12.7. The Four-Russians speedup

2016.04.26

조현진

Index

- Four-Russians speedup concept
- Block Edit Distance Algorithm
- Four-Russians Edit Distance Algorithm

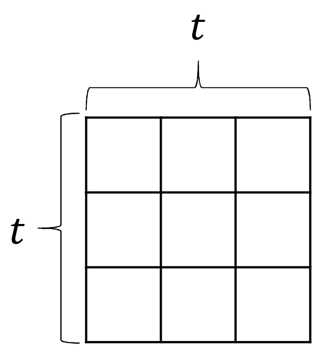
Definitions

- S_1 : String 1
- S_2 : String 2
- n : both length of S_1 and S_2
- i : character position of S_1 $(0 \le i \le n)$
- j : character position of S_2 $(0 \le j \le n)$
- $S_1(i)$: i'th character in S_1
- $S_2(j)$: j'th character in S_2
- σ : $alphabet \quad size \quad (Ex.\ \sigma\{A, C, G, T\} = 4\ )$
- $D(i, j)$: Edit Distance at (i, j)

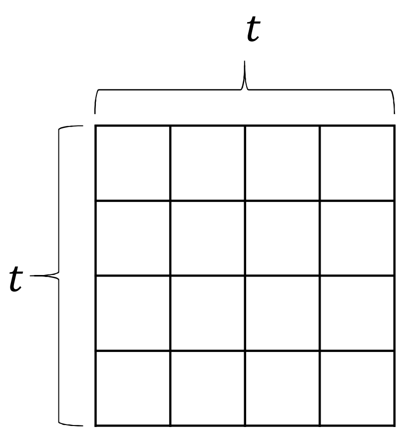
t -block

Definition

A ***t*-block** is a t by t square in the dynamic programming table.

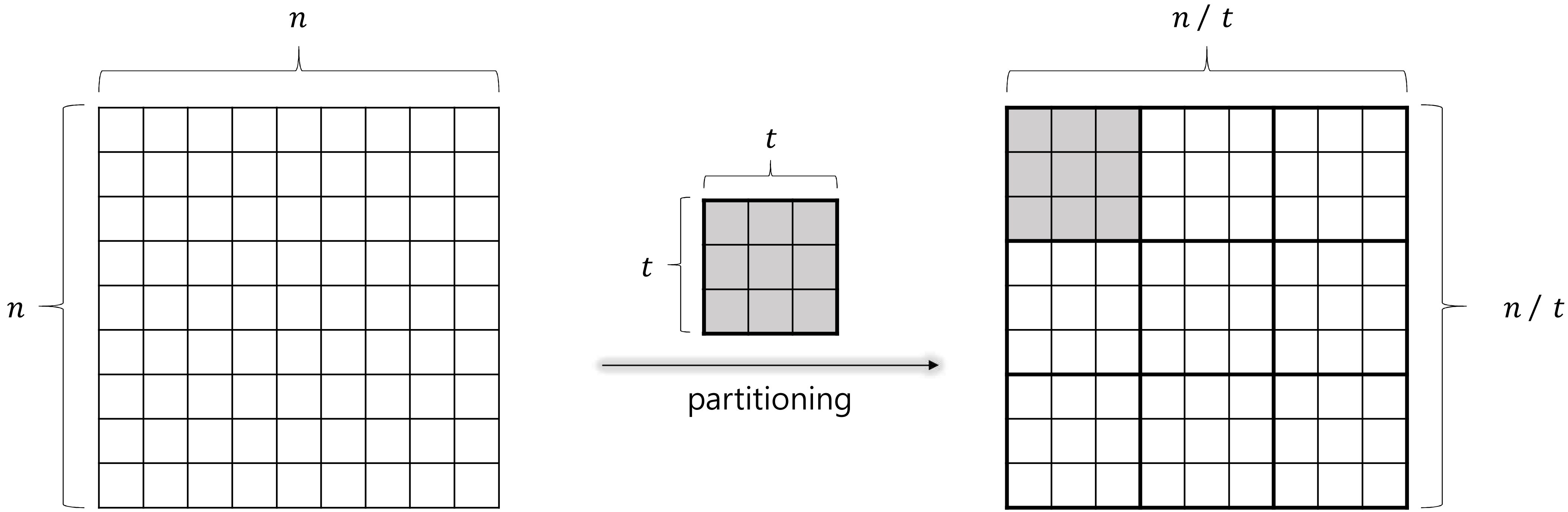


t =3

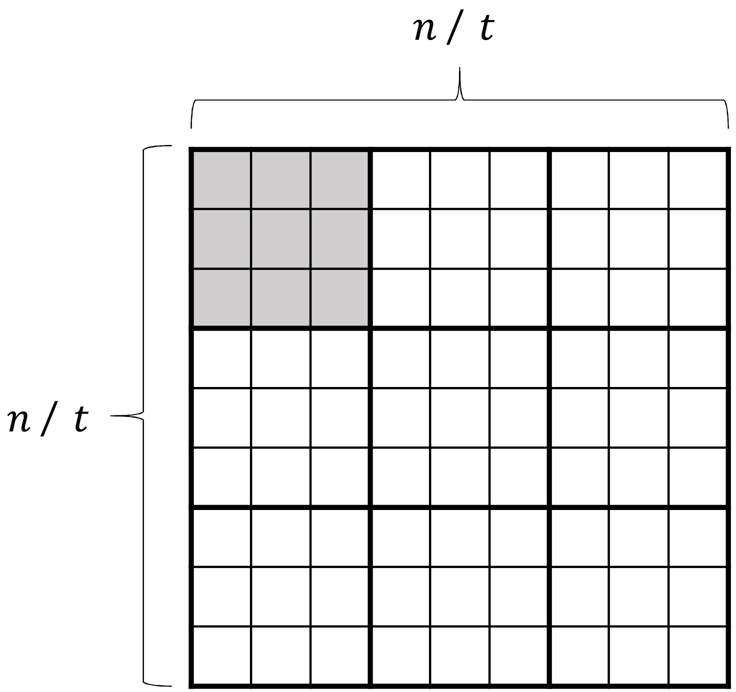


t =4

The Four-Russians speedup concept



The Four-Russians speedup concept



<Goal>

- $O(t)$ time per t-block

- Time Complexity $O(n^2)$ => $O(\frac{n^2}{\log n})$

Block function

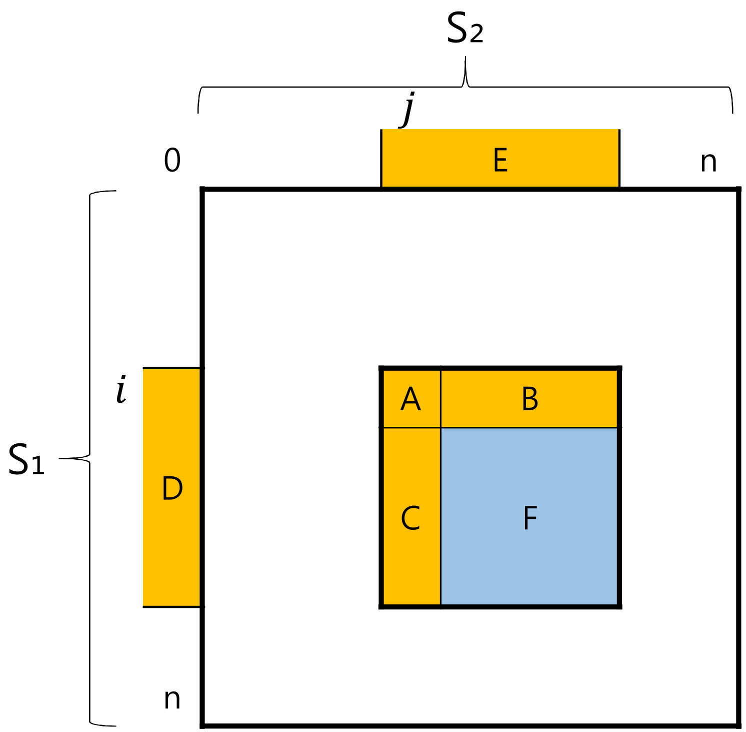
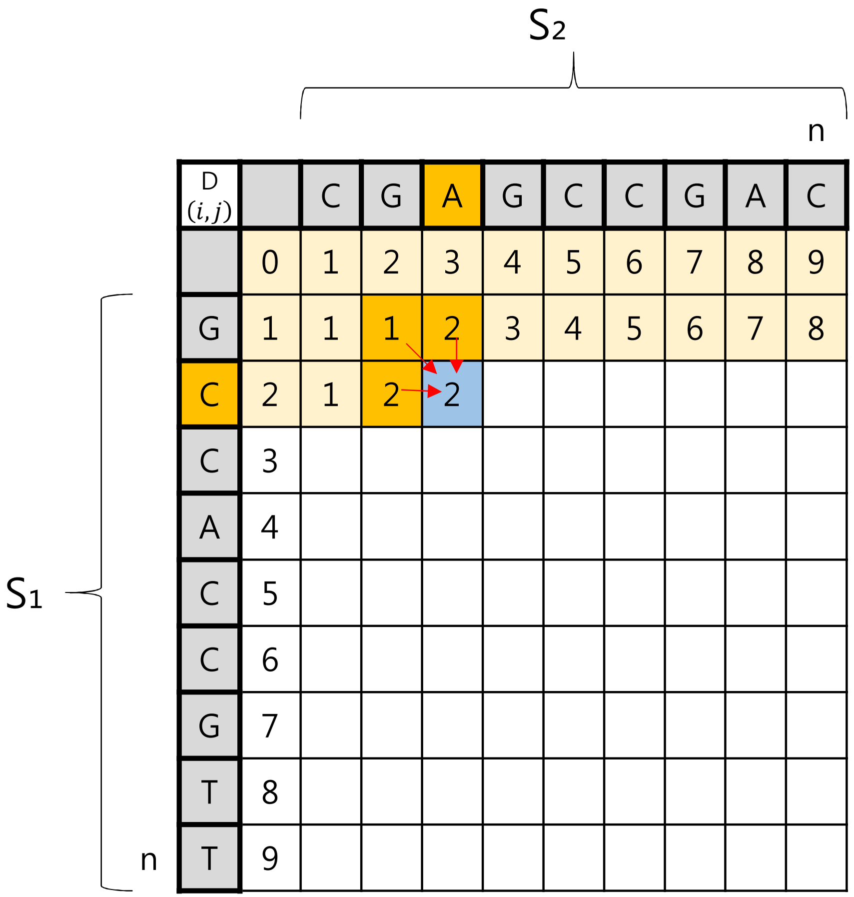


Figure 12.21

Lemma 12.7.1.

the distance values in a t-block starting in position (i, j) are a function of the values in its first row and column and the substrings S1$[i, i+1, \ldots, i+t-1]$ and S2$[j, j+1, \ldots, j+t-1]$.

