

# Teoria da Computação

---

## Introdução e Máquinas de Turing

Guilherme Meira

# Agenda

1. Apresentação da disciplina

2. Introdução

3. Máquinas de Turing

# Apresentação da disciplina

**Disciplina** Teoria da Computação

**Professor** Guilherme Meira

**Horário** Sexta-feiras, das 18:50h às 22:00h

- Intervalo de 10 minutos por volta das 20:20h
- Chamada ao final da aula

**Avaliação** Duas provas + duas avaliações multidisciplinares

- 7 pontos de prova ( $P_1$  e  $P_2$ )
- 3 pontos de Avaliação Multidisciplinar ( $T_1$  e  $T_2$ )
- $M_P = \frac{P_1 + T_1 + P_2 + T_2}{2}$
- $M_P \geq 7$ : aprovado
- $M_P < 7$ : prova final ( $P_F$ )
- $M_F = \frac{M_P + P_F}{2}$
- $M_F \geq 5$ : aprovado
- $M_F < 5$ : gostou tanto da matéria que vai fazer de novo

# Apresentação da disciplina

## Trabalhos Avaliação Multidisciplinar

- Individual
- Questões objetivas

**Livro** Introdução à Teoria da Computação (Michael Sipser - 3ª edição)

- Outros**
- Honestidade acadêmica
  - Comportamento em sala
  - Feedback!

# Agenda

1. Apresentação da disciplina

2. Introdução

3. Máquinas de Turing

# Introdução

Pra que serve Teoria da Computação?

# Introdução

Pra que serve Teoria da Computação?

Serve pra nada.  
Meu negócio é programar!



## Introdução

Considere uma coleção de dominós:

$$\left\{ \left[ \frac{b}{ca} \right], \left[ \frac{a}{ab} \right], \left[ \frac{ca}{a} \right], \left[ \frac{abc}{c} \right] \right\}$$

É possível organizar os dominós (com repetição) de forma que a sequência de cima seja igual à de baixo?



## Introdução

Considere uma coleção de dominós:

$$\left\{ \left[ \frac{b}{ca} \right], \left[ \frac{a}{ab} \right], \left[ \frac{ca}{a} \right], \left[ \frac{abc}{c} \right] \right\}$$

É possível organizar os dominós (com repetição) de forma que a sequência de cima seja igual à de baixo? Neste caso, sim:

$$\left[ \frac{a}{ab} \right], \left[ \frac{b}{ca} \right], \left[ \frac{ca}{a} \right], \left[ \frac{a}{ab} \right], \left[ \frac{abc}{c} \right]$$

## Introdução

Considere uma coleção de dominós:

$$\left\{ \left[ \frac{b}{ca} \right], \left[ \frac{a}{ab} \right], \left[ \frac{ca}{a} \right], \left[ \frac{abc}{c} \right] \right\}$$

É possível organizar os dominós (com repetição) de forma que a sequência de cima seja igual à de baixo? Neste caso, sim:

$$\left[ \frac{a}{ab} \right], \left[ \frac{b}{ca} \right], \left[ \frac{ca}{a} \right], \left[ \frac{a}{ab} \right], \left[ \frac{abc}{c} \right]$$

Escreva um programa que determine se é possível para qualquer conjunto de dominós.

## Introdução

Considere uma coleção de dominós:

$$\left\{ \left[ \frac{b}{ca} \right], \left[ \frac{a}{ab} \right], \left[ \frac{ca}{a} \right], \left[ \frac{abc}{c} \right] \right\}$$

É possível organizar os dominós (com repetição) de forma que a sequência de cima seja igual à de baixo? Neste caso, sim:

$$\left[ \frac{a}{ab} \right], \left[ \frac{b}{ca} \right], \left[ \frac{ca}{a} \right], \left[ \frac{a}{ab} \right], \left[ \frac{abc}{c} \right]$$

Escreva um programa que determine se é possível para qualquer conjunto de dominós. (Dica: **é impossível**)

# Introdução



## Introdução

Qual dos algoritmos é melhor para ordenar uma sequência de números?

```
void sort1(int arr[], int n) {  
    int i, j;  
    for (i = 0; i < n-1; i++)  
        for (j = 0; j < n-i-1; j++)  
            if (arr[j] > arr[j+1])  
                swap(&arr[j], &arr[j+1]);  
}
```

# Introdução

Qual dos algoritmos é melhor para ordenar uma sequência de números?

```
void merge(int arr[], int l, int m, int r) {
    int i, j, k, n1 = m - l + 1, n2 = r - m;
    int L[n1], R[n2];
    for (i = 0; i < n1; i++) L[i] = arr[l + i];
    for (j = 0; j < n2; j++) R[j] = arr[m + 1 + j];
    i = 0; j = 0; k = l;
    while (i < n1 && j < n2) {
        if (L[i] <= R[j]) {
            arr[k] = L[i]; i++;
        } else {
            arr[k] = R[j]; j++;
        }
        k++;
    }
    while (i < n1) {
        arr[k] = L[i];
        i++; k++;
    }
    while (j < n2) {
        arr[k] = R[j];
        j++; k++;
    }
}
```

```
void sort2(int arr[], int l, int r) {
    if (l < r) {
        int m = l+(r-l)/2;

        sort2(arr, l, m);
        sort2(arr, m+1, r);

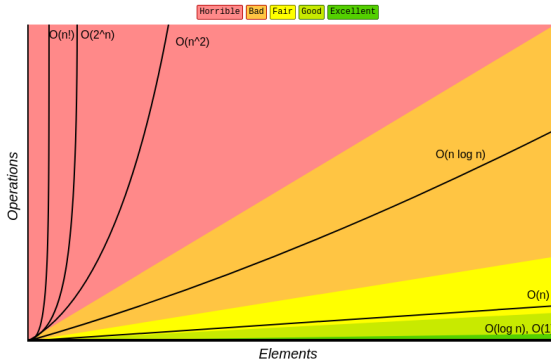
        merge(arr, l, m, r);
    }
}
```

## Introdução

- Algoritmo 1 é o **Bubble Sort**
- Algoritmo 2 é o **Merge Sort**

# Introdução

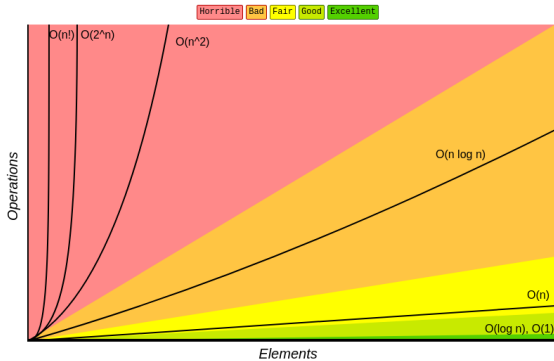
- Algoritmo 1 é o **Bubble Sort**
- Algoritmo 2 é o **Merge Sort**
- A complexidade do Algoritmo 1 é  $O(n^2)$
- A complexidade do Algoritmo 2 é  $O(n \log n)$





# Introdução

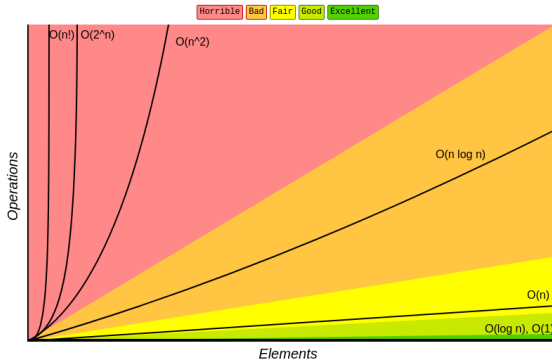
E se você tem restrição de memória?



# Introdução

E se você tem restrição de memória?

- A complexidade de espaço do Algoritmo 1 é  $O(1)$
- A complexidade de espaço do Algoritmo 2 é  $O(n)$



# Introdução

- Estudaremos duas grandes áreas da Teoria da Computação:
  - **Computabilidade:** O que computadores podem e não podem fazer
  - **Complexidade:** O quão eficiente um computador pode ser em um problema
- Conhecimento básico de Teoria da Computação pode ser uma ferramenta valiosa!
- Nossa primeira pergunta: **o que é um computador?**

# Agenda

1. Apresentação da disciplina
2. Introdução
3. Máquinas de Turing

# Máquinas de Turing

- Propostas por Alan Turing em 1936
- É um modelo matemático de um computador
- Tudo que um computador real pode fazer, uma Máquina de Turing pode fazer

# Máquinas de Turing

*Alan Turing (1912-1954)*

- Considerado o pai da Ciência da Computação e da Inteligência Artificial
- Durante a Segunda Guerra, teve papel fundamental na quebra da criptografia utilizada pelos alemães, incluindo a máquina Enigma



# Máquinas de Turing

Alan Turing (1912-1954)

- Propôs o **Teste de Turing**, para determinar se uma máquina é inteligente
- Se interessou, também, por biologia matemática, prevendo reações químicas que só seriam observadas anos depois

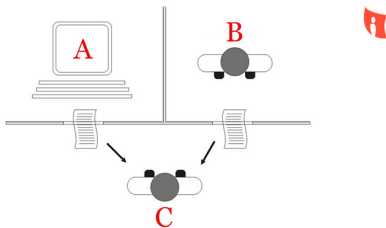


Image by [https://commons.wikimedia.org/wiki/File:Turing\\_test\\_diagram.png](https://commons.wikimedia.org/wiki/File:Turing_test_diagram.png)

# Máquinas de Turing

*Alan Turing (1912-1954)*

- Foi condenado por “indecência”, devido a sua homossexualidade
- Para não ser preso, aceitou se submeter a um “tratamento” que o deixou impotente e lhe causou o crescimento de seios
- Cometeu suicídio se envenenando com cianeto



# Máquinas de Turing

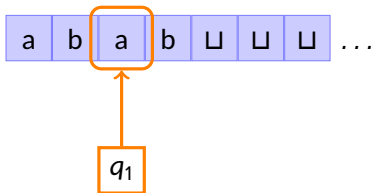
*Alan Turing (1912-1954)*

- Em 2009, o governo britânico se desculpou pelo tratamento dado a Turing
- Hoje, o Prêmio Turing, dado pela ACM, é considerado o prêmio Nobel da computação
- Em 2012, o Google homenageou seu 100º aniversário com um doodle



# Máquinas de Turing

- A memória da máquina consiste de uma fita infinita
- Uma cabeça de leitura e escrita percorre a fita
- A máquina armazena seu estado atual



# Máquinas de Turing

- Funcionamento:

- A entrada da máquina é colocada na fita e a máquina é iniciada
- A cabeça de leitura lê a fita
- Baseado no estado atual, a máquina escreve na posição atual da fita e move a cabeça para a esquerda ou para a direita e muda de estado
- Se o próximo estado é um estado **aceita** ou **rejeita**, a execução termina
- Caso contrário, o processo se repete

