NOTE on lab 5!

Two ways to count comparisons.

---

①
```
int count=0;
:
while( j>=1 && list[j] < list[j-1] ) {
        :
        count++;  <-- counts true instances
}
if( j>=1 ) {
     count++; <-- counts possible false
                  instance
}
:
```

---

②
```
int count=0;
:
while( j>=1 && ++count && list[j] < list[j-1]){
        :
}
```

counts all comparisons

CMPS 10 12-3-09

Turing Machines :

languages:

- Pseudo-code
- C++
- Circuits

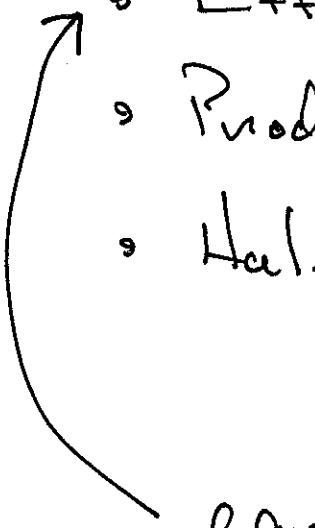
New language

- Turing Machines

Recall Defn of Algorithm:

A. Collection of operations which are

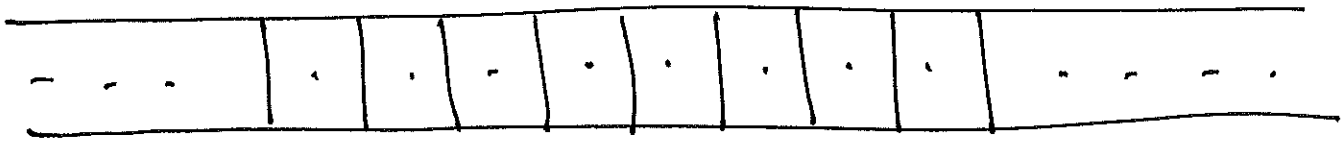
- well ordered
- unambiguous
- Effectively computable ?
- Produce a result
- Halt in finite time



can perform each operation.

Turing Machine consists of:

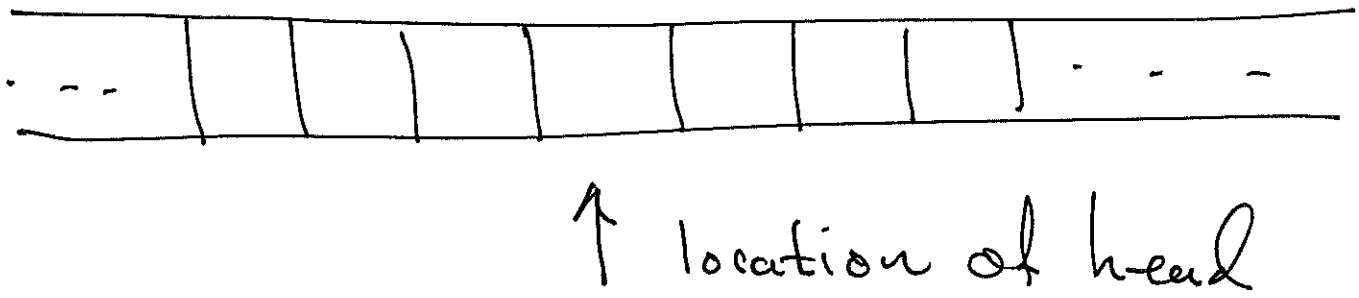
- Tape: Infinite in two Directions



Divided into cells.
each cell contains a symbol from a finite set called the Alphabet. Special symbol blank written 'b'.

- only finitely many non-blank symbols on tape.

