Lexical Analyzer

Node n	nullable(n)	firstpos(n)	lastpos(n)
n is a leaf node labeled ε	true	∅	∅
n is a leaf node labelled with position i	false	{ i }	{ i }
n is an or-node with left child c1 and right child c2	nullable(c1) or nullable(c2)	firstpos(c1) ∪ firstpos(c2)	lastpos(c1) ∪ lastpos(c2)
n is a cat-node with left child c1 and right child c2	nullable(c1) and nullable(c2)	If nullable(c1) then firstpos(c1) ∪ firstpos(c2) else firstpos(c1)	If nullable(c2) then lastpos(c2) ∪ lastpos(c1) else lastpos(c2)
n is a star-node with child node c1	true	firstpos(c1)	lastpos(c1)

Example

- Regular Expression:

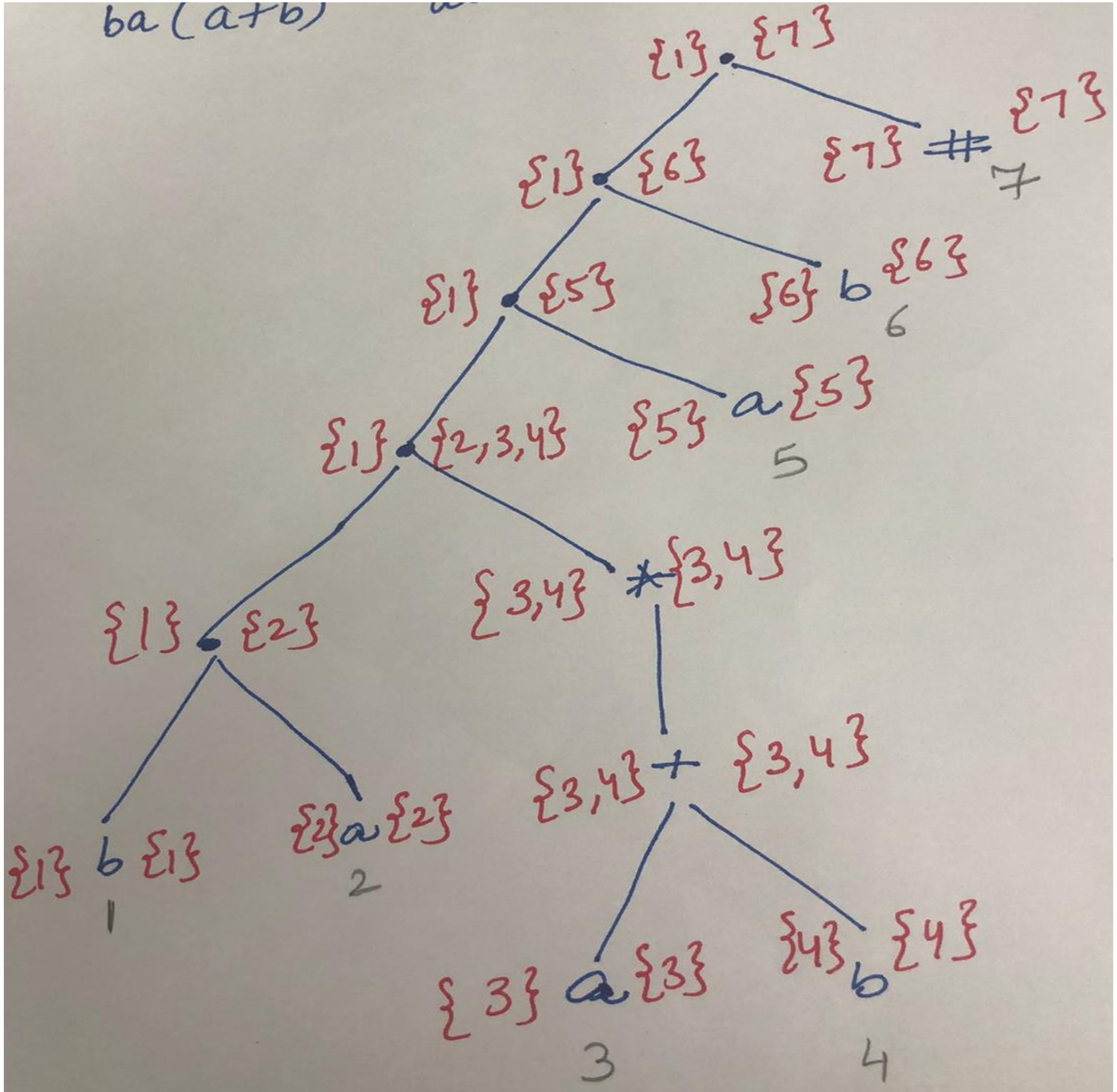
  ba(a+b)*ab

Step1: add # at the end.