Block function



D (i,j)		C	G	A	G	C	C	G	A	C
	0	1	2	3	4	5	6	7	8	9
G	1	1	1	2	3	4	5	6	7	8
C	2	1	2	2						
C	3									
A	4									
C	5									
C	6									
G	7									
T	8									
T	9									

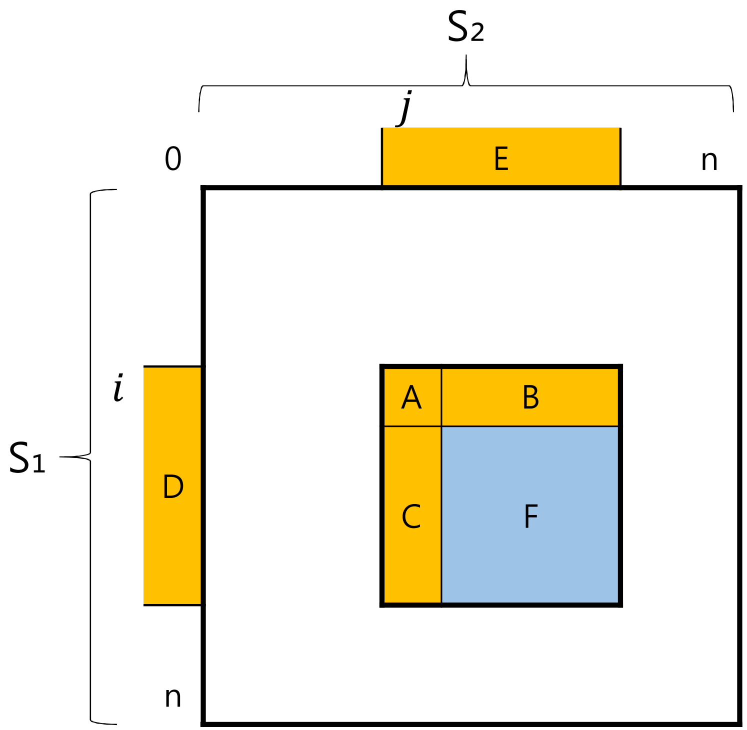


Figure 12.21

Block function

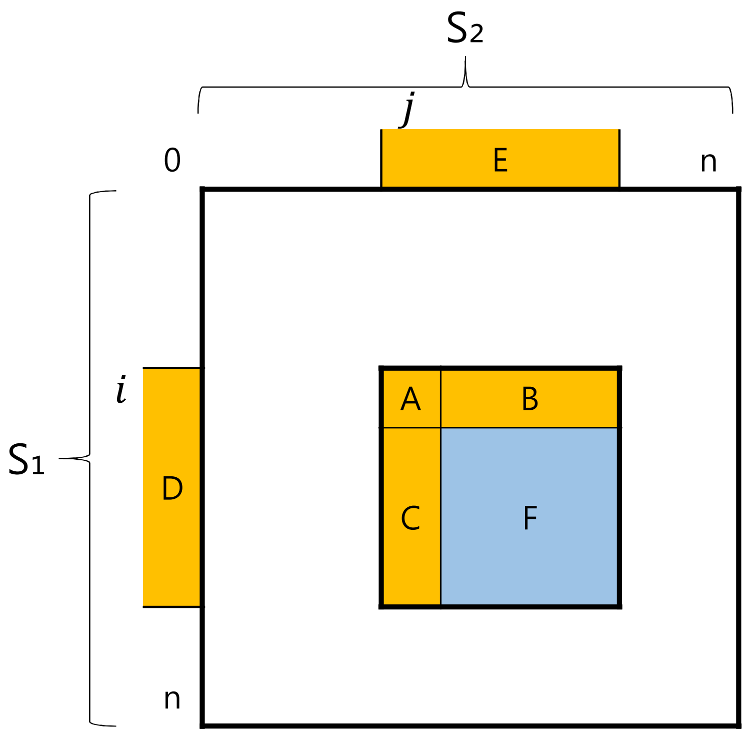


Figure 12.21

Definition

Given lemma 12.7.1, and using the notation shown in Figure 12.21, we define the block function as the function from the five inputs (A, B, C, D, E) to the output F.

{F} = blockFunction(A, B, C, D, E)

Restricted Block Function (RBF)

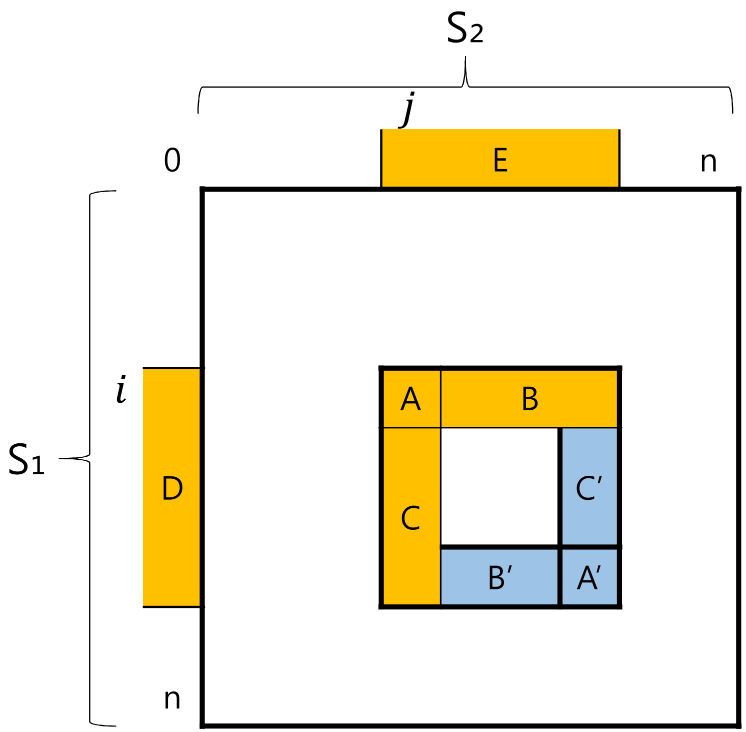


Figure 12.21

the values in the last row and column of a t-block are also a function of the inputs (A, B, C, D, E).

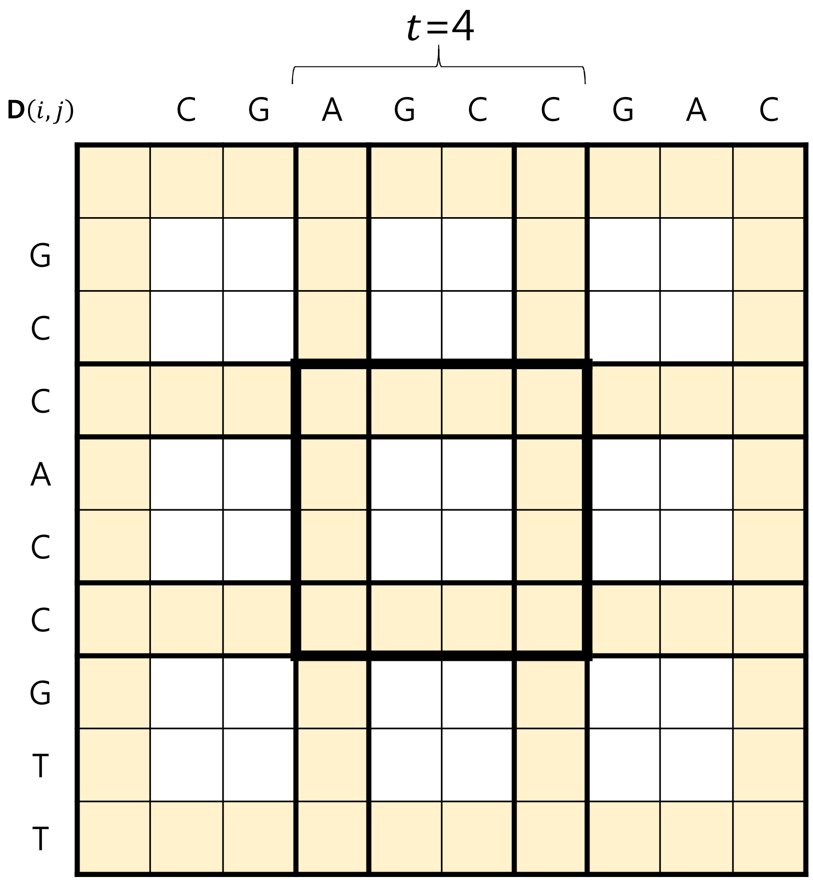
$$\{A', B', C'\} = \mathrm{RBF}(A, B, C, D, E)$$

Block Edit Distance Algorithm

$\mathbf{D}(i,j)$		C	G	A	G	C	C	G	A	C
G										
C										
C										
A										
C										
C										
G										
T										
T										

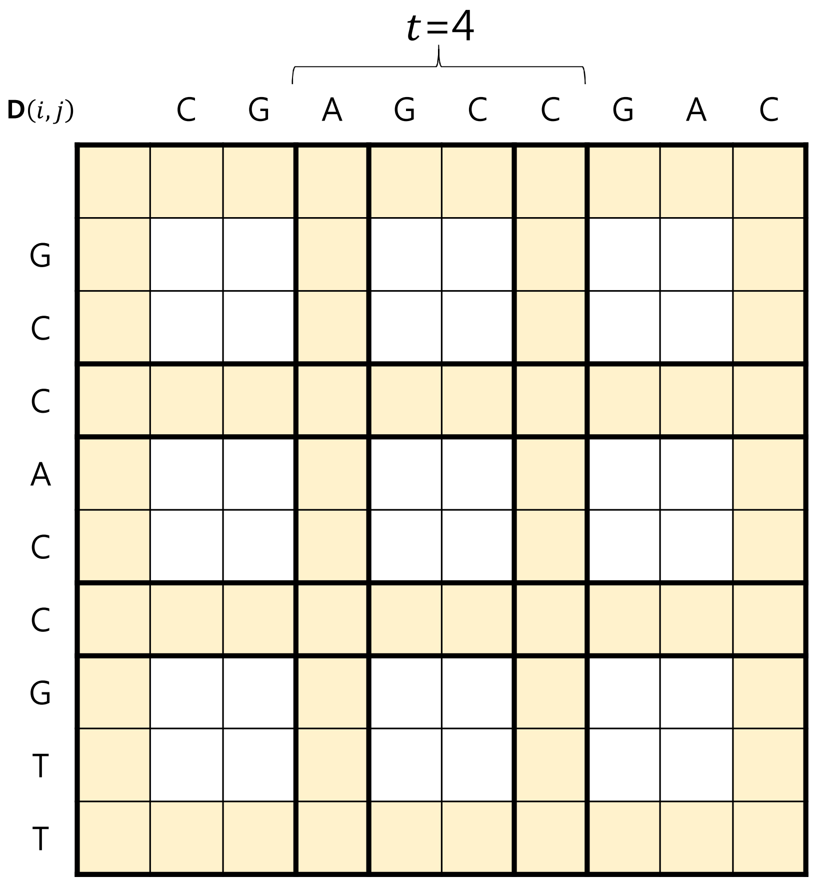
① Cover the $(n+1)$ by $(n+1)$ dynamic programming table with t-blocks that overlap by 1 row & 1 column

Block Edit Distance Algorithm



① Cover the $(n+1)$ by $(n+1)$ dynamic programming table with t-blocks that overlap by 1 row & 1 column

Block Edit Distance Algorithm



② Initialize the values in the first row and column of the full table according to the base conditions of the recurrence.

