1 M_2 M_2 \cdots\cdots M_n$$

Ex! $M_1 - 20 \times 10$

$M_2 - 10 \times 50$

$M_3 - 50 \times 5$

$M_4 - 5 \times 30$

$M_1 \cdot M_2 \cdot M_3 \cdot M_4$

$((M_1 M_2)(M_3 M_4))$

$(M_1(M_2M_3)M_4)$

$$(M_1 M_2 M_3 \cdots M_k)(M_{k+1} \cdots M_n)$$

$P(n)$ = # of alternative parenthesizations of n matrices

$P(k)$ $P(n-k)$

$$P(n) = \begin{cases} 1 & \text{if } n = 1 \\ \sum_{k=1}^{n-1} P(k)\,P(n-k) & \text{if } n \geq 2 \end{cases}$$

$P(n)$ - n^{th} catalan number

$$P(n) = \frac{1}{n}\binom{2(n-1)}{n-1} \geq \frac{4^{n-1}}{2n^2-n} = \Omega\left(\frac{4^n}{n^2}\right)$$

$n = 10$; $P(n) \geq \Omega\left(\frac{4^{10}}{10^2}\right) \sim 10^4$

$n = 20$; $P(n) \geq \Omega\left(\frac{4^{20}}{20^2}\right) \sim 10^9$

Dynamic programming solution

- m[i,j] is the number of multiplications performed using optimal parenthesization of

$$M_i M_{i+1} \cdots M_j$$

$$M_i M_{i+1} \cdots M_k M_{k+1} \cdots M_j$$

$d_{i-1} \times d_i$ $\quad$ $d_{k-1} \times d_k$ $\quad$ $d_k \times d_{k+1}$ $\quad$ $d_{j-1} \times d_j$

$$\begin{cases} m[i,j] = \min\limits_k \{ m[i,k] + m[k+1,j] + d_{i-1} \times d_k \times d_j \\ m[i,i] = 0 \end{cases}$$

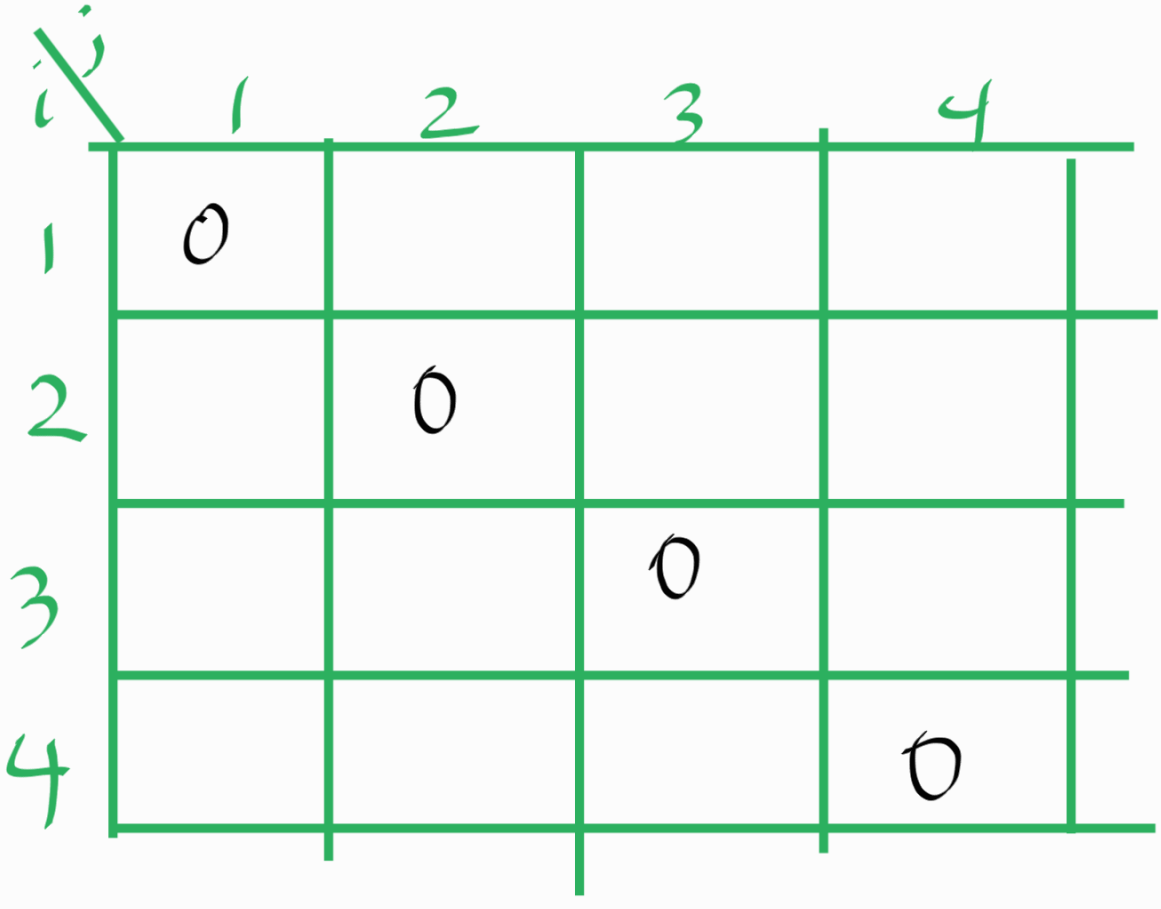
m[1,n] contains the optimal number of scalar multiplications for the product

