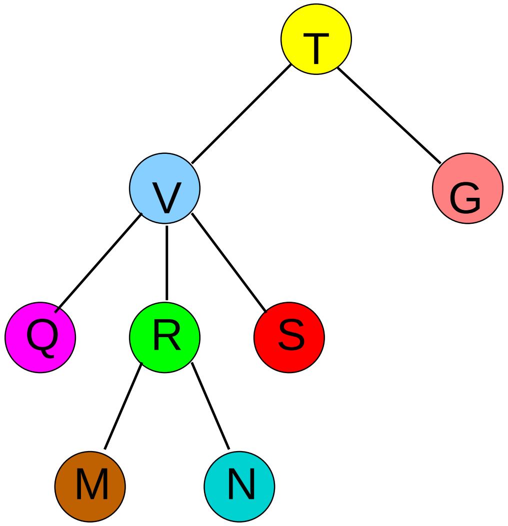
- linearizing datastructures

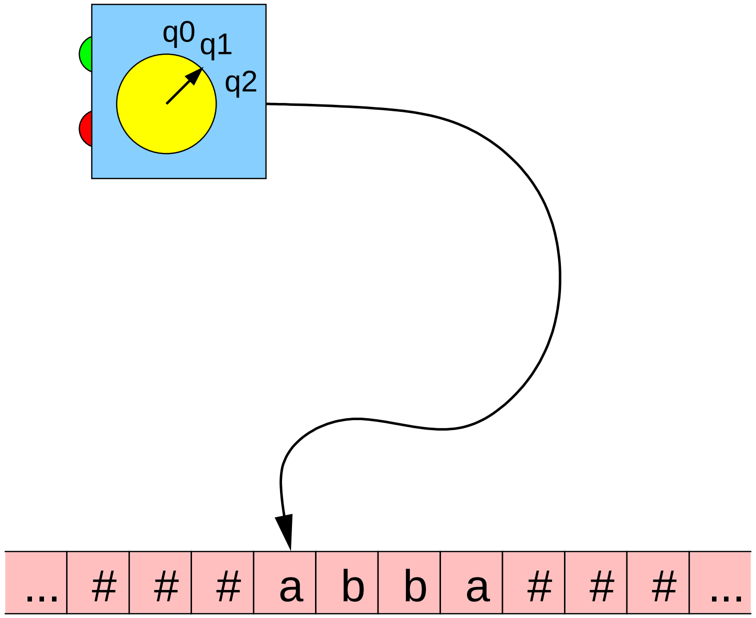


$$\begin{pmatrix} 15 & -4 & 6 & 104 \\ 6.33 & 0 & -61 & 3 \end{pmatrix}$$

15*-4*6*104**6.33*0*-61*3

(T)(V G)(Q R S)()()(M N)