Block Edit Distance Algorithm

$t=4$

$\mathbf{D}(i,j)$		C	G	A	G	C	C	G	A	C
	0	1	2	3	4	5	6	7	8	9
G	1									
C	2									
C	3									
A	4									
C	5									
C	6									
G	7									
T	8									
T	9									

② Initialize the values in the first row and column of the full table according to the base conditions of the recurrence.

Block Edit Distance Algorithm

$t=4$

$\mathbf{D}(i,j)$		C	G	A	G	C	C	G	A	C
	0	1	2	3	4	5	6	7	8	9
G	1									
C	2									
C	3									
A	4									
C	5									
C	6									
G	7									
T	8									
T	9									

③ In a row wise manner, use the restricted block function to successively determine the values in the last row and last column of each block.

Block Edit Distance Algorithm

$t=4$

$\mathbf{D}(i,j)$		C	G	A	G	C	C	G	A	C
	0	1	2	3	4	5	6	7	8	9
G	1			2						
C	2			2						
C	3	2	2	3						
A	4									
C	5									
C	6									
G	7									
T	8									
T	9									

③ In a row wise manner, use the restricted block function to successively determine the values in the last row and last column of each block.

Block Edit Distance Algorithm

t=4

$\mathbf{D}(i,j)$		C	G	A	G	C	C	G	A	C
	0	1	2	3	4	5	6	7	8	9
G	1			2			5			
C	2			2			4			
C	3	2	2	3	3	3	3			
A	4									
C	5									
C	6									
G	7									
T	8									
T	9									

③ In a row wise manner, use the restricted block function to successively determine the values in the last row and last column of each block.

Block Edit Distance Algorithm

t=4

$\mathbf{D}(i,j)$		C	G	A	G	C	C	G	A	C
	0	1	2	3	4	5	6	7	8	9
G	1			2			5			8
C	2			2			4			7
C	3	2	2	3	3	3	3	4	5	6
A	4									
C	5									
C	6									
G	7									
T	8									
T	9									

③ In a row wise manner, use the restricted block function to successively determine the values in the last row and last column of each block.

Block Edit Distance Algorithm

$t=4$

D(i,j)		C	G	A	G	C	C	G	A	C
	0	1	2	3	4	5	6	7	8	9
G	1			2			5			8
C	2			2			4			7
C	3	2	2	3	3	3	3	4	5	6
A	4			2						
C	5			3						
C	6	5	5	4						
G	7									
T	8									
T	9									

③ In a row wise manner, use the restricted block function to successively determine the values in the last row and last column of each block.

Block Edit Distance Algorithm

$t=4$

$\mathbf{D}(i,j)$		C	G	A	G	C	C	G	A	C
	0	1	2	3	4	5	6	7	8	9
G	1			2			5			8
C	2			2			4			7
C	3	2	2	3	3	3	3	4	5	6
A	4			2			4			5
C	5			3			4			4
C	6	5	5	4	4	3	3	4	5	5
G	7			5			4			5
T	8			6			5			5
T	9	8	7	7	6	6	6	5	5	5

③ In a row wise manner, use the restricted block function to successively determine the values in the last row and last column of each block.

Block Edit Distance Algorithm

t=4

D(i,j)		C	G	A	G	C	C	G	A	C
	0	1	2	3	4	5	6	7	8	9
G	1			2			5			8
C	2			2			4			7
C	3	2	2	3	3	3	3	4	5	6
A	4			2			4			5
C	5			3			4			4
C	6	5	5	4	4	3	3	4	5	5
G	7			5			4			5
T	8			6			5			5
T	9	8	7	7	6	6	6	5	5	5

④ the value in cell (n, n) is the edit distance of S_1 and S_2

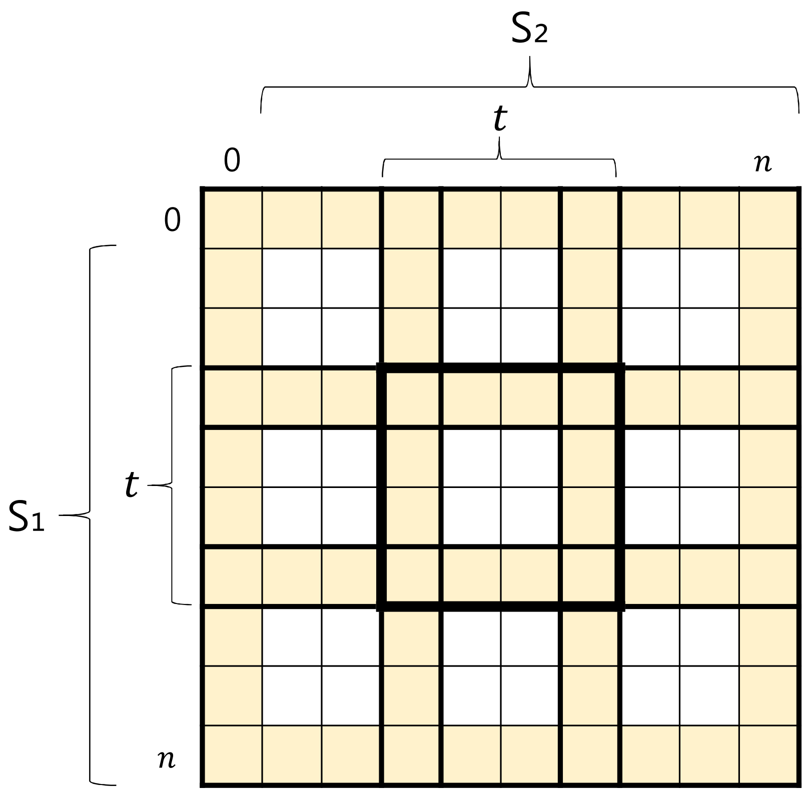
any instance of the restricted block function can be computed O(t^2) time, but that gains us nothing-!!

The Four-Russians idea for the RBF

- precomputation and storing subproblems.

Is it any faster than the original O(n^2) method?

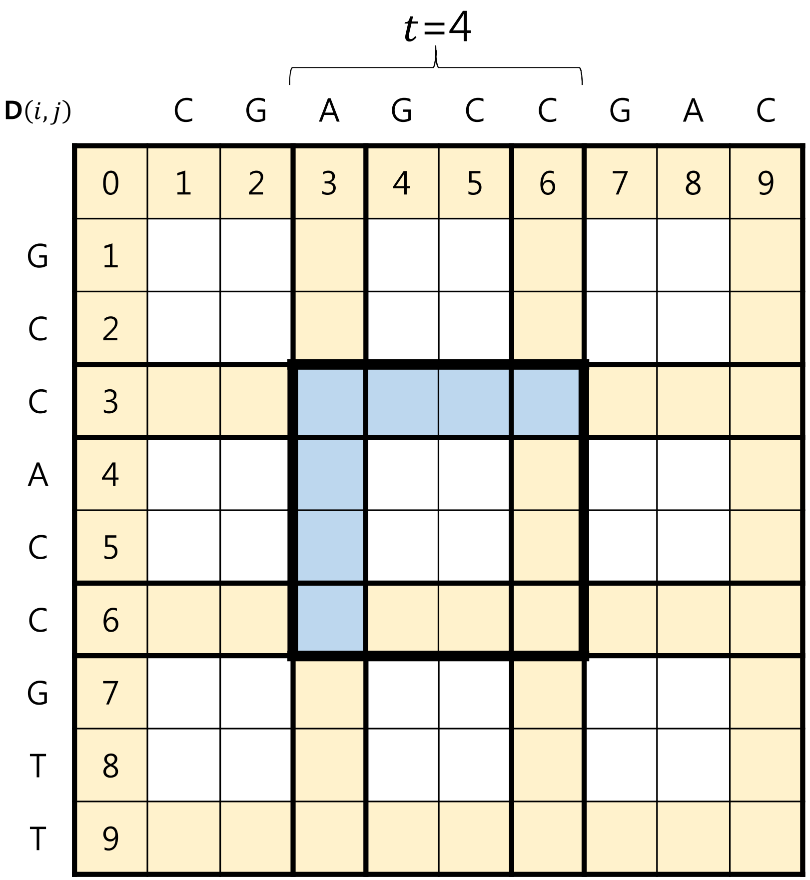
The Four-Russians idea for the RBF



- assume that precomputation is done
- the size of input and the output of the RBF are both $O(t)$
- thus, output can be retrieved in $O(t)$
- there are $\theta(\frac{n^2}{t^2})$ blocks
- total time = $O(\frac{n^2}{t})$
- set $t = \log n$, then $O(\frac{n^2}{\log n})$

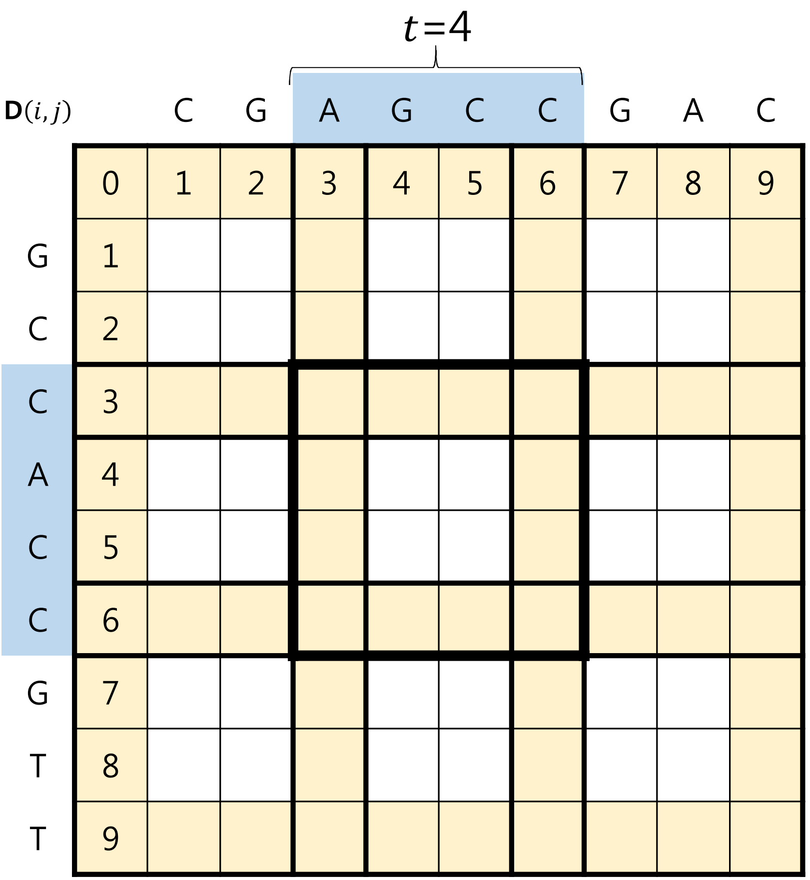
What about precomputation time?

