CIBB 2019

Modification of Valiant's Parsing Algorithm for String-Searching Problem

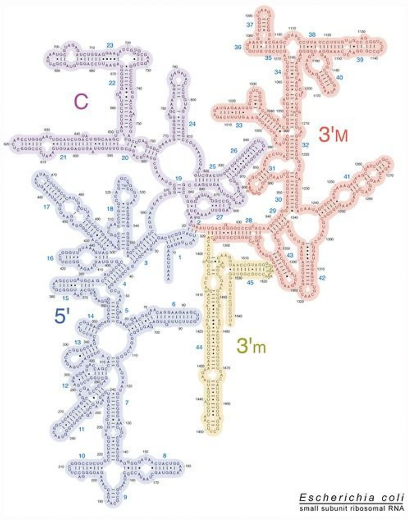
Yuliya Susanina, Semyon Grigorev, Anna Yaveyn

JetBrains Research, Programming Languages and Tools Lab
Saint Petersburg University

September 6, 2019

RNA Analysis

- RNA secondary structure prediction
- Applications: classification and recognition problems
- **String-searching problem**



Escherichia coli
small subunit ribosomal RNA

Formal Grammars and Languages

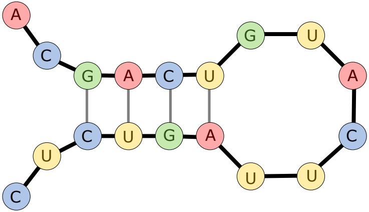
- $G = (\Sigma, N, R, s)$ — context-free grammar (CFG) in Chomsky normal form
  - $a \to bc$, where $a, b, c \in N$
  - $a \to A$, where $a \in N, A \in \Sigma$
  - $s \to \varepsilon$, where ε is an empty string
- $L_G(s) = \{\omega \mid s \Rightarrow^* \omega\}$, where $\omega \in \Sigma^*$
- Parsing — does ω belong to $L_G(s)$?

CFG-based Approach

- RNA sequences are treated as strings over $\Sigma = \{A, G, C, U\}$

  `CACGACUGUACUUAGUCUC...CUGGAUCACCUCCUU`

- CFG describes RNA secondary structure features



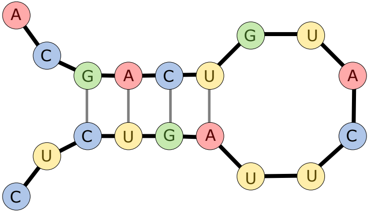
```
s: stem<s0>
s0: G U A C U U
stem1<s1>:
  A s1 U | U s1 A | C s1 G | G s1 C
stem<s1>:
  A stem<s1> U
  | U stem<s1> A
  | C stem<s1> G
  | G stem<s1> C
  | stem1<s1>
```

CFG-based Approach

- RNA sequences are treated as strings over $\Sigma = \{A, G, C, U\}$

  CACGACUGUACUUAGUCUC...CUGGAUCACCUCCUU

- CFG describes RNA secondary structure features



```
s: stem<s0>
s0: G U A C U U
stem1<s1>:
  A s1 U | U s1 A | C s1 G | G s1 C
stem<s1>:
  A stem<s1> U
  | U stem<s1> A
  | C stem<s1> G
  | G stem<s1> C
  | stem1<s1>
```

- Parsing as method to find all substrings with specific secondary structure features
- **String-searching problem:** for input string of length $n = 2^p - 1$ find all substrings of length m which belong to $L_G(s)$

Problems

$$\left.\begin{array}{l}\bullet\ \text{Long sequences}\\ \bullet\ \text{Large amount of data}\\ \bullet\ \text{Complex models}\end{array}\right\}\text{computational complexity}$$

$\Longrightarrow$ development of efficient parsing algorithms

Tabular Parsing Algorithms

- Input:
  - Grammar $G = (\Sigma, N, R, s)$ in Chomsky normal form
  - String $\omega = \omega_1\omega_2 \dots \omega_n$, $\omega_i \in \Sigma$
- Parsing table T:
  - $T_{i,j} = \{a \mid a \in N, \omega_{i+1} \dots \omega_j \in L_G(a)\} \quad \forall i < j$
  - $\omega \in L_G(s) \iff s \in T_{0,n}$

Tabular Parsing Algorithms

- Input:
  - Grammar $G = (\Sigma, N, R, s)$ in Chomsky normal form
  - String $\omega = \omega_1\omega_2 \dots \omega_n$, $\omega_i \in \Sigma$
- Parsing table T:
  - $T_{i,j} = \{a \mid a \in N, \omega_{i+1} \dots \omega_j \in L_G(a)\} \quad \forall i < j$
  - $\omega \in L_G(s) \iff s \in T_{0,n}$
- Process of filling:
  - $T_{i-1,i} = \{a \mid a \to \omega_i \in R\}$
  - $T_{i,j} = f(P_{i,j})$, where $P_{i,j} = \bigcup\limits_{k=i+1}^{j-1} T_{i,k} \times T_{k,j}$

$$f(P_{i,j}) = \{a \mid \exists a \to bc \in R : (b, c) \in P_{i,j}\}$$

Tabular Parsing Algorithms

- Input:
  - Grammar $G = (\Sigma, N, R, s)$ in Chomsky normal form
  - String $\omega = \omega_1\omega_2 \dots \omega_n$, $\omega_i \in \Sigma$
- Parsing table T:
  - $T_{i,j} = \{a \mid a \in N, \omega_{i+1} \dots \omega_j \in L_G(a)\} \quad \forall i < j$
  - $\omega \in L_G(s) \iff s \in T_{0,n}$
- Process of filling:
  - $T_{i-1,i} = \{a \mid a \to \omega_i \in R\}$
  - $T_{i,j} = f(P_{i,j})$, where $P_{i,j} = \boxed{\bigcup_{k=i+1}^{j-1} T_{i,k} \times T_{k,j}}$

