Introduction to computational linguistics

Exercise session 1: “Algorithms for matching”

Thursday May 7 2009

Theoretical exercises

a. Show how to compute Z_i stepwise for $i > 1$ (using the notion of Z-boxes) for the following strings:

   i. AABCAABXAAZ

   ii. ABCDXABCYABDXY

b. Apply the Knuth-Morris-Pratt algorithm to find occurrences of ABXYABXZ in XABXYABXYABXZABXZABXYABXZA

a.i.) Z_i for AABCAABXAAZ

S = AABCAABXAAZ

Step 0)

Compute $Z_2(S)$ by comparing left-to-right S[2..|S|] and S[1..|S|] until a mismatch is found; $Z_2(S)$ is the length of that string. If $Z_2(S) > 0$ then r=r2=$Z_2(S)$+1 and l=2, else l=r=0

S	A	A	B	C	A	A	B	X	A	A	Z
	1	2	3	4	5	6	7	8	9	10	11
$Z_i(S)$	--	1									

$Z_2(S)$=1: { *A* **A** B ... } so l=2, r=$Z_2(S)$+1=1+1=2

a.i.) Z_i for AABCAABXAAZ

a.i.) Z_i for AABCAABXAAZ

Step 1)

a.i.) Z_i for AABCAABXAAZ

Step 1)

k > r: 3 > (r=2) so find $Z_3(S)$ by comparing S[3...|S|] to S[1..|S|] until a mismatch is found; if $Z_3(S) > 0$ then l=3, r=3+$Z_3(S)$-1

a.i.) Z_i for AABCAABXAAZ

Step 1)

k > r: 3 > (r=2) so find $Z_3(S)$ by comparing $S[3...|S|]$ to $S[1..|S|]$ until a mismatch is found; if $Z_3(S) > 0$ then l=3, $r=3+Z_3(S)-1$

S(3)='B' ≠ S(1)='A', hence $Z_3(S)=0$, l and r remain as they are: l=r=2

a.i.) Z_i for AABCAABXAAZ

Step 1)

k > r: 3 > (r=2) so find $Z_3(S)$ by comparing S[3...|S|] to S[1..|S|] until a mismatch is found; if $Z_3(S) > 0$ then l=3, r=3+$Z_3(S)$-1

S(3)='B' ≠ S(1)='A', hence $Z_3(S)$=0, l and r remain as they are: l=r=2

S	A	A	B	C	A	A	B	X	A	A	Z
	1	2	3	4	5	6	7	8	9	10	11
$Z_i(S)$	--	1	0								

$Z_3(S)$=0 so l=2, r=2

a.i.) Z_i for AABCAABXAAZ

a.i.) Z_i for AABCAABXAAZ

Step 1)

a.i.) Z_i for AABCAABXAAZ

Step 1)

k > r: 4 > (r=2) so find $Z_4(S)$ by comparing $S[4...|S|]$ to $S[1..|S|]$ until a mismatch is found; if $Z_4(S) > 0$ then l=4, r=4+$Z_4(S)$-1

a.i.) Z_i for AABCAABXAAZ

Step 1)

k > r: 4 > (r=2) so find $Z_4(S)$ by comparing S[4...|S|] to S[1..|S|] until a mismatch is found; if $Z_4(S) > 0$ then l=4, r=4+$Z_4(S)$-1

S(4)='C' ≠ S(1)='A', hence $Z_4(S)$=0, l and r remain as they are: l=r=2

a.i.) Z_i for AABCAABXAAZ

Step 1)

k > r: 4 > (r=2) so find $Z_4(S)$ by comparing S[4...|S|] to S[1..|S|] until a mismatch is found; if $Z_4(S) > 0$ then l=4, r=4+$Z_4(S)$-1

S(4)='C' ≠ S(1)='A', hence $Z_4(S)=0$, l and r remain as they are: l=r=2

S	A	A	B	C	A	A	B	X	A	A	Z
	1	2	3	4	5	6	7	8	9	10	11
$Z_i(S)$	--	1	0	0							

$Z_4(S)=0$ so l=2, r=2

a.i.) Z_i for AABCAABXAAZ

a.i.) Z_i for AABCAABXAAZ

Step 1)

a.i.) Z_i for AABCAABXAAZ

Step 1)

k > r: 5 > (r=2) so find $Z_5(S)$ by comparing S[5...|S|] to S[1..|S|] until a mismatch is found; if $Z_5(S) > 0$ then l=5, r=5+$Z_5(S)$-1

a.i.) Z_i for AABCAABXAAZ

Step 1)

k > r: 5 > (r=2) so find $Z_5(S)$ by comparing S[5...|S|] to S[1..|S|] until a mismatch is found; if $Z_5(S) > 0$ then l=5, r=5+$Z_5(S)$-1

S[5..7]=”A A B” matches S[1..3]=”A A B”, hence $Z_5(S)$=3, and l and r are set as follows: l=5, r=5+$Z_5(S)$-1=5+3-1=7

a.i.) Z_i for AABCAABXAAZ

Step 1)

k > r: 5 > (r=2) so find $Z_5(S)$ by comparing S[5...|S|] to S[1..|S|] until a mismatch is found; if $Z_5(S) > 0$ then l=5, r=5+$Z_5(S)$-1

S[5..7]=”A A B” matches S[1..3]=”A A B”, hence $Z_5(S)$=3, and l and r are set as follows: l=5, r=5+$Z_5(S)$-1=5+3-1=7

S	A	A	B	C	A	A	B	X	A	A	Z
	1	2	3	4	5	6	7	8	9	10	11
$Z_i(S)$	--	1	0	0	3						

$Z_5(S)$=3 so l=5, r=7

a.i.) Z_i for AABCAABXAAZ

a.i.) Z_i for AABCAABXAAZ

Step 2)

a.i.) Z_i for AABCAABXAAZ

Step 2)

6 ≤ (r=7): position k=6 is contained in a Z-box (namely, ”AAB”=S[5..7], with S(6)=’A’).

a.i.) Z_i for AABCAABXAAZ

Step 2)

6 ≤ (r=7): position k=6 is contained in a Z-box (namely, ”AAB”=S[5..7], with S(6)=’A’).

Hence S(6) also appears in k’=k-l=6-5+1=2: S(6)=S(2)=’A’

a.i.) Z_i for AABCAABXAAZ

Step 2)

6 ≤ (r=7): position k=6 is contained in a Z-box (namely, ”AAB”=S[5..7], with S(6)=’A’).

Hence S(6) also appears in k’=k-l=6-5+1=2: S(6)=S(2)=’A’

Therefore, S[6..7] must match S[2..3], which it does

a.i.) Z_i for AABCAABXAAZ

Step 2)

6 ≤ (r=7): position k=6 is contained in a Z-box (namely, ”AAB”=S[5..7], with S(6)=’A’).

Hence S(6) also appears in k’=k-l=6-5+1=2: S(6)=S(2)=’A’

Therefore, S[6..7] must match S[2..3], which it does

Furthermore, there must be a match to a prefix of S of length minimum [Z_2(S), lS[2..3]l], i.e. minimum [1,r-k+1=2] = 2

a.i.) Z_i for AABCAABXAAZ

Step 2)

6 ≤ (r=7): position k=6 is contained in a Z-box (namely, ”AAB”=S[5..7], with S(6)=’A’).

Hence S(6) also appears in k’=k-l=6-5+1=2: S(6)=S(2)=’A’

Therefore, S[6..7] must match S[2..3], which it does

Furthermore, there must be a match to a prefix of S of length minimum [Z_2(S), |S[2..3]|], i.e. minimum [1,r-k+1=2] = 2

Step 2a)

a.i.) Z_i for AABCAABXAAZ

Step 2)

$6 \le (r=7)$: position k=6 is contained in a Z-box (namely, ”AAB”=S[5..7], with S(6)=’A’).

Hence S(6) also appears in k’=k-l=6-5+1=2: S(6)=S(2)=’A’

Therefore, S[6..7] must match S[2..3], which it does

Furthermore, there must be a match to a prefix of S of length minimum [Z_2(S), |S[2..3]|], i.e. minimum [1,r-k+1=2] = 2

Step 2a)

Z_6(S)=Z_2(S)=1 which is smaller than the length of S[2..3], hence l and r stay the same

a.i.) Z_i for AABCAABXAAZ

Step 2)

6 ≤ (r=7): position k=6 is contained in a Z-box (namely, ”AAB”=S[5..7], with S(6)=’A’).

Hence S(6) also appears in k’=k-l=6-5+1=2: S(6)=S(2)=’A’

Therefore, S[6..7] must match S[2..3], which it does

Furthermore, there must be a match to a prefix of S of length minimum [Z_2(S), |S[2..3]|], i.e. minimum [1,r-k+1=2] = 2

Step 2a)

Z_6(S)=Z_2(S)=1 which is smaller than the length of S[2..3], hence l and r stay the same

S	A	A	B	C	A	A	B	X	A	A	Z
	1	2	3	4	5	6	7	8	9	10	11
Z_i(S)	--	1	0	0	3	1					

Z_6(S)=Z_2(S)=1 so l and r remain the same: l=5, r=7

a.i.) Z_i for AABCAABXAAZ

a.i.) Z_i for AABCAABXAAZ

Step 2)

a.i.) Z_i for AABCAABXAAZ

Step 2)

7 ≤ (r=7): position k=7 is contained in S[5..7], with S(7)='B'.

a.i.) Z_i for AABCAABXAAZ

Step 2)

7 ≤ (r=7): position k=7 is contained in S[5..7], with S(7)='B'.

Hence S(7) also appears in k'=k-l=7-5+1=3: S(7)=S(3)='B'

a.i.) Z_i for AABCAABXAAZ

Step 2)

7 ≤ (r=7): position k=7 is contained in S[5..7], with S(7)=’B’.

Hence S(7) also appears in k’=k-l=7-5+1=3: S(7)=S(3)=’B’

Therefore, S[7..7] must match S[3..3], i.e. S(7)=S(3), which it does

a.i.) Z_i for AABCAABXAAZ

Step 2)

7 ≤ (r=7): position k=7 is contained in S[5..7], with S(7)='B'.

Hence S(7) also appears in k'=k-l=7-5+1=3: S(7)=S(3)='B'

Therefore, S[7..7] must match S[3..3], i.e. S(7)=S(3), which it does

Furthermore, there must be a match to a prefix of S of length minimum [Z_3(S), |S[3..3]|], i.e. minimum [0,r-k+1=1] = 1

a.i.) Z_i for AABCAABXAAZ

Step 2)

7 ≤ (r=7): position k=7 is contained in S[5..7], with S(7)='B'.