Precomputation time



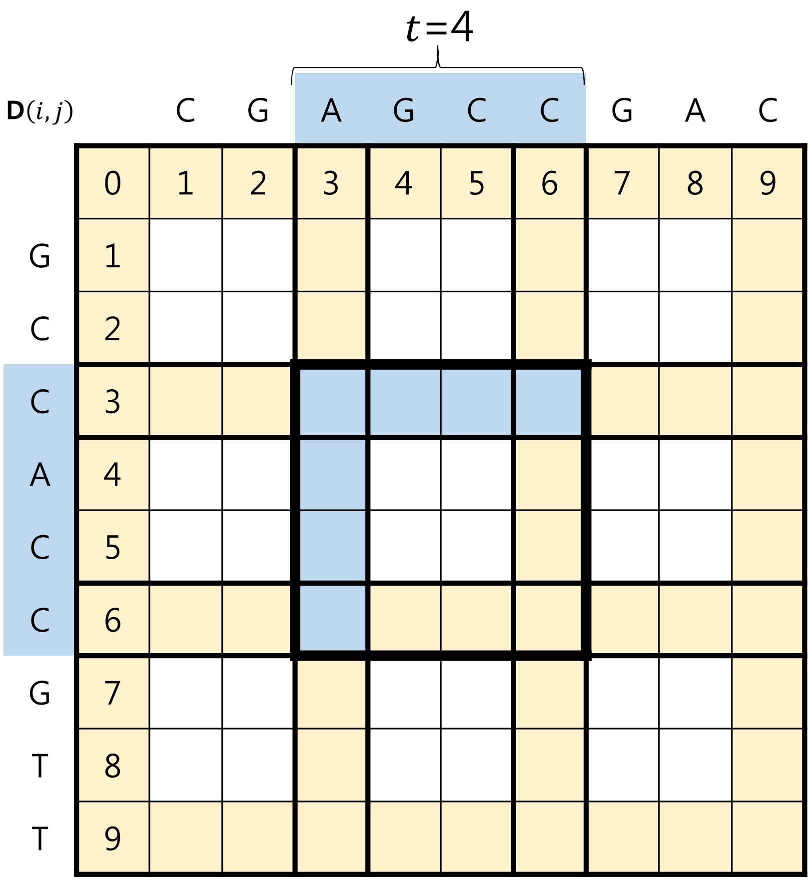
- Every cell has $0 \sim n$ values, so there are $(n+1)^t$ possible values for any t -length row or column

Precomputation time



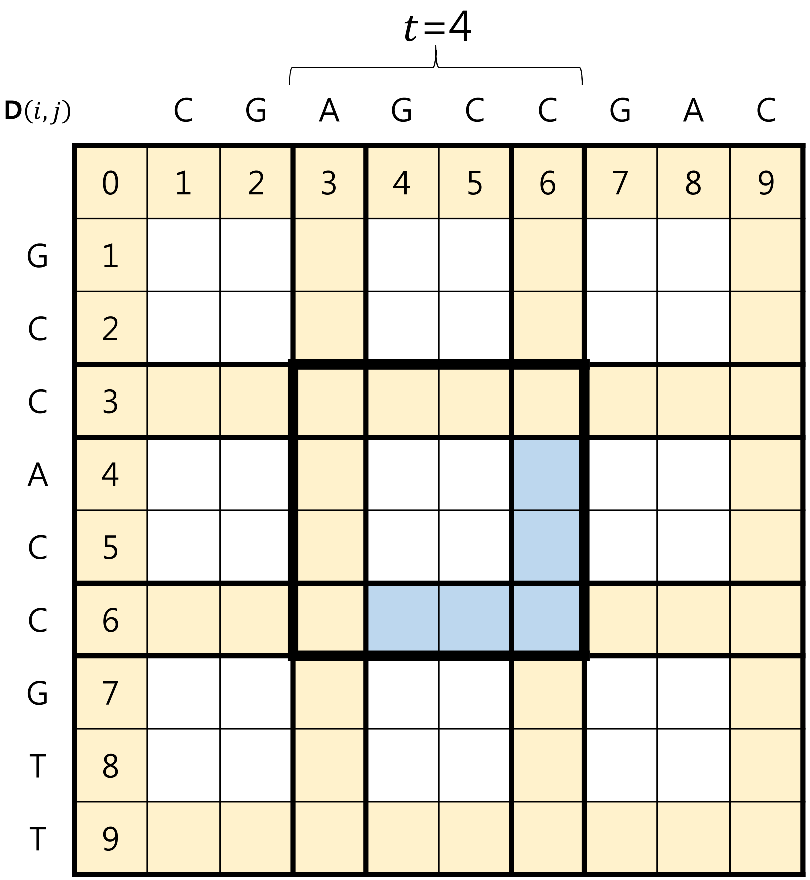
- Every cell has $0 \sim n$ values, so there are $(n+1)^t$ possible values for any t-length row or column
- If alphabet has size is σ, then there are σ^t possible substrings of length t

Precomputation time



- Every cell has 0 ~ n values, so there are $(n+1)^t$ possible values for any t -length row or column
- If alphabet has size is σ, $then$ there are σ^t possible substrings of length t
- Therefore, there are $(n+1)^{2t}\sigma^{2t}$ distinct input combinations for RBF

Precomputation time



- For each input, it takes $\theta(t^2)$ time to evaluate the last row and column of the resulting t-block (by running standard dynamic programming)

Precomputation time

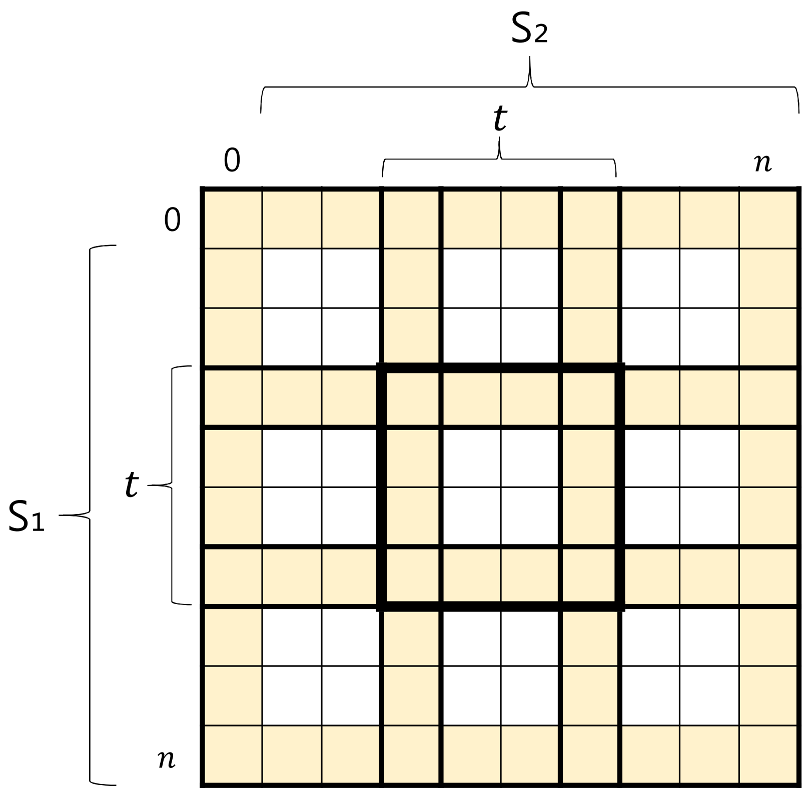
t=4

$\mathbf{D}(i,j)$		C	G	A	G	C	C	G	A	C
	0	1	2	3	4	5	6	7	8	9
G	1									
C	2									
C	3									
A	4									
C	5									
C	6									
G	7									
T	8									
T	9									

- For each input, it takes $\theta(t^2)$ time to evaluate the last row and column of the resulting t-block (by running standard dynamic programming)
- Thus, total precomputation time is $\theta((n+1)^{2t}\sigma^{2t}\ t^2)$
- t must be at least 1, so $\Omega(n^2)$ is used

we need another trick-!!

The Four-Russians idea for the RBF

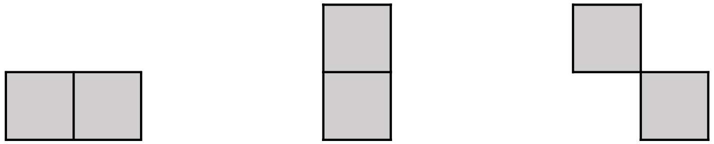


- assume that precomputation is done
- the size of input and the output of the RBF are both $O(t)$
- thus, output can be retrieved in $O(t)$
- there are $\theta(\frac{n^2}{t^2})$ blocks
- total time = $O(\frac{n^2}{t})$
- set $t = \log n$, then $O(\frac{n^2}{\log n})$

The trick : offset encoding

Lemma 12.7.2.

In any row, column, or diagonal of the dynamic programming table for edit distance, two adjacent cells can have a value that differs by at most one.



The trick : offset encoding

Example)

$\mathbf{D}(i,j)$		C	G	A	G	C	C	G	A	C
	0	1	2	3	4	5	6	7	8	9
G	1	1	1	2	3	4	5	6	7	8
C	2	1	2	2	3	3	4	5	6	7
C	3	2	2	3	3	3	3	4	5	6
A	4	3	3	2	3	4	4	4	4	5
C	5	4	4	3	3	3	4	5	5	4
C	6	5	5	4	4	3	3	4	5	5
G	7	6	5	5	4	4	4	3	4	5
T	8	7	6	6	5	5	5	4	4	5
T	9	8	7	7	6	6	6	5	5	5

The trick : offset encoding

Proof 

- Certainly, D(i, j) $\leq$ D$(i, j-1)$ + 1 $\cdots$ ①

$$\mathrm{D}(i,j) = \min \begin{cases} \mathrm{D}(i-1,j)+1 \\ \mathrm{D}(i,j-1)+1 \\ \mathrm{D}(i-1,j-1)+t(i,j) \end{cases}$$

The trick : offset encoding

Proof **Case 1.** $S_2(j)$ is matched with some character of $S_1[1..i]$

$D(i,j)$	$D(i,j-1)$

$S_1[1..i]$: A T − C G A A
                  | | | |
$S_2[1..j]$: A C C C G − −
	$S_1[1..i]$: A T − C G A A
	                  | | | | |
	$S_2[1..j]$: A C C C − − −

$S_1[1..i]$: A T − C G A A
                  | | | | |
$S_2[1..j]$: A C C G G − −
	$S_1[1..i]$: A T − C G A A
	                  | | | | |
	$S_2[1..j]$: A C C − G − −

$D(i,j-1) = D(i,j) + 1$

$D(i,j-1) = D(i,j)$

② $\boxed{D(i,j-1) \leq D(i,j) + 1}$

The trick : offset encoding

Proof **Case 2.** $S_2(j)$ is not matched with $S_1[1..i]$

$D(i,j)$	$D(i,j-1)$
$S_1[1..i]$: A T − C G A A |			| $S_2[1..j]$: A C C C T − −	$S_1[1..i]$: A T − C G A A |			| $S_2[1..j]$: A C C C − − −
$S_1[1..i]$: A T T C G A − − 				 $S_2[1..j]$: A − C C G A C G	$S_1[1..i]$: A T T C G A − 			 $S_2[1..j]$: A − C C G A C

$D(i,j-1) = D(i,j)$

$D(i,j-1) = D(i,j) - 1$

③ $\boxed{D(i,j-1) \geq D(i,j) - 1}$

The trick : offset encoding

Proof

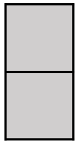
① $D(i,j) \leq D(i,j-1) + 1 \Longrightarrow D(i,j) \leq D(i,j-1) + 1$