    $f(P_{i,j}) = \{a \mid \exists a \to bc \in R : (b, c) \in P_{i,j}\}$

To Valiant's Parsing Algorithm

CYK: $\mathcal{O}(|G|n^3)$

Younger, D. H. "Context-free language processing in time n^3" 1966

$\Downarrow$

Reduction to matrix multiplication

$\Downarrow$

Reduction to Boolean matrix multiplication

$\Downarrow$

Valiant: $\mathcal{O}(|G|BMM(n)log(n))$

Valiant, L. G. "General context-free recognition in less than cubic time" 1975

To Valiant's Parsing Algorithm

CYK: $\mathcal{O}(|G|n^3)$

Younger, D. H. "Context-free language processing in time n^3" 1966

$\Downarrow$

Reduction to matrix multiplication

$\Downarrow$

Reduction to Boolean matrix multiplication

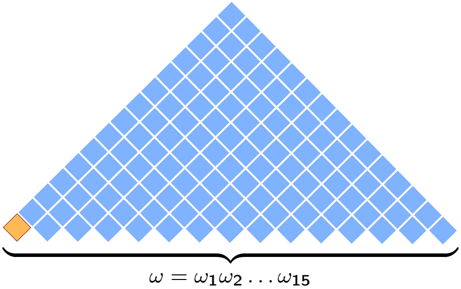
$\Downarrow$

Valiant: $\mathcal{O}(|G|BMM(n)log(n))$

Valiant, L. G. "General context-free recognition in less than cubic time" 1975

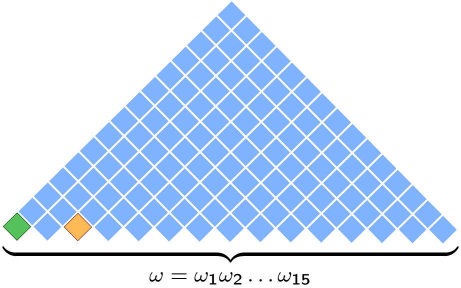
Valiant's Algorithm

- (+):
  - Utilization of parallel techniques and highly-efficient libraries
  - Generalization to more powerful classes of formal grammars: conjunctive and Boolean
- (−):
  - Not suitable for string-searching problem
    It is necessary to calculate at least 2 triangle submatrices of size $\frac{n}{2}$



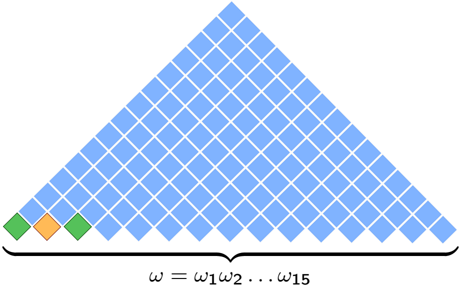
Valiant's Algorithm

- (+):
  - Utilization of parallel techniques and highly-efficient libraries
  - Generalization to more powerful classes of formal grammars: conjunctive and Boolean
- (−):
  - Not suitable for string-searching problem
    It is necessary to calculate at least 2 triangle submatrices of size $\frac{n}{2}$



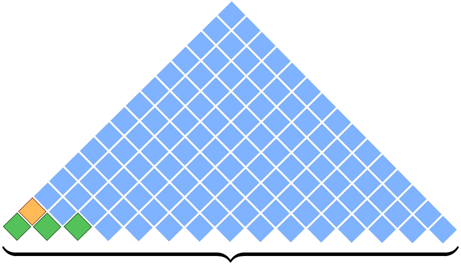
Valiant's Algorithm

- (+):
  - Utilization of parallel techniques and highly-efficient libraries
  - Generalization to more powerful classes of formal grammars: conjunctive and Boolean
- (−):
  - Not suitable for string-searching problem
    It is necessary to calculate at least 2 triangle submatrices of size $\frac{n}{2}$



Valiant's Algorithm

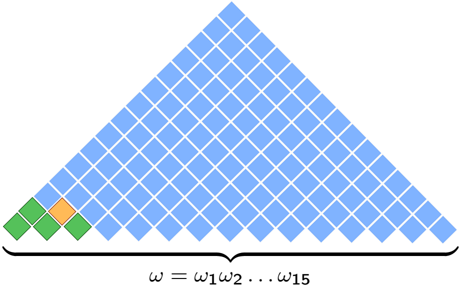
- (+):
  - Utilization of parallel techniques and highly-efficient libraries
  - Generalization to more powerful classes of formal grammars: conjunctive and Boolean
- (−):
  - Not suitable for string-searching problem
    It is necessary to calculate at least 2 triangle submatrices of size $\frac{n}{2}$



