I Lexic — char a, ... z, A, ... Z
digits 0, 1, ..., 9
spec char +, -, <, >, (,), {, }, #, &

II Syntax

II Semantics

5 categories of lexical items:

while

(,), 4, 7, ;

a, b

+, <, =

3, 2

Keywords

1) Reserved words: while, for, if, else, ...

2.) Separators (delimitors): (,), {, }, [,], ,, ;, (new line)

„white separators": (new line)
(white space)
(tab)

3.) Operators — logical op: and, or, not
— arithmetic op: +, -, /, *
— relational op: <, >, <=
— assignment op:

4.) Identifiers → RULE

5.) Constants - integer
- real / float
- char
- string
- pointer
- bool : true, false

Syntax

- syntax rules

BNF (Backus - Naur - Form)

EBNF (extended)

- primitives (lex tokens) 'for'
- syntactical constructions (if stmt, condition
- connectives | , [ ] 0/1
↳ appear 0 or one times
{ } 0/ any

```
int a;
float b;
int a = 20;
int a,b;
```

List declaration == decl | decl ";" List decl

decl == Type | d ":"

List decl == decl ":" List decl | C]

decl == Type ⊔ d)

Type == "int" | "float"

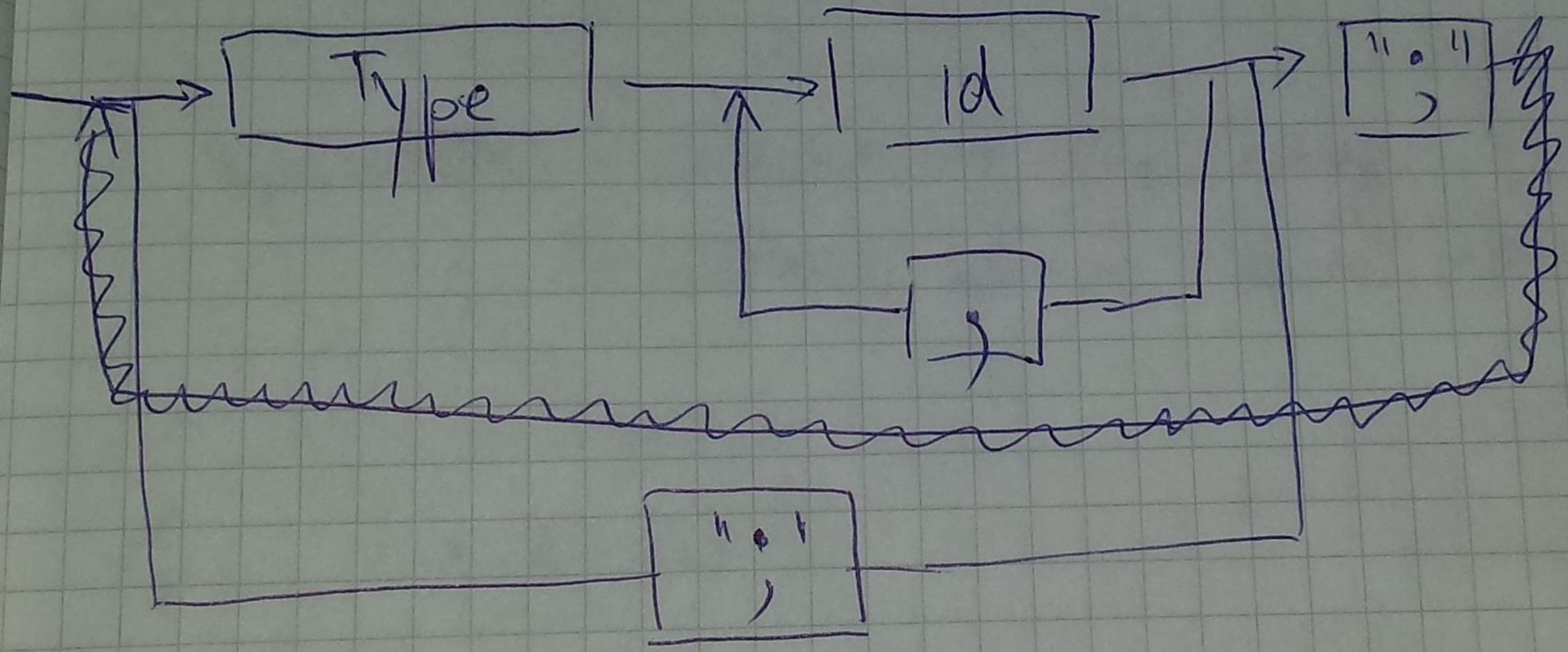
L id == Δ id "," L id | [ ]