Hence S(7) also appears in k'=k-l=7-5+1=3: S(7)=S(3)='B'

Therefore, S[7..7] must match S[3..3], i.e. S(7)=S(3), which it does

Furthermore, there must be a match to a prefix of S of length minimum [Z_3(S), |S[3..3]|], i.e. minimum [0,r-k+1=1] = 1

Step 2a)

a.i.) Z_i for AABCAABXAAZ

Step 2)

7 ≤ (r=7): position k=7 is contained in S[5..7], with S(7)='B'.

Hence S(7) also appears in k'=k-l=7-5+1=3: S(7)=S(3)='B'

Therefore, S[7..7] must match S[3..3], i.e. S(7)=S(3), which it does

Furthermore, there must be a match to a prefix of S of length minimum [Z_3(S), |S[3..3]|], i.e. minimum [0,r-k+1=1] = 1

Step 2a)

Z_7(S)=Z_3(S)=0 which is smaller than the length of S[3..3], hence l and r stay the same

a.i.) Z_i for AABCAABXAAZ

Step 2)

7 ≤ (r=7): position k=7 is contained in S[5..7], with S(7)='B'.

Hence S(7) also appears in k'=k-l=7-5+1=3: S(7)=S(3)='B'

Therefore, S[7..7] must match S[3..3], i.e. S(7)=S(3), which it does

Furthermore, there must be a match to a prefix of S of length minimum [Z_3(S), |S[3..3]|], i.e. minimum [0,r-k+1=1] = 1

Step 2a)

Z_7(S)=Z_3(S)=0 which is smaller than the length of S[3..3], hence l and r stay the same

S	A	A	B	C	A	A	B	X	A	A	Z
	1	2	3	4	5	6	7	8	9	10	11
Z_i(S)	--	1	0	0	3	1	0				

Z_7(S)=Z_3(S)=0 so l and r remain the same: l=5, r=7

a.i.) Z_i for AABCAABXAAZ

k=8 > (r=7) so step 1:

match S[8..|S|] to S[1..|S|]: mismatch, so Z_8(S)=0, l and r remain the same

S	A	A	B	C	A	A	B	X	A	A	Z
	1	2	3	4	5	6	7	8	9	10	11
Z_i(S)	--	1	0	0	3	1	0	0			

Z_8(S)=0 so l=5, r=7

k=9 > (r=7) so step 1:

match S[9..|S|] to S[1..|S|]: match S[9..10]=S[1..2], so Z_9(S)=2, l=9 and r=10

S	A	A	B	C	A	A	B	X	A	A	Z
	1	2	3	4	5	6	7	8	9	10	11
Z_i(S)	--	1	0	0	3	1	0	0	2		

Z_9(S)=2 so l=9, r=10

a.i.) Z_i for AABCAABXAAZ

k=10 ≤ (r=10) so step 2:

S(10) contained in S[9..10]; S(10) matches S(10-9+1)=S(2)='A'; $Z_2(S)=1 \geq |S[10..10]|=10-10+1=1$, hence **Step 2b)** but mismatch

S	A	A	B	C	A	A	B	X	A	A	Z
	1	2	3	4	5	6	7	8	9	10	11
$Z_i(S)$	--	1	0	0	3	1	0	0	2	1	

$Z_{10}(S)=1$

k=11 > (r=10) so step 1:

match S[11..|S|] to S[1..|S|]: mismatch so $Z_{11}(S)=0$

S	A	A	B	C	A	A	B	X	A	A	Z
	1	2	3	4	5	6	7	8	9	10	11
$Z_i(S)$	--	1	0	0	3	1	0	0	2	1	0

$Z_{11}(S)=0$

a.ii) Z_i for ABCDXABCYABDXY

a.ii) Z_i for ABCDXABCYABDXY

$Z_2(S)$: $S(2) \neq S(1)$ so $Z_2(S)=0$, $r=l=0$

a.ii) Z_i for ABCDXABCYABDXY

$Z_2(S)$: S(2) ≠ S(1) so $Z_2(S)$=0, r=l=0

S	A	B	C	D	X	A	B	C	Y	A	B	D	X	Y
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$Z_i(S)$	--	0												

$Z_2(S)$=0

a.ii) Z_i for ABCDXABCYABDXY

$Z_2(S)$: $S(2) \neq S(1)$ so $Z_2(S)=0$, r=l=0

S	A	B	C	D	X	A	B	C	Y	A	B	D	X	Y
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$Z_i(S)$	--	0												

$Z_2(S)=0$

i=3..5: $Z_i(S)$: $S(i) \neq S(1)$ so $Z_i(S)=0$, r=l=0

a.ii) Z_i for ABCDXABCYABDXY

$Z_2(S)$: $S(2) \neq S(1)$ so $Z_2(S)=0$, r=l=0

S	A	B	C	D	X	A	B	C	Y	A	B	D	X	Y
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$Z_i(S)$	--	0												

$Z_2(S)=0$

i=3..5: $Z_i(S)$: $S(i) \neq S(1)$ so $Z_i(S)=0$, r=l=0

S	A	B	C	D	X	A	B	C	Y	A	B	D	X	Y
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$Z_i(S)$	--	0	0	0	0									

$Z_{i=3..5}(S)=0$

a.ii) Z_i for ABCDXABCYABDXY

$Z_2(S)$: $S(2) \neq S(1)$ so $Z_2(S)=0$, r=l=0

S	A	B	C	D	X	A	B	C	Y	A	B	D	X	Y
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$Z_i(S)$	--	0												

$Z_2(S)=0$

i=3..5: $Z_i(S)$: $S(i) \neq S(1)$ so $Z_i(S)=0$, r=l=0

S	A	B	C	D	X	A	B	C	Y	A	B	D	X	Y
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$Z_i(S)$	--	0	0	0	0									

$Z_{i=3..5}(S)=0$

$Z_6(S)$: $S(6) = S(1)$: S[6..8] matches S[1..3], so $Z_6(S)=3$, l=6 and r=8

a.ii) Z_i for ABCDXABCYABDXY

$Z_2(S)$: $S(2) \neq S(1)$ so $Z_2(S)=0$, r=l=0

S	A	B	C	D	X	A	B	C	Y	A	B	D	X	Y
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$Z_i(S)$	--	0												

$Z_2(S)=0$

i=3..5: $Z_i(S)$: $S(i) \neq S(1)$ so $Z_i(S)=0$, r=l=0

S	A	B	C	D	X	A	B	C	Y	A	B	D	X	Y
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$Z_i(S)$	--	0	0	0	0									

$Z_{i=3..5}(S)=0$

$Z_6(S)$: $S(6) = S(1)$: $S[6..8]$ matches $S[1..3]$, so $Z_6(S)=3$, l=6 and r=8

S	A	B	C	D	X	A	B	C	Y	A	B	D	X	Y
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$Z_i(S)$	--	0	0	0	0	3								

$Z_6(S)=3$

a.ii) Z_i for ABCDXABCYABDXY

a.ii) Z_i for ABCDXABCYABDXY

$Z_7(S)$: 7 ≤ (r=8) hence S(7)=S(7-6+1)=S(2)='B', $Z_2(S)$=0 whereas lS[7..8]l=2, hence $Z_7(S)$=$Z_2(S)$=0 and l and remain as they are: l=6 and r=8

a.ii) Z_i for ABCDXABCYABDXY

$Z_7(S)$: 7 ≤ (r=8) hence S(7)=S(7-6+1)=S(2)='B', $Z_2(S)$=0 whereas |S[7..8]|=2, hence $Z_7(S)=Z_2(S)$=0 and l and remain as they are: l=6 and r=8

S	A	B	C	D	X	A	B	C	Y	A	B	D	X	Y
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$Z_i(S)$	--	0	0	0	0	3	0							

$Z_7(S)$=0

a.ii) Z_i for ABCDXABCYABDXY

Z_7(S): 7 ≤ (r=8) hence S(7)=S(7-6+1)=S(2)='B', Z_2(S)=0 whereas |S[7..8]|=2, hence Z_7(S)=Z_2(S)=0 and l and remain as they are: l=6 and r=8

S	A	B	C	D	X	A	B	C	Y	A	B	D	X	Y
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Z_i(S)	--	0	0	0	0	3	0							

Z_7(S)=0

Z_8(S): 8 ≤ (r=8) hence S(8)=S(8-6+1)=S(3)='C', Z_3(S)=0 whereas |S[8..8]|=1, hence Z_8(S)=Z_3(S)=0 and l and remain as they are: l=6 and r=8

a.ii) Z_i for ABCDXABCYABDXY

Z_7(S): 7 ≤ (r=8) hence S(7)=S(7-6+1)=S(2)='B', Z_2(S)=0 whereas |S[7..8]|=2, hence Z_7(S)=Z_2(S)=0 and l and remain as they are: l=6 and r=8

S	A	B	C	D	X	A	B	C	Y	A	B	D	X	Y
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Z_i(S)	--	0	0	0	0	3	0							

Z_7(S)=0

Z_8(S): 8 ≤ (r=8) hence S(8)=S(8-6+1)=S(3)='C', Z_3(S)=0 whereas |S[8..8]|=1, hence Z_8(S)=Z_3(S)=0 and l and remain as they are: l=6 and r=8

S	A	B	C	D	X	A	B	C	Y	A	B	D	X	Y
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Z_i(S)	--	0	0	0	0	3	0	0						

Z_8(S)=0

a.ii) Z_i for ABCDXABCYABDXY

$Z_7(S)$: 7 ≤ (r=8) hence S(7)=S(7-6+1)=S(2)='B', $Z_2(S)$=0 whereas |S[7..8]|=2, hence $Z_7(S)=Z_2(S)$=0 and l and remain as they are: l=6 and r=8

S	A	B	C	D	X	A	B	C	Y	A	B	D	X	Y
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$Z_i(S)$	--	0	0	0	0	3	0							

$Z_7(S)=0$

$Z_8(S)$: 8 ≤ (r=8) hence S(8)=S(8-6+1)=S(3)='C', $Z_3(S)$=0 whereas |S[8..8]|=1, hence $Z_8(S)=Z_3(S)$=0 and l and remain as they are: l=6 and r=8

S	A	B	C	D	X	A	B	C	Y	A	B	D	X	Y
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$Z_i(S)$	--	0	0	0	0	3	0	0						