② $D(i,j-1) \leq D(i,j) + 1 \Rightarrow D(i,j-1) - 1 \leq D(i,j)$

③ $D(i,j-1) \geq D(i,j) - 1 \Longrightarrow D(i,j) \leq D(i,j-1) + 1$

$$\boxed{D(i,j-1) - 1 \leq D(i,j) \leq D(i,j-1) + 1}$$

The trick : offset encoding

Proof  is same as 

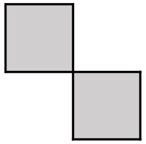
① $\mathrm{D}(i,j) \leq \mathrm{D}(i-1,j) + 1 \Rightarrow \mathrm{D}(i,j) \leq \mathrm{D}(i-1,j) + 1$

② $\mathrm{D}(i-1,j) \leq \mathrm{D}(i,j) + 1 \Rightarrow \mathrm{D}(i-1,j) - 1 \leq \mathrm{D}(i,j)$

③ $\mathrm{D}(i-1,j) \geq \mathrm{D}(i,j) - 1 \Rightarrow \mathrm{D}(i,j) \leq \mathrm{D}(i-1,j) + 1$

$$\boxed{\mathrm{D}(i-1,j) - 1 \leq \mathrm{D}(i,j) \leq \mathrm{D}(i-1,j) + 1}$$

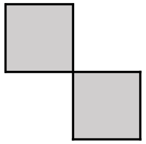
The trick : offset encoding

Proof 

- Certainly, $\mathrm{D}(i,j) \leq \mathrm{D}(i-1,j-1) + 1$ … ①

$$\mathrm{D}(i,j) = \min \begin{cases} \mathrm{D}(i-1,j)+1 \\ \mathrm{D}(i,j-1)+1 \\ \mathrm{D}(i-1,j-1)+t(i,j) \end{cases}$$

The trick : offset encoding

Proof  **Case 1.** optimal alignment of $S_1[1..i]$ and $S_2[1..j]$ aligns i against j

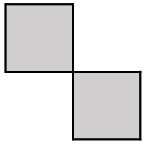
$D(i,j)$	$D(i-1,j-1)$
$S_1[1..i]$: A T − C G A | | $S_2[1..j]$: A C C C G A	$S_1[1..i]$: A T − C G | | $S_2[1..j]$: A C C C G
$S_1[1..i]$: A T − C G A | | | $S_2[1..j]$: A C C C G C	$S_1[1..i]$: A T − C G | | $S_2[1..j]$: A C C C G

$D(i-1,j-1) = D(i,j)$

$D(i-1,j-1) = D(i,j) - 1$

② $\boxed{D(i-1,j-1) \leq D(i,j)}$

The trick : offset encoding

Proof  **Case 2.** optimal alignment doesn't align i against j

$\mathrm{D}(i,j)$	$\mathrm{D}(i-1,j-1)$
$S_1[1..i]$: A T T C G A C T | | | | $S_2[1..j]$: A − C C G A − −	$S_1[1..i]$: A T T C G A C | | | | $S_2[1..j]$: A − C C G − −
$S_1[1..i]$: A T T C G A C T | | | | | $S_2[1..j]$: A − C C G C − −	$S_1[1..i]$: A T T C G A C | | | | $S_2[1..j]$: A − C C G − −

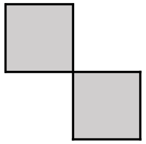
$\mathrm{D}(i-1,j-1) = \mathrm{D}(i,j)$

$\mathrm{D}(i-1,j-1) = \mathrm{D}(i,j) - 1$

③ $\boxed{\mathrm{D}(i-1,j-1) \leq \mathrm{D}(i,j)}$

The trick : offset encoding

Proof



① $D(i,j) \le D(i-1,j-1)+1$ → $D(i,j) \le D(i-1,j-1)+1$

② $D(i-1,j-1) \le D(i,j)$ → $D(i-1,j-1) \le D(i,j)$

③ $D(i-1,j-1) \le D(i,j)$ → $D(i-1,j-1) \le D(i,j)$

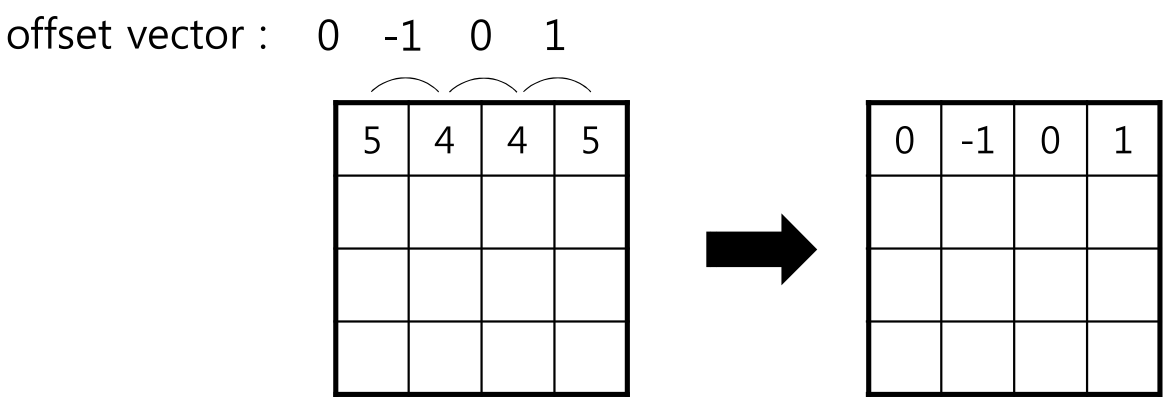
$$\boxed{D(i-1,j-1) \le D(i,j) \le D(i-1,j-1)+1}$$

Lemma 12.7.2. has been proven-!!

offset vector

Definition

The *offset vector* is a t-length vector of values from {-1, 0, 1}, where the first entry must be zero.



offset vector

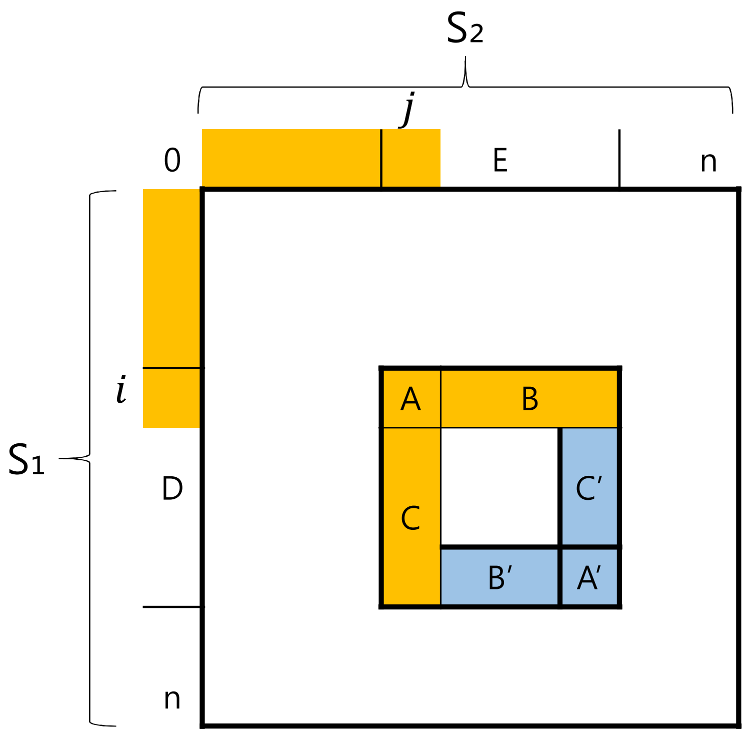


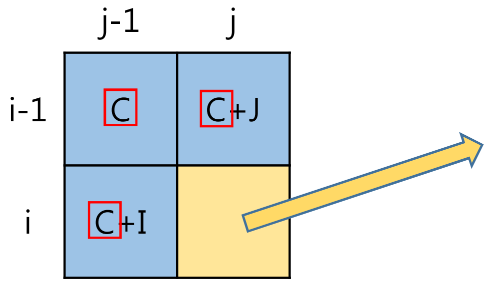
Figure 12.21

Theorem 12.7.1.

Consider a t-block with upper left corner in position (i, j). The two offset vectors for the last row and last column of the block can be determined from the two offset vectors for the first row and column of the block and from substrings $S_1[1..i]$ and $S_2[1..j]$. *That is, no Edit Distance value is needed in the input in order to determine the offset vectors in the last row and column of the block*.

offset vector

Proof



$$\min \begin{cases} C + I + 1 \\ C + J + 1 \\ C + t(i,j) \end{cases}$$

offset vector

Proof

	j	j+1	j+2	j+3
i	C	C+J	C+J+J′	C+…
i+1	C+I	C+…	C+…	C+…
i+2	C+I+I′	C+…	C+…	C+…
i+3	C+…	C+…	C+…	C+…

-C

	j	j+1	j+2	j+3
i	0	J	J+J′	…
i+1	I	…	…	…
i+2	I+I′	…	…	…
i+3	…	…	…	…

Theorem 12.7.1. has been proven-!!