Δ id == Id | Id "=" Const

Δ id == Id ["=" Const]

Id == letter { letter | digit }

letter == "a" | ... "z"



S3 (sem 8-10 , 12-14)

TC

Seu 2.

```
while (a <= 10)
{  b = a + 1;
   a = a + 2;
}
```

codif: 1 - identificatori
2 constante
3 if
4 Num
5 while
...
10 +
11 -
12 *
...
15 =
16 <
17 <=
...
25 (
26)
27 {
28 }
...
35 ;

PIF

5	-1
25	-1
1	1
17	-1
2	2
26	-1
27	-1
1	3
15	-1
1	1
11	-1
2	4
35	-1
1	1
15	-1
1	1
10	-1
2	5
35	-1
28	-1

ST

1	a
2	10
3	b
4	1
5	2

```
while (a <= 10)
{  b = a - 1;
   a = a + 2;
}
```

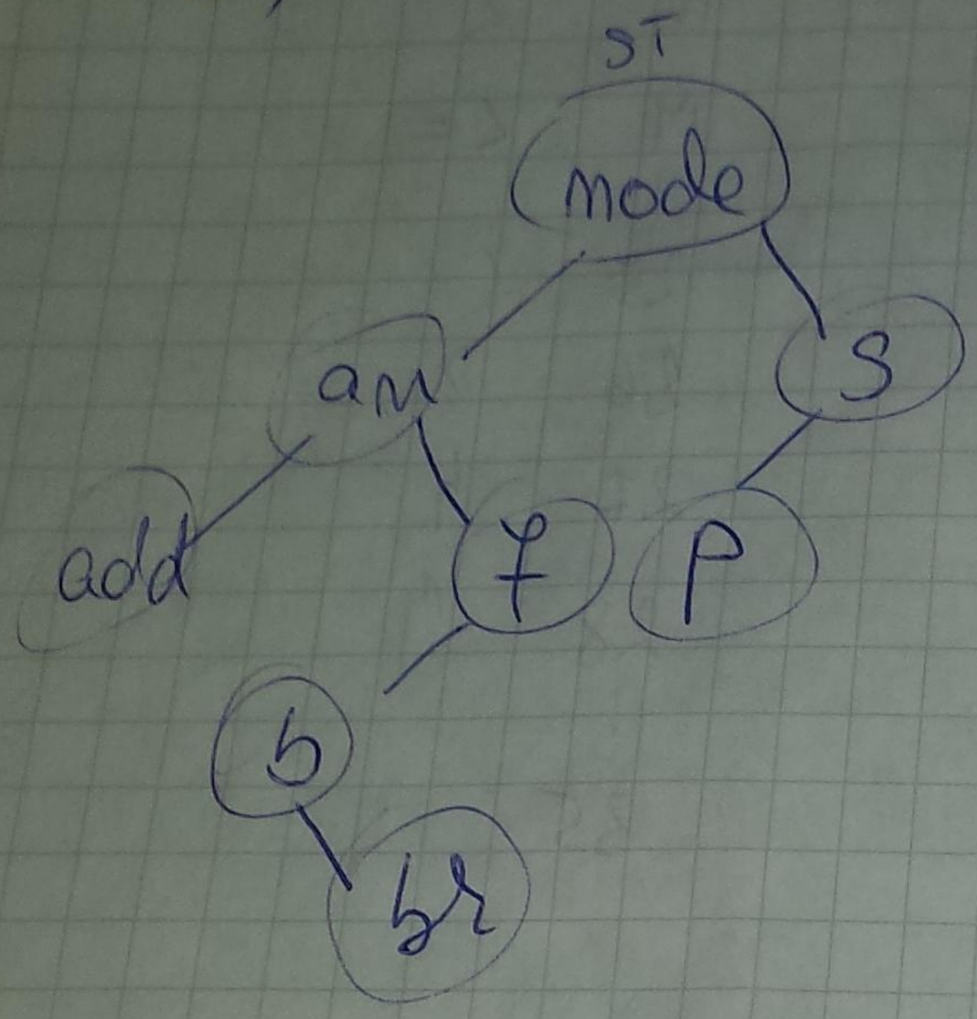
1.) lexicographically sorted table
2.) binary search tree
3.) hash table