$\omega = \omega_1 \omega_2 \dots \omega_{15}$

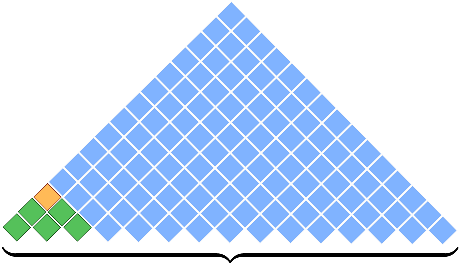
Valiant's Algorithm

- (+):
  - Utilization of parallel techniques and highly-efficient libraries
  - Generalization to more powerful classes of formal grammars: conjunctive and Boolean
- (−):
  - Not suitable for string-searching problem
    It is necessary to calculate at least 2 triangle submatrices of size $\frac{n}{2}$



Valiant's Algorithm

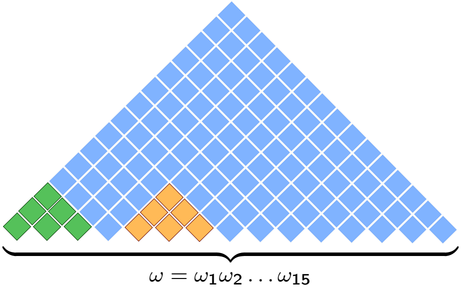
- (+):
  - Utilization of parallel techniques and highly-efficient libraries
  - Generalization to more powerful classes of formal grammars: conjunctive and Boolean
- (−):
  - Not suitable for string-searching problem
    It is necessary to calculate at least 2 triangle submatrices of size $\frac{n}{2}$



$\omega = \omega_1\omega_2 \dots \omega_{15}$

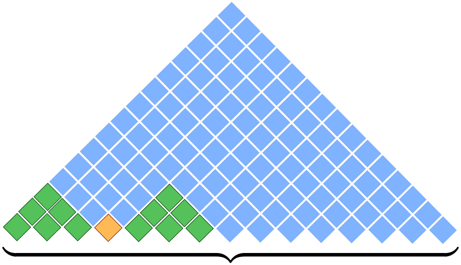
Valiant's Algorithm

- (+):
  - Utilization of parallel techniques and highly-efficient libraries
  - Generalization to more powerful classes of formal grammars: conjunctive and Boolean
- (−):
  - Not suitable for string-searching problem
    It is necessary to calculate at least 2 triangle submatrices of size $\frac{n}{2}$



Valiant's Algorithm

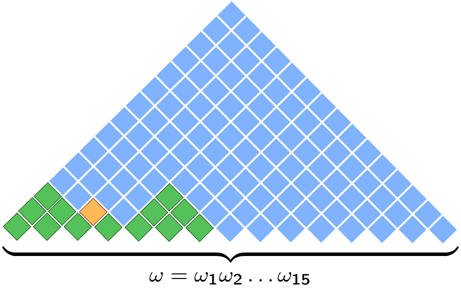
- (+):
  - Utilization of parallel techniques and highly-efficient libraries
  - Generalization to more powerful classes of formal grammars: conjunctive and Boolean
- (−):
  - Not suitable for string-searching problem
    It is necessary to calculate at least 2 triangle submatrices of size $\frac{n}{2}$



$\omega = \omega_1\omega_2\dots\omega_{15}$

Valiant's Algorithm

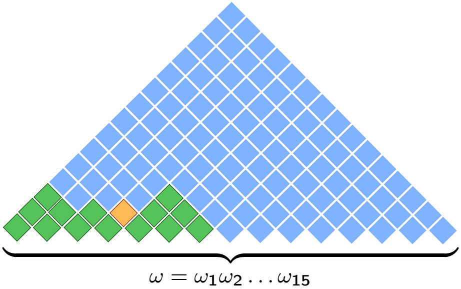
- (+):
  - Utilization of parallel techniques and highly-efficient libraries
  - Generalization to more powerful classes of formal grammars: conjunctive and Boolean
- (−):
  - Not suitable for string-searching problem
    It is necessary to calculate at least 2 triangle submatrices of size $\frac{n}{2}$



$\omega = \omega_1\omega_2\ldots\omega_{15}$

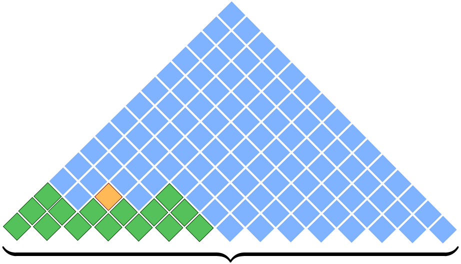
Valiant's Algorithm

- (+):
  - Utilization of parallel techniques and highly-efficient libraries
  - Generalization to more powerful classes of formal grammars: conjunctive and Boolean
- (−):
  - Not suitable for string-searching problem
    It is necessary to calculate at least 2 triangle submatrices of size $\frac{n}{2}$



Valiant's Algorithm

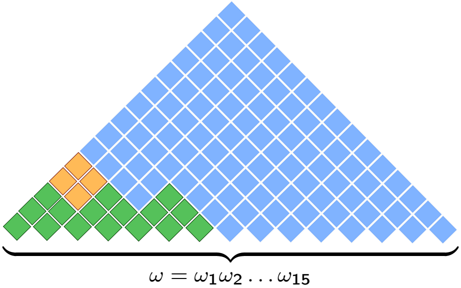
- (+):
  - Utilization of parallel techniques and highly-efficient libraries
  - Generalization to more powerful classes of formal grammars: conjunctive and Boolean
- (−):
  - Not suitable for string-searching problem
    It is necessary to calculate at least 2 triangle submatrices of size $\frac{n}{2}$