# Máquinas de Turing

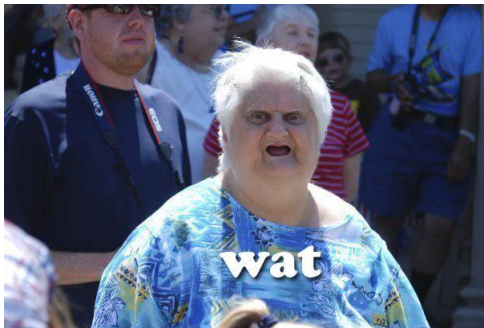
Descreva uma Máquina de Turing que verifica se uma palavra pertence à linguagem:

$$B = \{w\#w \mid w \in \{0,1\}^*\}$$

## Máquinas de Turing

Descreva uma Máquina de Turing que verifica se uma palavra pertence à linguagem:

$$B = \{w\#w \mid w \in \{0,1\}^*\}$$



# Máquinas de Turing

Descreva uma Máquina de Turing que verifica se uma palavra pertence à linguagem:

$$B = \{w\#w \mid w \in \{0,1\}^*\}$$

- Uma **linguagem** é um conjunto de palavras
- Exemplo: *idades* = {*vitória*, *vila velha*, *serra*}
- Aqui, geralmente vamos descrever linguagens matematicamente

# Máquinas de Turing

Descreva uma Máquina de Turing que verifica se uma palavra pertence à linguagem:

$$B = \{w\#w \mid w \in \{0,1\}^*\}$$

- Notação de conjuntos. Vamos descrever todas as palavras da nossa linguagem dentro das chaves

# Máquinas de Turing

Descreva uma Máquina de Turing que verifica se uma palavra pertence à linguagem:

$$B = \{w\#w \mid w \in \{0,1\}^*\}$$

- As palavras da nossa linguagem tem esse formato: queremos duas palavras iguais ( $w$ ) separadas por um caractere  $\#$
- O formato de  $w$  será mostrado a seguir
- Exemplo: se  $w$  é *dia*,  $w\#w$  é *dia#dia*



# Máquinas de Turing

Descreva uma Máquina de Turing que verifica se uma palavra pertence à linguagem:

$$B = \{w\#w \mid w \in \{0,1\}^*\}$$

- Lê-se “tal que”

# Máquinas de Turing

Descreva uma Máquina de Turing que verifica se uma palavra pertence à linguagem:

$$B = \{w\#w \mid w \in \{0,1\}^*\}$$

- Lê-se “pertence”

# Máquinas de Turing

Descreva uma Máquina de Turing que verifica se uma palavra pertence à linguagem:

$$B = \{w\#w \mid w \in \{0,1\}^*\}$$

- $\{0,1\}$  representa um número que pode ser 0 ou 1
- O asterisco diz que podemos ter zero ou mais ocorrências do que vem antes dele
- Então,  $w$  é qualquer sequência de zeros e uns

# Máquinas de Turing

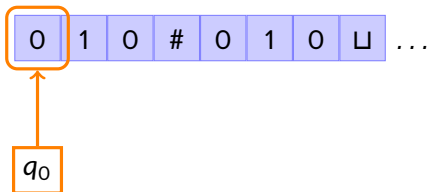
Descreva uma Máquina de Turing que verifica se uma palavra pertence à linguagem:

$$B = \{w\#w \mid w \in \{0,1\}^*\}$$

- Nossa linguagem é composta de todas as palavras compostas por duas sequências **iguais** de zeros e uns separadas por um #
- Exemplos:
  - **001#001** pertence à linguagem
  - **#** pertence à linguagem
  - **010101#010101** pertence à linguagem
  - **001#100** **não** pertence à linguagem

# Máquinas de Turing

- Queremos uma Máquina de Turing que diga se uma palavra pertence a essa linguagem
- A palavra inicialmente é colocada na fita
- Lembre-se o que nossa máquina pode fazer:
  - Mover a cabeça para a direita ou esquerda
  - Escrever na fita
  - Alterar o próprio estado

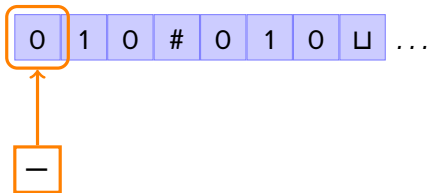


# Máquinas de Turing

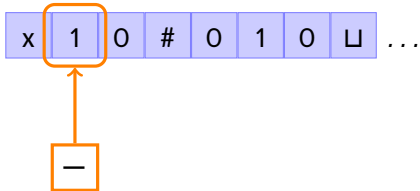
$M_1$  = “Para uma string de entrada  $w$ :

1. Faça um zig-zag pela fita indo de um lado para o outro do símbolo # e verifique se eles são iguais. Se eles não forem iguais ou se não existir um #, **rejeite**. Marque os símbolos que forem verificados ao longo do processo.
2. Quando todos os símbolos à esquerda do # tiverem sido marcados, verifique se sobraram símbolos à direita do #. Se sobraram, **rejeite**, caso contrário, **aceite**.”

## Máquinas de Turing

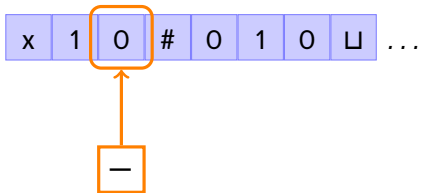


## Máquinas de Turing

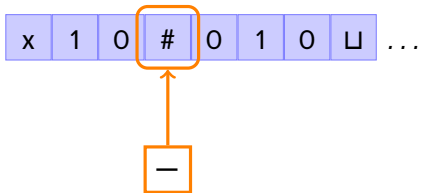




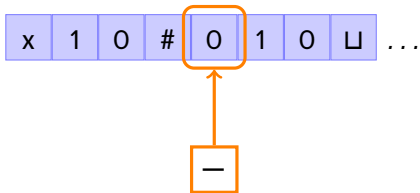
## Máquinas de Turing



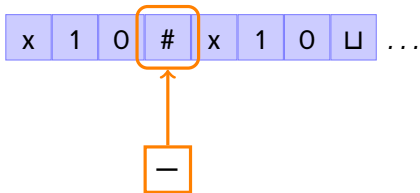
## Máquinas de Turing



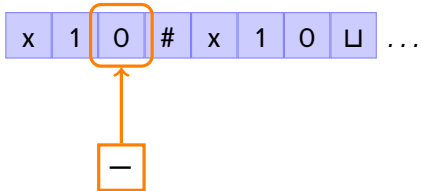
## Máquinas de Turing



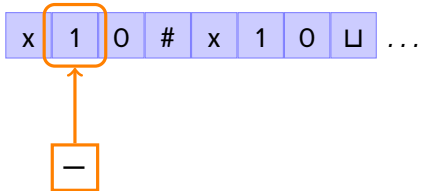
## Máquinas de Turing



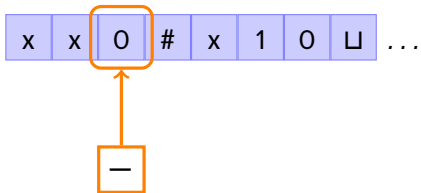
## Máquinas de Turing



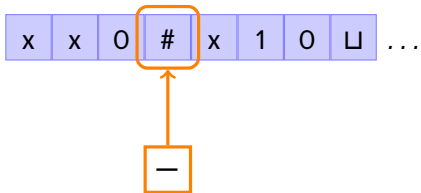
## Máquinas de Turing



## Máquinas de Turing

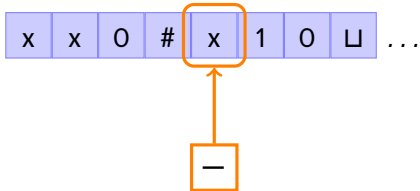


## Máquinas de Turing

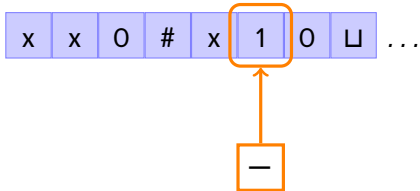




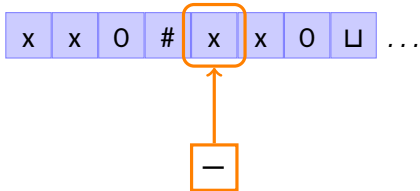
## Máquinas de Turing



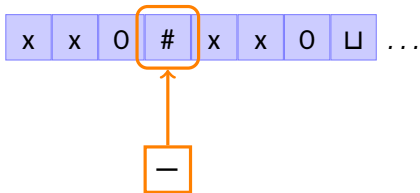
## Máquinas de Turing



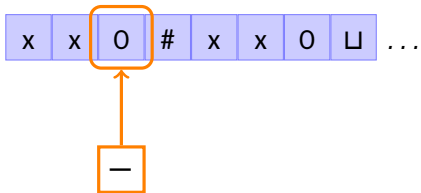
## Máquinas de Turing



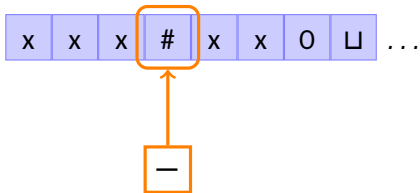
## Máquinas de Turing



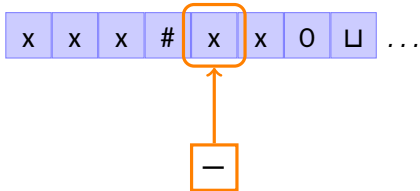
## Máquinas de Turing



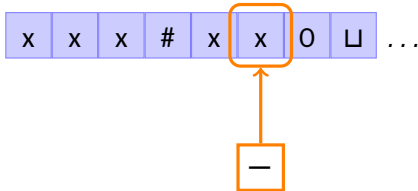
## Máquinas de Turing



## Máquinas de Turing

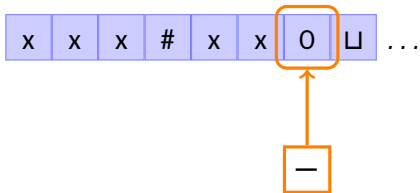


## Máquinas de Turing

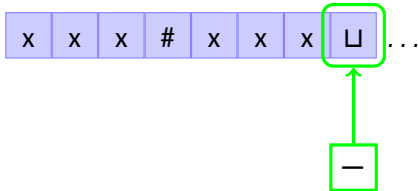




## Máquinas de Turing



## Máquinas de Turing



# Máquinas de Turing

Neste curso, descreveremos Máquinas de Turing de três maneiras:

- **Descrição do algoritmo:** descrevemos os passos que um algoritmo realiza para resolver o problema, sem nos preocuparmos com a implementação em uma Máquina de Turing
- **Descrição em alto nível da Máquina de Turing:** descrevemos com palavras os passos realizados pela Máquina de Turing para resolver o problema. Acabamos de ver uma descrição assim
- **Descrição formal da Máquina de Turing:** descrevemos matematicamente a Máquina de Turing

Raramente utilizamos a descrição formal por ser muito trabalhosa, mas vamos ver como ela funciona.