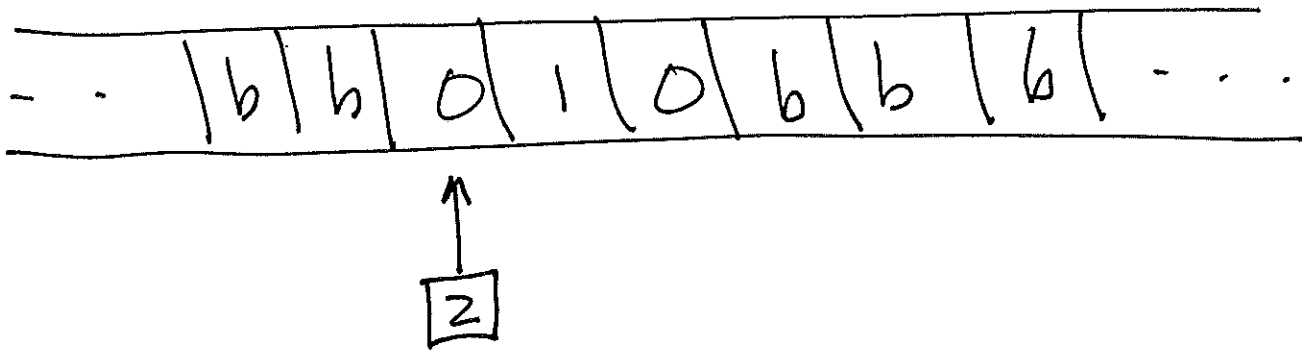
- Device for Reading from and writing to Tape : head.



↑ location of head

- Finite set of Internal states $\{1, 2, 3, \ldots, n\}$

Ex.



in state 2, reading symbol 0.

- Performs one primitive op. 3 actions in succession.
  1.) write a symbol to a cell (overwriting symbol there)
  2.) go to a new state (could be same state)
  3.) move head Right (R) or Left (L).
- Each instruction has form

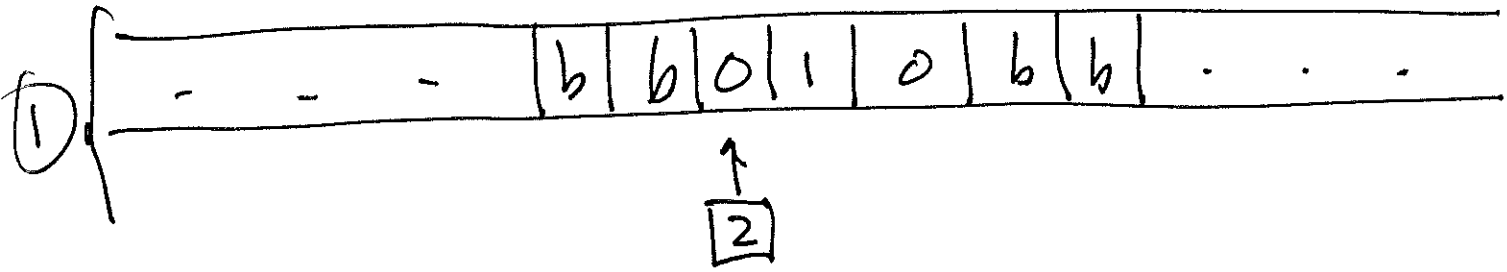
  if (in state i) and (reading symb j)
  write symbol k
  enter state s
  move in Direction D

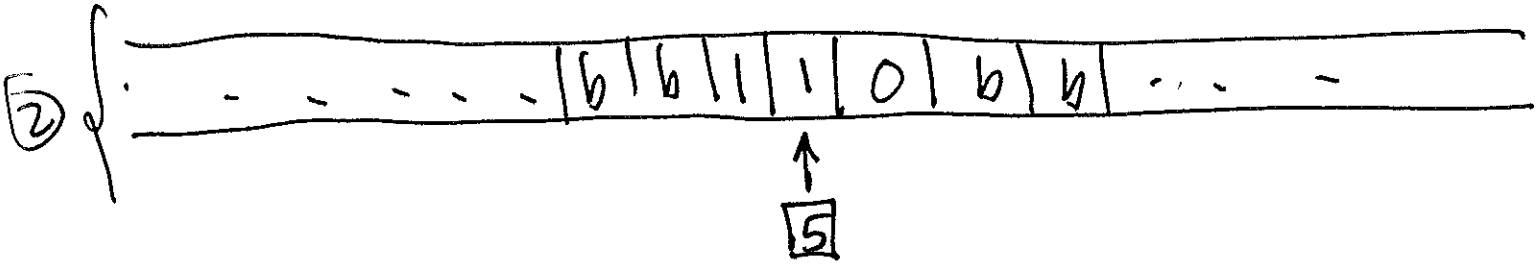
Such an instruction is specified by a 5-tuple

(i, j, K, s, D)

Ex. $(2, 0, 1, 5, R)$

Says: if (in state 2) and (reading '0')
write '1'
enter state 5
move R





Configuration: head location, symbol, state.