$\omega = \omega_1\omega_2\ldots\omega_{15}$

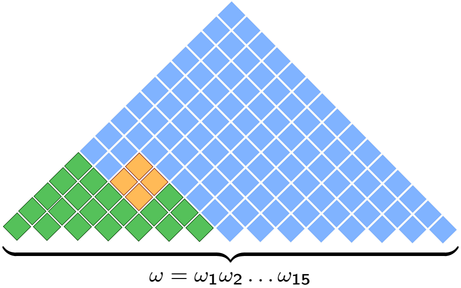
Valiant's Algorithm

- (+):
  - Utilization of parallel techniques and highly-efficient libraries
  - Generalization to more powerful classes of formal grammars: conjunctive and Boolean
- (−):
  - Not suitable for string-searching problem
    It is necessary to calculate at least 2 triangle submatrices of size $\frac{n}{2}$



Valiant's Algorithm

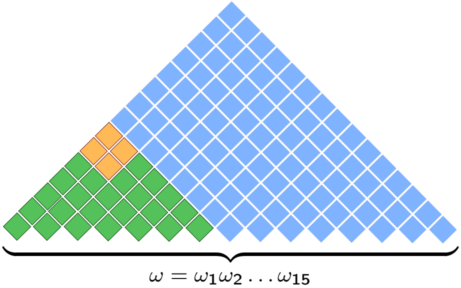
- (+):
  - Utilization of parallel techniques and highly-efficient libraries
  - Generalization to more powerful classes of formal grammars: conjunctive and Boolean
- (−):
  - Not suitable for string-searching problem
    It is necessary to calculate at least 2 triangle submatrices of size $\frac{n}{2}$



$\underbrace{\qquad\qquad}_{\omega = \omega_1\omega_2\ldots\omega_{15}}$

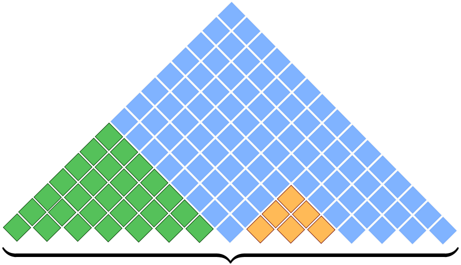
Valiant's Algorithm

- (+):
  - Utilization of parallel techniques and highly-efficient libraries
  - Generalization to more powerful classes of formal grammars: conjunctive and Boolean
- (−):
  - Not suitable for string-searching problem
    It is necessary to calculate at least 2 triangle submatrices of size $\frac{n}{2}$



Valiant's Algorithm

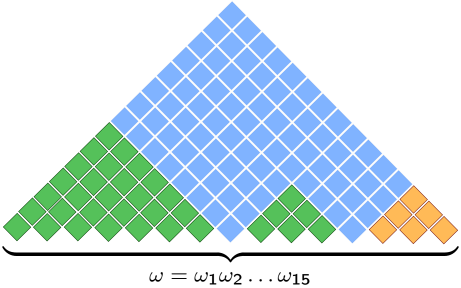
- (+):
  - Utilization of parallel techniques and highly-efficient libraries
  - Generalization to more powerful classes of formal grammars: conjunctive and Boolean
- (−):
  - Not suitable for string-searching problem
    It is necessary to calculate at least 2 triangle submatrices of size $\frac{n}{2}$



$\omega = \omega_1\omega_2 \ldots \omega_{15}$

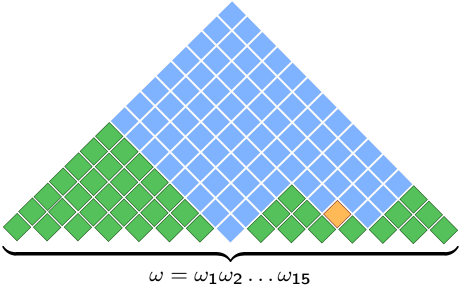
Valiant's Algorithm

- (+):
  - Utilization of parallel techniques and highly-efficient libraries
  - Generalization to more powerful classes of formal grammars: conjunctive and Boolean
- (−):
  - Not suitable for string-searching problem
    It is necessary to calculate at least 2 triangle submatrices of size $\frac{n}{2}$



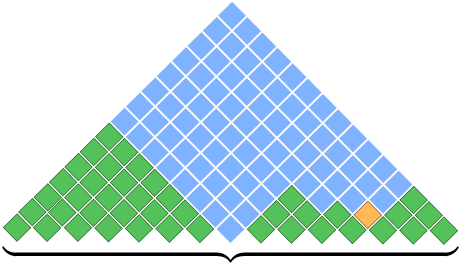
Valiant's Algorithm

- (+):
  - Utilization of parallel techniques and highly-efficient libraries
  - Generalization to more powerful classes of formal grammars: conjunctive and Boolean
- (−):
  - Not suitable for string-searching problem
    It is necessary to calculate at least 2 triangle submatrices of size $\frac{n}{2}$



Valiant's Algorithm

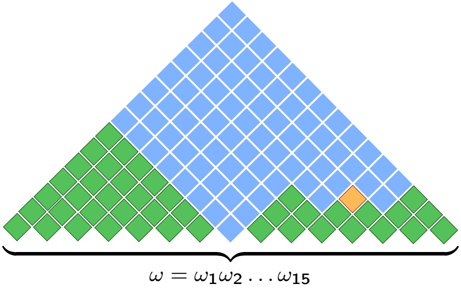
- (+):
  - Utilization of parallel techniques and highly-efficient libraries
  - Generalization to more powerful classes of formal grammars: conjunctive and Boolean
- (−):
  - Not suitable for string-searching problem
    It is necessary to calculate at least 2 triangle submatrices of size $\frac{n}{2}$