- Turing machine architecture



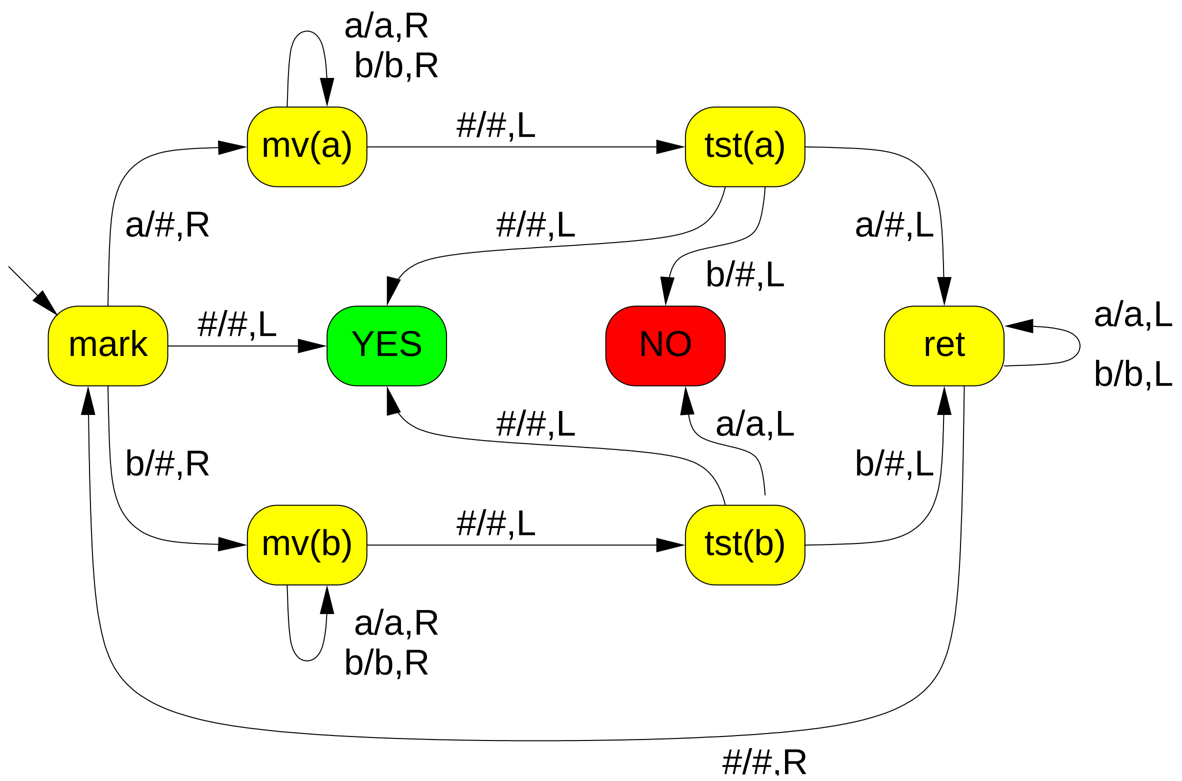
```
-- Turing-machine computing binary increment

-- q0 find rightmost digit
   d(q0,0) = (q0,0,R)
   d(q0,1) = (q0,1,R)
   d(q0,#) = (q1,#,L)

-- q1 increment rightmost digit
   d(q1,0) = (HALT,1,R)
   d(q1,1) = (q2,0,L)

-- q2 process carry
   d(q2,0) = (HALT,1,L)
   d(q2,1) = (q2,0,L)
   d(q2,#) = (HALT,1,L)
```

- a state-transition diagram



```
-- q0 mark symbol
   d(q0,a) = (q1,#,R)
   d(q0,b) = (q3,#,R)
   d(q0,#) = (YES,#,L)

-- q1 move right, remember 'a'
   d(q1,a) = (q1,a,R)
   d(q1,b) = (q1,b,R)
   d(q1,#) = (q2,#,L)

-- q2 test if there is an 'a'
   d(q2,a) = (q5,#,L)
   d(q2,b) = (NO,#,L)
   d(q2,#) = (YES,#,L)

-- q3 move right, remember 'b'
   d(q3,a) = (q3,a,R)
   d(q3,b) = (q3,b,R)
   d(q3,#) = (q4,#,L)

-- q4 check if there is an 'b'
   d(q4,a) = (NO,#,L)
   d(q4,b) = (q5,#,L)
   d(q4,#) = (YES,#,L)

-- q5 return to leftmost nonblank
   d(q5,a) = (q5,a,L)
   d(q5,b) = (q5,b,L)
   d(q5,#) = (q0,#,R)
```

- addition of decimal numbers

`digit-is-0`, . . . , `digit-is-9`,
`sum-is-0-nocarry`, . . . , `sum-is-9-nocarry`,
`sum-is-0-carry`, . . . , `sum-is-9-carry`

- computation of 736+63519

```
#######736+63519#
   ...
####15=73#+6351##
   ...
=64255=###+######
```

computed result between '=' markers

- Turing machine 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, \texttt{\#}, F)$
  - Q states, q_0 start state, $F \subseteq Q$ final states
  - Σ input symbols, $\Gamma \supseteq \Sigma$ tape symbols, $\texttt{\#} \in \Gamma \backslash \Sigma$ blank
  - transition function $\delta : Q \backslash F \times \Gamma \to Q \times \Gamma \times \{\texttt{L}, \texttt{R}\}$

typically $F = \{\mathsf{HALT}\}$ or $F = \{\mathsf{YES}, \mathsf{NO}\}$

- instanteneous description $xqay$
  - xay part of tape from leftmost to rightmost non-blank
  - tapehead at symbol a
  - machine in state q

one-step computation

$xqby \vdash^* xcq'y$ if $\delta(q,b) = (q',c,R)$

$xaqby \vdash^* xq'acy$ and $qby \vdash^* q'\#cy$ if $\delta(q,b) = (q',c,L)$

language accepted by Turing machine M

$\mathcal{L}(M) = \{\ w \in \Sigma^* \mid \exists q \in F, x, y \in \Gamma^* \colon q_0 w \vdash^* xqy\ \}$

