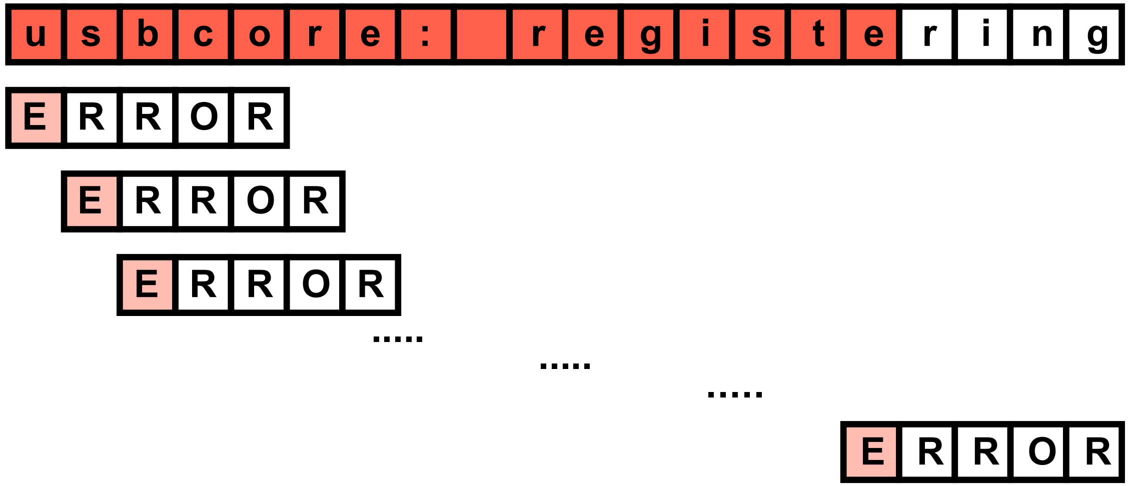
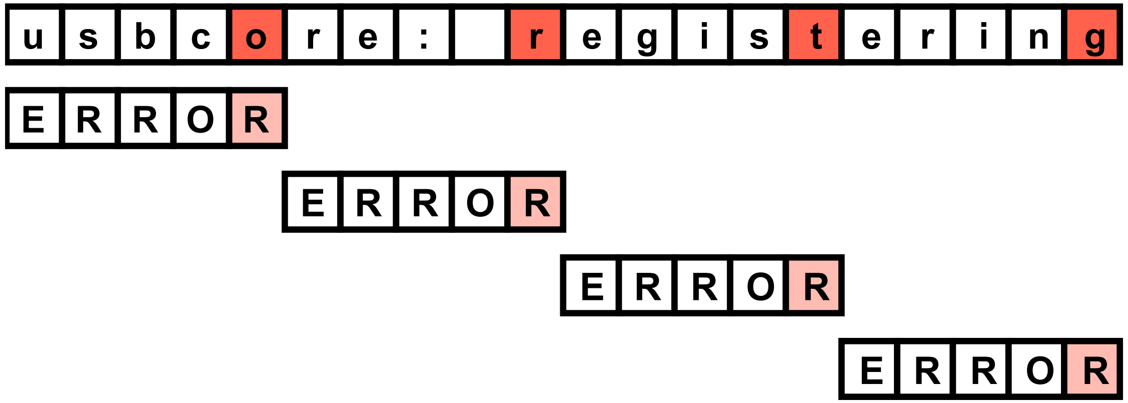
Boyer Moore Algorithm
for
Pattern Searching