$Z_8(S)=0$

$Z_9(S)$: 9 > (r=8) but S(9) ≠ S(1) hence $Z_9(S)$=0 and l and remain as they are: l=6 and r=8

a.ii) Z_i for ABCDXABCYABDXY

$Z_7(S)$: 7 ≤ (r=8) hence S(7)=S(7-6+1)=S(2)='B', $Z_2(S)=0$ whereas $|S[7..8]|=2$, hence $Z_7(S)=Z_2(S)=0$ and l and remain as they are: l=6 and r=8

S	A	B	C	D	X	A	B	C	Y	A	B	D	X	Y
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$Z_i(S)$	--	0	0	0	0	3	0							

$Z_7(S)=0$

$Z_8(S)$: 8 ≤ (r=8) hence S(8)=S(8-6+1)=S(3)='C', $Z_3(S)=0$ whereas $|S[8..8]|=1$, hence $Z_8(S)=Z_3(S)=0$ and l and remain as they are: l=6 and r=8

S	A	B	C	D	X	A	B	C	Y	A	B	D	X	Y
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$Z_i(S)$	--	0	0	0	0	3	0	0						

$Z_8(S)=0$

$Z_9(S)$: 9 > (r=8) but S(9) ≠ S(1) hence $Z_9(S)=0$ and l and remain as they are: l=6 and r=8

S	A	B	C	D	X	A	B	C	Y	A	B	D	X	Y
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$Z_i(S)$	--	0	0	0	0	3	0	0	0					

$Z_9(S)=0$

a.ii) Z_i for ABCDXABCYABDXY

a.ii) Z_i for ABCDXABCYABDXY

$Z_{10}(S)$: 10 > (r=8), S(10)=S(1), match S[10..1] with S[1..2], hence $Z_{10}(S)$=2 and l=10 and r=11

a.ii) Z_i for ABCDXABCYABDXY

$Z_{10}(S)$: 10 > (r=8), S(10)=S(1), match S[10..1] with S[1..2], hence $Z_{10}(S)$=2 and l=10 and r=11

S	A	B	C	D	X	A	B	C	Y	A	B	D	X	Y
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$Z_i(S)$	--	0	0	0	0	3	0	0	0	2				

$Z_{10}(S)$=0

a.ii) Z_i for ABCDXABCYABDXY

Z_{10}(S): 10 > (r=8), S(10)=S(1), match S[10..1] with S[1..2], hence Z_{10}(S)=2 and l=10 and r=11

S	A	B	C	D	X	A	B	C	Y	A	B	D	X	Y
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Z_i(S)	--	0	0	0	0	3	0	0	0	2				

Z_{10}(S)=0

Z_{11}(S): 11 ≤ (r=11) hence S(11)=S(11-10+1)=S(2)='B', Z_2(S)=0 whereas lS[11..11]l=1, hence Z_{11}(S)=Z_2(S)=0 and l and remain as they are: l=10 and r=11

a.ii) Z_i for ABCDXABCYABDXY

$Z_{10}(S)$: 10 > (r=8), S(10)=S(1), match S[10..1] with S[1..2], hence $Z_{10}(S)$=2 and l=10 and r=11

S	A	B	C	D	X	A	B	C	Y	A	B	D	X	Y
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$Z_i(S)$	--	0	0	0	0	3	0	0	0	2				

$Z_{10}(S)=0$

$Z_{11}(S)$: 11 ≤ (r=11) hence S(11)=S(11-10+1)=S(2)='B', $Z_2(S)$=0 whereas lS[11..11]l=1, hence $Z_{11}(S)=Z_2(S)$=0 and l and remain as they are: l=10 and r=11

S	A	B	C	D	X	A	B	C	Y	A	B	D	X	Y
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$Z_i(S)$	--	0	0	0	0	3	0	0	0	2	0			

$Z_{11}(S)=0$

a.ii) Z_i for ABCDXABCYABDXY

$Z_{10}(S)$: 10 > (r=8), S(10)=S(1), match S[10..1] with S[1..2], hence $Z_{10}(S)$=2 and l=10 and r=11

S	A	B	C	D	X	A	B	C	Y	A	B	D	X	Y
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$Z_i(S)$	--	0	0	0	0	3	0	0	0	2				

$Z_{10}(S)$=0

$Z_{11}(S)$: 11 ≤ (r=11) hence S(11)=S(11-10+1)=S(2)=’B’, $Z_2(S)$=0 whereas lS[11..11]l=1, hence $Z_{11}(S)$=$Z_2(S)$=0 and l and remain as they are: l=10 and r=11

S	A	B	C	D	X	A	B	C	Y	A	B	D	X	Y
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$Z_i(S)$	--	0	0	0	0	3	0	0	0	2	0			

$Z_{11}(S)$=0

i=12..14: $Z_i(S)$=0

a.ii) Z_i for ABCDXABCYABDXY

Z_{10}(S): 10 > (r=8), S(10)=S(1), match S[10..1] with S[1..2], hence Z_{10}(S)=2 and l=10 and r=11

S	A	B	C	D	X	A	B	C	Y	A	B	D	X	Y
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Z_i(S)	--	0	0	0	0	3	0	0	0	2				

Z_{10}(S)=0

Z_{11}(S): 11 ≤ (r=11) hence S(11)=S(11-10+1)=S(2)='B', Z_2(S)=0 whereas |S[11..11]|=1, hence Z_{11}(S)=Z_2(S)=0 and l and remain as they are: l=10 and r=11

S	A	B	C	D	X	A	B	C	Y	A	B	D	X	Y
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Z_i(S)	--	0	0	0	0	3	0	0	0	2	0			

Z_{11}(S)=0

i=12..14: Z_i(S)=0

S	A	B	C	D	X	A	B	C	Y	A	B	D	X	Y
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Z_i(S)	--	0	0	0	0	3	0	0	0	2	0	0	0	0

i=12..14: Z_i(S)=0

b) Knuth-Morris-Pratt

- "Apply the Knuth-Morris-Pratt algorithm to find occurrences of ABXYABXZ in XABXYABXYABXZABXZABXYABXZA"
- Pre-processing
  - For each position i in the pattern we need to define $sp_i(P)$ to be the length of the longest proper suffix of P[1..i] that matches a prefix of P.
  - *Optimization*: let $sp'_i(P)$ be $sp_i(P)$ with the added condition that characters P(i+1) and $P(sp'_i+1)$ are unequal
  - Compute $sp'_i(P)$ on the basis of the Z-values for the pattern; compute the failure function on the basis of the $sp'_i(P)$ values

The Knuth-Morris-Pratt algorithm

- Basic idea
  - Shift smarter than the naive method does

	1	2	3	4	5	6	7	8	9
P=	A	B	C	X	A	B	C	D	E

  - A mismatch with P(8) means we can shift 4 places
  - Deduction on P alone: no need to know T, or how P and T are aligned
- Complexity of the algorithm
  - The algorithm is linear, not -possibly- sublinear like Boyer-Moore
  - Extension: the Aho-Corasick algorithm for matching sets of patterns

The Knuth-Morris-Pratt algorithm

- Basic idea
  - Shift smarter than the naive method does

	1	2	3	4	5	6	7	8	9
P=	A	B	C	X	A	B	C	D	E
								⇧	

  - A mismatch with P(8) means we can shift 4 places
  - Deduction on P alone: no need to know T, or how P and T are aligned
- Complexity of the algorithm
  - The algorithm is linear, not -possibly- sublinear like Boyer-Moore
  - Extension: the Aho-Corasick algorithm for matching sets of patterns

The Knuth-Morris-Pratt algorithm

- Basic idea
  - Shift smarter than the naive method does

	1	2	3	4	5	6	7	8	9
P=	A	B	C	X	A	B	C	D	E

  - A mismatch with P(8) means we can shift 4 places
  - Deduction on P alone: no need to know T, or how P and T are aligned
- Complexity of the algorithm
  - The algorithm is linear, not -possibly- sublinear like Boyer-Moore
  - Extension: the Aho-Corasick algorithm for matching sets of patterns

The Knuth-Morris-Pratt algorithm

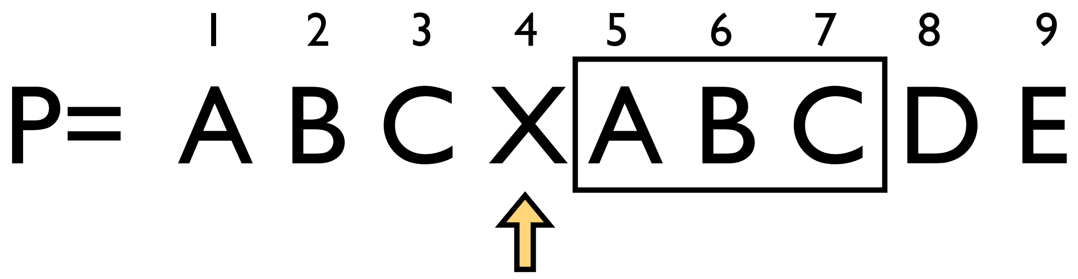
- Basic idea
  - Shift smarter than the naive method does

	1	2	3	4	5	6	7	8	9
P=	A	B	C	X	A	B	C	D	E
				⇧					

  - A mismatch with P(8) means we can shift 4 places
  - Deduction on P alone: no need to know T, or how P and T are aligned
- Complexity of the algorithm
  - The algorithm is linear, not -possibly- sublinear like Boyer-Moore
  - Extension: the Aho-Corasick algorithm for matching sets of patterns

The Knuth-Morris-Pratt algorithm

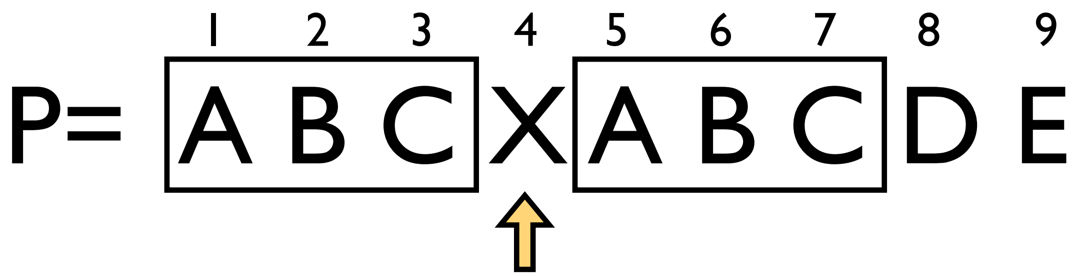
- Basic idea
  - Shift smarter than the naive method does