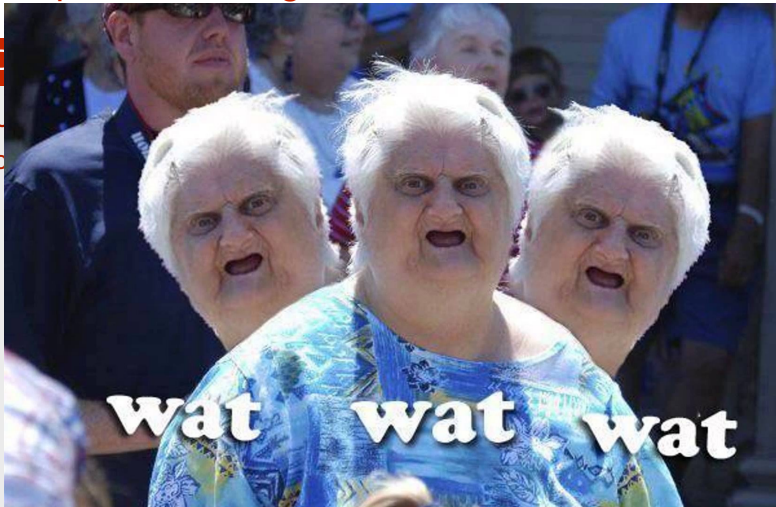
# Máquinas de Turing

## Definição

Uma Máquina de Turing é uma 7-tupla  $(Q, \Sigma, \Gamma, \delta, q_0, q_{aceita}, q_{rejeita})$  onde  $Q$ ,  $\Sigma$  e  $\Gamma$  são conjuntos finitos e:

- $Q$  é o conjunto de estados
- $\Sigma$  é o alfabeto de entrada, não incluindo o símbolo vazio  $\sqcup$
- $\Gamma$  é o alfabeto da fita, onde  $\sqcup \in \Gamma$  e  $\Sigma \subseteq \Gamma$
- $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{D, E\}$  é a função de transição
- $q_0 \in Q$  é o estado inicial
- $q_{aceita} \in Q$  é o estado aceita
- $q_{rejeita} \in Q$  é o estado rejeita, onde  $q_{aceita} \neq q_{rejeita}$

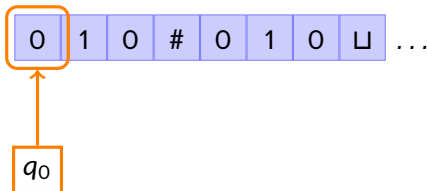
## Máquinas de Turing



• grejta e q e o estado grejta, onde qaceita 7 grejta

# Máquinas de Turing

$Q$  é o conjunto dos estados que a Máquina de Turing pode estar.

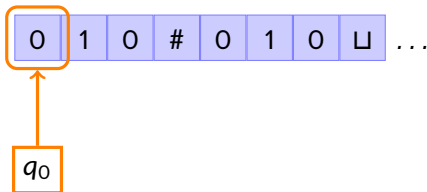


Se a Máquina pode estar nos estados  $q_0$ ,  $q_1$  e  $q_2$ , então:

$$Q = \{q_0, q_1, q_2\}$$

## Máquinas de Turing

$\Sigma$  (sigma) é o conjunto de caracteres de entrada. Contém todos os caracteres que podem estar na fita quando a máquina é iniciada.



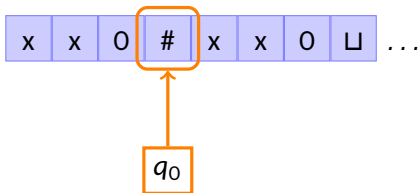
No nosso exemplo anterior:

$$\Sigma = \{0, 1, \#\}$$

Não incluímos aqui o caractere vazio ( $\sqcup$ ).

## Máquinas de Turing

$\Gamma$  (gama) é o alfabeto da fita. Inclui todos os caracteres que podem ser escritos na fita em algum ponto do algoritmo.



No nosso exemplo anterior, além dos caracteres da entrada, usamos o caractere  $x$  para marcar as posições já visitadas.

$$\Gamma = \{0, 1, \#, x, \sqcup\}$$

Aqui, incluímos o caractere vazio ( $\sqcup$ ).



# Máquinas de Turing

$q_0$ ,  $q_{aceita}$  e  $q_{rejeita}$  são estados pertencentes a  $Q$  que possuem significado especial.

- $q_0$  é o estado inicial. Quando a máquina é iniciada, ela está nesse estado
- $q_{aceita}$  é o estado aceita. Se a máquina chega nesse estado, ela para de executar e informa que **aceitou** aquele valor de entrada
- $q_{rejeita}$  é o estado final. Quando a máquina chega nesse estado, ela para de executar e informa que **rejeitou** aquele valor de entrada

## Máquinas de Turing

$\delta$  (delta) é a função de transição da Máquina de Turing. Ela que define qual será o comportamento da máquina.

A função  $\delta$  recebe os parâmetros  $(q, a)$ :

$q$  o estado atual da máquina

$a$  o caractere na fita na posição atual da cabeça da máquina

A função  $\delta$  retorna como saída os valores  $(r, b, E)$ :

$r$  o novo estado da máquina

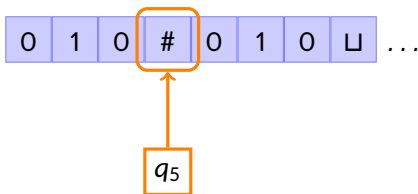
$b$  o valor a ser escrito na fita

$E$  a direção para movimentar a cabeça de leitura (direita ou esquerda)

- Se já estamos no início da fita e a função retorna  $E$ , permanecemos no início da fita

## Máquinas de Turing

Podemos representar uma “foto” de uma Máquina de Turing em determinado ponto da execução. Essa “foto” é chamada de **configuração**. Usamos uma notação especial para representar configurações:



Representamos essa configuração como:

010  $q_5$  #010

## Máquinas de Turing

Dizemos que uma configuração  $C_1$  **produz** uma configuração  $C_2$  se é possível ir da configuração  $C_1$  para a configuração  $C_2$  em um único passo. Exemplo:

A configuração

$$ua\ q_i\ bv$$

**produz**

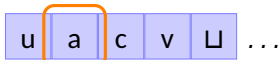
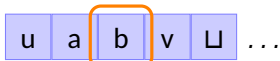
$$u\ q_j\ acv$$

se a função de transição

$$\delta(q_i, b) = (q_j, c, E)$$

# Máquinas de Turing

$$\delta(q_i, b) = (q_j, c, E)$$



# Máquinas de Turing

## Exercícios

- É possível  $ua q_i bv$  produzir  $q_j uabv$  ?

# Máquinas de Turing

## Exercícios

- É possível  $ua q_i bv$  produzir  $q_j uabv$  ?
  - Não. Pois a cabeça moveu duas posições de uma configuração para a outra

# Máquinas de Turing

## Exercícios

- É possível  $ua q_i bv$  produzir  $q_j uabv$  ?
  - Não. Pois a cabeça moveu duas posições de uma configuração para a outra
- É possível  $ua q_i bv$  produzir  $ua q_j bv$  ?



# Máquinas de Turing

## Exercícios

- É possível  $ua q_i bv$  produzir  $q_j uabv$  ?
  - Não. Pois a cabeça moveu duas posições de uma configuração para a outra
- É possível  $ua q_i bv$  produzir  $ua q_j bv$  ?
  - Não. Pois a cabeça não se moveu

# Máquinas de Turing

## Exercícios

- É possível  $ua q_i bv$  produzir  $q_j uabv$  ?
  - Não. Pois a cabeça moveu duas posições de uma configuração para a outra
- É possível  $ua q_i bv$  produzir  $ua q_j bv$  ?
  - Não. Pois a cabeça não se moveu
- É possível  $ua q_i bv$  produzir  $tab q_j v$  ?

# Máquinas de Turing

## Exercícios

- É possível  $ua q_i bv$  produzir  $q_j uabv$  ?
  - Não. Pois a cabeça moveu duas posições de uma configuração para a outra
- É possível  $ua q_i bv$  produzir  $ua q_j bv$  ?
  - Não. Pois a cabeça não se moveu
- É possível  $ua q_i bv$  produzir  $tab q_j v$  ?
  - Não, pois só pode haver mudança na fita na posição da cabeça

# Máquinas de Turing

## Exercícios

- É possível  $ua q_i bv$  produzir  $q_j uabv$  ?
  - Não. Pois a cabeça moveu duas posições de uma configuração para a outra
- É possível  $ua q_i bv$  produzir  $ua q_j bv$  ?
  - Não. Pois a cabeça não se moveu
- É possível  $ua q_i bv$  produzir  $tab q_j v$  ?
  - Não, pois só pode haver mudança na fita na posição da cabeça
- É possível  $ua q_i bv$  produzir  $uab q_i v$  ?

# Máquinas de Turing

## Exercícios

- É possível  $ua q_i bv$  produzir  $q_j uabv$  ?
  - Não. Pois a cabeça moveu duas posições de uma configuração para a outra
- É possível  $ua q_i bv$  produzir  $ua q_j bv$  ?
  - Não. Pois a cabeça não se moveu
- É possível  $ua q_i bv$  produzir  $tab q_j v$  ?
  - Não, pois só pode haver mudança na fita na posição da cabeça
- É possível  $ua q_i bv$  produzir  $uab q_i v$  ?
  - Sim. O estado da máquina não precisa necessariamente mudar a cada transição.

# Máquinas de Turing

## Exercícios

- É possível  $ua q_i bv$  produzir  $q_j uabv$  ?
  - Não. Pois a cabeça moveu duas posições de uma configuração para a outra
- É possível  $ua q_i bv$  produzir  $ua q_j bv$  ?
  - Não. Pois a cabeça não se moveu
- É possível  $ua q_i bv$  produzir  $tab q_j v$  ?
  - Não, pois só pode haver mudança na fita na posição da cabeça
- É possível  $ua q_i bv$  produzir  $uab q_i v$  ?
  - Sim. O estado da máquina não precisa necessariamente mudar a cada transição.  $\delta(q_i, b) = (q_i, b, D)$

# Máquinas de Turing

## Exercícios

- É possível  $ua q_i bv$  produzir  $q_j uabv$  ?
  - Não. Pois a cabeça moveu duas posições de uma configuração para a outra
- É possível  $ua q_i bv$  produzir  $ua q_j bv$  ?
  - Não. Pois a cabeça não se moveu
- É possível  $ua q_i bv$  produzir  $tab q_j v$  ?
  - Não, pois só pode haver mudança na fita na posição da cabeça
- É possível  $ua q_i bv$  produzir  $uab q_i v$  ?
  - Sim. O estado da máquina não precisa necessariamente mudar a cada transição.  $\delta(q_i, b) = (q_i, b, D)$
- É possível  $q_i uabv$  produzir  $q_j uabv$  ?

# Máquinas de Turing

## Exercícios

- É possível  $ua q_i bv$  produzir  $q_j uabv$  ?
  - Não. Pois a cabeça moveu duas posições de uma configuração para a outra
- É possível  $ua q_i bv$  produzir  $ua q_j bv$  ?
  - Não. Pois a cabeça não se moveu
- É possível  $ua q_i bv$  produzir  $tab q_j v$  ?
  - Não, pois só pode haver mudança na fita na posição da cabeça
- É possível  $ua q_i bv$  produzir  $uab q_i v$  ?
  - Sim. O estado da máquina não precisa necessariamente mudar a cada transição.  $\delta(q_i, b) = (q_i, b, D)$
- É possível  $q_i uabv$  produzir  $q_j uabv$  ?
  - Sim. A cabeça da máquina está no início da fita.