Left to Right

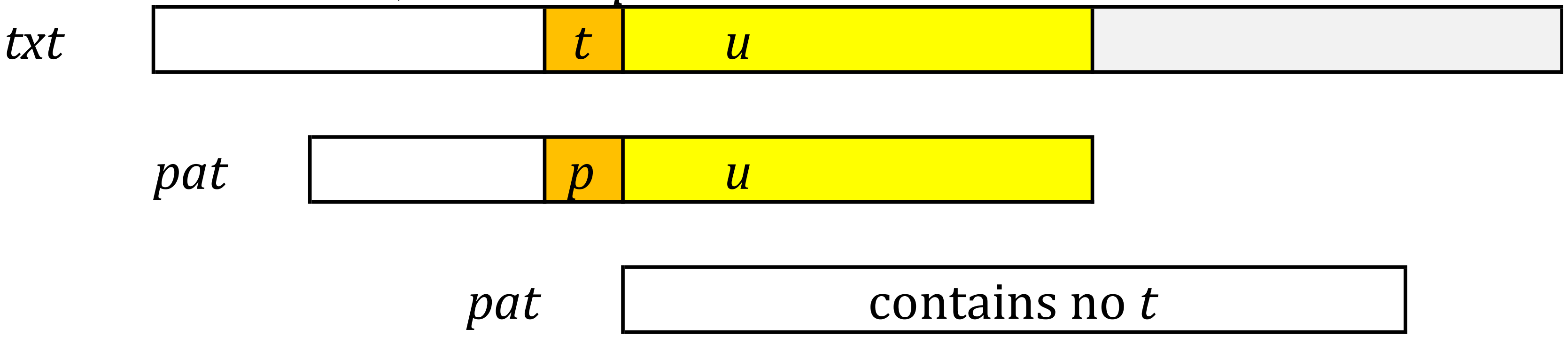


Right to Left



Bad character shift:

1) if pat does not contain t, we slide pat next to t

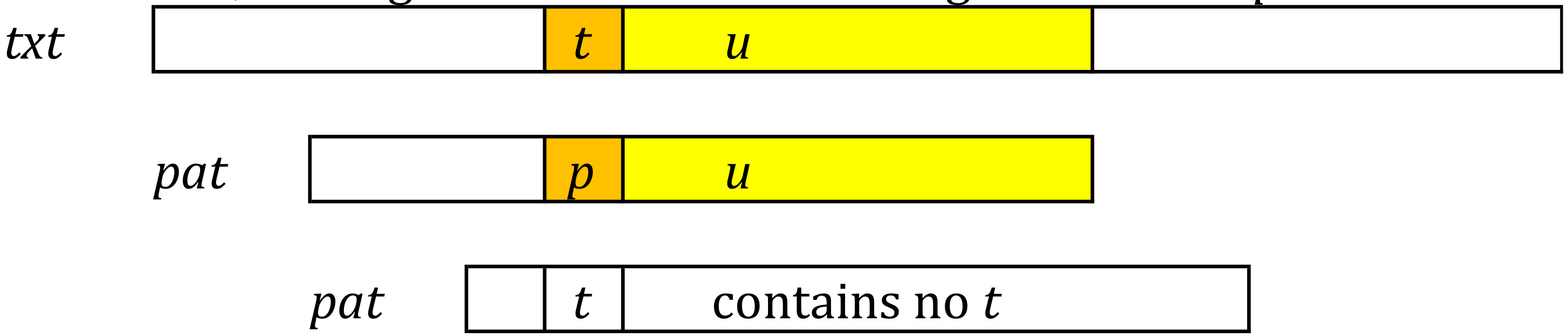


txt t u

pat p u

pat contains no t

2) if pat containts t, we align t of the txt with the right-most t of pat



txt t u

pat p u

pat t contains no t

Bad character table:

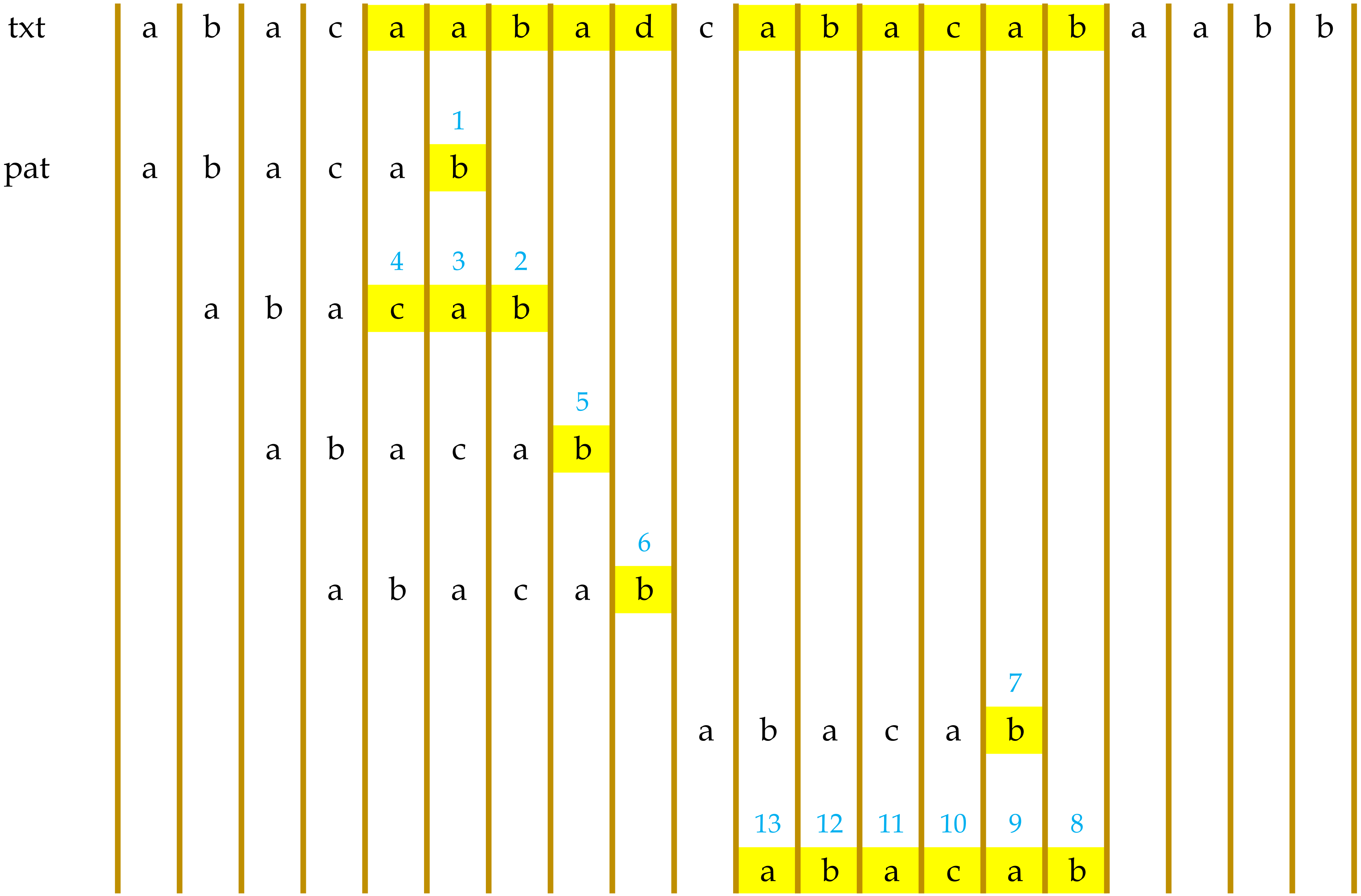
- Initialize all characters with shift of *m* (=length of pattern)
- Moving from right-to-left, update shifts of all but last character with the number of jumps required to reach the right-most character.
- If *t* is the character in *txt* that mismatched with the corresponding character of *pat*, d_1 is computed as following:

  $d_1 = bmBc(t) - k$, where *k* is the number of characters already matched.
- Example:

  pat = ABACAB

c	**A**	**B**	**C**	*****
bmBc(*c*)	1	4	2	6

Bad character table:



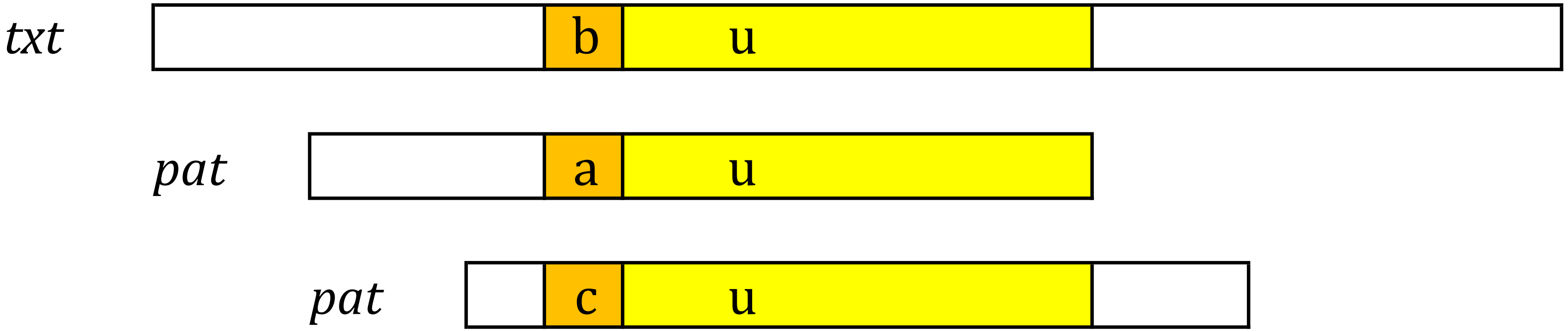
txt	a	b	a	c	a	a	b	a	d	c	a	b	a	c	a	b	a	a	b	b
						1														
pat	a	b	a	c	a	b														
					4	3	2													
		a	b	a	c	a	b													
								5												
			a	b	a	c	a	b												
									6											
				a	b	a	c	a	b											
															7					
										a	b	a	c	a	b					
											13	12	11	10	9	8				
											a	b	a	c	a	b				

Bad character table

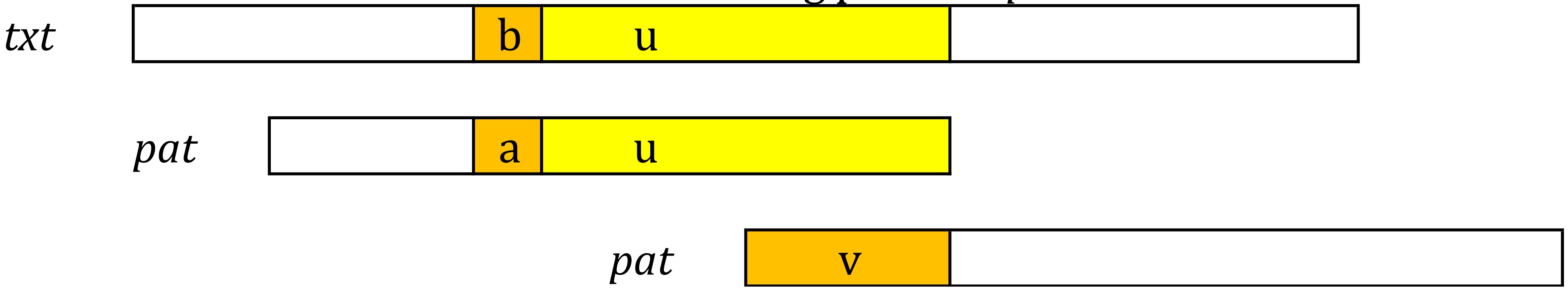
c	a	b	c	d
$bmBc(c)$	1	4	2	6

Good suffix shift:

1) u re-occurs preceeded by a character $a \neq c$



2) only a suffix of u re-occurs in x and as a matching prefix in *pat*



Good suffix table:

k	**Pattern**	d_2	***Requirement***
1	$\overline{\boldsymbol{G}}\ C\ A\ G\ A\ G\ A\ \boldsymbol{G}$	7	*G* not preceded by *A*
2	$G\ C\ \overline{\boldsymbol{A}\ \boldsymbol{G}}\ A\ G\ \boldsymbol{A}\ \boldsymbol{G}$	4	*AG* not preceded by *G*
3	$\overline{\boldsymbol{G}}\ C\ A\ G\ A\ \boldsymbol{G}\ \boldsymbol{A}\ \boldsymbol{G}$	7	*GAG* not preceded by *A*
4	$G\ C\ \overline{\boldsymbol{A}\ \boldsymbol{G}\ \boldsymbol{A}\ \boldsymbol{G}}\ \boldsymbol{A}\ \boldsymbol{G}$	2	*AGAG* not preceded by G
5	$\overline{\boldsymbol{G}}\ C\ A\ \boldsymbol{G}\ \boldsymbol{A}\ \boldsymbol{G}\ \boldsymbol{A}\ \boldsymbol{G}$	7	*GAGAG* not preceded by A
6	$\overline{\boldsymbol{G}}\ C\ \boldsymbol{A}\ \boldsymbol{G}\ \boldsymbol{A}\ \boldsymbol{G}\ \boldsymbol{A}\ \boldsymbol{G}$	7	*AGAGAG* not preceded by C
7	$\overline{\boldsymbol{G}}\ \boldsymbol{C}\ \boldsymbol{A}\ \boldsymbol{G}\ \boldsymbol{A}\ \boldsymbol{G}\ \boldsymbol{A}\ \boldsymbol{G}$	7	*CAGAGAG* not preceded by G

Good suffix table:

k	Pattern	d_2
1	$A\ B\ C\ \overline{\boldsymbol{B}}\ A\ B$	2
2	$\overline{\boldsymbol{A}\ \boldsymbol{B}}\ C\ B\ A\ B$	4
3	$\overline{\boldsymbol{A}\ \boldsymbol{B}}\ C\ B\ A\ B$	4
4	$\overline{\boldsymbol{A}\ \boldsymbol{B}}\ C\ B\ A\ B$	4
5	$\overline{\boldsymbol{A}}\ B\ C\ B\ A\ B$	4

k	Pattern	d_2
1	$B\ A\ O\ \overline{\boldsymbol{B}}\ A\ B$	2
2	$\overline{\boldsymbol{B}}\ A\ O\ B\ A\ B$	5
3	$\overline{\boldsymbol{B}}\ A\ O\ B\ A\ B$	5
4	$\overline{\boldsymbol{B}}\ A\ O\ B\ A\ B$	5
5	$\overline{\boldsymbol{B}}\ A\ O\ B\ A\ B$	5

Good suffix table:

k	Pattern	d_2
1	$A\,J\,\overline{\mathbf{I}}\,J\,\mathbf{I}$	2
2	$A\,\overline{\mathbf{J}\,\mathbf{I}}\,\mathbf{J}\,\mathbf{I}$	2
3	$A\,J\,\mathbf{I}\,\mathbf{J}\,\mathbf{I}$	5
4	$A\,\mathbf{J}\,\mathbf{I}\,\mathbf{J}\,\mathbf{I}$	5

Not optimal as $a = c$ (= I)