  - A mismatch with P(8) means we can shift 4 places
  - Deduction on P alone: no need to know T, or how P and T are aligned
- Complexity of the algorithm
  - The algorithm is linear, not -possibly- sublinear like Boyer-Moore
  - Extension: the Aho-Corasick algorithm for matching sets of patterns

The Knuth-Morris-Pratt algorithm

- Basic idea
  - Shift smarter than the naive method does



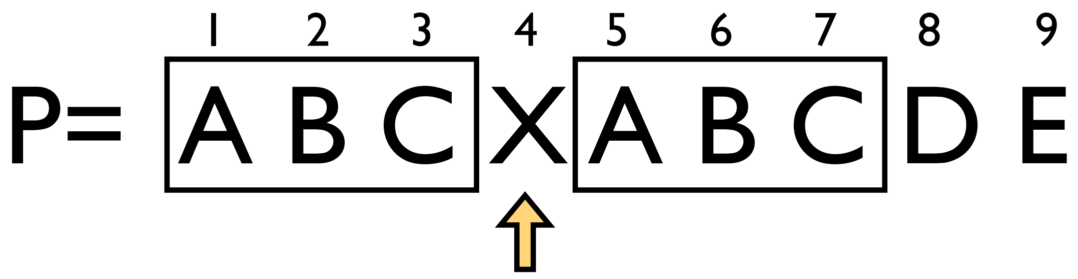
1 2 3 4 5 6 7 8 9

P= A B C X A B C D E

  - A mismatch with P(8) means we can shift 4 places
  - Deduction on P alone: no need to know T, or how P and T are aligned
- Complexity of the algorithm
  - The algorithm is linear, not -possibly- sublinear like Boyer-Moore
  - Extension: the Aho-Corasick algorithm for matching sets of patterns

The Knuth-Morris-Pratt algorithm

- Basic idea
  - Shift smarter than the naive method does



P= A B C X A B C D E

  - A mismatch with P(8) means we can shift 4 places (*like good suffix rule!*)
  - Deduction on P alone: no need to know T, or how P and T are aligned
- Complexity of the algorithm
  - The algorithm is linear, not -possibly- sublinear like Boyer-Moore
  - Extension: the Aho-Corasick algorithm for matching sets of patterns

Formalization

- Definition

For each position i in P, define $sp_i(P)$ to be the length of the longest proper suffix of P[1...i] that matches a prefix of P.

$$P= \begin{array}{ccccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\ A & B & C & A & E & A & B & C & A & B & D \end{array}$$

- Optimization

For each position i in P, define $sp'_i(P)$ to be the length of the longest proper suffix of P[1...i] that matches a prefix of P, with the added condition that characters P(i+1) and $P(sp'_i+1)$ are unequal.

Formalization

- Definition

For each position i in P, define $sp_i(P)$ to be the length of the longest proper suffix of P[1...i] that matches a prefix of P.

	1	2	3	4	5	6	7	8	9	10	11
P=	A	B	C	A	E	A	B	C	A	B	D
sp	0	0	0	1	0	1	2	3	4	2	0

- Optimization

For each position i in P, define $sp'_i(P)$ to be the length of the longest proper suffix of P[1...i] that matches a prefix of P, with the added condition that characters P(i+1) and $P(sp'_i+1)$ are unequal.

The Knuth-Morris-Pratt shift rule

- Alignment of P and T, left-to-right matching
- The shift rule:

For any alignment of P and T, if the first mismatch (comparing from left to right) occurs in position i+1 of P and position k of T, then shift P to the right (relative to T) so that P[1..sp_i’] aligns with T[k-sp_i’..k-1]. In other words, shift P exactly i+1-(sp_i’+1)=i-sp_i’ places to the right, so that character sp_i’+1 of P will align with character k in T. In the case that an occurrence of P has been found (no mismatch), shift P by n-sp_i’ places.

Computing the sp_i’ values

- Preprocessing using the Z values

  Position j > 1 maps to i if $i=j+Z_j(P)-1$. That is, j maps to i if i is the right end of a Z-box starting at j.

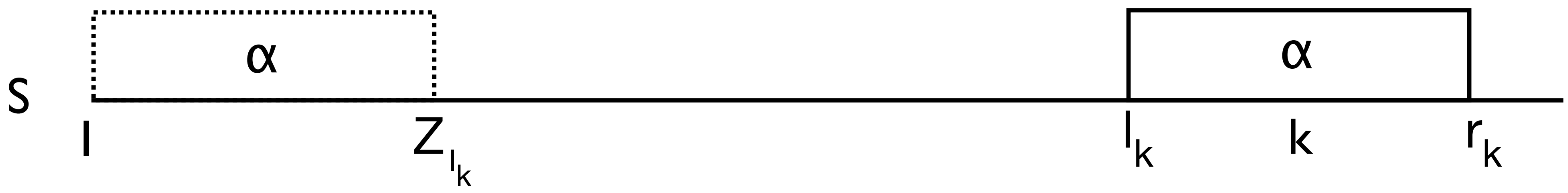
- Z-based Knuth-Morris-Pratt

```
for i := 1 to n do
    $sp_i$’ := 0;
for j := n downto 2 do
    i := j + $Z_j$(P) -1;
    $sp_i$’ := $Z_j$
```

Computing the sp_i’ values

- Preprocessing using the Z values

Position $j > 1$ maps to i if $i=j+Z_j(P)-1$. That is, j maps to i if i is the right end of a Z-box starting at j.



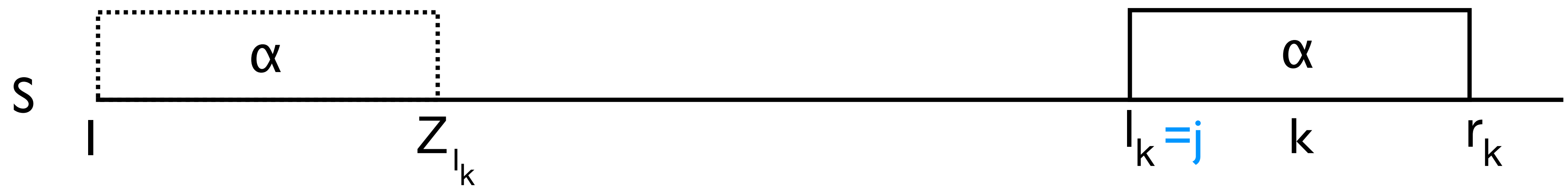
- Z-based Knuth-Morris-Pratt

```
for $i := 1$ to $n$ do
      $sp_i' := 0$;
for $j := n$ downto 2 do
      $i := j + Z_j(P) - 1$;
      $sp_i' := Z_j$
```

Computing the sp_i' values

- Preprocessing using the Z values

Position j > 1 maps to i if $i=j+Z_j(P)-1$. That is, j maps to i if i is the right end of a Z-box starting at j.



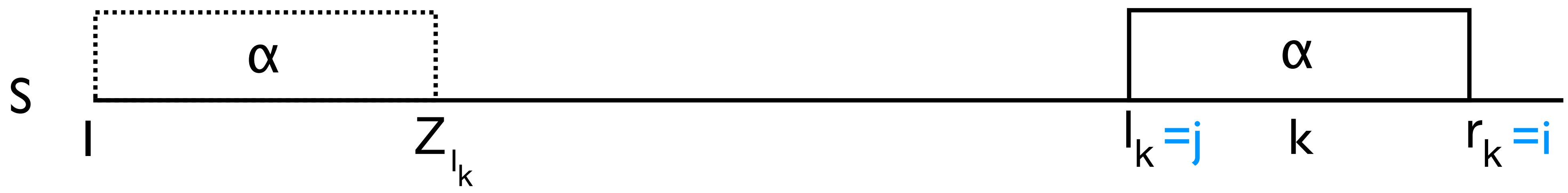
- Z-based Knuth-Morris-Pratt

```
for i := 1 to n do
    $sp_i$' := 0;
for j := n downto 2 do
    i := j + $Z_j$(P) -1;
    $sp_i$' := $Z_j$
```

Computing the sp_i' values

- Preprocessing using the Z values

Position $j > 1$ maps to i if $i=j+Z_j(P)-1$. That is, j maps to i if i is the right end of a Z-box starting at j.



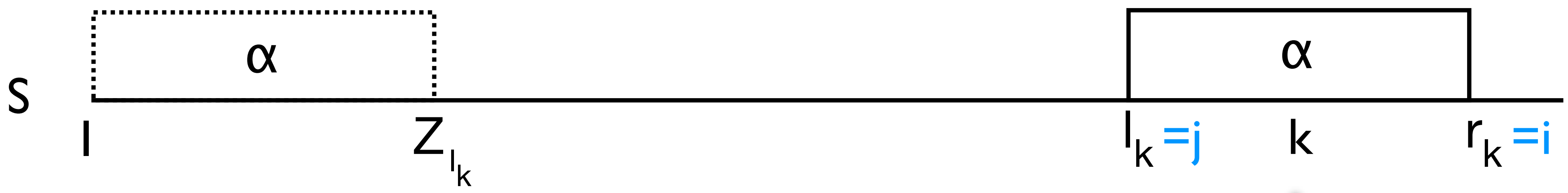
- Z-based Knuth-Morris-Pratt

```
for i := 1 to n do
    $sp_i' := 0$;
for j := n downto 2 do
    $i := j + Z_j(P) - 1$;
    $sp_i' := Z_j$
```

Computing the sp_i' values

- Preprocessing using the Z values

Position j > 1 maps to i if $i=j+Z_j(P)-1$. That is, j maps to i if i is the right end of a Z-box starting at j.