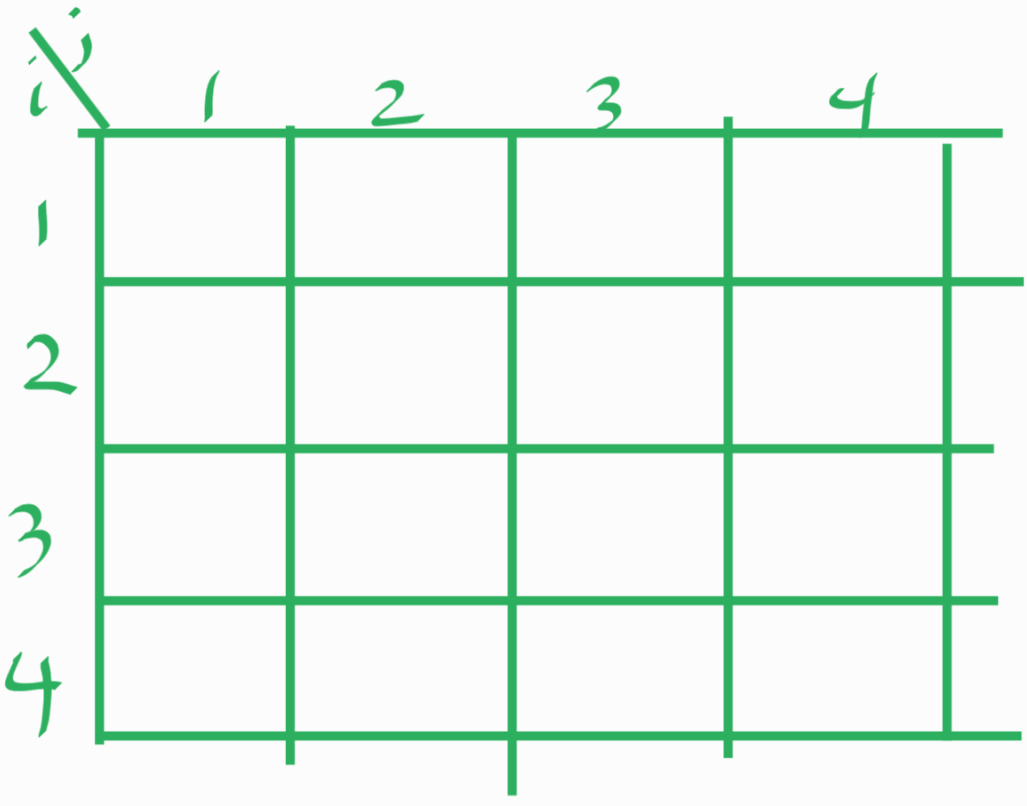
$M_1 - 20 \times 10$

$M_2 - 10 \times 50$

$M_3 - 50 \times 5$

$M_4 - 5 \times 30$





$M_1 - 20 \times 10$

$M_2 - 10 \times 50$

$M_3 - 50 \times 5$

$M_4 - 5 \times 30$

$$m[2,4] = \min \begin{cases} k=2 \\ m[2,2] + m[3,4] + d_{i-1} \times d_k \times d_j \\ 0 + 7500 + 10 \times 50 \times 30 \\ 7500 + 15000 \\ k=3 \quad m[2,3] + m[4,4] + d_{i-1} \times d_k \times d_j \\ 2500 + 0 + 10 \times 5 \times 30 \\ 2500 + 1500 \\ 4000 \end{cases}$$