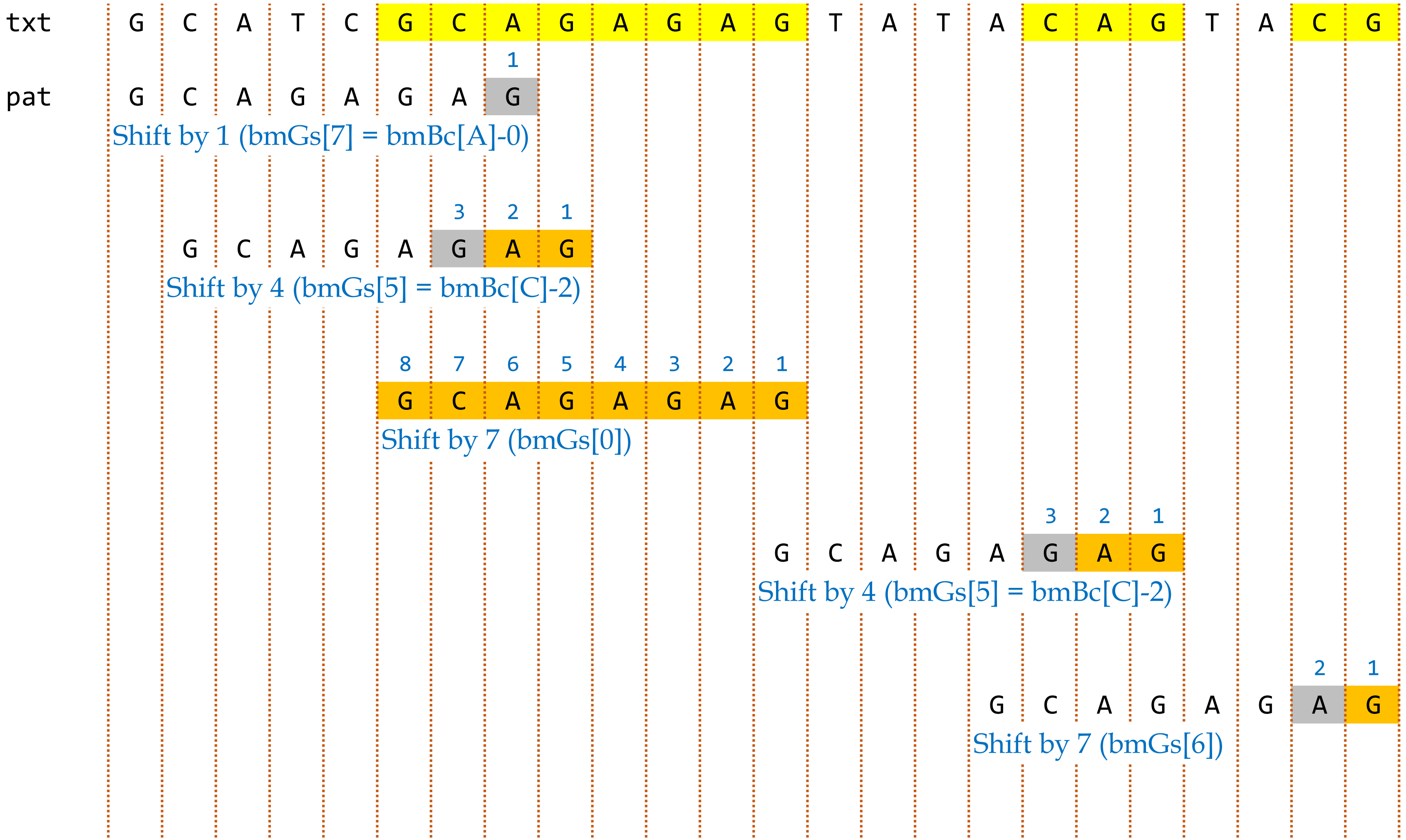
k	Pattern	d_2
1	$D\,R\,\overline{\mathbf{I}}\,D\,\mathbf{I}$	2
2	$D\,R\,I\,\mathbf{D}\,\mathbf{I}$	5
3	$D\,R\,\mathbf{I}\,\mathbf{D}\,\mathbf{I}$	5
4	$D\,\mathbf{R}\,\mathbf{I}\,\mathbf{D}\,\mathbf{I}$	5

k	Pattern	d_2
1	$S\,T\,O\,\mathbf{R}$	4
2	$S\,T\,\underline{\mathbf{O}\,\mathbf{R}}$	4
3	$S\,\underline{\mathbf{T}\,\mathbf{O}\,\mathbf{R}}$	4

Good suffix table:

k	**Pattern**	d_2	***Requirement***
1	$\overline{\boldsymbol{G}}$ *C A G A G A* ***G***	7	*G* not preceded by *A*
2	*G C* $\overline{\boldsymbol{A}\,\boldsymbol{G}}$ *A G* ***A G***	4	*AG* not preceded by *G*
3	$\overline{\boldsymbol{G}}$ *C A G A* ***G A G***	7	*GAG* not preceded by *A*
4	*G C* $\overline{\boldsymbol{A}\,\boldsymbol{G}\,\boldsymbol{A}\,\boldsymbol{G}}$ ***A G***	2	*AGAG* not preceded by G
5	$\overline{\boldsymbol{G}}$ *C A* ***G A G A G***	7	*GAGAG* not preceded by A
6	$\overline{\boldsymbol{G}}$ *C* ***A G A G A G***	7	*AGAGAG* not preceded by C
7	$\overline{\boldsymbol{G}}$ ***C A G A G A G***	7	*CAGAGAG* not preceded by G