- Z-based Knuth-Morris-Pratt

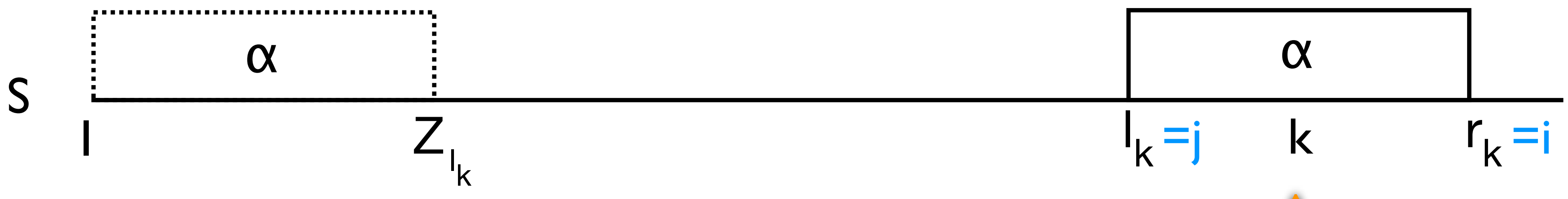
```
for i := 1 to n do
    sp_i' := 0;
for j := n downto 2 do
    i := j + Z_j(P) -1;
    sp_i' := Z_j
```

$sp'_i(P)$ is the length of the longest proper suffix of P[1...i] i.e. the length of the Z-box that starts at j (the suffix)

Computing the sp_i’ values

- Preprocessing using the Z values

Position j > 1 maps to i if $i=j+Z_j(P)-1$. That is, j maps to i if i is the right end of a Z-box starting at j.



- Z-based Knuth-Morris-Pratt

```
for i := 1 to n do
    $sp_i$’ := 0;
for j := n downto 2 do
    i := j + $Z_j(P)$ -1;
    $sp_i$’ := $Z_j$
```

$sp'_i(P)$ is the length of the longest proper suffix of P[1...i] i.e. the length of the Z-box that starts at j (the suffix)

The algorithm

- Preliminaries
  - Shifts through pointers: p points into P, c points into T
  - For each position i from 1 to n+1, define the failure function F'(i) to be $sp'_{i-1} + 1$ (and define $F(i)=sp_{i-1} + 1$); let sp_0' and sp_0 be 0.
- The algorithm

```
preprocess P to find $F'(k)=sp'_{k-1} + 1$ for k from 1 to n+1
    c := 1;
    p := 1;
    while c + (n-p) ≤ m do
        while P(p) = T(c) and p ≤ n
            p := p+1;
            c := c+1;
        if p = n+1 then
            report an occurrence of P starting at position c-n of T
        if p = 1 then c:=c+1
        p := F'(p)
```

Compute sp'$_i$(P) from Z for P

The Z-values are as follows, we only have a Z-box starting at l=5: Z_5(S)=3

Compute $sp'_i(P)$ from Z for P

The Z-values are as follows, we only have a Z-box starting at l=5: $Z_5(S)=3$

S	A	B	X	Y	A	B	X	Z
	1	2	3	4	5	6	7	8
$Z_i(S)$	--	0	0	0	3	0	0	0

Compute sp'$_i$(P) from Z for P

The Z-values are as follows, we only have a Z-box starting at l=5: Z_5(S)=3

S	A	B	X	Y	A	B	X	Z
	1	2	3	4	5	6	7	8
Z_i(S)	--	0	0	0	3	0	0	0

Z-based Knuth-Morris-Pratt

```
for i := 1 to n do
    $sp_i'$ := 0;
for j := n downto 2 do
    i := j + $Z_j$(P) -1;
    $sp_i'$ := $Z_j$
```

Compute sp'$_i$(P) from Z for P

The Z-values are as follows, we only have a Z-box starting at l=5: $Z_5(S)=3$

S	A	B	X	Y	A	B	X	Z
	1	2	3	4	5	6	7	8
$Z_i(S)$	--	0	0	0	3	0	0	0

Z-based Knuth-Morris-Pratt

```
for i := 1 to n do
    $sp_i' := 0$;
for j := n downto 2 do
    $i := j + Z_j(P) - 1$;
    $sp_i' := Z_j$
```

j

$Z_j(S)$

i

sp_i'

Compute $sp'_i(P)$ from Z for P

The Z-values are as follows, we only have a Z-box starting at l=5: $Z_5(S)=3$

S	A	B	X	Y	A	B	X	Z
	1	2	3	4	5	6	7	8
$Z_i(S)$	--	0	0	0	3	0	0	0

Z-based Knuth-Morris-Pratt

```
for i := 1 to n do
    $sp_i'$ := 0;
for j := n downto 2 do
    i := j + $Z_j(P)$ -1;
    $sp_i'$ := $Z_j$
```

j								
$Z_j(S)$								
i								
sp_i'								

Compute $sp'_i(P)$ from Z for P

The Z-values are as follows, we only have a Z-box starting at I=5: $Z_5(S)=3$

S	A	B	X	Y	A	B	X	Z
	1	2	3	4	5	6	7	8
$Z_i(S)$	--	0	0	0	3	0	0	0

Z-based Knuth-Morris-Pratt

```
for i := 1 to n do
    $sp_i'$ := 0;
for j := n downto 2 do
    i := j + $Z_j(P)$ -1;
    $sp_i'$ := $Z_j$
```

j	8							
$Z_j(S)$	0							
i	8+0-1							
sp_i'	0							

Compute $sp'_i(P)$ from Z for P

The Z-values are as follows, we only have a Z-box starting at l=5: $Z_5(S)=3$

S	A	B	X	Y	A	B	X	Z
	I	2	3	4	5	6	7	8
$Z_i(S)$	--	0	0	0	3	0	0	0

Z-based Knuth-Morris-Pratt

```
for i := 1 to n do
    $sp_i'$ := 0;
for j := n downto 2 do
    i := j + $Z_j(P)$ -1;
    $sp_i'$ := $Z_j$
```

j	8	7						
$Z_j(S)$	0	0						
i	8+0-I	7+0-I						
sp_i'	0	3						

Compute $sp'_i(P)$ from Z for P

The Z-values are as follows, we only have a Z-box starting at l=5: $Z_5(S)=3$

S	A	B	X	Y	A	B	X	Z
	1	2	3	4	5	6	7	8
$Z_i(S)$	--	0	0	0	3	0	0	0

Z-based Knuth-Morris-Pratt

```
for i := 1 to n do
    $sp_i'$ := 0;
for j := n downto 2 do
    i := j + $Z_j(P)$ -1;
    $sp_i'$ := $Z_j$
```

j	8	7	6					
$Z_j(S)$	0	0	0					
i	8+0-1	7+0-1	6+0-1					
sp_i'	0	3	0					

Compute $sp'_i(P)$ from Z for P

The Z-values are as follows, we only have a Z-box starting at l=5: $Z_5(S)=3$

S	A	B	X	Y	A	B	X	Z
	1	2	3	4	5	6	7	8
$Z_i(S)$	--	0	0	0	3	0	0	0

Z-based Knuth-Morris-Pratt

```
for i := 1 to n do
    sp_i' := 0;
for j := n downto 2 do
    i := j + Z_j(P) -1;
    sp_i' := Z_j
```

j	8	7	6	5				
$Z_j(S)$	0	0	0	3				
i	8+0-1	7+0-1	6+0-1	5+3-1				
sp_i'	0	3	0	0				

Compute sp'$_i$(P) from Z for P

The Z-values are as follows, we only have a Z-box starting at l=5: $Z_5(S)=3$

S	A	B	X	Y	A	B	X	Z
	1	2	3	4	5	6	7	8
$Z_i(S)$	--	0	0	0	3	0	0	0

Z-based Knuth-Morris-Pratt

```
for i := 1 to n do
    $sp_i'$ := 0;
for j := n downto 2 do
    i := j + $Z_j(P)$ -1;
    $sp_i'$ := $Z_j$
```

j	8	7	6	5	4			
$Z_j(S)$	0	0	0	3	0			
i	8+0-1	7+0-1	6+0-1	5+3-1	4+0-1			
sp_i'	0	3	0	0	0			

Compute $sp'_i(P)$ from Z for P

The Z-values are as follows, we only have a Z-box starting at l=5: $Z_5(S)=3$

S	A	B	X	Y	A	B	X	Z
	1	2	3	4	5	6	7	8
$Z_i(S)$	--	0	0	0	3	0	0	0

Z-based Knuth-Morris-Pratt

```
for i := 1 to n do
    $sp_i'$ := 0;
for j := n downto 2 do
    i := j + $Z_j(P)$ -1;
    $sp_i'$ := $Z_j$
```

j	8	7	6	5	4	3		
$Z_j(S)$	0	0	0	3	0	0		
i	8+0-1	7+0-1	6+0-1	5+3-1	4+0-1	3+0-1		
sp_i'	0	3	0	0	0	0		

Compute $sp'_i(P)$ from Z for P

The Z-values are as follows, we only have a Z-box starting at l=5: $Z_5(S)=3$

S	A	B	X	Y	A	B	X	Z
	1	2	3	4	5	6	7	8
$Z_i(S)$	--	0	0	0	3	0	0	0

Z-based Knuth-Morris-Pratt

```
for i := 1 to n do
    $sp_i'$ := 0;
for j := n downto 2 do
    i := j + $Z_j(P)$ -1;
    $sp_i'$ := $Z_j$
```

j	8	7	6	5	4	3	2	
$Z_j(S)$	0	0	0	3	0	0	0	
i	8+0-1	7+0-1	6+0-1	5+3-1	4+0-1	3+0-1	2+0-1	
sp_i'	0	3	0	0	0	0	0	

Compute $sp'_i(P)$ from Z for P

The Z-values are as follows, we only have a Z-box starting at l=5: $Z_5(S)=3$

S	A	B	X	Y	A	B	X	Z
	1	2	3	4	5	6	7	8
$Z_i(S)$	--	0	0	0	3	0	0	0

Z-based Knuth-Morris-Pratt

```
for i := 1 to n do
    $sp_i'$ := 0;
for j := n downto 2 do
    i := j + $Z_j(P)$ -1;
    $sp_i'$ := $Z_j$
```

j	8	7	6	5	4	3	2	1
$Z_j(S)$	0	0	0	3	0	0	0	0
i	8+0-1	7+0-1	6+0-1	5+3-1	4+0-1	3+0-1	2+0-1	--
sp_i'	0	3	0	0	0	0	0	0

Compute $sp'_i(P)$ from Z for P

The Z-values are as follows, we only have a Z-box starting at l=5: $Z_5(S)=3$

S	A	B	X	Y	A	B	X	Z
	1	2	3	4	5	6	7	8
$Z_i(S)$	--	0	0	0	3	0	0	0

Z-based Knuth-Morris-Pratt

```
for i := 1 to n do
    $sp_i'$ := 0;
for j := n downto 2 do
    i := j + $Z_j(P)$ -1;
    $sp_i'$ := $Z_j$
```

j	8	7	6	5	4	3	2	1
$Z_j(S)$	0	0	0	3	0	0	0	0
i	8+0-1	7+0-1	6+0-1	5+3-1	4+0-1	3+0-1	2+0-1	--
sp_i'	0	3	0	0	0	0	0	0