ba(a+b)*ab#

ba(a+b)*ab#
12 3 4   56

Step 2: syntax tree
Step 3: compute nullable, firstpos and lastpos



Step 4: compute follow pos

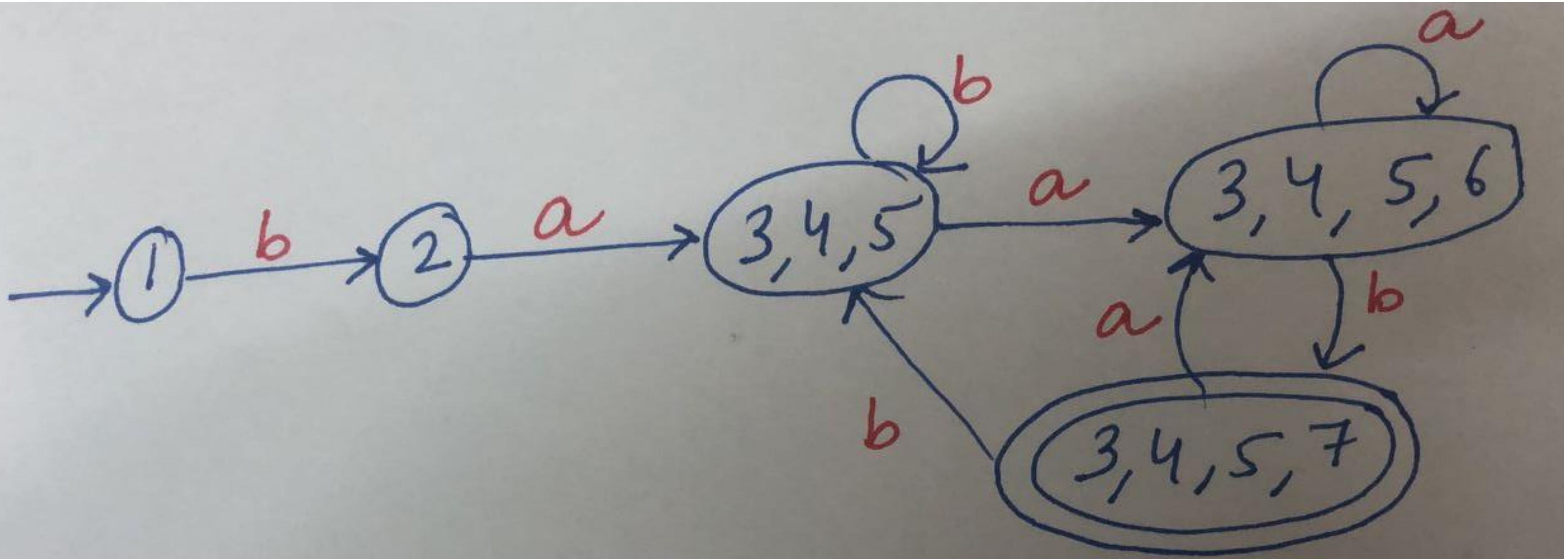
NODE	followpos
1	{2}
2	{3, 4, 5}
3	{3, 4, 5}
4	{3, 4, 5}
5	{6}
6	{7}
7	∅

