Scanner Data Structures for Wiki Example — First Token

Ch Loc	Matching Token Set	Best Match	Length
1: 1: p	{ pqrs, opqr, endsq, twosmallwords, IGNORE }	pqrs	1
1: 2: q	{ pqrs, opqr, endsq, twosmallwords }	pqrs	2
1: 3: r	{ pqrs, opqr, endsq, twosmallwords }	pqrs	3
1: 4: p	{ pqrs, opqr, endsq, twosmallwords }	pqrs	4
1: 5: q	{ pqrs, opqr, endsq }	pqrs	5
⋮	⋮	⋮	
1:17: r	{ pqrs, opqr, endsq }	pqrs	17
1:18: r	{ pqrs, opqr, endsq }	pqrs	18
1:19:\n	{ endsq }	pqrs	18
⋮	⋮	⋮	
2:17: p	{ endsq }	pqrs	18
2:18:SP	{ endsq }	pqrs	18
2:19: \	∅	pqrs	18
emit pqrs pqrpqrpppqqqpprrrr 1 1			

Scanner Data Structures for Wiki Example — First Token

Ch Loc	Matching Token Set	Best Match	Length
1: 1: p	{ pqrs, opqr, endsq, twosmallwords, IGNORE }	pqrs	1
1: 2: q	{ pqrs, opqr, endsq, twosmallwords }	pqrs	2
1: 3: r	{ pqrs, opqr, endsq, twosmallwords }	pqrs	3
1: 4: p	{ pqrs, opqr, endsq, twosmallwords }	pqrs	4
1: 5: q	{ pqrs, opqr, endsq }	pqrs	5
⋮	⋮	⋮	
1:17: r	{ pqrs, opqr, endsq }	pqrs	17
1:18: r	{ pqrs, opqr, endsq }	pqrs	18
1:19:\n	{ endsq }	pqrs	18
⋮	⋮	⋮	
2:17: p	{ endsq }	pqrs	18
2:18:SP	{ endsq }	pqrs	18
2:19: \	∅	pqrs	18
emit pqrs pqrpqrpppqqqpprrrr 1 1			

Notice that we could not be sure the match was a `pqrs` token of length 18 until the `endsq` matching failed. Now we have to start over again at line 1 character 19. . .

A Common Misconception

Here is a **common misconception** (misunderstanding?):

> *Token ids are assigned based on which DFAs did not fail during scanning.*

Why is this statement wrong?

A Common Misconception

Here is a **common misconception** (misunderstanding?):

> *Token ids are assigned based on which DFAs did not fail during scanning.*

Why is this statement wrong? Because **all the DFAs fail** before a decision is made![1]

Token ids are assigned based on **match length (and definition file ordering in the case of match length ties)**.

[1] Technically there is an edge case where this does not hold — but it is true most of the time.

Token Detection from Line 1 Character 19 (Second Token)

Ch Loc	Matching Token Set	Best Match	Length
1:19:\n	{ endsq, IGNORE }	IGNORE	1
2: 1: r	{ endsq }	IGNORE	1
⋮	⋮	⋮	
2:17: p	{ endsq }	IGNORE	1
2:18:SP	{ endsq }	IGNORE	1
2:19: \	∅	IGNORE	1
emit IGNORE x0a 1 19			

Token Detection from Line 1 Character 19 (Second Token)

Ch Loc	Matching Token Set	Best Match	Length
1:19:\n	{ endsq, IGNORE }	IGNORE	1
2: 1: r	{ endsq }	IGNORE	1
⋮	⋮	⋮	
2:17: p	{ endsq }	IGNORE	1
2:18:SP	{ endsq }	IGNORE	1
2:19: \	∅	IGNORE	1
emit IGNORE x0a 1 19			

And we rewind some 19 characters and begin matching again at line 2 character 1. . .

Token Detection from Line 2 Character 1 (Third Token)

Ch Loc	Matching Token Set	Best Match	Length
2: 1: r	{ pqrs, opqr, endsq, twosmallwords, IGNORE }	pqrs	1
2: 2: o	{ opqr, endsq, twosmallwords }	opqr	2
2: 3: p	{ opqr, endsq, twosmallwords }	opqr	3
2: 4:SP	{ endsq, twosmallwords }	opqr	3
2: 5: r	{ endsq, twosmallwords }	opqr	3
2: 6: o	{ endsq, twosmallwords }	opqr	3
2: 7: p	{ endsq, twosmallwords }	opqr	3
2: 8:SP	{ endsq, twosmallwords }	twosmallwords	8
2: 9: r	{ endsq }	twosmallwords	8
2:10: o	{ endsq }	twosmallwords	8
⋮	⋮	⋮	
2:18:SP	{ endsq }	twosmallwords	8
2:19: \	∅	twosmallwords	8
emit twosmallwords ropx20ropx20 2 1			