offset function

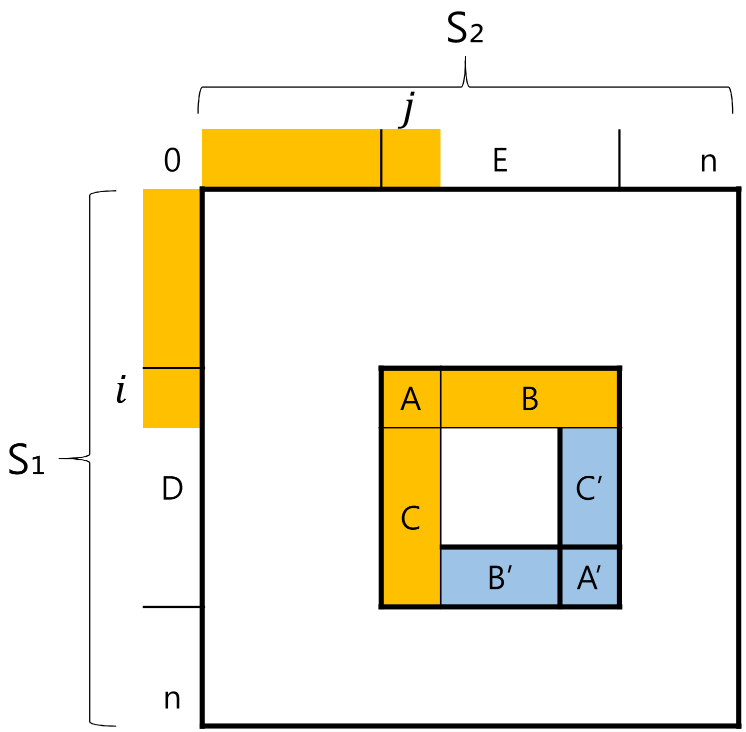
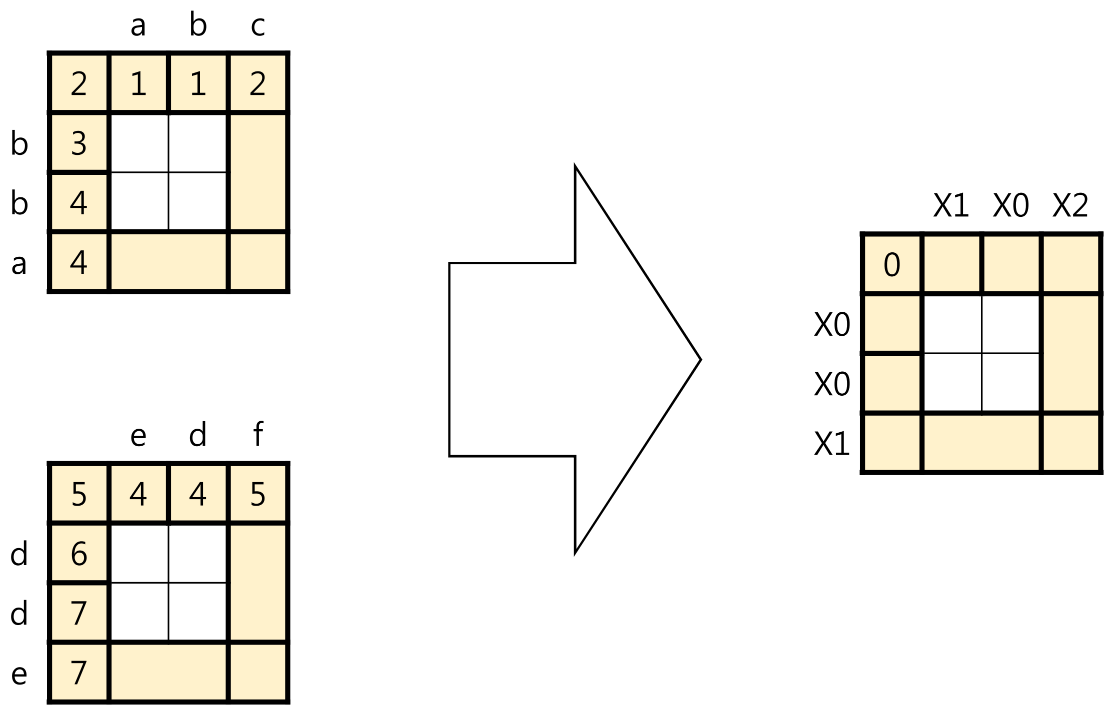


Figure 12.21

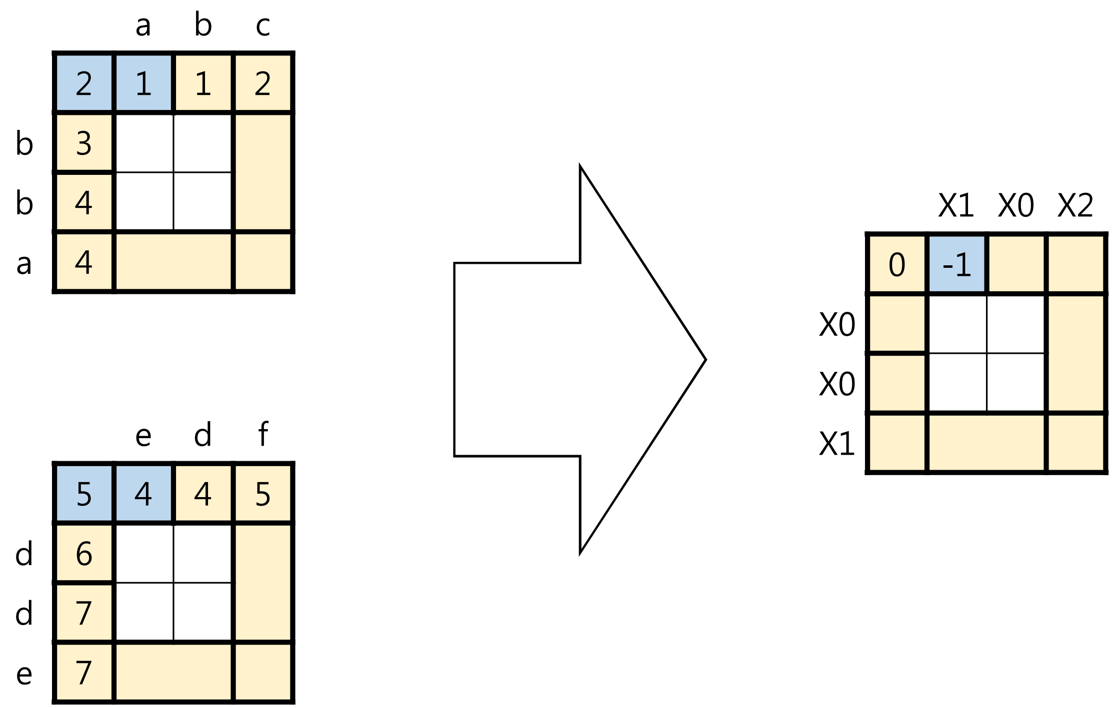
Definition

The function that determines the two offset vectors for the last row and last column from the two offset vectors for the first row and column of a block together with substrings $S_1[1..i]$and $S_2[1..j]$ is called the *offset function*.