$\omega = \omega_1\omega_2 \ldots \omega_{15}$

Valiant's Algorithm

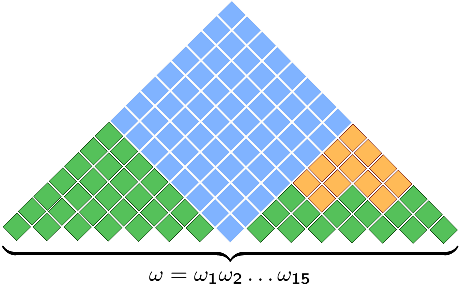
- (+):
  - Utilization of parallel techniques and highly-efficient libraries
  - Generalization to more powerful classes of formal grammars: conjunctive and Boolean
- (−):
  - Not suitable for string-searching problem
    It is necessary to calculate at least 2 triangle submatrices of size $\frac{n}{2}$



$\omega = \omega_1\omega_2 \ldots \omega_{15}$

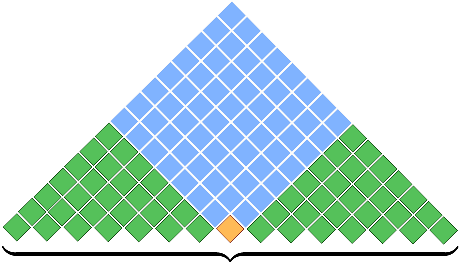
Valiant's Algorithm

- (+):
  - Utilization of parallel techniques and highly-efficient libraries
  - Generalization to more powerful classes of formal grammars: conjunctive and Boolean
- (−):
  - Not suitable for string-searching problem
    It is necessary to calculate at least 2 triangle submatrices of size $\frac{n}{2}$



Valiant's Algorithm

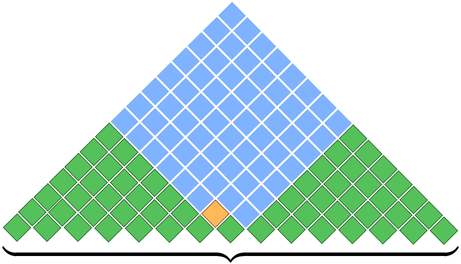
- (+):
  - Utilization of parallel techniques and highly-efficient libraries
  - Generalization to more powerful classes of formal grammars: conjunctive and Boolean
- (−):
  - Not suitable for string-searching problem
    It is necessary to calculate at least 2 triangle submatrices of size $\frac{n}{2}$



$\omega = \omega_1\omega_2\ldots\omega_{15}$

Valiant's Algorithm

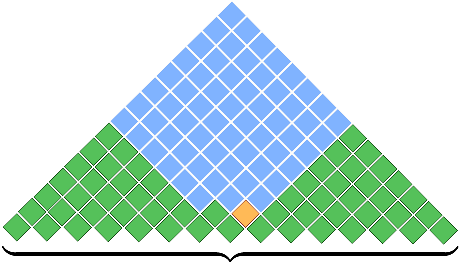
- (+):
  - Utilization of parallel techniques and highly-efficient libraries
  - Generalization to more powerful classes of formal grammars: conjunctive and Boolean
- (−):
  - Not suitable for string-searching problem
    It is necessary to calculate at least 2 triangle submatrices of size $\frac{n}{2}$



$\omega = \omega_1\omega_2 \dots \omega_{15}$

Valiant's Algorithm

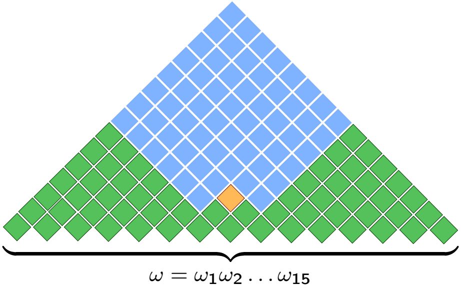
- (+):
  - Utilization of parallel techniques and highly-efficient libraries
  - Generalization to more powerful classes of formal grammars: conjunctive and Boolean
- (−):
  - Not suitable for string-searching problem
    It is necessary to calculate at least 2 triangle submatrices of size $\frac{n}{2}$



$\omega = \omega_1\omega_2\ldots\omega_{15}$

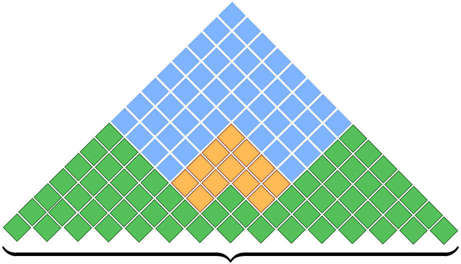
Valiant's Algorithm

- (+):
  - Utilization of parallel techniques and highly-efficient libraries
  - Generalization to more powerful classes of formal grammars: conjunctive and Boolean
- (−):
  - Not suitable for string-searching problem
    It is necessary to calculate at least 2 triangle submatrices of size $\frac{n}{2}$