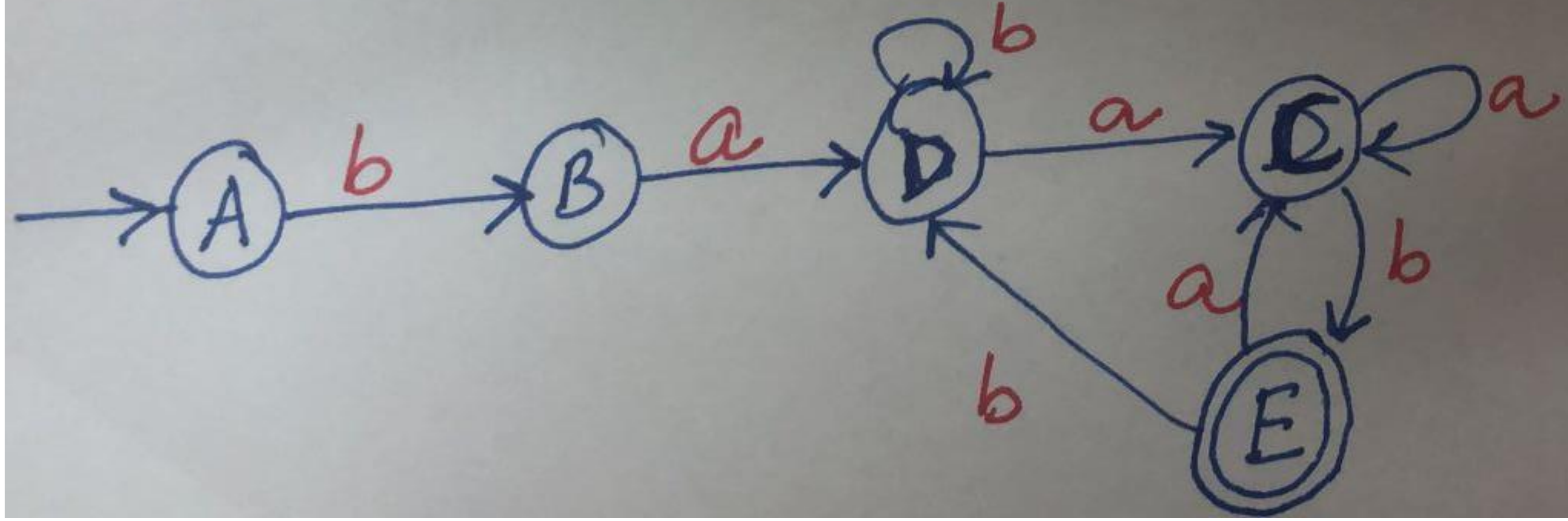
Construct states

- Root {1} initial state named as A.
- A-> {1} = FP(1) = {2} named as B.
- B-> {2} = FP(2) = {3,4,5}, 3 and 5 point to same symbol ‘a’ so FP(3)UFP(5) = {3, 4, 5, 6} named as C. FP(4) = {3,4,5} named as D.
- C-> {3, 4, 5, 6}, 3 and 5 point to same symbol ‘a’ so FP(3)UFP(5) = {3, 4, 5, 6} already there as C. 4 and 6 point to same symbol ‘b’ so FP(4)UFP(6) = {3, 4, 5, 7} named as E.
- D-> {3,4,5}, 3 and 5 point to same symbol ‘a’ so FP(3)UFP(5) = {3, 4, 5, 6} already there as C. FP(4) = {3,4,5} already as D.
- E-> {3,4,5,7}, 3 and 5 point to same symbol ‘a’ so FP(3)UFP(5) = {3, 4, 5, 6} already there as C. FP(4) = {3,4,5} already as D. FP(7) = ϕ. So **E is the final state**.

Transition Table

	Input	
State	**a**	**b**
⇢ A {1}	-	B
B {2}	D	-
D {3,4,5}	C	D
C {3,4,5,6)	C	E
E {3,4,5,7}	C	D


1
b
2
a
3,4,5
b
a
3,4,5,6
a
b
a
b
3,4,5,7


A
b
B
a
D
b
a
C
a
b
a
b
E

Example

- Regular Expression:

  (a|b)*.(c|d*).a.d

Step 1: add # at the end.