# Máquinas de Turing

## Exercícios

- É possível  $ua q_i bv$  produzir  $q_j uabv$  ?
  - Não. Pois a cabeça moveu duas posições de uma configuração para a outra
- É possível  $ua q_i bv$  produzir  $ua q_j bv$  ?
  - Não. Pois a cabeça não se moveu
- É possível  $ua q_i bv$  produzir  $tab q_j v$  ?
  - Não, pois só pode haver mudança na fita na posição da cabeça
- É possível  $ua q_i bv$  produzir  $uab q_i v$  ?
  - Sim. O estado da máquina não precisa necessariamente mudar a cada transição.  $\delta(q_i, b) = (q_i, b, D)$
- É possível  $q_i uabv$  produzir  $q_j uabv$  ?
  - Sim. A cabeça da máquina está no início da fita.  $\delta(q_i, u) = (q_j, u, E)$

# Máquinas de Turing

Uma Máquina de Turing  $M$  **aceita** uma entrada  $w$  se existe uma sequência de configurações  $C_1, C_2, \dots, C_k$  tal que:

- $C_1$  é a configuração inicial de  $M$  para uma entrada  $w$
- Cada  $C_i$  produz  $C_{i+1}$
- $C_k$  é uma configuração de aceitação

# Máquinas de Turing

O conjunto de palavras que  $M$  aceita é a linguagem de  $M$  ou a linguagem reconhecida por  $M$ , denotada como  $L(M)$ .

# Máquinas de Turing

O conjunto de palavras que  $M$  aceita é a linguagem de  $M$  ou a linguagem reconhecida por  $M$ , denotada como  $L(M)$ .

## Definição

Uma linguagem é Turing-reconhecível se existe uma Máquina de Turing que a reconheça.

# Máquinas de Turing

Ao iniciar uma Máquina de Turing, três coisas podem acontecer:

- A máquina **aceita** a entrada
- A máquina **rejeita** a entrada
- A máquina **entra em loop**

Uma máquina que sempre dá uma resposta (nunca entra em loop) é chamada de **decisor**.

# Máquinas de Turing

Ao iniciar uma Máquina de Turing, três coisas podem acontecer:

- A máquina **aceita** a entrada
- A máquina **rejeita** a entrada
- A máquina **entra em loop**

Uma máquina que sempre dá uma resposta (nunca entra em loop) é chamada de **decisor**.

## Definição

Uma linguagem é **Turing-decidível** se existe uma Máquina de Turing que a decida.

## Máquinas de Turing

Descreva uma Máquina de Turing  $M_2$  que decida a linguagem:

$$A = \{0^{2^n} \mid n \geq 0\}$$

## Máquinas de Turing

Descreva uma Máquina de Turing  $M_2$  que decida a linguagem:

$$A = \{0^{2^n} \mid n \geq 0\}$$

Esta linguagem consiste de todas as palavras formadas por zeros cujo tamanho seja uma potência de 2.

Vamos começar com uma descrição em alto nível da máquina.  
Idéias?



## Máquinas de Turing

$M_2$  = “Para uma string de entrada  $w$ :

1. Varra a fita da esquerda para a direita, marcando um zero sim e outro não
2. Se no estágio 1 a fita continha apenas um zero, **aceite**
3. Se no estágio 1 a fita continha mais de um zero e o número de zeros foi ímpar, **rejeite**
4. Volte a cabeça para o início da fita
5. Vá para o estágio 1”

## Máquinas de Turing

$M_2$  = “Para uma string de entrada  $w$ :

1. Varra a fita da esquerda para a direita, marcando um zero sim e outro não
2. Se no estágio 1 a fita continha apenas um zero, **aceite**
3. Se no estágio 1 a fita continha mais de um zero e o número de zeros foi ímpar, **rejeite**
4. Volte a cabeça para o início da fita
5. Vá para o estágio 1”

Como voltar para o início da fita?

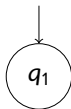
## Máquinas de Turing

$M_2$  = “Para uma string de entrada  $w$ :

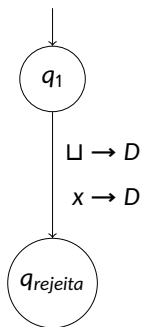
1. Varra a fita da esquerda para a direita, marcando um zero sim e outro não
2. Se no estágio 1 a fita continha apenas um zero, **aceite**
3. Se no estágio 1 a fita continha mais de um zero e o número de zeros foi ímpar, **rejeite**
4. Volte a cabeça para o início da fita
5. Vá para o estágio 1”

Como voltar para o início da fita? Vamos começar definindo  $\delta$ . As outras partes da descrição ficam fáceis depois.