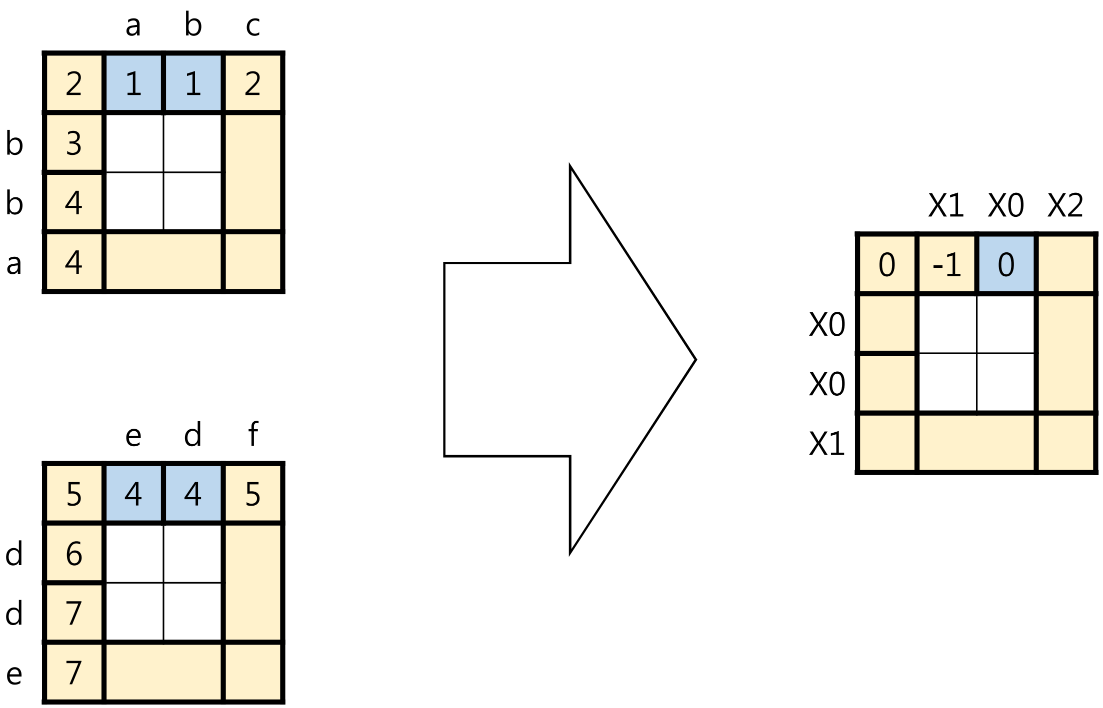
Precomputation using offset encoding



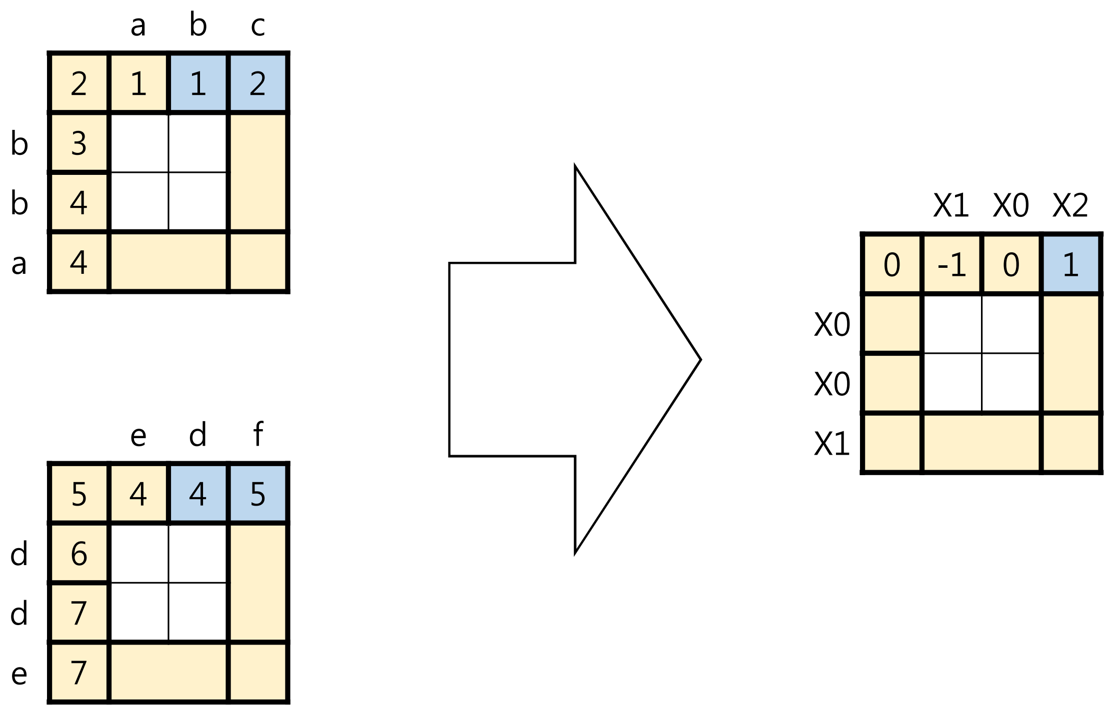
Precomputation using offset encoding



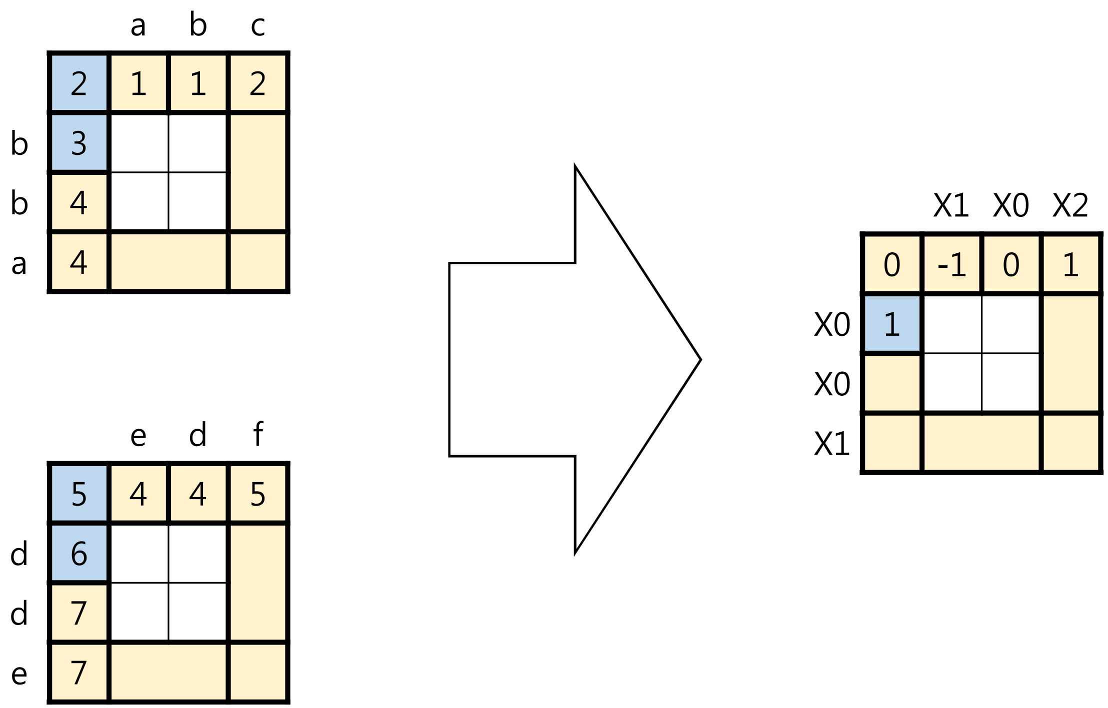
Precomputation using offset encoding



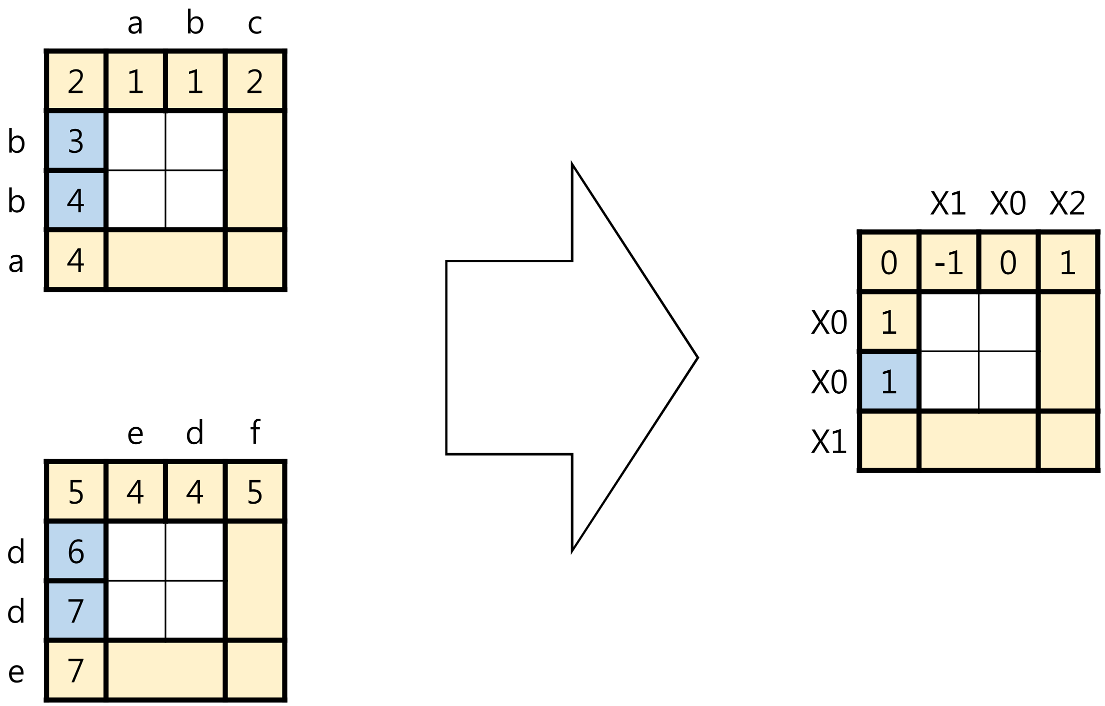
Precomputation using offset encoding



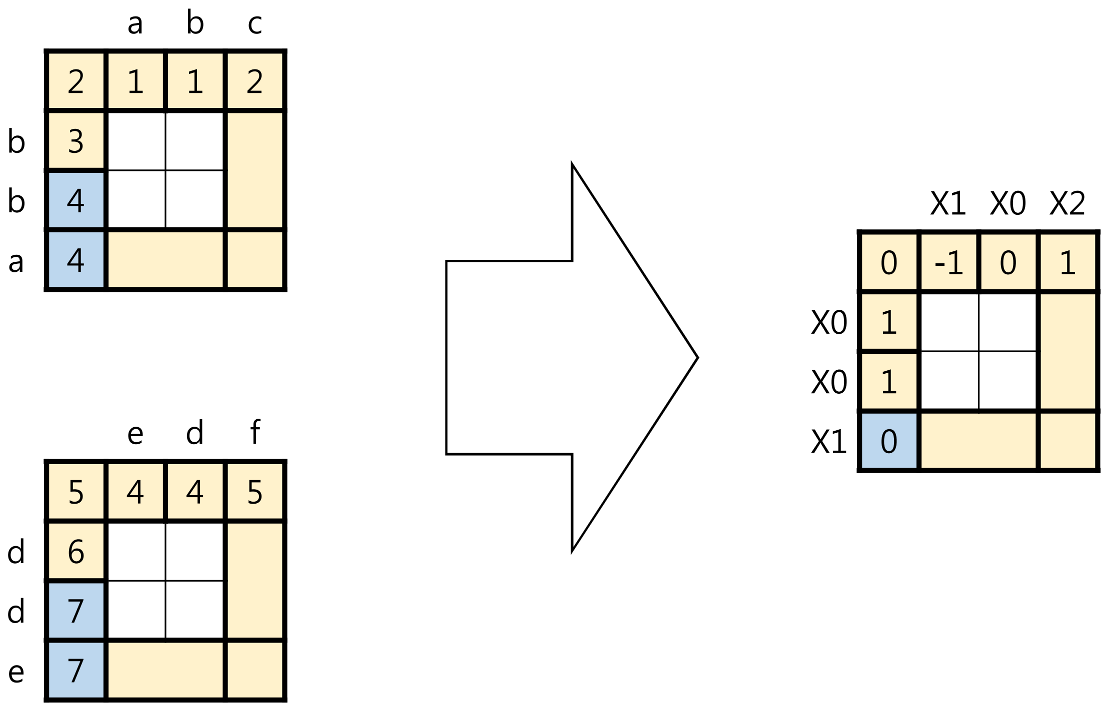
Precomputation using offset encoding



Precomputation using offset encoding



Precomputation using offset encoding



Precomputation using offset encoding

		X1	X0	X2
	0	-1	0	1
X0	1			
X0	1			
X1	0			

$$\theta((n+1)^{2t}\sigma^{2t}\ t^2)$$

$$\Downarrow$$

$$\theta((3)^{2t}\sigma^{2t}\ t^2)$$

$$\Big\downarrow\ t = \frac{\log_{3\sigma} n}{2}$$

$$\theta\left((3\sigma)^{2\left(\frac{\log_{3\sigma} n}{2}\right)} \left(\frac{\log_{3\sigma} n}{2}\right)^2 \right) \longrightarrow \mathrm{O}(n(\log n)^2)$$

Precomputation table

precomputation table

$D(i,j)$		X1	X0	X2
	0	1	1	1
X0	1	…	…	…
X1	1	…	…	…
X1	1	…	…	…

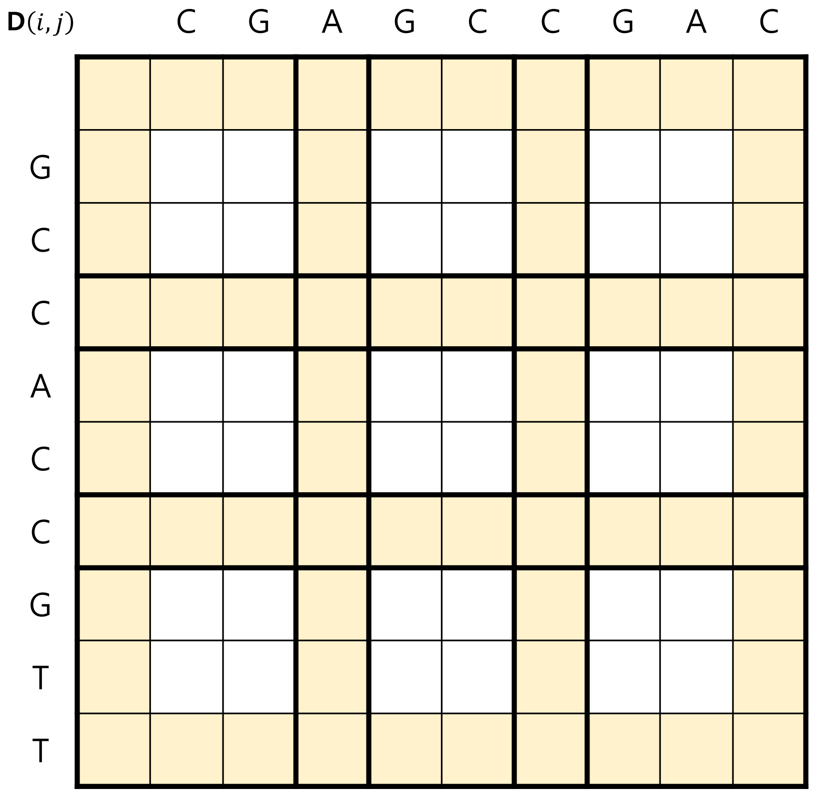
$D(i,j)$		X0	X1	X1
	0	1	1	1
X0	-1	…	…	…
X1	0	…	…	…
X1	1	…	…	…

Four-Russians Edit Distance Algorithm

D(i,j)		C	G	A	G	C	C	G	A	C
G										
C										
C										
A										
C										
C										
G										
T										
T										

① Cover the $(n+1)$ by $(n+1)$ dynamic programming table with t-blocks that overlap by 1 row & 1 column

Four-Russians Edit Distance Algorithm



① Cover the $(n+1)$ by $(n+1)$ dynamic programming table with t-blocks that overlap by 1 row & 1 column

Four-Russians Edit Distance Algorithm

D(i,j)		C	G	A	G	C	C	G	A	C
G										
C										
C										
A										
C										
C										
G										
T										
T										

② Initialize the values in the first row and column of the full table according to the base conditions of the recurrence. Compute the offset values in the first row and column.

Four-Russians Edit Distance Algorithm

D(i,j)		C	G	A	G	C	C	G	A	C
	0	1	2	3	4	5	6	7	8	9
G	1									
C	2									
C	3									
A	4									
C	5									
C	6									
G	7									
T	8									
T	9									

② Initialize the values in the first row and column of the full table according to the base conditions of the recurrence. Compute the offset values in the first row and column.