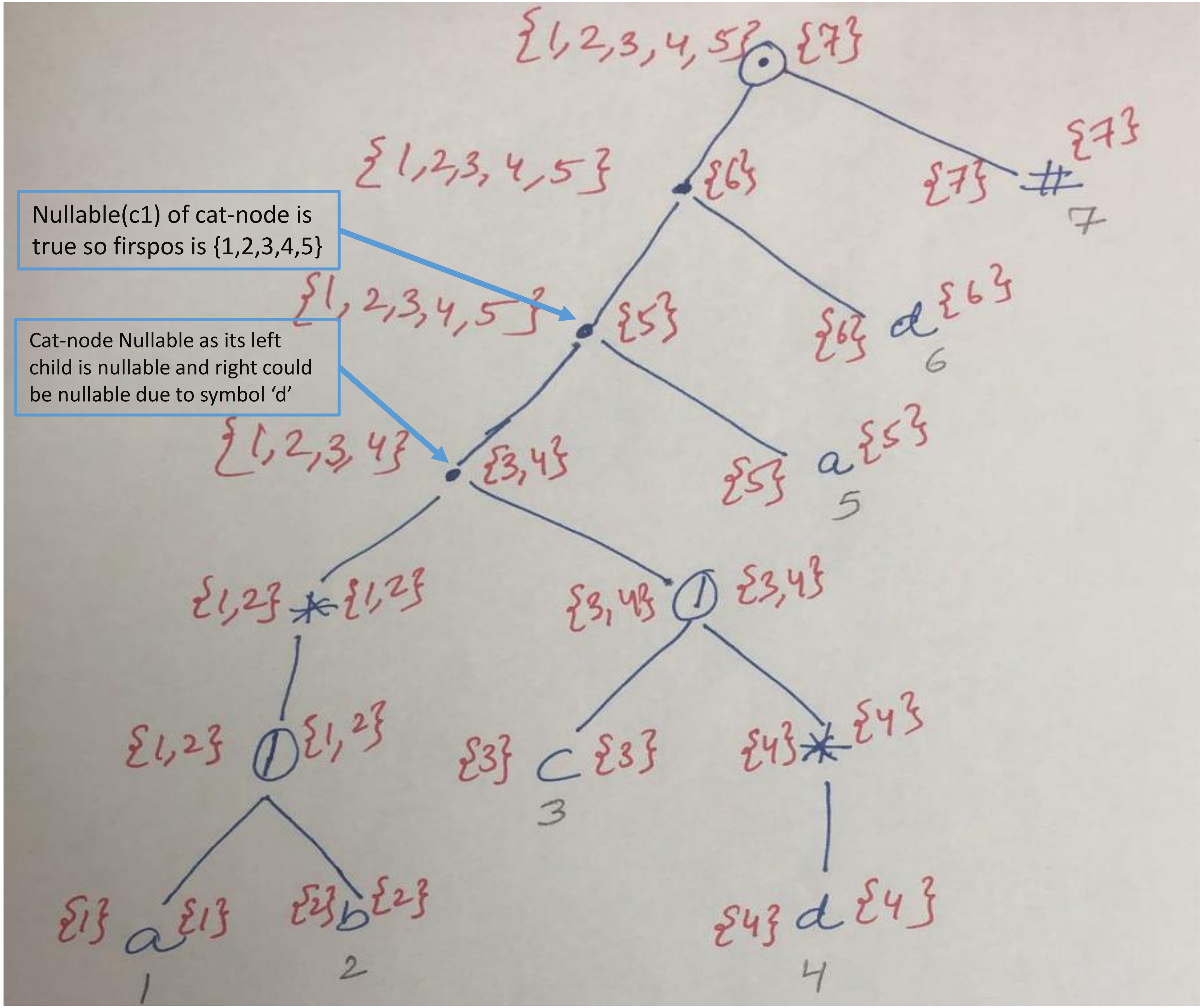
(a|b)*.(c|d*).a.d#

(a|b)*.(c|d*).a.d#

1 2 3 4 5 67

Step 2: syntax tree

Step 3: compute nullable, firstpos and lastpos



Nullable(c1) of cat-node is true so firspos is {1,2,3,4,5}

Cat-node Nullable as its left child is nullable and right could be nullable due to symbol ‘d’

Follow pos

NODE	followpos
1	{1, 2, 3, 4, 5}
2	{1, 2, 3, 4, 5}
3	{5}
4	{4, 5}
5	{6}
6	{7}
7	∅