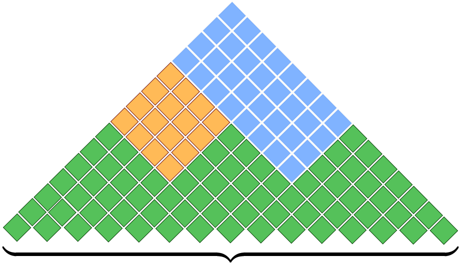
Valiant's Algorithm

- (+):
  - Utilization of parallel techniques and highly-efficient libraries
  - Generalization to more powerful classes of formal grammars: conjunctive and Boolean
- (−):
  - Not suitable for string-searching problem
    It is necessary to calculate at least 2 triangle submatrices of size $\frac{n}{2}$



Valiant's Algorithm

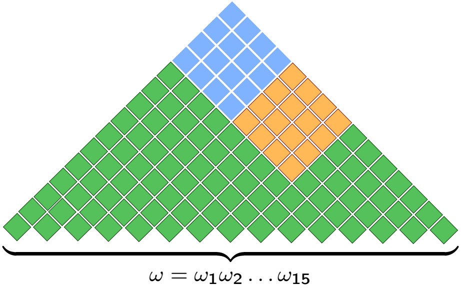
- (+):
  - Utilization of parallel techniques and highly-efficient libraries
  - Generalization to more powerful classes of formal grammars: conjunctive and Boolean
- (−):
  - Not suitable for string-searching problem
    It is necessary to calculate at least 2 triangle submatrices of size $\frac{n}{2}$



$$\omega = \omega_1\omega_2\dots\omega_{15}$$

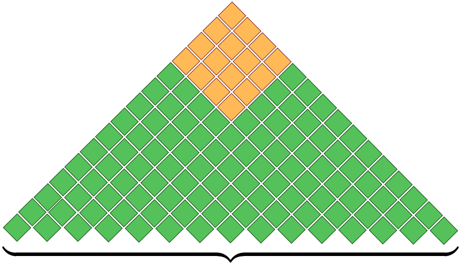
Valiant's Algorithm

- (+):
  - Utilization of parallel techniques and highly-efficient libraries
  - Generalization to more powerful classes of formal grammars: conjunctive and Boolean
- (−):
  - Not suitable for string-searching problem
    It is necessary to calculate at least 2 triangle submatrices of size $\frac{n}{2}$



Valiant's Algorithm

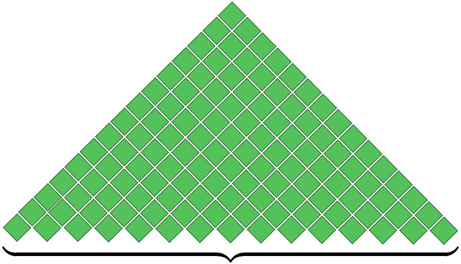
- (+):
  - Utilization of parallel techniques and highly-efficient libraries
  - Generalization to more powerful classes of formal grammars: conjunctive and Boolean
- (−):
  - Not suitable for string-searching problem
    It is necessary to calculate at least 2 triangle submatrices of size $\frac{n}{2}$



$\omega = \omega_1\omega_2 \dots \omega_{15}$

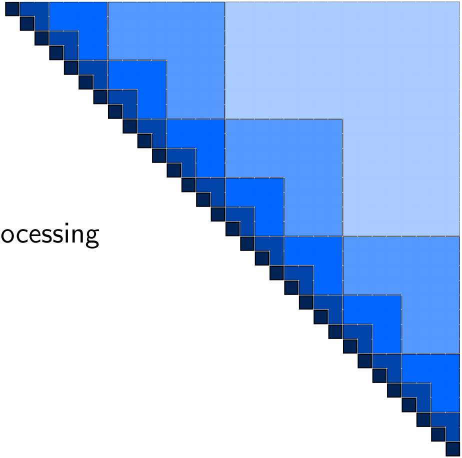
Valiant's Algorithm

- (+):
  - Utilization of parallel techniques and highly-efficient libraries
  - Generalization to more powerful classes of formal grammars: conjunctive and Boolean
- (−):
  - Not suitable for string-searching problem
    It is necessary to calculate at least 2 triangle submatrices of size $\frac{n}{2}$



$\omega = \omega_1\omega_2 \dots \omega_{15}$

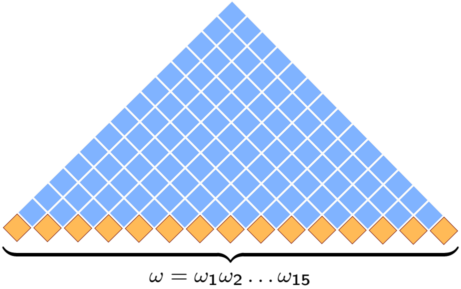
Layered Submatrices Processing (1)



- Rearranging the submatrices processing
- Division the parsing table into layers of disjoint submatrices

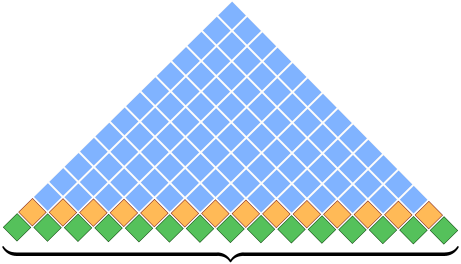
Layered Submatrices Processing (2)

- Each matrix in the layer can be handled independently
- Increasing the lever of parallelism:
  - Matrix multiplication
  - Each matrix in layer
  - Each pair of nonterminals



Layered Submatrices Processing (2)