A Turing Machine is A finite collection of 5-tuples

Ex. $\mathcal{A} = \{b, 0, 1\}$, states $= \{1, 2, 3\}$

$(1, 0, 1, 2, R)$

$(1, 1, 0, 2, R)$

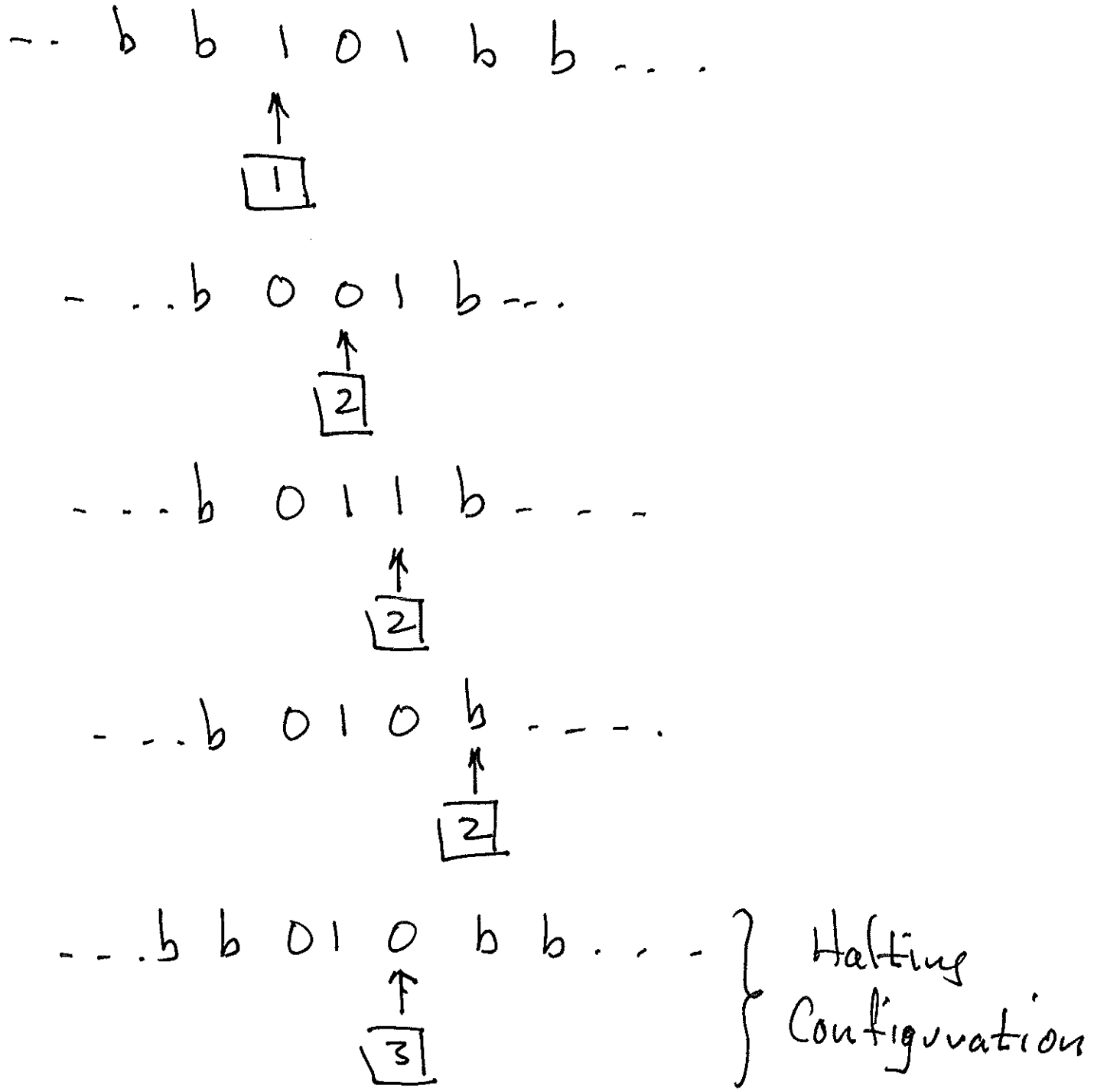
$(2, 0, 1, 2, R)$

$(2, 1, 0, 2, R)$

$(2, b, b, 3, L)$

{ initial state: always 1
initial head position: leftmost non-blank

Initial Tape



Exercise : Same TM

b b 1 0 1 1 0 b b

Exercise : same TM

.. b b 0 0 1 0 1 b b .. ..

Correspondance:

Algorithm $\longleftrightarrow$ Turing Machine