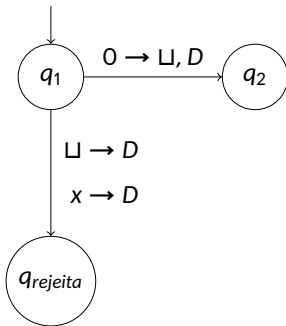
# Máquinas de Turing



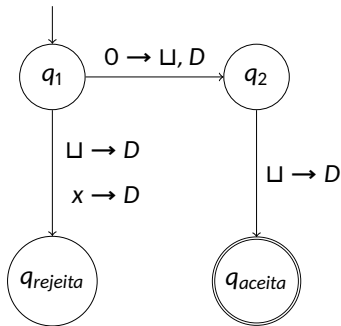
# Máquinas de Turing



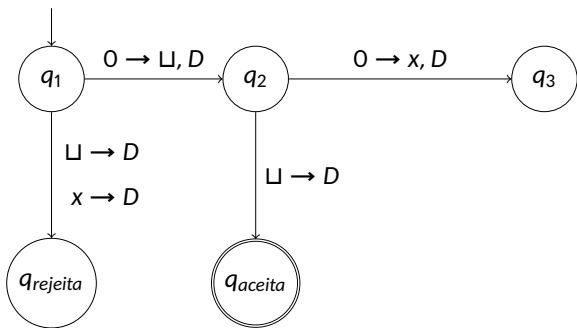
# Máquinas de Turing



## Máquinas de Turing

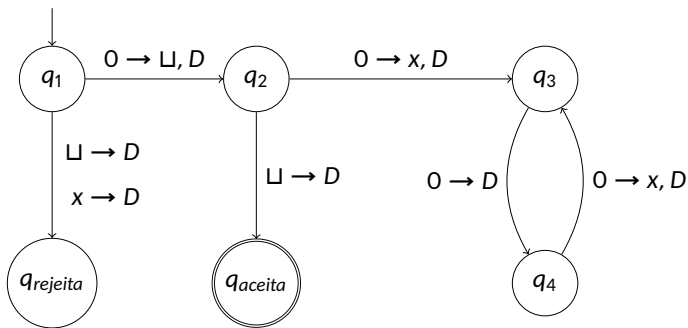


# Máquinas de Turing

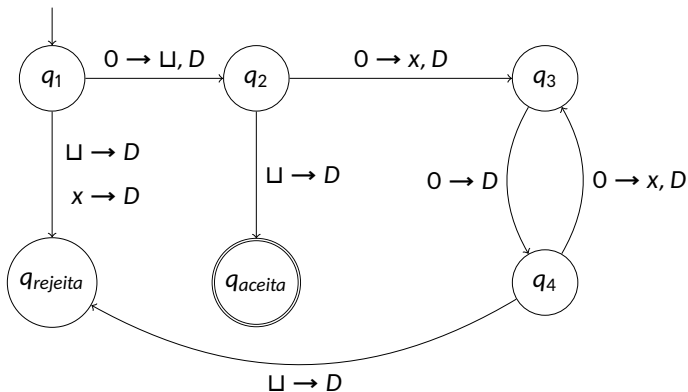




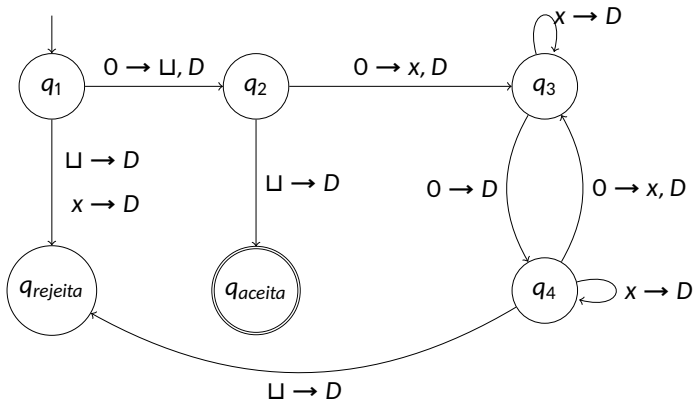
# Máquinas de Turing



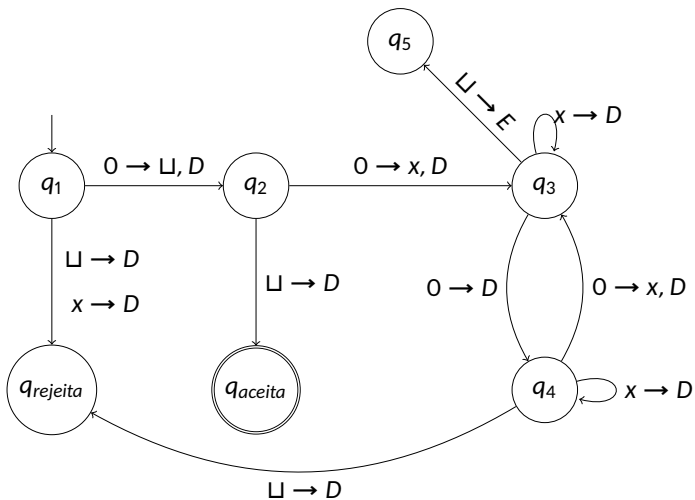
## Máquinas de Turing



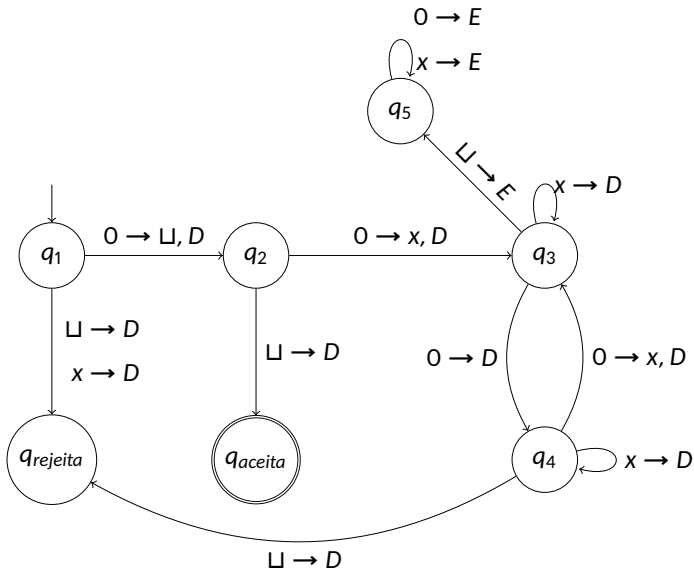
## Máquinas de Turing



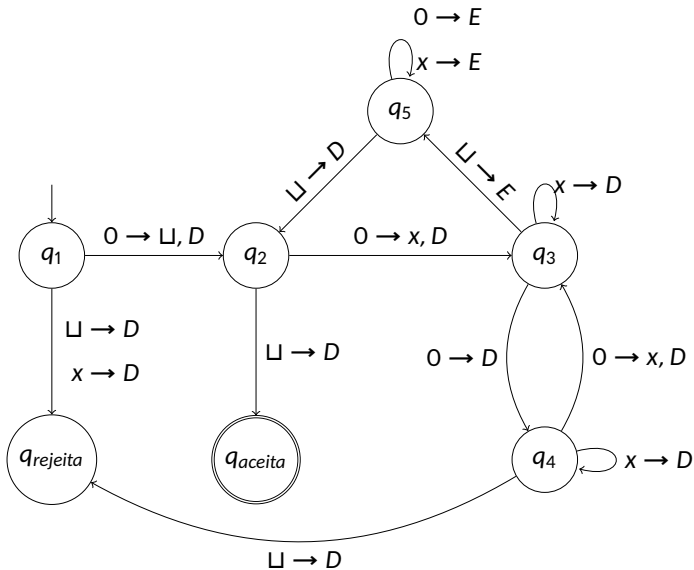
# Máquinas de Turing



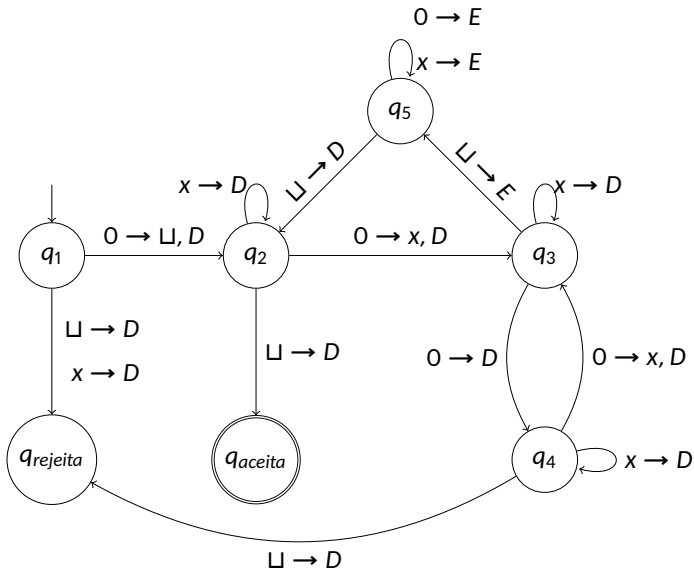
# Máquinas de Turing



# Máquinas de Turing



# Máquinas de Turing



## Máquinas de Turing

Lembre-se: em cada estado deve haver uma transição para cada uma das possibilidades do alfabeto da fita  $\Gamma$ .



# Máquinas de Turing

Lembre-se: em cada estado deve haver uma transição para cada uma das possibilidades do alfabeto da fita  $\Gamma$ .

Ainda faltam as 6 outras partes da definição da Máquina!

- $Q =$

## Máquinas de Turing

Lembre-se: em cada estado deve haver uma transição para cada uma das possibilidades do alfabeto da fita  $\Gamma$ .

Ainda faltam as 6 outras partes da definição da Máquina!

- $Q = \{q_1, q_2, q_3, q_4, q_5, q_{aceita}, q_{rejeita}\}$

# Máquinas de Turing

Lembre-se: em cada estado deve haver uma transição para cada uma das possibilidades do alfabeto da fita  $\Gamma$ .

Ainda faltam as 6 outras partes da definição da Máquina!

- $Q = \{q_1, q_2, q_3, q_4, q_5, q_{aceita}, q_{rejeita}\}$
- $\Sigma =$

# Máquinas de Turing

Lembre-se: em cada estado deve haver uma transição para cada uma das possibilidades do alfabeto da fita  $\Gamma$ .

Ainda faltam as 6 outras partes da definição da Máquina!

- $Q = \{q_1, q_2, q_3, q_4, q_5, q_{aceita}, q_{rejeita}\}$
- $\Sigma = \{0\}$

# Máquinas de Turing

Lembre-se: em cada estado deve haver uma transição para cada uma das possibilidades do alfabeto da fita  $\Gamma$ .

Ainda faltam as 6 outras partes da definição da Máquina!

- $Q = \{q_1, q_2, q_3, q_4, q_5, q_{aceita}, q_{rejeita}\}$
- $\Sigma = \{0\}$
- $\Gamma =$

# Máquinas de Turing

Lembre-se: em cada estado deve haver uma transição para cada uma das possibilidades do alfabeto da fita  $\Gamma$ .

Ainda faltam as 6 outras partes da definição da Máquina!

- $Q = \{q_1, q_2, q_3, q_4, q_5, q_{aceita}, q_{rejeita}\}$
- $\Sigma = \{0\}$
- $\Gamma = \{0, x, \sqcup\}$

# Máquinas de Turing

Lembre-se: em cada estado deve haver uma transição para cada uma das possibilidades do alfabeto da fita  $\Gamma$ .

Ainda faltam as 6 outras partes da definição da Máquina!

- $Q = \{q_1, q_2, q_3, q_4, q_5, q_{aceita}, q_{rejeita}\}$
- $\Sigma = \{0\}$
- $\Gamma = \{0, x, \sqcup\}$
- $q_1$  é o estado inicial

# Máquinas de Turing

Lembre-se: em cada estado deve haver uma transição para cada uma das possibilidades do alfabeto da fita  $\Gamma$ .

Ainda faltam as 6 outras partes da definição da Máquina!

- $Q = \{q_1, q_2, q_3, q_4, q_5, q_{aceita}, q_{rejeita}\}$
- $\Sigma = \{0\}$
- $\Gamma = \{0, x, \sqcup\}$
- $q_1$  é o estado inicial
- $q_{aceita}$  é o estado aceita



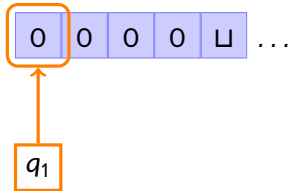
# Máquinas de Turing

Lembre-se: em cada estado deve haver uma transição para cada uma das possibilidades do alfabeto da fita  $\Gamma$ .

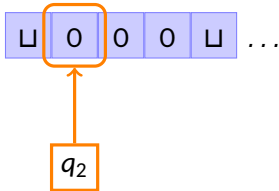
Ainda faltam as 6 outras partes da definição da Máquina!

- $Q = \{q_1, q_2, q_3, q_4, q_5, q_{aceita}, q_{rejeita}\}$
- $\Sigma = \{0\}$
- $\Gamma = \{0, x, \sqcup\}$
- $q_1$  é o estado inicial
- $q_{aceita}$  é o estado aceita
- $q_{rejeita}$  é o estado rejeita

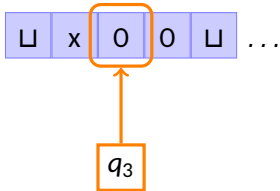
# Máquinas de Turing



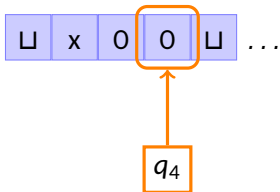
## Máquinas de Turing



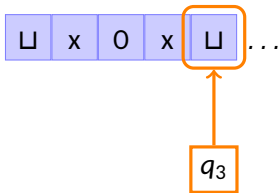
## Máquinas de Turing



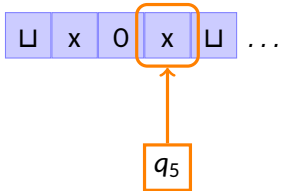
## Máquinas de Turing



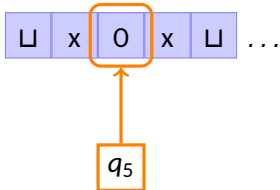
## Máquinas de Turing



## Máquinas de Turing

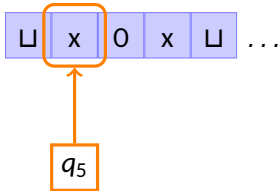


## Máquinas de Turing

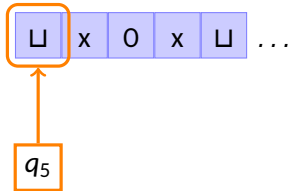




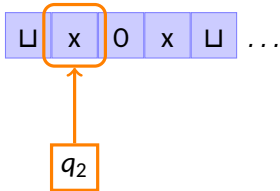
## Máquinas de Turing



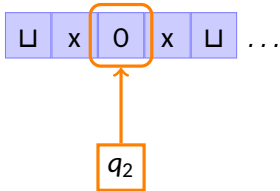
## Máquinas de Turing



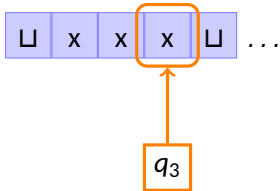
## Máquinas de Turing



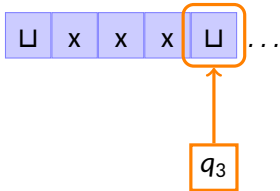
## Máquinas de Turing



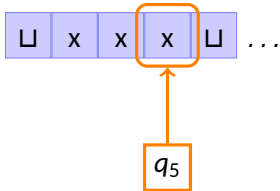
## Máquinas de Turing



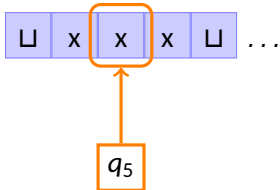
## Máquinas de Turing



## Máquinas de Turing

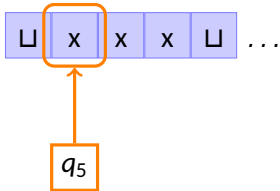


# Máquinas de Turing

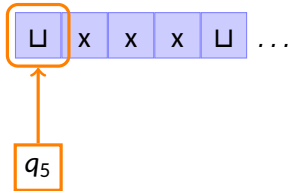




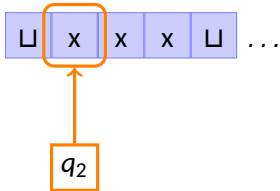
# Máquinas de Turing



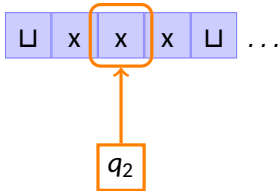
## Máquinas de Turing



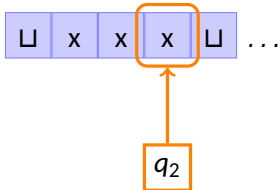
## Máquinas de Turing



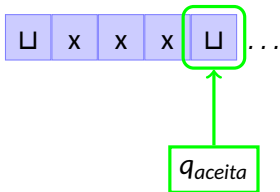
## Máquinas de Turing



## Máquinas de Turing



## Máquinas de Turing



# Máquinas de Turing

## Exercício

Agora, vamos fazer a descrição formal da primeira máquina que vimos no curso.

Queremos uma Máquina de Turing que decida a linguagem:

$$B = \{w\#w \mid w \in \{0,1\}^*\}$$

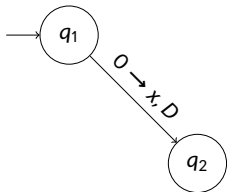
Nossa estratégia era fazer zigue-zague pela fita marcando os elementos antes e depois do #.