node, am, f, s, b, add, br, p

1.) node am
node f
node

am — f — node

2) BST



Balanced BST

Index	Node	Left	Right
1	node	am	s
2	am	add	f
3	add	—	—
4	f	b	—
5	b	—	br

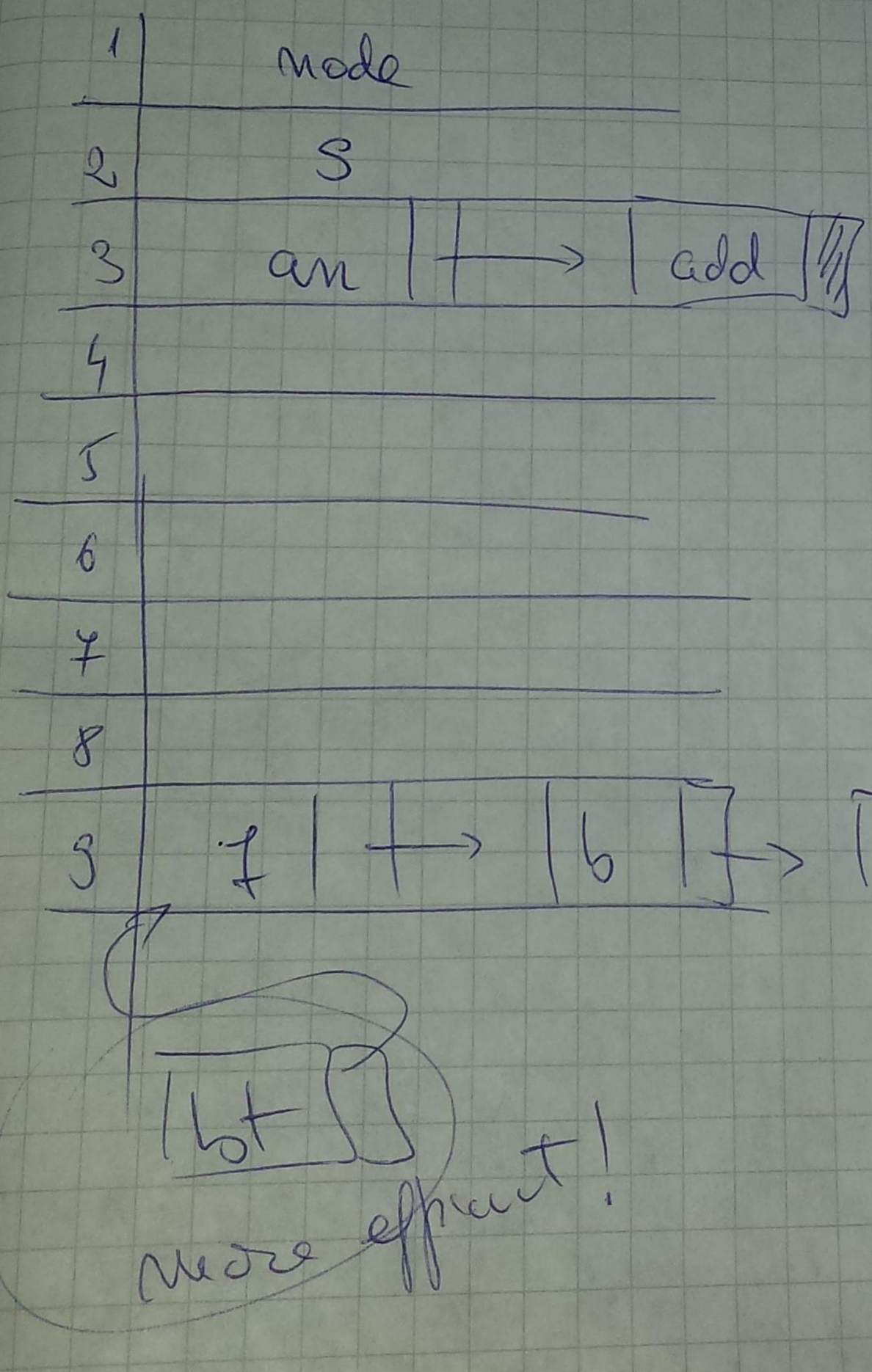