Instance of Problem that Algorithm solves $\longleftrightarrow$ Input Tape

Ex. A SIMPLER MACHINE THAT ALSO Reverses bit: states = {1}, alpha = {b, 0, 1}

(1, 0, 1, 1, R)

(1, 1, 0, 1, R)

Exercise: Try this on preceding Tapes.

Ex. ODD Parity TM :

appends 0 or 1 to end of a bit string so as to make # of 1's to be odd.

alpha = { b, 0, 1 }

states = { 1, 2, 3 }

(1, 1, 1, 2, R)

(1, 0, 0, 1, R)

(2, 1, 1, 1, R)

(2, 0, 0, 2, R)

(1, b, 1, 3, R)

(2, b, 0, 3, R)

EXERCISE:

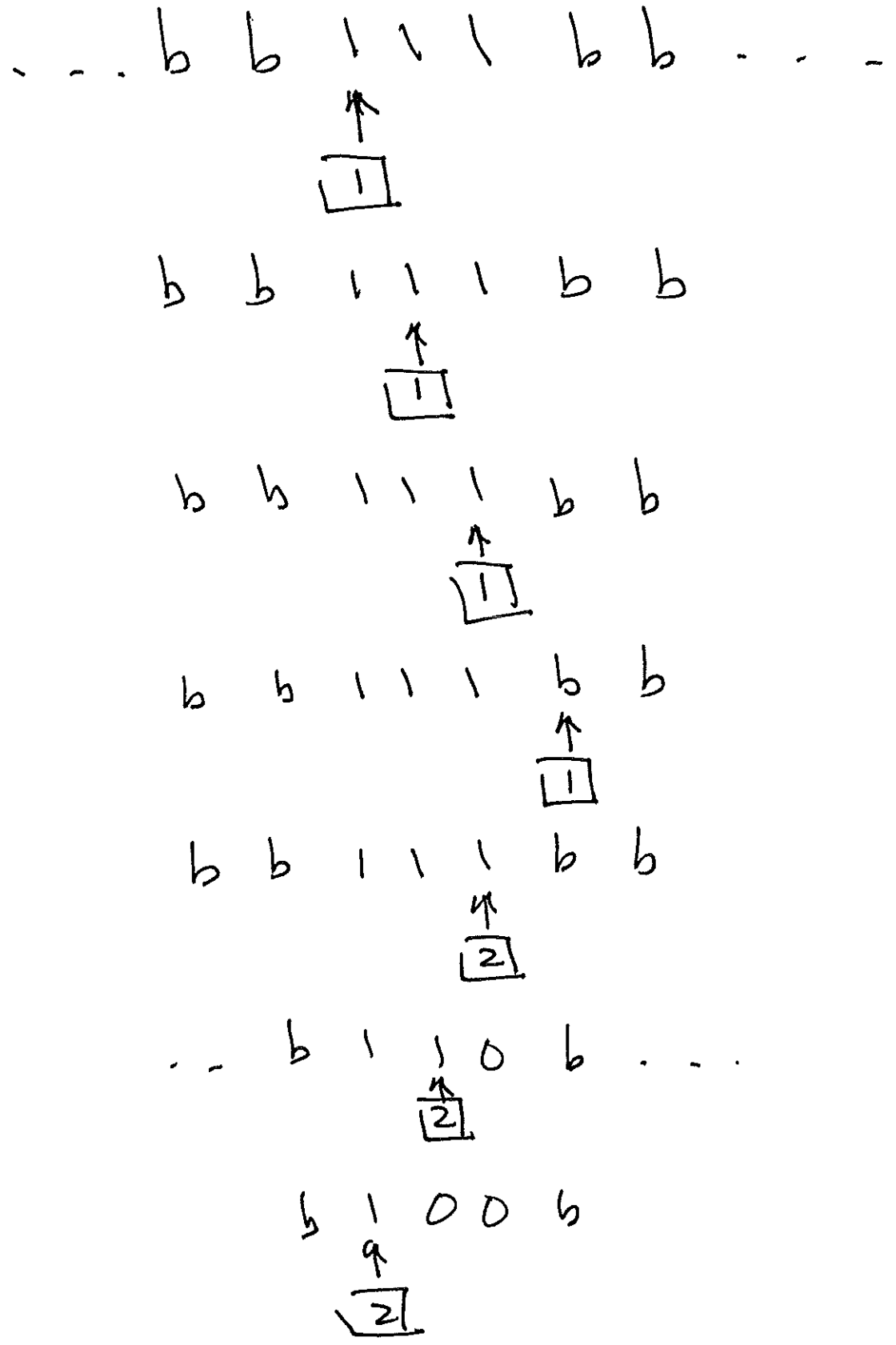
Run this TM on some examples.