Complete example

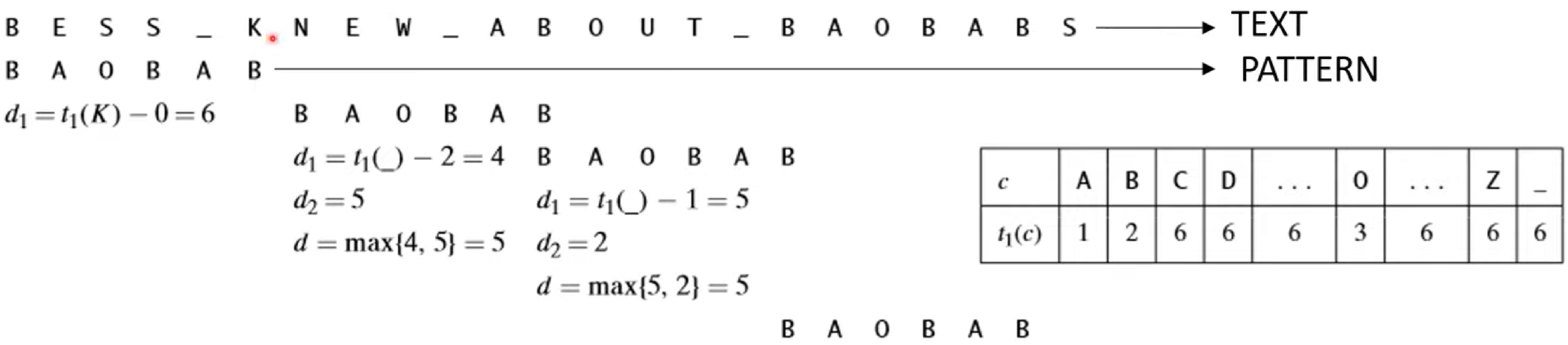


```
txt  G C A T C G C A G A G A G T A T A C A G T A C G
                   1
pat  G C A G A G A G
     Shift by 1 (bmGs[7] = bmBc[A]-0)

                 3 2 1
       G C A G A G A G
       Shift by 4 (bmGs[5] = bmBc[C]-2)

               8 7 6 5 4 3 2 1
               G C A G A G A G
               Shift by 7 (bmGs[0])

                                       3 2 1
                             G C A G A G A G
                             Shift by 4 (bmGs[5] = bmBc[C]-2)

                                                 2 1
                                     G C A G A G A G
                                     Shift by 7 (bmGs[6])
```

c	A	C	G	T
bmBc [*c*]	1	6	2	8

i	0	1	2	3	4	5	6	7
x [*i*]	G	C	A	G	A	G	A	G
suff [*i*]	1	0	0	2	0	4	0	8
bmGs [*i*]	7	7	7	2	7	4	7	1

Shift $k = \max\{d_1, d_2\}$



B E S S _ K N E W _ A B O U T _ B A O B A B S → TEXT

B A O B A B → PATTERN

$d_1 = t_1(K) - 0 = 6$ B A O B A B

$d_1 = t_1(\_) - 2 = 4$ B A O B A B

$d_2 = 5$ $d_1 = t_1(\_) - 1 = 5$

$d = \max\{4, 5\} = 5$ $d_2 = 2$

$d = \max\{5, 2\} = 5$

B A O B A B

c	A	B	C	D	...	O	...	Z	_
$t_1(c)$	1	2	6	6	6	3	6	6	6

c	**B**	**A**	**O**	*****
***bmBc*(*c*)**	1	4	3	6

$d_1 = bmBc(t) - k$

$d = \max\{d_1, d_2\}$

k	**Pattern**	***d*₂**
1	$B\ A\ O\ \overline{\boldsymbol{B}}\ A\ B$	2
2	$\overline{\boldsymbol{B}}\ A\ O\ B\ A\ B$	5
3	$\overline{\boldsymbol{B}}\ A\ O\ B\ A\ B$	5
4	$\overline{\boldsymbol{B}}\ A\ O\ B\ A\ B$	5
5	$\overline{\boldsymbol{B}}\ A\ O\ B\ A\ B$	5

See: https://www.youtube.com/watch?v=JITD8C2wLQY

Complexities?