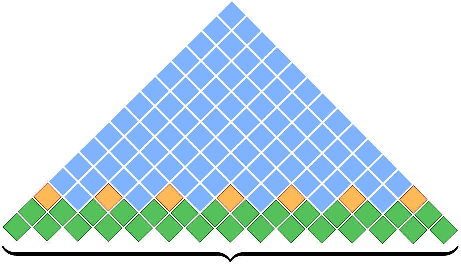
- Each matrix in the layer can be handled independently
- Increasing the lever of parallelism:
  - Matrix multiplication
  - Each matrix in layer
  - Each pair of nonterminals



$\omega = \omega_1\omega_2 \ldots \omega_{15}$

Layered Submatrices Processing (2)

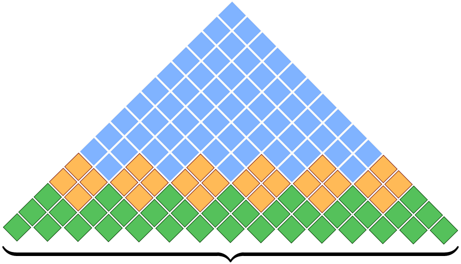
- Each matrix in the layer can be handled independently
- Increasing the lever of parallelism:
  - Matrix multiplication
  - Each matrix in layer
  - Each pair of nonterminals



$\omega = \omega_1\omega_2\ldots\omega_{15}$

Layered Submatrices Processing (2)

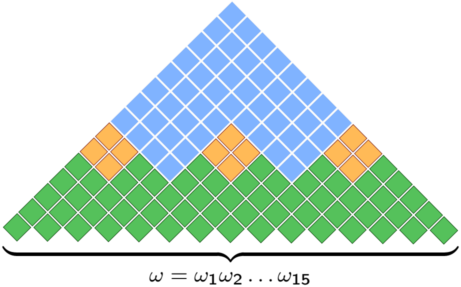
- Each matrix in the layer can be handled independently
- Increasing the lever of parallelism:
  - Matrix multiplication
  - Each matrix in layer
  - Each pair of nonterminals



$\omega = \omega_1\omega_2 \dots \omega_{15}$

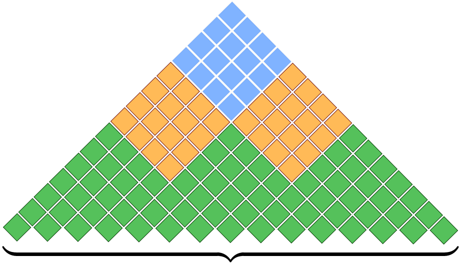
Layered Submatrices Processing (2)

- Each matrix in the layer can be handled independently
- Increasing the lever of parallelism:
  - Matrix multiplication
  - Each matrix in layer
  - Each pair of nonterminals


$\omega = \omega_1\omega_2 \ldots \omega_{15}$

Layered Submatrices Processing (2)

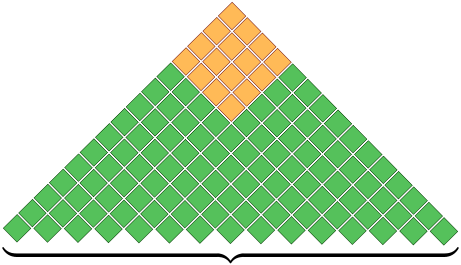
- Each matrix in the layer can be handled independently
- Increasing the lever of parallelism:
  - Matrix multiplication
  - Each matrix in layer
  - Each pair of nonterminals



$\omega = \omega_1\omega_2 \dots \omega_{15}$

Layered Submatrices Processing (2)

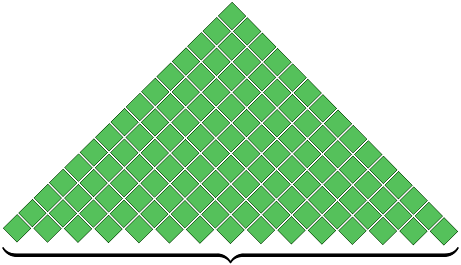
- Each matrix in the layer can be handled independently
- Increasing the lever of parallelism:
    - Matrix multiplication
    - Each matrix in layer
    - Each pair of nonterminals



$\omega = \omega_1\omega_2 \ldots \omega_{15}$

Layered Submatrices Processing (2)

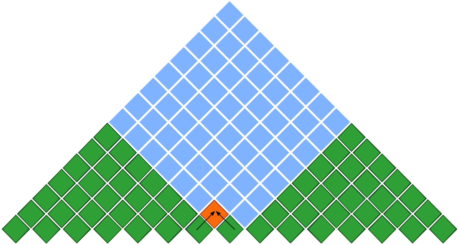
- Each matrix in the layer can be handled independently
- Increasing the lever of parallelism:
  - Matrix multiplication
  - Each matrix in layer
  - Each pair of nonterminals



$\omega = \omega_1\omega_2 \ldots \omega_{15}$

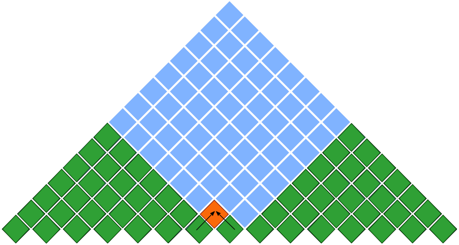
Application for the String-Searching Problem

- **Problem:** for input string of length $n = 2^p - 1$ find all substrings of length m which belong to $L_G(s)$
- **Valiant's algorithm:** it is necessary to calculate at least 2 triangle submatrices of size $\frac{n}{2}$
  Time complexity: $\mathcal{O}(|G|BMM(2^{p-1})(p-2))$