# Máquinas de Turing

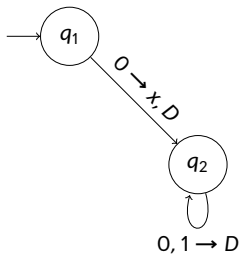




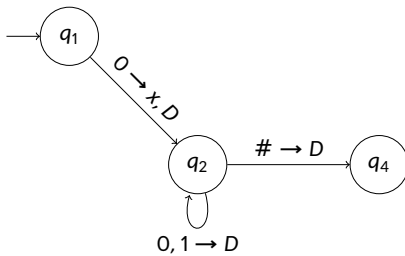
# Máquinas de Turing



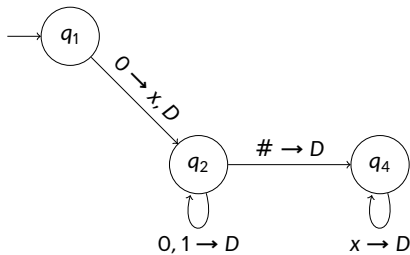
# Máquinas de Turing



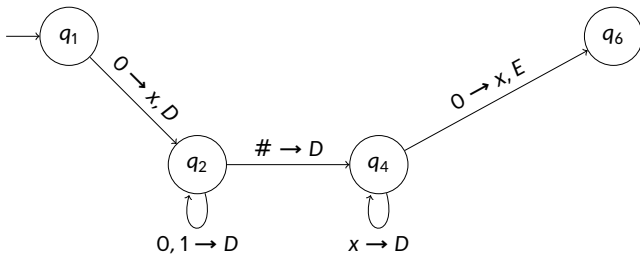
# Máquinas de Turing



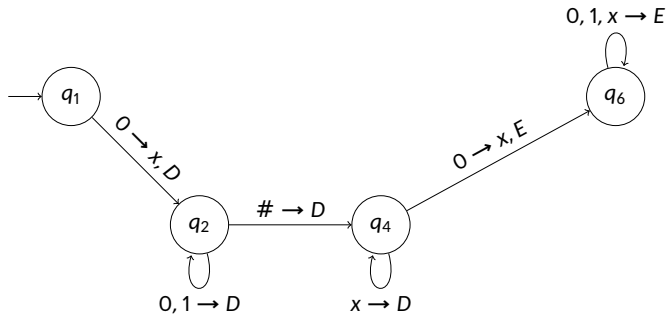
# Máquinas de Turing



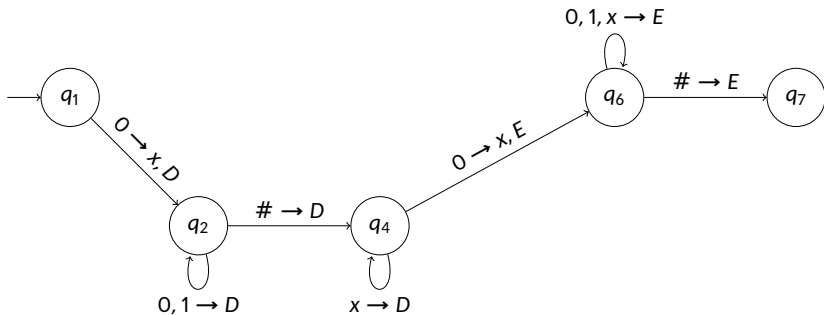
# Máquinas de Turing



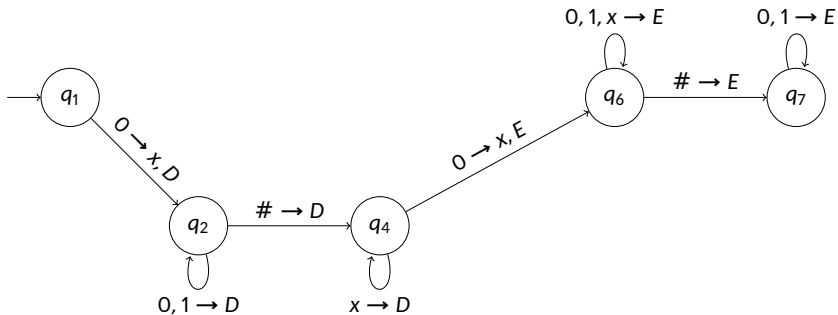
# Máquinas de Turing



# Máquinas de Turing

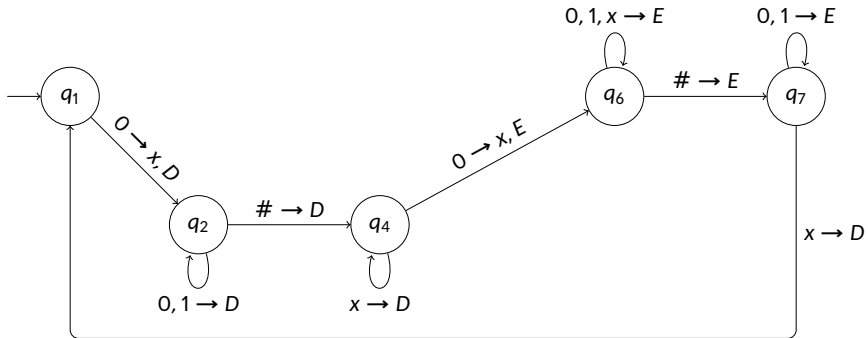


# Máquinas de Turing

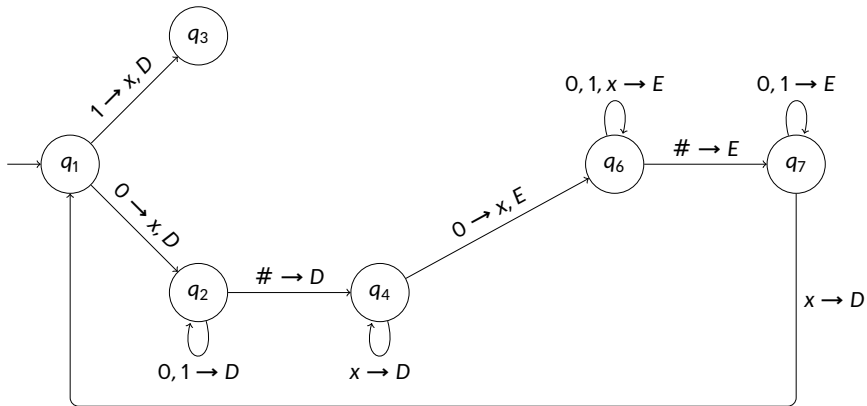




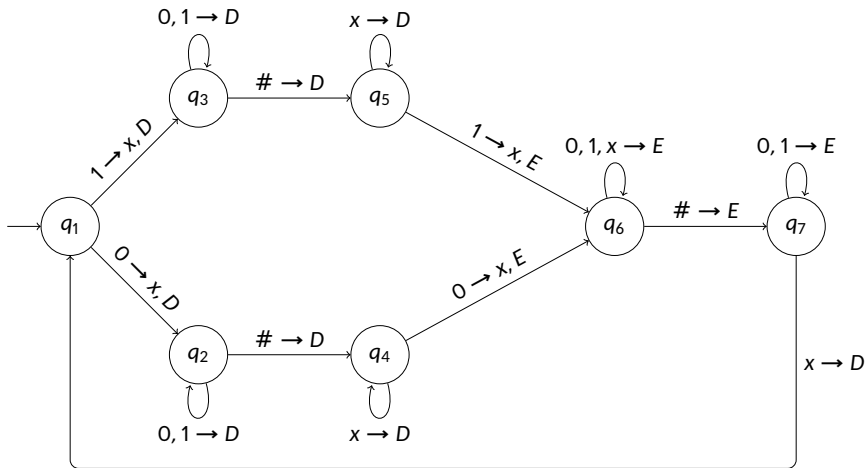
# Máquinas de Turing



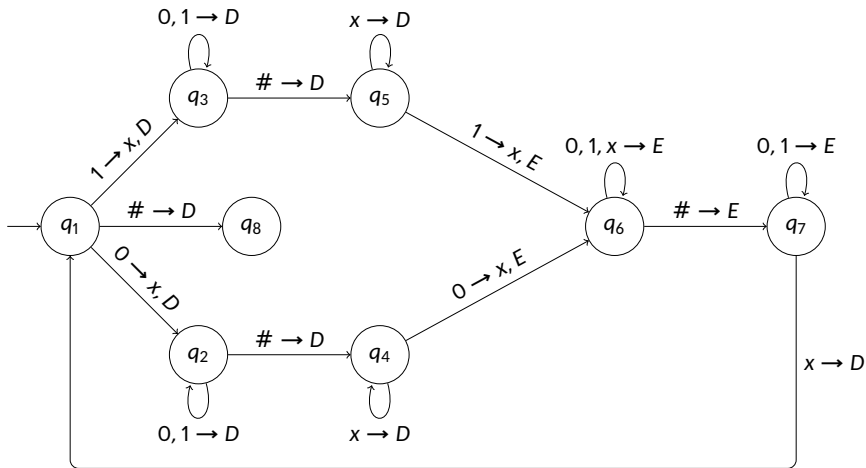
# Máquinas de Turing



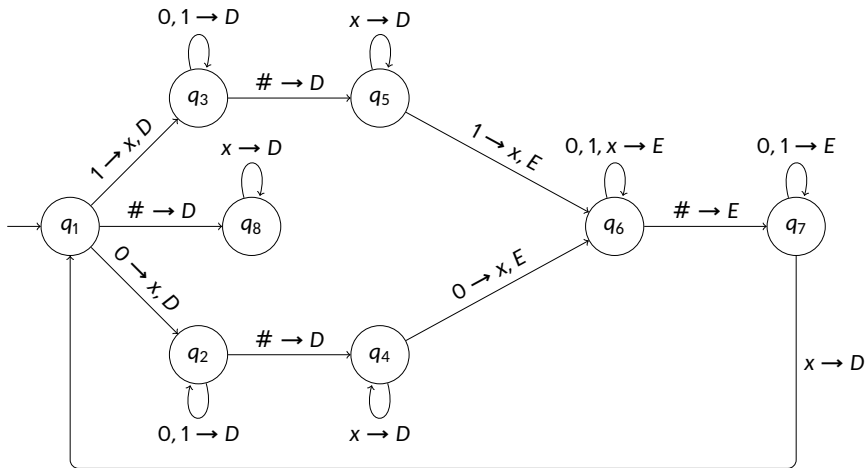
# Máquinas de Turing



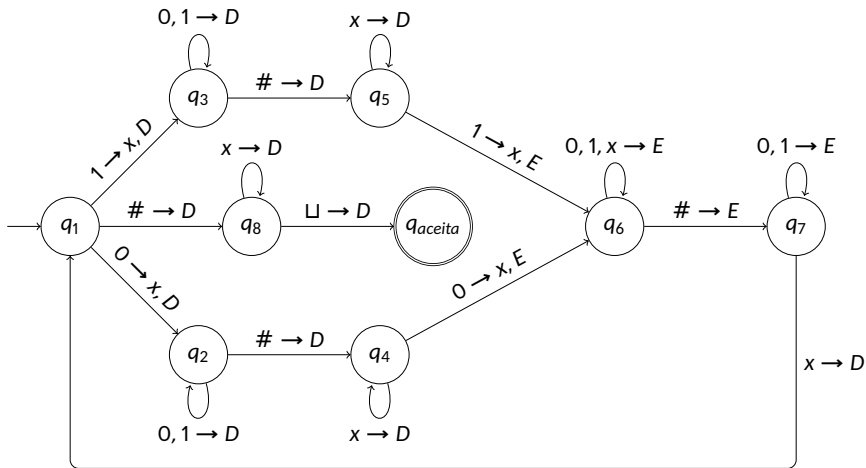
# Máquinas de Turing



# Máquinas de Turing



# Máquinas de Turing



## Máquinas de Turing

No diagrama anterior, todas as transições não representadas vão levar para  $q_{rejeita}$ .