Token Detection from Line 2 Character 1 (Third Token)

Ch Loc	Matching Token Set	Best Match	Length
2: 1: r	{ pqrs, opqr, endsq, twosmallwords, IGNORE }	pqrs	1
2: 2: o	{ opqr, endsq, twosmallwords }	opqr	2
2: 3: p	{ opqr, endsq, twosmallwords }	opqr	3
2: 4:SP	{ endsq, twosmallwords }	opqr	3
2: 5: r	{ endsq, twosmallwords }	opqr	3
2: 6: o	{ endsq, twosmallwords }	opqr	3
2: 7: p	{ endsq, twosmallwords }	opqr	3
2: 8:SP	{ endsq, twosmallwords }	twosmallwords	8
2: 9: r	{ endsq }	twosmallwords	8
2:10: o	{ endsq }	twosmallwords	8
⋮	⋮	⋮	
2:18:SP	{ endsq }	twosmallwords	8
2:19: \	∅	twosmallwords	8
emit twosmallwords ropx20ropx20 2 1			

Where do we “rewind” to in the token stream? How is this calculated from the data in the scanning trace?

Token Detection from Line 2 Character 1 (Third Token)

Ch Loc	Matching Token Set	Best Match	Length
2: 1: r	{ pqrs, opqr, endsq, twosmallwords, IGNORE }	pqrs	1
2: 2: o	{ opqr, endsq, twosmallwords }	opqr	2
2: 3: p	{ opqr, endsq, twosmallwords }	opqr	3
2: 4:SP	{ endsq, twosmallwords }	opqr	3
2: 5: r	{ endsq, twosmallwords }	opqr	3
2: 6: o	{ endsq, twosmallwords }	opqr	3
2: 7: p	{ endsq, twosmallwords }	opqr	3
2: 8:SP	{ endsq, twosmallwords }	twosmallwords	8
2: 9: r	{ endsq }	twosmallwords	8
2:10: o	{ endsq }	twosmallwords	8
⋮	⋮	⋮	
2:18:SP	{ endsq }	twosmallwords	8
2:19: \	∅	twosmallwords	8
emit twosmallwords ropx20ropx20 2 1			

Where do we “rewind” to in the token stream? How is this calculated from the data in the scanning trace?

. . . we rewind to the 8th character past 2:1:r.

Fourth Token from Line 2 Character 9

Ch Loc	Matching Token Set	Best Match	Length
2: 9: r	{ pqrs, opqr, endsq, twosmallwords, IGNORE }	pqrs	1
2:10: o	{ opqr, endsq, twosmallwords }	opqr	2
2:11: p	{ opqr, endsq, twosmallwords }	opqr	3
2:12:SP	{ endsq, twosmallwords }	opqr	3
2:13:SP	{ endsq, twosmallwords }	opqr	3
2:14:SP	{ endsq, twosmallwords }	opqr	3
2:15: r	{ endsq, twosmallwords }	opqr	3
2:16: o	{ endsq, twosmallwords }	opqr	3
2:17: p	{ endsq, twosmallwords }	opqr	3
2:18:SP	{ endsq, twosmallwords }	twosmallwords	10
2:19: \	$\emptyset$	twosmallwords	10
emit twosmallwords ropx20x20x20ropx20 2 9			

Fifth Token from Line 2 Character 9

Ch Loc	Matching Token Set	Best Match	Length
2:19: \	{ whack, IGNORE }	IGNORE	1
2:20:SP	{ whack }	IGNORE	1
2:21: r	{ whack }	IGNORE	1
2:22: o	{ whack }	IGNORE	1
2:23: p	{ whack }	IGNORE	1
2:24:SP	{ whack }	IGNORE	1
2:25:SP	{ whack }	IGNORE	1
2:26: r	{ whack }	IGNORE	1
2:27: o	{ whack }	IGNORE	1
2:28: p	{ whack }	IGNORE	1
2:29:SP	{ whack }	IGNORE	1
2:30: \	{ whack }	whack	12
2:31:SP	$\emptyset$	whack	12
emit	whack x5cooox5cx20x5cooox5c 2 19		

Fifth Token from Line 2 Character 9

Ch Loc	Matching Token Set	Best Match	Length
2:19: \	{ whack, IGNORE }	IGNORE	1
2:20:SP	{ whack }	IGNORE	1
2:21: r	{ whack }	IGNORE	1
2:22: o	{ whack }	IGNORE	1
2:23: p	{ whack }	IGNORE	1
2:24:SP	{ whack }	IGNORE	1
2:25:SP	{ whack }	IGNORE	1
2:26: r	{ whack }	IGNORE	1
2:27: o	{ whack }	IGNORE	1
2:28: p	{ whack }	IGNORE	1
2:29:SP	{ whack }	IGNORE	1
2:30: \	{ whack }	whack	12
2:31:SP	∅	whack	12
emit	whack x5cooox5cx20x5cooox5c 2 19		

Recall that the `whack` token has explicit data associated with it in `scan.u`, this is why the source data does not appear in the emitted output.