$sp_7'=Z_8(P)=0$; $sp_6'=Z_7(P)=0$;

$sp_5'=Z_6(P)=0$;**$sp_7'=Z_5(P)=3$**;

$sp_3'=Z_4(P)=0$;...;$sp_1'=Z_2(P)=0$

Compute F'(k) from sp'$_i$(P)

- Failure function F'(k)=sp'$_{k-1}$+1 for k from 1 to n+1

k	--	1	2	3	4	5	6	7	8	9
sp$_i$'	0	0	0	0	0	2	2	0	0	--
F'(k)	--	1	1	1	1	1	1	1	4	1

The Knuth-Morris-Pratt algorithm

```
preprocess P to find $F'(k)=sp'_{k-1} + 1$ for k from 1 to n+1
    c := 1;
    p := 1;
    while $c + (n-p) \leq m$ do
        while $P(p) = T(c)$ and $p \leq n$
            p := p+1;
            c := c+1;
        if p = n+1 then
            report an occurrence of P starting at position c-n of T
        if p = 1 then c:=c+1
        p := F'(p)
```

The Knuth-Morris-Pratt algorithm

```
preprocess P to find $F'(k)=sp'_{k-1} + 1$ for k from 1 to n+1
    c := 1;
    p := 1;
    while $c + (n-p) \leq m$ do
        while $P(p) = T(c)$ and $p \leq n$                MatchChar (MC)
            p := p+1;
            c := c+1;
        if p = n+1 then
            report an occurrence of P starting at position c-n of T
        if p = 1 then c:=c+1
        p := F'(p)
```

The Knuth-Morris-Pratt algorithm

```
preprocess P to find $F'(k)=sp'_{k-1} + 1$ for k from 1 to n+1
    c := 1;
    p := 1;
    while $c + (n-p) \leq m$ do
        while $P(p) = T(c)$ and $p \leq n$                                  MatchChar (MC)
            p := p+1;
            c := c+1;
                                                                   MatchPattern (MP)
        if p = n+1 then
            report an occurrence of P starting at position c-n of T
        if p = 1 then c:=c+1
        p := F'(p)
```

The Knuth-Morris-Pratt algorithm

preprocess P to find $F'(k)=sp'_{k-1} + 1$ for k from 1 to n+1

```
c := 1;
p := 1;
while c + (n-p) ≤ m do
    while P(p) = T(c) and p ≤ n                      MatchChar (MC)
        p := p+1;
        c := c+1;
    if p = n+1 then                                  MatchPattern (MP)
        report an occurrence of P starting at position c-n of T
    if p = 1 then c:=c+1                             MismatchStart (F0)
    p := F'(p)
```

The Knuth-Morris-Pratt algorithm

```
preprocess P to find $F'(k)=sp'_{k-1} + 1$ for k from 1 to $n+1$
    $c := 1$;
    $p := 1$;
    while $c + (n-p) \leq m$ do
        while $P(p) = T(c)$ and $p \leq n$                    MatchChar (MC)
            $p := p+1$;
            $c := c+1$;
        if $p = n+1$ then                                    MatchPattern (MP)
            report an occurrence of P starting at position $c-n$ of T
        if $p = 1$ then $c:=c+1$                             MismatchStart (F0)
        $p := F'(p)$                                         MismatchInternal (Fi)
```

K-M-P matching P against T

c=1, p=1: F0

c=2, p=1: MC

c=3, p=2: MC

c=4, p=3: MC

c=5, p=4: MC

c=6, p=5: MC

c=7, p=6: MC

c=8, p=7: MC

c=9, p=8: Fi: F’(8)=4 ⇒ p=4

c=9, p=4: MC

c=10, p=5: MC

c=11, p=6: MC

c=12, p=7: MC

c=13, p=8: MC

c=14, p=9: MP

K-M-P matching P against T

c=1, p=1: F0

c=2, p=1: MC

c=3, p=2: MC

c=4, p=3: MC

c=5, p=4: MC

c=6, p=5: MC

c=7, p=6: MC

c=8, p=7: MC

c=9, p=8: Fi: F’(8)=4 ⇒ p=4

c=9, p=4: MC

c=10, p=5: MC

c=11, p=6: MC

c=12, p=7: MC

c=13, p=8: MC

c=14, p=9: MP

K-M-P matching P against T

	X	A	B	X	Y	A	B	X	Y	A	B	X	Z	A	B	X	Z	A	B	X	Y	A	B	X	Z
c=1, p=1: F0																									
c=2, p=1: MC																									
c=3, p=2: MC																									
c=4, p=3: MC																									
c=5, p=4: MC																									
c=6, p=5: MC																									
c=7, p=6: MC																									
c=8, p=7: MC																									
c=9, p=8: Fi: F'(8)=4 ⇒ p=4																									
c=9, p=4: MC																									
c=10, p=5: MC																									
c=11, p=6: MC																									
c=12, p=7: MC																									
c=13, p=8: MC																									
c=14, p=9: MP																									

K-M-P matching P against T

	X	A	B	X	Y	A	B	X	Y	A	B	X	Z	A	B	X	Z	A	B	X	Y	A	B	X	Z
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
c=1, p=1: F0																									
c=2, p=1: MC																									
c=3, p=2: MC																									
c=4, p=3: MC																									
c=5, p=4: MC																									
c=6, p=5: MC																									
c=7, p=6: MC																									
c=8, p=7: MC																									
c=9, p=8: Fi: F’(8)=4 ⇒ p=4																									
c=9, p=4: MC																									
c=10, p=5: MC																									
c=11, p=6: MC																									
c=12, p=7: MC																									
c=13, p=8: MC																									
c=14, p=9: MP																									

K-M-P matching P against T

	X	A	B	X	Y	A	B	X	Y	A	B	X	Z	A	B	X	Z	A	B	X	Y	A	B	X	Z
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
c=1, p=1: F0	**A**	B	X	Y	A	B	X	Z																	
c=2, p=1: MC																									
c=3, p=2: MC																									
c=4, p=3: MC																									
c=5, p=4: MC																									
c=6, p=5: MC																									
c=7, p=6: MC																									
c=8, p=7: MC																									
c=9, p=8: Fi: F'(8)=4 ⇒ p=4																									
c=9, p=4: MC																									
c=10, p=5: MC																									
c=11, p=6: MC																									
c=12, p=7: MC																									
c=13, p=8: MC																									
c=14, p=9: MP																									

K-M-P matching P against T

	X	A	B	X	Y	A	B	X	Y	A	B	X	Z	A	B	X	Z	A	B	X	Y	A	B	X	Z
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
c=1, p=1: F0	**A**	B	X	Y	A	B	X	Z																	
c=2, p=1: MC		**A**	B	X	Y	A	B	X	Z																
c=3, p=2: MC																									
c=4, p=3: MC																									
c=5, p=4: MC																									
c=6, p=5: MC																									
c=7, p=6: MC																									
c=8, p=7: MC																									
c=9, p=8: Fi: F'(8)=4 ⇒ p=4																									
c=9, p=4: MC																									
c=10, p=5: MC																									
c=11, p=6: MC																									
c=12, p=7: MC																									
c=13, p=8: MC																									
c=14, p=9: MP																									

K-M-P matching P against T

	X	A	B	X	Y	A	B	X	Y	A	B	X	Z	A	B	X	Z	A	B	X	Y	A	B	X	Z
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
c=1, p=1: F0	**A**	B	X	Y	A	B	X	Z																	
c=2, p=1: MC		**A**	B	X	Y	A	B	X	Z																
c=3, p=2: MC		A	**B**	X	Y	A	B	X	Z																
c=4, p=3: MC																									
c=5, p=4: MC																									
c=6, p=5: MC																									
c=7, p=6: MC																									
c=8, p=7: MC																									
c=9, p=8: Fi: F'(8)=4 ⇒ p=4																									
c=9, p=4: MC																									
c=10, p=5: MC																									
c=11, p=6: MC																									
c=12, p=7: MC																									
c=13, p=8: MC																									
c=14, p=9: MP																									

K-M-P matching P against T

	X	A	B	X	Y	A	B	X	Y	A	B	X	Z	A	B	X	Z	A	B	X	Y	A	B	X	Z
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
c=1, p=1: F0	**A**	B	X	Y	A	B	X	Z																	
c=2, p=1: MC		**A**	B	X	Y	A	B	X	Z																
c=3, p=2: MC		A	**B**	X	Y	A	B	X	Z																
c=4, p=3: MC		A	B	**X**	Y	A	B	X	Z																
c=5, p=4: MC																									
c=6, p=5: MC																									
c=7, p=6: MC																									
c=8, p=7: MC																									
c=9, p=8: Fi: F’(8)=4 ⇒ p=4																									
c=9, p=4: MC																									
c=10, p=5: MC																									
c=11, p=6: MC																									
c=12, p=7: MC																									
c=13, p=8: MC																									
c=14, p=9: MP																									

K-M-P matching P against T

	X	A	B	X	Y	A	B	X	Y	A	B	X	Z	A	B	X	Z	A	B	X	Y	A	B	X	Z
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
c=1, p=1: F0	**A**	B	X	Y	A	B	X	Z																	
c=2, p=1: MC		**A**	B	X	Y	A	B	X	Z																
c=3, p=2: MC		A	**B**	X	Y	A	B	X	Z																
c=4, p=3: MC		A	B	**X**	Y	A	B	X	Z																
c=5, p=4: MC		A	B	X	**Y**	A	B	X	Z																
c=6, p=5: MC																									
c=7, p=6: MC																									
c=8, p=7: MC																									
c=9, p=8: Fi: F'(8)=4 ⇒ p=4																									
c=9, p=4: MC																									
c=10, p=5: MC																									
c=11, p=6: MC																									
c=12, p=7: MC																									
c=13, p=8: MC																									
c=14, p=9: MP																									

K-M-P matching P against T

	X	A	B	X	Y	A	B	X	Y	A	B	X	Z	A	B	X	Z	A	B	X	Y	A	B	X	Z
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
c=1, p=1: F0	**A**	B	X	Y	A	B	X	Z																	
c=2, p=1: MC		**A**	B	X	Y	A	B	X	Z																
c=3, p=2: MC		A	**B**	X	Y	A	B	X	Z																
c=4, p=3: MC		A	B	**X**	Y	A	B	X	Z																
c=5, p=4: MC		A	B	X	**Y**	A	B	X	Z																
c=6, p=5: MC		A	B	X	Y	**A**	B	X	Z																
c=7, p=6: MC																									
c=8, p=7: MC																									
c=9, p=8: Fi: F'(8)=4 ⇒ p=4																									
c=9, p=4: MC																									
c=10, p=5: MC																									
c=11, p=6: MC																									
c=12, p=7: MC																									
c=13, p=8: MC																									
c=14, p=9: MP																									