Four-Russians Edit Distance Algorithm

$\mathbf{D}(i,j)$		C	G	A	G	C	C	G	A	C
	0_0	1_1	1_2	1_3	1_4	1_5	1_6	1_7	1_8	1_9
G	1_1									
C	1_2									
C	1_3									
A	1_4									
C	1_5									
C	1_6									
G	1_7									
T	1_8									
T	1_9									

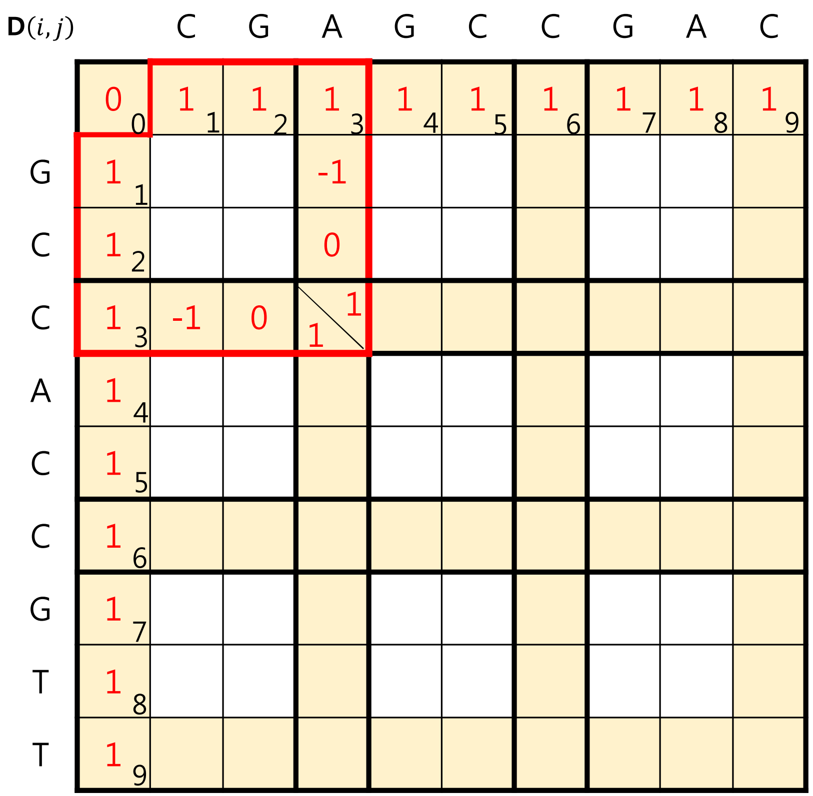
② Initialize the values in the first row and column of the full table according to the base conditions of the recurrence. Compute the offset values in the first row and column.

Four-Russians Edit Distance Algorithm

$\mathbf{D}(i,j)$		C	G	A	G	C	C	G	A	C
	0_{0}	1_{1}	1_{2}	1_{3}	1_{4}	1_{5}	1_{6}	1_{7}	1_{8}	1_{9}
G	1_{1}									
C	1_{2}									
C	1_{3}									
A	1_{4}									
C	1_{5}									
C	1_{6}									
G	1_{7}									
T	1_{8}									
T	1_{9}									

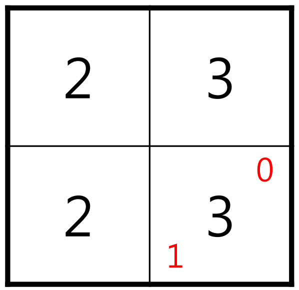
③ In a row wise manner, use the ~~restricted block function~~ offset block function to successively determine the offset values in the last row and last column of each block.

Four-Russians Edit Distance Algorithm



③ In a row wise manner, use the ~~restricted block function~~ offset block function to successively determine the offset values in the last row and last column of each block.

Ex)



Four-Russians Edit Distance Algorithm

$\mathbf{D}(i,j)$		C	G	A	G	C	C	G	A	C
	0_0	1_1	1_2	1_3	1_4	1_5	1_6	1_7	1_8	1_9
G	1_1			-1			-1			
C	1_2			0			-1			
C	1_3	-1	0	1 / 1	0	0	-1 / 0			
A	1_4									
C	1_5									
C	1_6									
G	1_7									
T	1_8									
T	1_9									

③ In a row wise manner, use the ~~restricted block function~~ offset block function to successively determine the offset values in the last row and last column of each block.

Four-Russians Edit Distance Algorithm

D(i, j)		C	G	A	G	C	C	G	A	C
	0_{0}	1_{1}	1_{2}	1_{3}	1_{4}	1_{5}	1_{6}	1_{7}	1_{8}	1_{9}
G	1_{1}			-1			-1			-1
C	1_{2}			0			-1			-1
C	1_{3}	-1	0	1 / 1	0	0	-1 / 0	1	1	-1 / 1
A	1_{4}									
C	1_{5}									
C	1_{6}									
G	1_{7}									
T	1_{8}									
T	1_{9}									

③ In a row wise manner, use the ~~restricted block function~~ offset block function to successively determine the offset values in the last row and last column of each block.

Four-Russians Edit Distance Algorithm

D(i,j)		C	G	A	G	C	C	G	A	C
	0 $_0$	1 $_1$	1 $_2$	1 $_3$	1 $_4$	1 $_5$	1 $_6$	1 $_7$	1 $_8$	1 $_9$
G	1 $_1$			-1			-1			-1
C	1 $_2$			0			-1			-1
C	1 $_3$	-1	0	1 1	0	0	-1 0	1	1	-1 1
A	1 $_4$			-1						
C	1 $_5$			1						
C	1 $_6$	-1	0	1 -1						
G	1 $_7$									
T	1 $_8$									
T	1 $_9$									

③ In a row wise manner, use the ~~restricted block function~~ offset block function to successively determine the offset values in the last row and last column of each block.

Four-Russians Edit Distance Algorithm

$\mathbf{D}(i,j)$		C	G	A	G	C	C	G	A	C
	0 $_0$	1 $_1$	1 $_2$	1 $_3$	1 $_4$	1 $_5$	1 $_6$	1 $_7$	1 $_8$	1 $_9$
G	1 $_1$			-1			-1			-1
C	1 $_2$			0			-1			-1
C	1 $_3$	-1	0	1 \ 1	0	0	0 \ -1	1	1	1 \ -1
A	1 $_4$			-1			1			-1
C	1 $_5$			1			0			-1
C	1 $_6$	-1	0	-1 \ 1	0	-1	0 \ -1	1	1	0 \ 1
G	1 $_7$			1			1			0
T	1 $_8$			1			1			0
T	1 $_9$	-1	-1	0 \ 1	-1	0	0 \ 1	-1	0	0 \ 0

③ In a row wise manner, use the ~~restricted block function~~ offset block function to successively determine the offset values in the last row and last column of each block.

Four-Russians Edit Distance Algorithm

$\mathbf{D}(i,j)$		C	G	A	G	C	C	G	A	C
	0_{0}	1_{1}	1_{2}	1_{3}	1_{4}	1_{5}	1_{6}	1_{7}	1_{8}	1_{9}
G	1_{1}			-1			-1			-1
C	1_{2}			0			-1			-1
C	1_{3}	-1	0	1 / 1	0	0	0 / -1	1	1	1 / -1
A	1_{4}			-1			1			-1
C	1_{5}			1			0			-1
C	1_{6}	-1	0	-1 / 1	0	-1	0 / -1	1	1	0 / 1
G	1_{7}			1			1			0
T	1_{8}			1			1			0
T	1_{9}	-1	-1	0 / 1	-1	0	0 / 1	-1	0	0 / 0

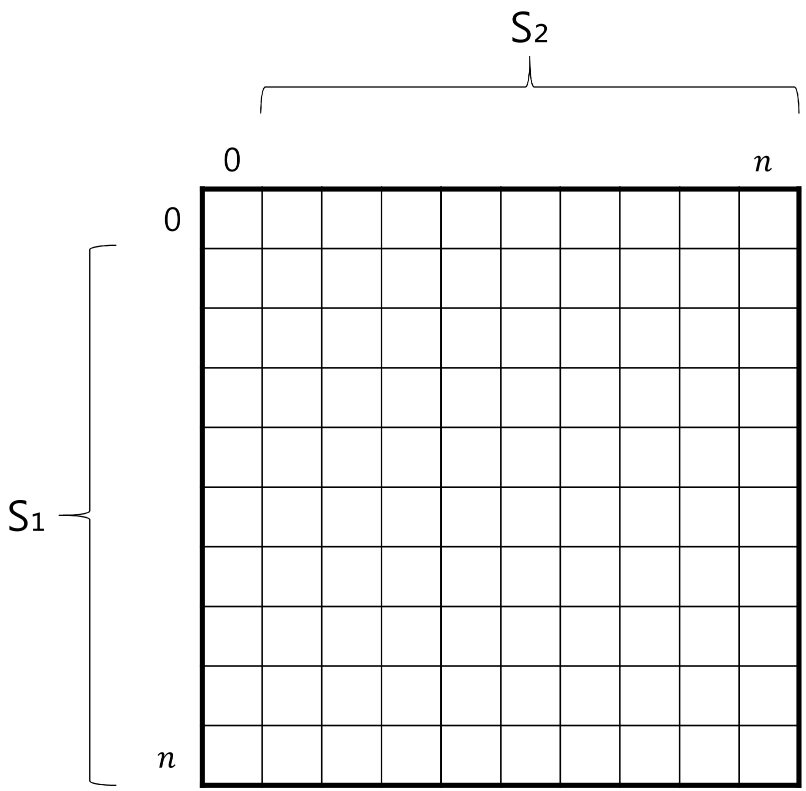
④ ~~The value in cell (n, n) is the edit distanace of S_1 and S_2.~~ Let Q be the total of the offset values computed for cells in row n. D(n, n) = D(n, 0) + Q = n + Q

Four-Russians Edit Distance Algorithm

D(i, j)		C	G	A	G	C	C	G	A	C
	0_{0}	1_{1}	1_{2}	1_{3}	1_{4}	1_{5}	1_{6}	1_{7}	1_{8}	1_{9}
G	1_{1}			-1			-1			-1
C	1_{2}			0			-1			-1
C	1_{3}	-1	0	1 / 1	0	0	-1 / 0	1	1	-1 / 1
A	1_{4}			-1			1			-1
C	1_{5}			1			0			-1
C	1_{6}	-1	0	1 / -1	0	-1	-1 / 0	1	1	1 / 0
G	1_{7}			1			1			0
T	1_{8}			1			1			0
T	1_{9}	-1_{8}	-1_{7}	1 / 0 $_{7}$	-1_{6}	0_{6}	1 / 0 $_{6}$	-1_{5}	0_{5}	0 / 0 $_{5}$

④ ~~The value in cell (n, n) is the edit distanace of S_1 and S_2.~~ Let Q be the total of the offset values computed for cells in row n. D(n, n) = D(n, 0) + Q = n + Q

Summary



<Goal>

- $O(t)$ time per t-block ✓
- Time Complexity $O(n^2)$ => $O(\frac{n^2}{\log n})$ ✓