Sixth Token from Line 2 Character 31

Ch Loc	Matching Token Set	Best Match	Length
2:31:SP	{ endsq, IGNORE }	IGNORE	1
2:32:\n	{ endsq }	IGNORE	1
3: 1: p	{ endsq }	IGNORE	1
3: 2:\n	{ endsq }	IGNORE	1
4: 1: q	{ endsq }	endsq	5
4: 2:SP	{ endsq }	endsq	5
4: 3: r	{ endsq }	endsq	5
⋮	⋮	⋮	
6: 3: q	{ endsq }	endsq	17
6: 4:SP	{ endsq }	endsq	17
6: 5: \	∅	endsq	17
emit endsq x20x0apx0aqx20rx20sx20rx0aqx0apx20q 2 31			

Sixth Token from Line 2 Character 31

Ch Loc	Matching Token Set	Best Match	Length
2:31:SP	{ endsq, IGNORE }	IGNORE	1
2:32:\n	{ endsq }	IGNORE	1
3: 1: p	{ endsq }	IGNORE	1
3: 2:\n	{ endsq }	IGNORE	1
4: 1: q	{ endsq }	endsq	5
4: 2:SP	{ endsq }	endsq	5
4: 3: r	{ endsq }	endsq	5
⋮	⋮	⋮	
6: 3: q	{ endsq }	endsq	17
6: 4:SP	{ endsq }	endsq	17
6: 5: \	∅	endsq	17
emit endsq x20x0apx0aqx20rx20sx20rx0aqx0apx20q 2 31			

More complicated? where do we “rewind” to in the token stream?

Sixth Token from Line 2 Character 31

Ch Loc	Matching Token Set	Best Match	Length
2:31:SP	{ endsq, IGNORE }	IGNORE	1
2:32:\n	{ endsq }	IGNORE	1
3: 1: p	{ endsq }	IGNORE	1
3: 2:\n	{ endsq }	IGNORE	1
4: 1: q	{ endsq }	endsq	5
4: 2:SP	{ endsq }	endsq	5
4: 3: r	{ endsq }	endsq	5
⋮	⋮	⋮	
6: 3: q	{ endsq }	endsq	17
6: 4:SP	{ endsq }	endsq	17
6: 5: \	∅	endsq	17
emit	endsq x20x0apx0aqx20rx20sx20rx0aqx0apx20q 2 31		

More complicated? where do we “rewind” to in the token stream?

. . . we rewind to 17 characters past the 2:31:SP where the match began.

Seventh and Eighth Tokens

Ch Loc	Matching Token Set	Best Match	Length
6:4: SP	{ endsq, IGNORE }	IGNORE	1
6:5: \	∅	IGNORE	1
emit IGNORE x20 6 4			

Ch Loc	Matching Token Set	Best Match	Length
6: 5: \	{ whack, IGNORE }	IGNORE	1
6: 6: p	{ whack }	IGNORE	1
6: 7: o	{ whack }	IGNORE	1
6: 8: s	{ whack }	IGNORE	1
6: 9:SP	{ whack }	IGNORE	1
6:10: r	{ whack }	IGNORE	1
6:11: o	{ whack }	IGNORE	1
6:12: s	{ whack }	IGNORE	1
6:13: p	{ whack }	IGNORE	1
6:14: \	{ whack }	whack	10
6:15:SP	∅	whack	10
emit whack x5cooox5cx20x5cooox5c 6 5			

Nineth, Tenth, and Final Token

Ch Loc	Matching Token Set	Best Match	Length
6:15:SP	{ endsq, IGNORE }	IGNORE	1
6:16: r	{ endsq }	IGNORE	1
6:17:\n	{ endsq }	IGNORE	1
emit IGNORE x20 6 15			

Ch Loc	Matching Token Set	Best Match	Length
6:16: r	{ pqrs, opqr, endsq, twosmallwords, IGNORE }	pqrs	1
6:17:\n	{ endsq }	pqrs	1
emit pqrs r 6 16			

Ch Loc	Matching Token Set	Best Match	Length
6:17:\n	{ endsq, IGNORE }	IGNORE	1
emit IGNORE x0a 6 17			

These are missing the last "∅" row of previous traces because there is no final character that eliminates the "Matching Token Set" — it is the **end-of-file** that forces the "Best Match" to be emitted.

(I could have faked the slides to appear more similar, but it seems useful to highlight the destinction in the two logic branches.)