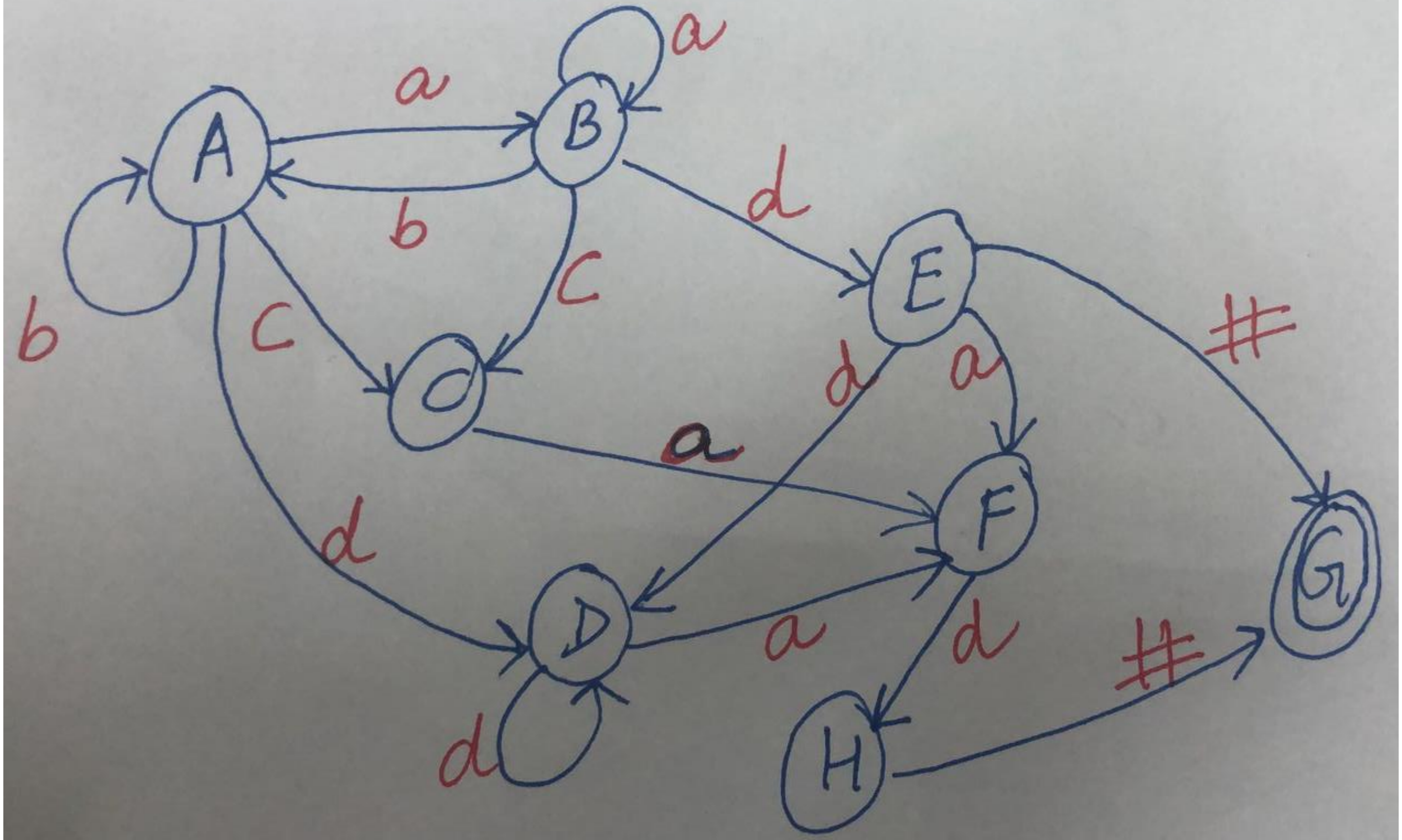
Follow pos of 3 i.e. symbol ‘c’ is only 5 not 4 because its or operator (c or d). If we choose c then d can’t be next symbol.

Follow pos of 4 i.e. symbol ‘d’ is both 4 and 5 because ‘d’ could be more than once in the string and next position could be 5th symbol i.e. ‘a’

States

States	
A {1,2,3,4,5}	FP(1)UFP(5)= {1,2,3,4,5,6} named as B FP(2) = {1,2,3,4,5} already there A FP(3) = {5} named as C FP(4) = {4,5} named as D
B {1,2,3,4,5,6}	FP(1)UFP(5) ={1,2,3,4,5} already B FP(2) = {1,2,3,4,5} already A FP(3) = {5} already C FP(4)UFP(6)= {4,5,7} named as E
C {5}	FP(5) = {6} named as F

States	
D {4,5}	FP(4) = {4,5} already D FP(5) = {6} already F
E {4,5,7}	FP(4) = {4,5} already D FP(5) = {6} already F FP(7) = ϕ named as G
F {6}	FP(6)= {7} named as H
H {7}	FP(7) = ϕ Final State



a
a
A
B
b
d
b
c
c
E
#
C
d
a
a
F
d
G
D
a
d
#
d
H

Error Handling Routine

In the compiler design process error may occur in all the below-given phases:

Lexical analyzer:

- Wrongly spelled tokens,

  e.g. int 4num;

- Exceeding length of identifier,

  e.g. int abcdefgh23456h5ghghghg4556h6ghghghghghghghghghgg

- Illegal character,

  e.g. printf(“hello”);#

- Strings that don’t match

  e.g. starting of comment but no ending..

Error Recovery

- Removes one character from the remaining input
- In panic mode recovery method, successive characters from the input are removed one at a time until a designated set of synchronizing tokens is found. Synchronizing tokens are delimiters such as; or }
- The advantage is that it is easy to implement and do not go into an infinite loop but using this a considerable amount of input is skipped without checking it for additional errors.
- By inserting the missing character into the remaining input
- Replace a character with another character
- Transpose two serial characters