Application for the String-Searching Problem

- **Problem:** for input string of length $n = 2^p - 1$ find all substrings of length m which belong to $L_G(s)$
- **Valiant's algorithm:** it is necessary to calculate at least 2 triangle submatrices of size $\frac{n}{2}$
  Time complexity: $\mathcal{O}(|G|BMM(2^{p-1})(p-2))$



- **Modification:** it is necessary to compute layers with submatrices of size not greater than 2^r, where $2^{r-2} < m \leq 2^{r-1}$
  Time complexity: $\mathcal{O}(|G|2^{2(p-r)-1}BMM(2^r)(r-1))$

Evaluation

- Implementation
  - CPU-based: M4RI library
  - GPU-based: CUDA C
- Grammars
  - D_2:

    ```
    s: s s | ( s ) | [ s ] | ε
    ```

  - *BIO*:

    ```
    s: stem<s0>
    any_str: any_smb*[2..10]
    s0: any_str | any_str stem<s0> s0
    any_smb: A | U | C | G
    stem1<s1>: A s1 U | G s1 C | U s1 A | C s1 G
    stem2<s1>: stem1<stem1<s1>>
    stem<s1>:
          A stem<s1> U
        | U stem<s1> A
        | C stem<s1> G
        | G stem<s1> C
        | stem1<stem2<s1>>
    ```

Data Preprocessing

- Example: n = 15, m = 6

$$\underbrace{\texttt{AAGCUU AAGCUU AAGCUU}}_{\text{len} = 12}$$

Data Preprocessing

- Example: n = 15, m = 6

$$\underbrace{\texttt{AAGCUU AAGCUU AAGCUU }}_{\text{len} = 12}$$

$$\underbrace{\texttt{AAGCUUGAAGCUUGAAGCUUG}}_{\text{len} = 15}$$

Results: Comparative Analysis

n	Time (sec)							
	Grammar D_2				Grammar *BIO*			
	valCPU	modCPU	valGPU	modGPU	valCPU	modCPU	valGPU	modGPU
127	0.08	0.08	0.20	0.10	1.35	1.34	0.19	0.10
255	0.28	0.30	0.52	0.13	5.40	5.50	0.53	0.14
511	1.21	1.18	1.90	0.25	21.97	22.35	1.99	0.26
1023	4.90	4.78	7.88	0.54	88.70	90.32	7.89	0.60
2047	19.61	19.38	33.50	1.50	363.32	374.20	34.01	1.70
4095	78.36	78.28	140.47	4.45	1467.68	1480.59	141.10	5.47
8191	315.67	315.08	-	13.65	-	-	-	18.04

Results: Comparative Analysis

n	Time (sec)							
	Grammar D_2				Grammar *BIO*			
	valCPU	modCPU	valGPU	modGPU	valCPU	modCPU	valGPU	modGPU
127	0.08	0.08	0.20	0.10	1.35	1.34	0.19	0.10
255	0.28	0.30	0.52	0.13	5.40	5.50	0.53	0.14
511	1.21	1.18	1.90	0.25	21.97	22.35	1.99	0.26
1023	4.90	4.78	7.88	0.54	88.70	90.32	7.89	0.60
2047	19.61	19.38	33.50	1.50	363.32	374.20	34.01	1.70
4095	78.36	78.28	140.47	4.45	1467.68	1480.59	141.10	5.47
8191	315.67	315.08	-	13.65	-	-	-	18.04

Results: Comparative Analysis

n	Time (sec)							
	Grammar D_2				Grammar *BIO*			
	valCPU	modCPU	valGPU	modGPU	valCPU	modCPU	valGPU	modGPU
127	0.08	0.08	0.20	0.10	1.35	1.34	0.19	0.10
255	0.28	0.30	0.52	0.13	5.40	5.50	0.53	0.14
511	1.21	1.18	1.90	0.25	21.97	22.35	1.99	0.26
1023	4.90	4.78	7.88	0.54	88.70	90.32	7.89	0.60
2047	19.61	19.38	33.50	1.50	363.32	374.20	34.01	1.70
4095	78.36	78.28	140.47	4.45	1467.68	1480.59	141.10	5.47
8191	315.67	315.08	-	13.65	-	-	-	18.04

Results: String-searching Problem

m	n	Time (sec)			
		valCPU	modCPU	valGPU	modGPU
250	1023	4.90	3.00	7.88	0.24
	2047	19.61	6.65	33.50	0.26
	4095	78.36	13.83	140.47	0.32
	8191	315.67	28.90	-	0.46
510	2047	19.61	12.18	33.50	0.58
	4095	78.36	26.58	140.47	0.65
	8191	315.67	56.70	-	0.88
1020	4095	78.36	48.31	140.47	1.59
	8191	315.67	108.38	-	1.95
2040	8191	315.67	197.32	-	5.10

Conclusion

- The modification of Valiant's algorithm was proposed
  - Layered submatrices processing
  - Effective utilization of parallel techniques and GPGPU
- The modification is applicable to the string-searching problem

Future Research

- Improvement of the existing implementation
- Evaluation on real-world data
- Extension for more expressive classes of formal languages (conjunctive, Boolean)

Contact Information

- Semyon Grigorev: semyon.grigorev@jetbrains.com
- Yuliya Susanina: jsusanina@gmail.com
- Anna Yaveyn: anya.ayveyn@yandex.ru

Thanks!