K-M-P matching P against T

	X	A	B	X	Y	A	B	X	Y	A	B	X	Z	A	B	X	Z	A	B	X	Y	A	B	X	Z
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
c=1, p=1: F0	**A**	B	X	Y	A	B	X	Z																	
c=2, p=1: MC		**A**	B	X	Y	A	B	X	Z																
c=3, p=2: MC		A	**B**	X	Y	A	B	X	Z																
c=4, p=3: MC		A	B	**X**	Y	A	B	X	Z																
c=5, p=4: MC		A	B	X	**Y**	A	B	X	Z																
c=6, p=5: MC		A	B	X	Y	**A**	B	X	Z																
c=7, p=6: MC		A	B	X	Y	A	**B**	X	Z																
c=8, p=7: MC																									
c=9, p=8: Fi: F’(8)=4 ⇒ p=4																									
c=9, p=4: MC																									
c=10, p=5: MC																									
c=11, p=6: MC																									
c=12, p=7: MC																									
c=13, p=8: MC																									
c=14, p=9: MP																									

K-M-P matching P against T

	X	A	B	X	Y	A	B	X	Y	A	B	X	Z	A	B	X	Z	A	B	X	Y	A	B	X	Z
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
c=1, p=1: F0	**A**	B	X	Y	A	B	X	Z																	
c=2, p=1: MC		**A**	B	X	Y	A	B	X	Z																
c=3, p=2: MC		A	**B**	X	Y	A	B	X	Z																
c=4, p=3: MC		A	B	**X**	Y	A	B	X	Z																
c=5, p=4: MC		A	B	X	**Y**	A	B	X	Z																
c=6, p=5: MC		A	B	X	Y	**A**	B	X	Z																
c=7, p=6: MC		A	B	X	Y	A	**B**	X	Z																
c=8, p=7: MC		A	B	X	Y	A	B	**X**	Z																
c=9, p=8: Fi: F’(8)=4 ⇒ p=4																									
c=9, p=4: MC																									
c=10, p=5: MC																									
c=11, p=6: MC																									
c=12, p=7: MC																									
c=13, p=8: MC																									
c=14, p=9: MP																									

K-M-P matching P against T

	X	A	B	X	Y	A	B	X	Y	A	B	X	Z	A	B	X	Z	A	B	X	Y	A	B	X	Z
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
c=1, p=1: F0	**A**	B	X	Y	A	B	X	Z																	
c=2, p=1: MC		**A**	B	X	Y	A	B	X	Z																
c=3, p=2: MC		A	**B**	X	Y	A	B	X	Z																
c=4, p=3: MC		A	B	**X**	Y	A	B	X	Z																
c=5, p=4: MC		A	B	X	**Y**	A	B	X	Z																
c=6, p=5: MC		A	B	X	Y	**A**	B	X	Z																
c=7, p=6: MC		A	B	X	Y	A	**B**	X	Z																
c=8, p=7: MC		A	B	X	Y	A	B	**X**	Z																
c=9, p=8: Fi: F’(8)=4 ⇒ p=4		A	B	X	Y	A	B	X	**Z**																
c=9, p=4: MC																									
c=10, p=5: MC																									
c=11, p=6: MC																									
c=12, p=7: MC																									
c=13, p=8: MC																									
c=14, p=9: MP																									

K-M-P matching P against T

	X	A	B	X	Y	A	B	X	Y	A	B	X	Z	A	B	X	Z	A	B	X	Y	A	B	X	Z
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
c=1, p=1: F0	**A**	B	X	Y	A	B	X	Z																	
c=2, p=1: MC		**A**	B	X	Y	A	B	X	Z																
c=3, p=2: MC		A	**B**	X	Y	A	B	X	Z																
c=4, p=3: MC		A	B	**X**	Y	A	B	X	Z																
c=5, p=4: MC		A	B	X	**Y**	A	B	X	Z																
c=6, p=5: MC		A	B	X	Y	**A**	B	X	Z																
c=7, p=6: MC		A	B	X	Y	A	**B**	X	Z																
c=8, p=7: MC		A	B	X	Y	A	B	**X**	Z																
c=9, p=8: Fi: F'(8)=4 ⇒ p=4		A	B	X	Y	A	B	X	**Z**																
c=9, p=4: MC						A	B	X	**Y**	A	B	X	Z												
c=10, p=5: MC																									
c=11, p=6: MC																									
c=12, p=7: MC																									
c=13, p=8: MC																									
c=14, p=9: MP																									

K-M-P matching P against T

	X	A	B	X	Y	A	B	X	Y	A	B	X	Z	A	B	X	Z	A	B	X	Y	A	B	X	Z
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
c=1, p=1: F0	**A**	B	X	Y	A	B	X	Z																	
c=2, p=1: MC		**A**	B	X	Y	A	B	X	Z																
c=3, p=2: MC		A	**B**	X	Y	A	B	X	Z																
c=4, p=3: MC		A	B	**X**	Y	A	B	X	Z																
c=5, p=4: MC		A	B	X	**Y**	A	B	X	Z																
c=6, p=5: MC		A	B	X	Y	**A**	B	X	Z																
c=7, p=6: MC		A	B	X	Y	A	**B**	X	Z																
c=8, p=7: MC		A	B	X	Y	A	B	**X**	Z																
c=9, p=8: Fi: F’(8)=4 ⇒ p=4		A	B	X	Y	A	B	X	**Z**																
c=9, p=4: MC						A	B	X	**Y**	A	B	X	Z												
c=10, p=5: MC						A	B	X	Y	**A**	B	X	Z												
c=11, p=6: MC																									
c=12, p=7: MC																									
c=13, p=8: MC																									
c=14, p=9: MP																									

K-M-P matching P against T

	X	A	B	X	Y	A	B	X	Y	A	B	X	Z	A	B	X	Z	A	B	X	Y	A	B	X	Z
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
c=1, p=1: F0	**A**	B	X	Y	A	B	X	Z																	
c=2, p=1: MC		**A**	B	X	Y	A	B	X	Z																
c=3, p=2: MC		A	**B**	X	Y	A	B	X	Z																
c=4, p=3: MC		A	B	**X**	Y	A	B	X	Z																
c=5, p=4: MC		A	B	X	**Y**	A	B	X	Z																
c=6, p=5: MC		A	B	X	Y	**A**	B	X	Z																
c=7, p=6: MC		A	B	X	Y	A	**B**	X	Z																
c=8, p=7: MC		A	B	X	Y	A	B	**X**	Z																
c=9, p=8: Fi: F’(8)=4 ⇒ p=4		A	B	X	Y	A	B	X	**Z**																
c=9, p=4: MC						A	B	X	**Y**	A	B	X	Z												
c=10, p=5: MC						A	B	X	Y	**A**	B	X	Z												
c=11, p=6: MC						A	B	X	Y	A	**B**	X	Z												
c=12, p=7: MC																									
c=13, p=8: MC																									
c=14, p=9: MP																									

K-M-P matching P against T

	X	A	B	X	Y	A	B	X	Y	A	B	X	Z	A	B	X	Z	A	B	X	Y	A	B	X	Z
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
c=1, p=1: F0	**A**	B	X	Y	A	B	X	Z																	
c=2, p=1: MC		**A**	B	X	Y	A	B	X	Z																
c=3, p=2: MC		A	**B**	X	Y	A	B	X	Z																
c=4, p=3: MC		A	B	**X**	Y	A	B	X	Z																
c=5, p=4: MC		A	B	X	**Y**	A	B	X	Z																
c=6, p=5: MC		A	B	X	Y	**A**	B	X	Z																
c=7, p=6: MC		A	B	X	Y	A	**B**	X	Z																
c=8, p=7: MC		A	B	X	Y	A	B	**X**	Z																
c=9, p=8: Fi: F’(8)=4 ⇒ p=4		A	B	X	Y	A	B	X	**Z**																
c=9, p=4: MC						A	B	X	**Y**	A	B	X	Z												
c=10, p=5: MC						A	B	X	Y	**A**	B	X	Z												
c=11, p=6: MC						A	B	X	Y	A	**B**	X	Z												
c=12, p=7: MC						A	B	X	Y	A	B	**X**	Z												
c=13, p=8: MC																									
c=14, p=9: MP																									

K-M-P matching P against T

	X	A	B	X	Y	A	B	X	Y	A	B	X	Z	A	B	X	Z	A	B	X	Y	A	B	X	Z
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
c=1, p=1: F0	**A**	B	X	Y	A	B	X	Z																	
c=2, p=1: MC		**A**	B	X	Y	A	B	X	Z																
c=3, p=2: MC		A	**B**	X	Y	A	B	X	Z																
c=4, p=3: MC		A	B	**X**	Y	A	B	X	Z																
c=5, p=4: MC		A	B	X	**Y**	A	B	X	Z																
c=6, p=5: MC		A	B	X	Y	**A**	B	X	Z																
c=7, p=6: MC		A	B	X	Y	A	**B**	X	Z																
c=8, p=7: MC		A	B	X	Y	A	B	**X**	Z																
c=9, p=8: Fi: F'(8)=4 ⇒ p=4		A	B	X	Y	A	B	X	**Z**																
c=9, p=4: MC						A	B	X	**Y**	A	B	X	Z												
c=10, p=5: MC						A	B	X	Y	**A**	B	X	Z												
c=11, p=6: MC						A	B	X	Y	A	**B**	X	Z												
c=12, p=7: MC						A	B	X	Y	A	B	**X**	Z												
c=13, p=8: MC						A	B	X	Y	A	B	X	**Z**												
c=14, p=9: MP																									