- separate input tape

- folding over an infinite tape

$\cdots$	b	$*$	a	5	b	7	$!$	a	c	a	$\#$	$\cdots$

represented by

$\triangleright$	7	b	$!$	5	a	a	c	$*$	a	b	$\#$	$\cdots$

requirement $\delta(q, \triangleright) = (q', \triangleright, R)$

- non-deterministic Turing machine

- basic functions

  zero functions $z(x_1, \ldots, x_n) = 0$

  successor function $s(x) = x + 1$

  projections $\pi_k^n(x_1, \ldots, x_n) = x_k$

- function schemes

  composition

  $f(x_1, \ldots, x_n) = h(g_1(x_1, \ldots, x_n), \ldots, g_m(x_1, \ldots, x_n))$

  primitive recursion

$$
\begin{aligned}
f(x_1, \ldots, x_n, 0) &= g(x_1, \ldots, x_n) \\
f(x_1, \ldots, x_n, y + 1) &= h(x_1, \ldots, x_n, y, f(x_1, \ldots, x_n, y))
\end{aligned}
$$

- addition

  $\mathit{add}(x, 0) = x$, $\mathit{add}(x, y+1) = s(\mathit{add}(x, y))$

- multiplication

  $\mathit{mult}(x, 0) = 0$, $\mathit{mult}(x, y+1) = \mathit{add}(x, \mathit{mult}(x, y))$

- predecessor

  $\mathit{pred}(0) = 0$, $\mathit{pred}(y+1) = y$

- monus

  $x \dot{-} 0 = x$, $x \dot{-} (y+1) = \mathit{pred}(x \dot{-} y)$

- not all total functions on $\mathbb{N}$ are primitive recursive

Ackermann's function

$$\begin{aligned} A(0,y) &= y \\ A(x+1,0) &= A(x,1) \\ A(x+1,y+1) &= A(x,A(x+1,y)) \end{aligned}$$

- minimalization scheme $\mu y[g(x_1,\ldots,x_n,y)]$

smallest y such that $g(x_1,\ldots,x_n,y)=0$

$\mathit{sqrt}(x) = \mu y[\,(y+1)^2 > x\,]$

$\mathit{is\_prime}(x) = \mu y[\,\mu z[\,\mathit{mult}(y,z) = x \wedge z > 1\,] < x\,]$

- class of general recursive functions is Turing complete

- counterprograms

$X, Y, Z, \ldots$ variables, $A, B, C, \ldots$ labels

instructions (with or without label)

A: $X \leftarrow 0$	set to zero
A: $X \leftarrow Y + 1$	assign increment
A: $X \leftarrow Y \dot{-} 1$	assign decrement
A: if $X = 0$ goto B	conditional jump

- multiplying non-negative integers

$$
\begin{array}{ll}
 & U \leftarrow 0 \\
 & Z \leftarrow 0 \\
A: & \texttt{if } X = 0 \texttt{ goto } C \\
 & X \leftarrow X - 1 \\
 & V \leftarrow Y + 1 \\
 & V \leftarrow V - 1 \\
B: & \texttt{if } V = 0 \texttt{ goto } A \\
 & V \leftarrow V - 1 \\
 & Z \leftarrow Z + 1 \\
 & \texttt{if } U = 0 \texttt{ goto } B
\end{array}
$$

- counter program simulating Turing machine

  binary representations, labels for each machine state

  ID $xqay$ corresponds to counter X, Y containing x, ay

$xqby \vdash^* xcq'y$

```
q : "X <- 2*X + c"
    "Y <- Y/2"
    if U=0 goto q'
```

$xaqby \vdash^* xq'acy$

```
q : "X <- X/2"
    "Y <- 2*Y + c"
    "Y <- 2*Y + a"
    if U=0 goto q'
```

- Turing machine simulating counter program

```
** U : 0 * Z : 1100 * X : #10 * Y : 11 ** ... scratch ...
```

  states for each instruction

  reset, increment, decrement and test on zero

- counter programs are Turing complete