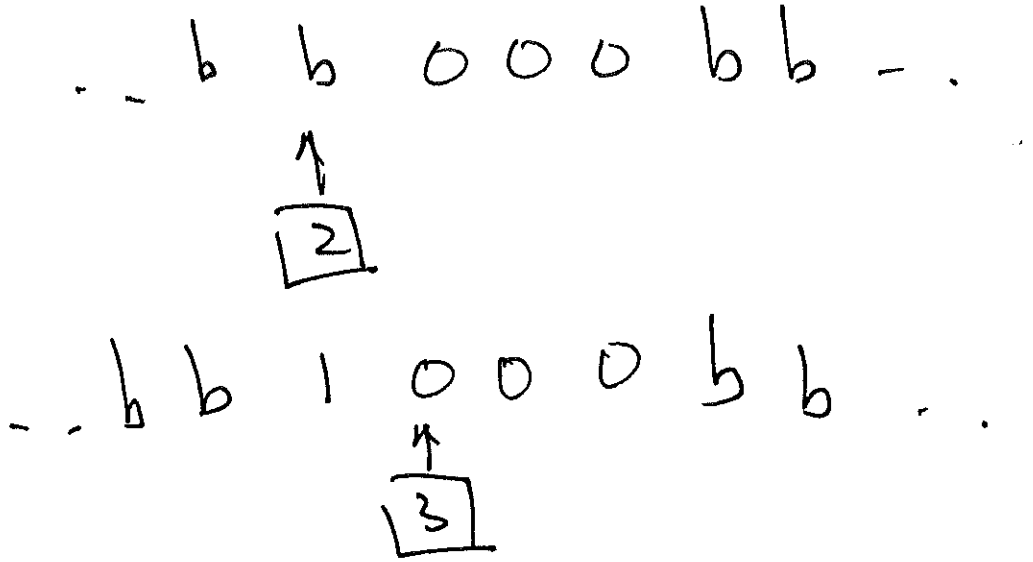
Ex. Binary Increment

alpha = {b, 0, 1}, states = {1, 2, 3}

1	1	1	1	R
1	0	0	1	R
1	b	b	2	L
2	0	1	3	L
2	b	1	3	R
2	1	0	2	L

Input Tape:


. . . b b 1 1 1 b b . . .
1
b b 1 1 1 b b
1
b b 1 1 1 b b
1
b b 1 1 1 b b
1
b b 1 1 1 b b
2
. . b 1 1 0 b . . .
2
b 1 0 0 b
q
2


. . b b o o o b b . .
2
. . b b 1 o o o b b . .
3