K-M-P matching P against T

	X	A	B	X	Y	A	B	X	Y	A	B	X	Z	A	B	X	Z	A	B	X	Y	A	B	X	Z
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
c=1, p=1: F0	**A**	B	X	Y	A	B	X	Z																	
c=2, p=1: MC		**A**	B	X	Y	A	B	X	Z																
c=3, p=2: MC		A	**B**	X	Y	A	B	X	Z																
c=4, p=3: MC		A	B	**X**	Y	A	B	X	Z																
c=5, p=4: MC		A	B	X	**Y**	A	B	X	Z																
c=6, p=5: MC		A	B	X	Y	**A**	B	X	Z																
c=7, p=6: MC		A	B	X	Y	A	**B**	X	Z																
c=8, p=7: MC		A	B	X	Y	A	B	**X**	Z																
c=9, p=8: Fi: F'(8)=4 ⇒ p=4		A	B	X	Y	A	B	X	**Z**																
c=9, p=4: MC						A	B	X	**Y**	A	B	X	Z												
c=10, p=5: MC						A	B	X	Y	**A**	B	X	Z												
c=11, p=6: MC						A	B	X	Y	A	**B**	X	Z												
c=12, p=7: MC						A	B	X	Y	A	B	**X**	Z												
c=13, p=8: MC						A	B	X	Y	A	B	X	**Z**												
c=14, p=9: MP						**A**	**B**	**X**	**Y**	**A**	**B**	**X**	**Z**												

K-M-P matching P against T

	X	A	B	X	Y	A	B	X	Y	A	B	X	Z	A	B	X	Z	A	B	X	Y	A	B	X	Z
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
c=1, p=1: F0	**A**	B	X	Y	A	B	X	Z																	
c=2, p=1: MC		**A**	B	X	Y	A	B	X	Z																
c=3, p=2: MC		A	**B**	X	Y	A	B	X	Z																
c=4, p=3: MC		A	B	**X**	Y	A	B	X	Z																
c=5, p=4: MC		A	B	X	**Y**	A	B	X	Z																
c=6, p=5: MC		A	B	X	Y	**A**	B	X	Z																
c=7, p=6: MC		A	B	X	Y	A	**B**	X	Z																
c=8, p=7: MC		A	B	X	Y	A	B	**X**	Z																
c=9, p=8: Fi: F’(8)=4 ⇒ p=4		A	B	X	Y	A	B	X	**Z**																
c=9, p=4: MC						A	B	X	**Y**	A	B	X	Z												
c=10, p=5: MC						A	B	X	Y	**A**	B	X	Z												
c=11, p=6: MC						A	B	X	Y	A	**B**	X	Z												
c=12, p=7: MC						A	B	X	Y	A	B	**X**	Z												
c=13, p=8: MC						A	B	X	Y	A	B	X	**Z**												
c=14, p=9: MP						**A**	**B**	**X**	**Y**	**A**	**B**	**X**	**Z**												

K-M-P matching P against T

	X	A	B	X	Y	A	B	X	Y	A	B	X	Z	A	B	X	Z	A	B	X	Y	A	B	X	Z
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
c=1, p=1: F0	**A**	B	X	Y	A	B	X	Z																	
c=2, p=1: MC		**A**	B	X	Y	A	B	X	Z																
c=3, p=2: MC		A	**B**	X	Y	A	B	X	Z																
c=4, p=3: MC		A	B	**X**	Y	A	B	X	Z																
c=5, p=4: MC		A	B	X	**Y**	A	B	X	Z																
c=6, p=5: MC		A	B	X	Y	**A**	B	X	Z																
c=7, p=6: MC		A	B	X	Y	A	**B**	X	Z																
c=8, p=7: MC		A	B	X	Y	A	B	**X**	Z																
c=9, p=8: Fi: F’(8)=4 ⇒ p=4		A	B	X	Y	A	B	X	**Z**																
c=9, p=4: MC						A	B	X	**Y**	A	B	X	Z												
c=10, p=5: MC						A	B	X	Y	**A**	B	X	Z												
c=11, p=6: MC						A	B	X	Y	A	**B**	X	Z												
c=12, p=7: MC						A	B	X	Y	A	B	**X**	Z												
c=13, p=8: MC						A	B	X	Y	A	B	X	**Z**												
c=14, p=9: MP						**A**	**B**	**X**	**Y**	**A**	**B**	**X**	**Z**												

K-M-P matching P against T

	X	A	B	X	Y	A	B	X	Y	A	B	X	Z	A	B	X	Z	A	B	X	Y	A	B	X	Z
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
c=1, p=1: F0	**A**	B	X	Y	A	B	X	Z																	
c=2, p=1: MC		**A**	B	X	Y	A	B	X	Z																
c=3, p=2: MC		A	**B**	X	Y	A	B	X	Z																
c=4, p=3: MC		A	B	**X**	Y	A	B	X	Z																
c=5, p=4: MC		A	B	X	**Y**	A	B	X	Z																
c=6, p=5: MC		A	B	X	Y	**A**	B	X	Z																
c=7, p=6: MC		A	B	X	Y	A	**B**	X	Z																
c=8, p=7: MC		A	B	X	Y	A	B	**X**	Z																
c=9, p=8: Fi: F’(8)=4 ⇒ p=4		A	B	X	Y	A	B	X	**Z**																
c=9, p=4: MC						A	B	X	**Y**	A	B	X	Z												
c=10, p=5: MC						A	B	X	Y	**A**	B	X	Z												
c=11, p=6: MC						A	B	X	Y	A	**B**	X	Z												
c=12, p=7: MC						A	B	X	Y	A	B	**X**	Z												
c=13, p=8: MC						A	B	X	Y	A	B	X	**Z**												
c=14, p=9: MP						**A**	**B**	**X**	**Y**	**A**	**B**	**X**	**Z**												
																		A	**B**	**X**	**Y**	**A**	**B**	**X**	**Z**

NaiveChars.java

```
public int match (String pattern, String text) {
    int p = 0;
    int s = 0;
    int t = 0;
    int matches = 0; // number of matches, return value
    while (t < text.length()) {
        if (p < pattern.length()) System.out.println("pattern("+p+") \t"+pattern.charAt(p));
        System.out.println("text("+t+") \t"+text.charAt(t));
        if (pattern.charAt(p) == text.charAt(t)) {
            // make sure to check against length-1 else OutOfRange exception!
            if (p < pattern.length()-1) {
                p = p+1;
                t = t+1;
            } else {
                System.out.println("Match found at position "+(s+1));
                p = 0;
                s = s+1;
                t = s;
                matches = matches+1;
            } // end if..else check for full occurrence of P
        } else {
            p = 1;
            s = s+1;
            t = s;
        } // end if..else check for character match
    } // end while over the text
    return matches;
} // end match
```

NaiveWords.java: initialization

```
public int match (String pattern, String text) {

    // ----------------------------------------
    // Initialization
    // ----------------------------------------
    int p = 0;
    int s = 0;
    int t = 0;
    int matches = 0; // number of matches, return value

    Vector patternVec = new Vector();
    Vector textVec = new Vector ();

    // Represent the pattern as a sequence of words

    StringTokenizer pst = new StringTokenizer(pattern);
    while (pst.hasMoreTokens()) {
        String word = (String)pst.nextToken();
        patternVec.addElement(word);
    } // end while

    // Represent the text as a sequence of words

    StringTokenizer tst = new StringTokenizer(text);
    while (tst.hasMoreTokens()) {
        String word = (String)tst.nextToken();
        textVec.addElement(word);
    } // end
```

NaiveWords.java: matching

```
    // ---------------------------------------
    // Loop
    // ---------------------------------------
    // Note that the conditions now refer to the vectors, not to the original strings.

    while (t < textVec.size()) {
        if (p < patternVec.size()) System.out.println("pattern("+p+") \t<"+(String)patternVec.elementAt(p)+">");
        System.out.println("text("+t+") \t<"+(String)textVec.elementAt(t)+">");
        if (((String)patternVec.elementAt(p)).equals((String)textVec.elementAt(t))) {
            // make sure to check against length-1 else OutOfRange exception!
            if (p < patternVec.size()-1) {
                p = p+1;
                t = t+1;
            } else {
                System.out.println("Match found at position "+(s+1));
                p = 0;
                s = s+1;
                t = s;
                matches = matches+1;
            } // end if..else check for full occurrence of P
        } else {
            p = 0;
            s = s+1;
            t = s;
        } // end if..else check for character match
    } // end while over the text
    return matches;
  } // end match
```

KeywordTreeNode.java

- records:
  - the label on the edge to the vertex *v*
  - the path, i.e. the concatenation of the words on the path to *v*: L(v)
  - the parent of the vertex
  - the children of a vertex i
- basic accessor methods for adding, getting and setting

KeywordTree.java: buildBranch

```
public KeywordTreeNode buildBranch (Vector pvec) {
   KeywordTreeNode broot  = new KeywordTreeNode ();
   KeywordTreeNode parent = broot;
   boolean rootSet = false;
   String path = "";
   Iterator pvIter = pvec.iterator();
   while (pvIter.hasNext()) {
      String word = (String)pvIter.next();
      // check whether the root has been set; if not, initialize
      // the root, otherwise create a new node, and add it to
      // the current parent.
      if (!rootSet) {
         broot.setEdge(word);
         rootSet = true;
         broot.setPath(word);
         path = word;
      } else {
         KeywordTreeNode node = new KeywordTreeNode(word);
         path = path+" "+word;
         node.setPath(path);
         node.setParent(parent);
         parent.addChild(node);
         // Set the parent to be the current node
         parent = node;
      } // end if..else check for root or child
   } // end while
   return broot;
} // end
```

KeywordTree.java: buildTree

```
        KeywordTreeNode branch = this.buildBranch(patternVec);

        // Next, go down the tree as far as possible to find the lowest attachment point for this branch. From the root
        // of the branch we go down tree, until we get to a point where none of the children on the branch would be
        // matched; that is where we insert the (remainder of the) branch.

        KeywordTreeNode branchnode = branch;   // the current branch node
        KeywordTreeNode attachment = treeroot; // the node where to attach
        boolean golower = (treeroot.isLeaf())?false:true;
        while (golower) {
            // cycle over the children of the current attachment node
            boolean matchfound = false;
            Iterator chIter = attachment.getChildren();
            while (chIter.hasNext() && !matchfound) {
                KeywordTreeNode child = (KeywordTreeNode) chIter.next();
                // if this child has the same edge, and the same
                // path, as the current node in the branch, then
                // decend one node down the branch and set the
                // current node as the attachment point for the
                // remainder of the branch.
                if (child.getEdge().equals(branchnode.getEdge()) && child.getPath().equals(branchnode.getPath())) {
                    matchfound = true;
                    attachment = child;
                    branchnode = branchnode.getFirstChild();
                } // end if.. check whether match found
            } // end while over children
            // continue if we found a matching child, and descended accordingly
            golower = matchfound;
        } // end while
        attachment.addChild(branchnode);
    } // end for
```