Com a definição do  $\delta$ , o restante da definição é simples:

- $Q =$

## Máquinas de Turing

No diagrama anterior, todas as transições não representadas vão levar para  $q_{rejeita}$ .

Com a definição do  $\delta$ , o restante da definição é simples:

- $Q = \{q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_{aceita}, q_{rejeita}\}$



## Máquinas de Turing

No diagrama anterior, todas as transições não representadas vão levar para  $q_{rejeita}$ .

Com a definição do  $\delta$ , o restante da definição é simples:

- $Q = \{q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_{aceita}, q_{rejeita}\}$
- $\Sigma =$

## Máquinas de Turing

No diagrama anterior, todas as transições não representadas vão levar para  $q_{rejeita}$ .

Com a definição do  $\delta$ , o restante da definição é simples:

- $Q = \{q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_{aceita}, q_{rejeita}\}$
- $\Sigma = \{0, 1, \#\}$

## Máquinas de Turing

No diagrama anterior, todas as transições não representadas vão levar para  $q_{rejeita}$ .

Com a definição do  $\delta$ , o restante da definição é simples:

- $Q = \{q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_{aceita}, q_{rejeita}\}$
- $\Sigma = \{0, 1, \#\}$
- $\Gamma =$

## Máquinas de Turing

No diagrama anterior, todas as transições não representadas vão levar para  $q_{rejeita}$ .

Com a definição do  $\delta$ , o restante da definição é simples:

- $Q = \{q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_{aceita}, q_{rejeita}\}$
- $\Sigma = \{0, 1, \#\}$
- $\Gamma = \{0, 1, \#, x, \sqcup\}$

## Máquinas de Turing

No diagrama anterior, todas as transições não representadas vão levar para  $q_{rejeita}$ .

Com a definição do  $\delta$ , o restante da definição é simples:

- $Q = \{q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_{aceita}, q_{rejeita}\}$
- $\Sigma = \{0, 1, \#\}$
- $\Gamma = \{0, 1, \#, x, \sqcup\}$
- $q_1$  é o estado inicial
- $q_{aceita}$  é o estado aceita
- $q_{rejeita}$  é o estado rejeita

# Máquinas de Turing

## Exercício

Faça uma **descrição de alto nível** de uma Máquina de Turing que decida a linguagem

$$C = \{a^i b^j c^k \mid i \times j = k \text{ e } i, j, k \geq 1\}$$

# Máquinas de Turing

## Exercício

Faça uma **descrição de alto nível** de uma Máquina de Turing que decida a linguagem

$$C = \{a^i b^j c^k \mid i \times j = k \text{ e } i, j, k \geq 1\}$$

Exemplos de palavras dessa linguagem:

- $i = 1, j = 3, k = 3$ : abbbccc
- $i = 3, j = 2, k = 6$ : aaabbcccccc
- $i = 4, j = 3, k = 12$ : aaaabbbcccccccccc

## Máquinas de Turing

$M_3$  = “Para uma string de entrada  $w$ :

1. Varra a fita da esquerda para a direita para determinar se a entrada está no formato  $a^+b^+c^+$  e **rejeite** se não estiver.



## Máquinas de Turing

$M_3$  = “Para uma string de entrada  $w$ :

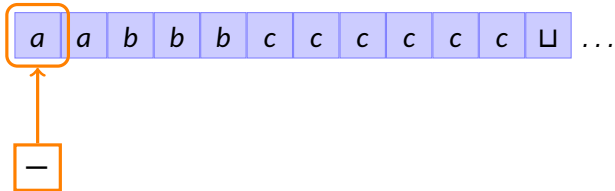
1. Varra a fita da esquerda para a direita para determinar se a entrada está no formato  $a^+b^+c^+$  e **rejeite** se não estiver.
  - Enquanto o  $*$  significa zero ou mais caracteres, o  $+$  significa um ou mais caracteres.

## Máquinas de Turing

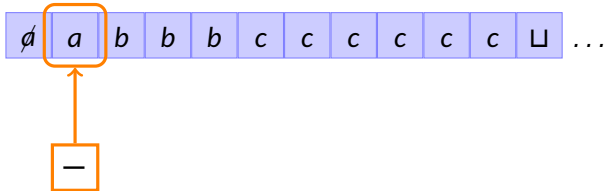
$M_3$  = “Para uma string de entrada  $w$ :

1. Varra a fita da esquerda para a direita para determinar se a entrada está no formato  $a^+b^+c^+$  e **rejeite** se não estiver.
2. Volte a cabeça da máquina para o início da fita
3. Risque um  $a$  e mova para a direita até encontrar um  $b$ . Faça um zigue-zague entre os  $b$ 's e os  $c$ 's riscando um  $b$  e um  $c$  até que acabem os  $b$ 's. Se todos os  $c$ 's forem riscados e ainda houver  $b$ 's, **rejeite**
4. Restaure todos os  $b$ 's e repita o estágio 3 se ainda houver  $a$ 's para serem riscados. Se todos os  $a$ 's forem riscados, verifique se todos os  $c$ 's também foram riscados. Se sim, **aceite**, caso contrário, **rejeite**”