6 br — —
f ∧ p —
g p — —

node	an	y	s	b	add	br	p
1	3	9	2	9	3	9	9

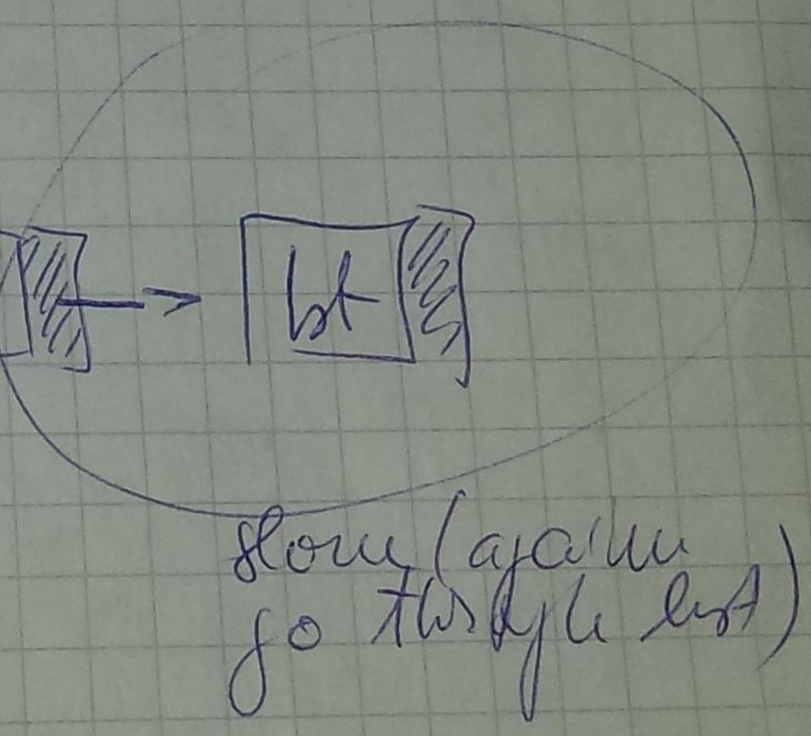
HT |HT| = 9 ; 0 dr

1.) Open Bucket / Open Addrress 4 strategies
2.) Closed Bucket

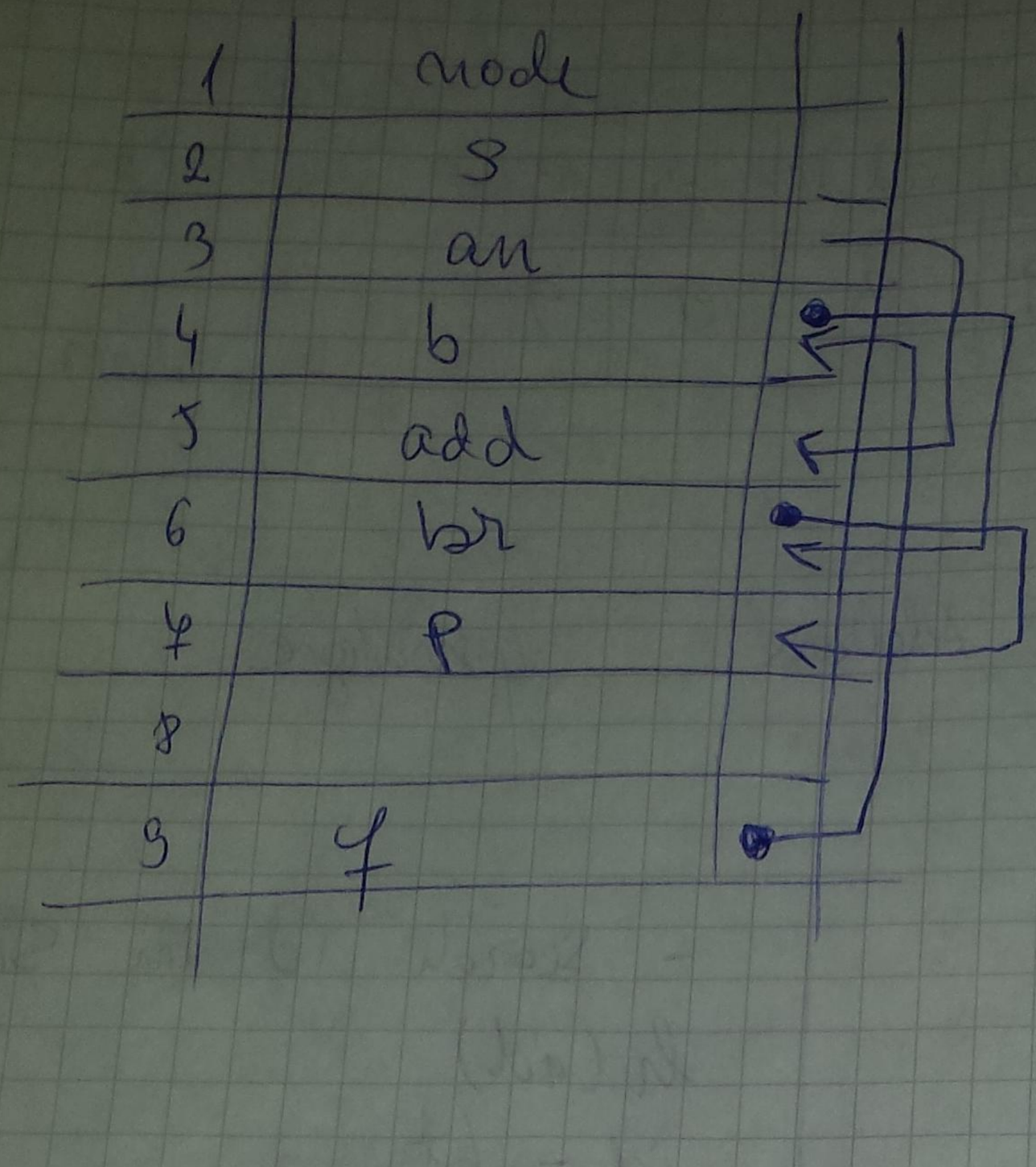
1.) Open Bucket



- Search id in ST a
h(aid)
id = bt
h(bt) = 9 // not found
// should insert



2.) Closed Bucket



b- search for
next free position

- search : same
- insert : load from
the other