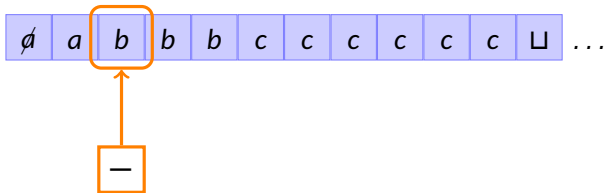
## Máquinas de Turing



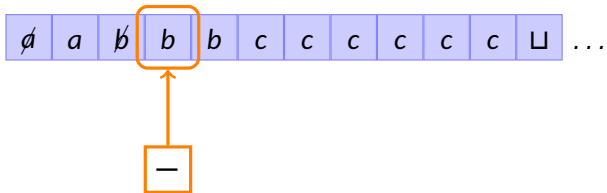
# Máquinas de Turing



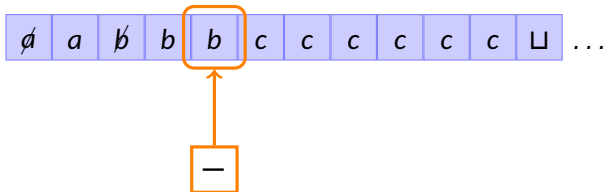
## Máquinas de Turing



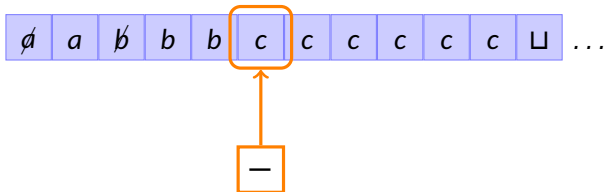
## Máquinas de Turing



## Máquinas de Turing

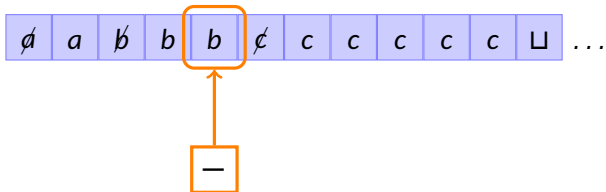


## Máquinas de Turing

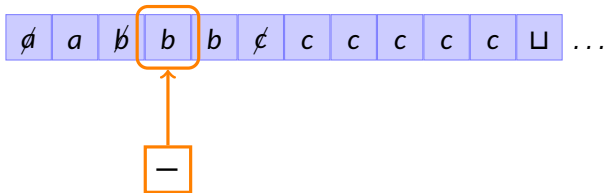




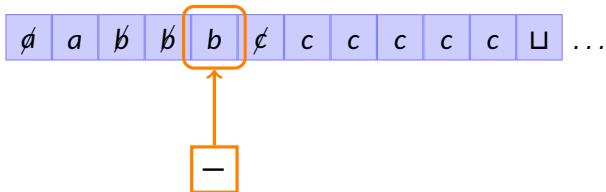
## Máquinas de Turing



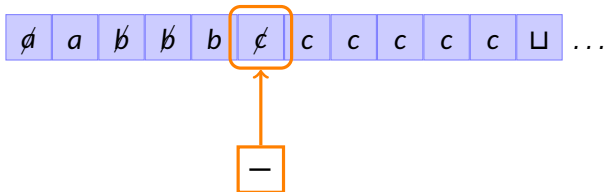
# Máquinas de Turing



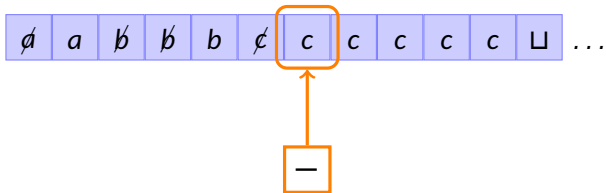
## Máquinas de Turing



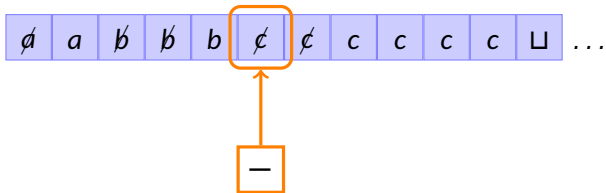
## Máquinas de Turing



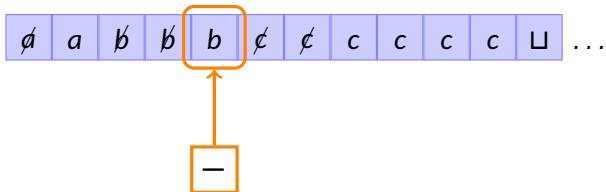
## Máquinas de Turing



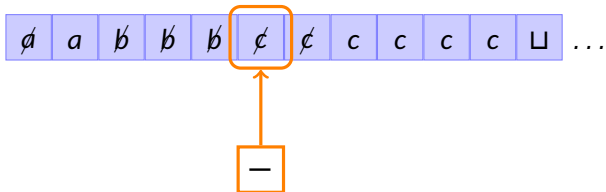
# Máquinas de Turing



# Máquinas de Turing

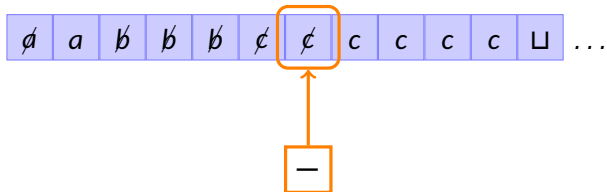


## Máquinas de Turing

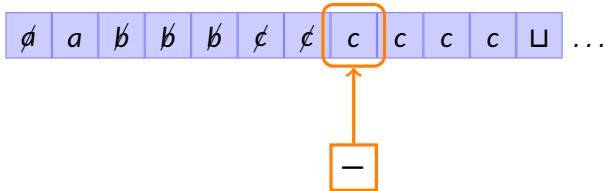




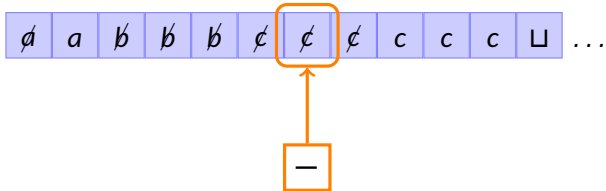
# Máquinas de Turing



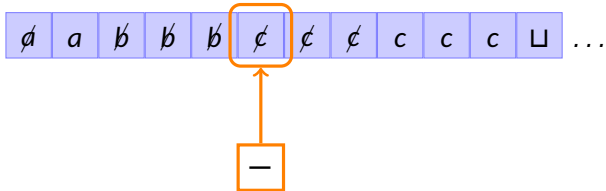
## Máquinas de Turing



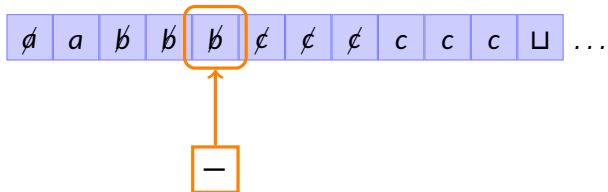
# Máquinas de Turing



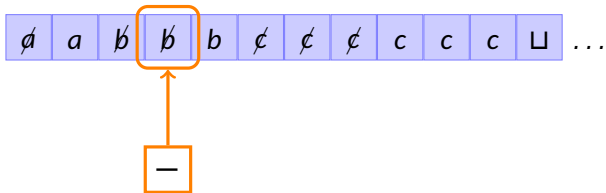
# Máquinas de Turing



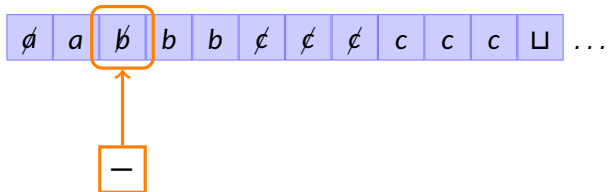
# Máquinas de Turing



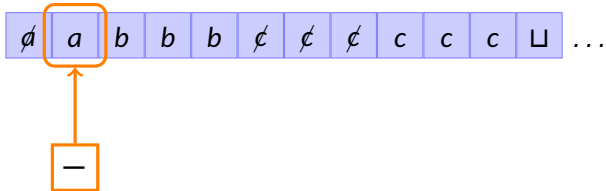
# Máquinas de Turing



# Máquinas de Turing

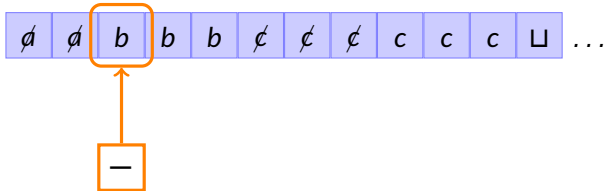


# Máquinas de Turing

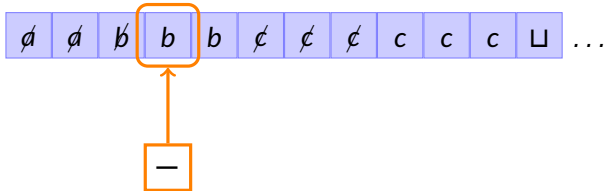




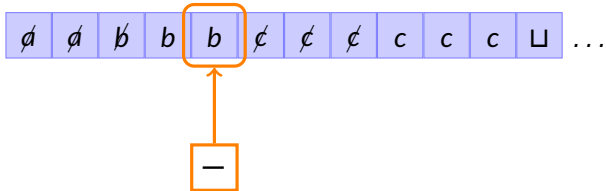
# Máquinas de Turing



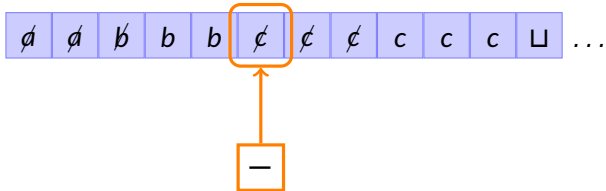
# Máquinas de Turing



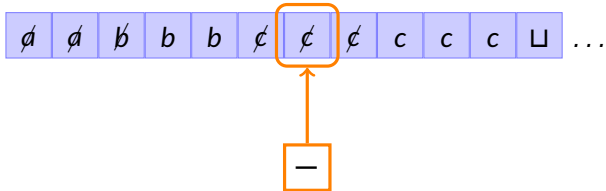
# Máquinas de Turing



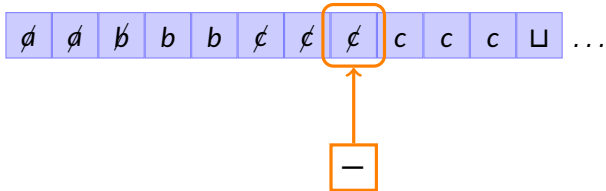
# Máquinas de Turing



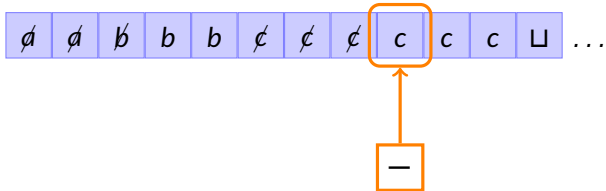
# Máquinas de Turing



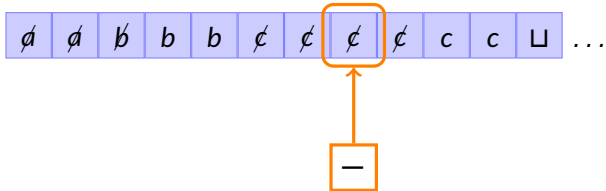
# Máquinas de Turing



# Máquinas de Turing

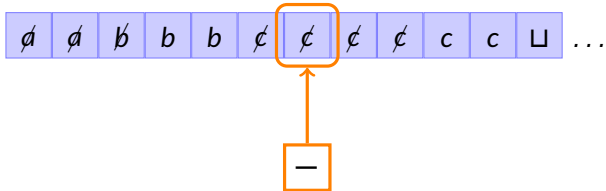


# Máquinas de Turing

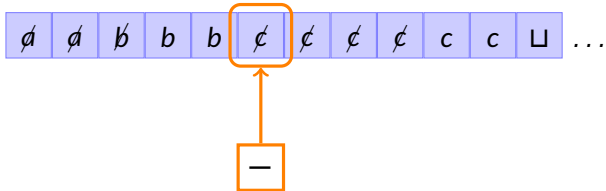




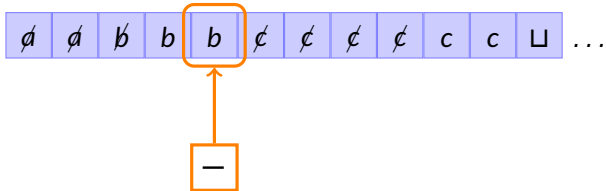
# Máquinas de Turing



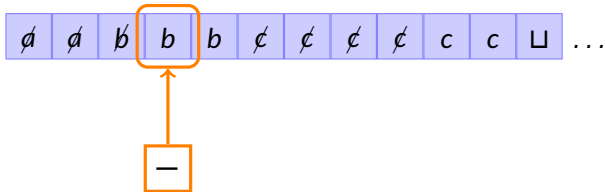
# Máquinas de Turing



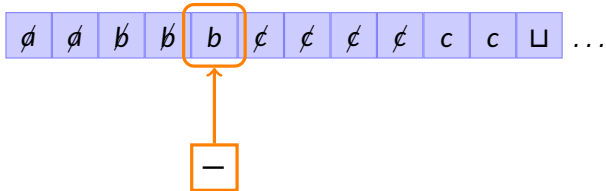
# Máquinas de Turing



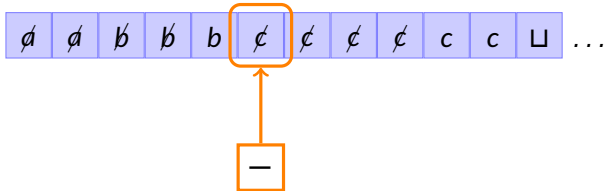
# Máquinas de Turing



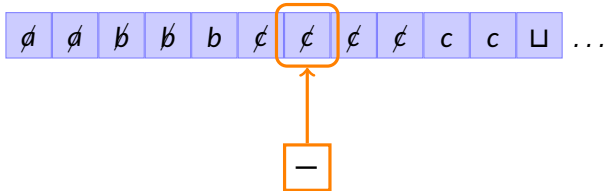
# Máquinas de Turing



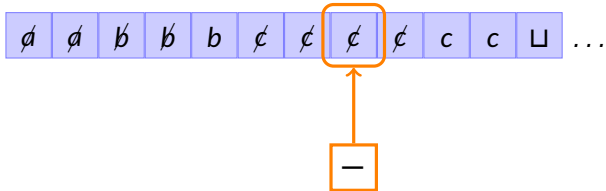
# Máquinas de Turing



# Máquinas de Turing

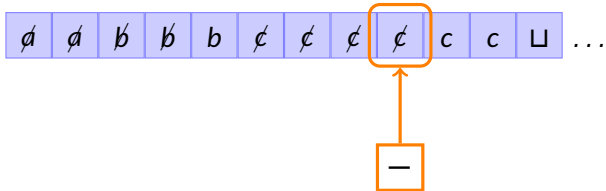


# Máquinas de Turing

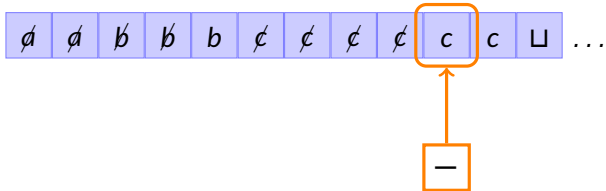




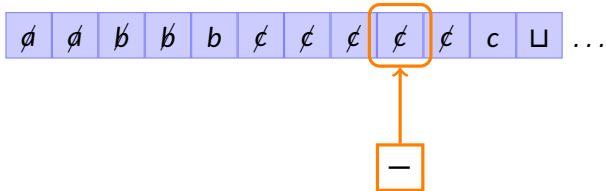
# Máquinas de Turing



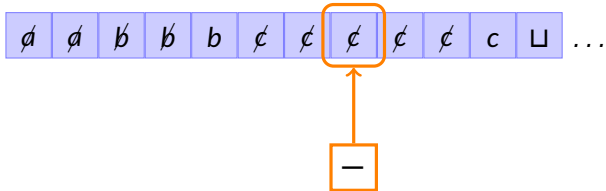
# Máquinas de Turing



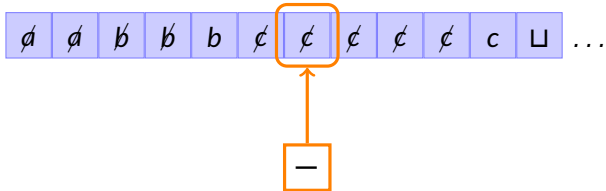
# Máquinas de Turing



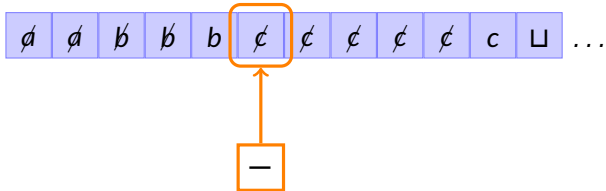
# Máquinas de Turing



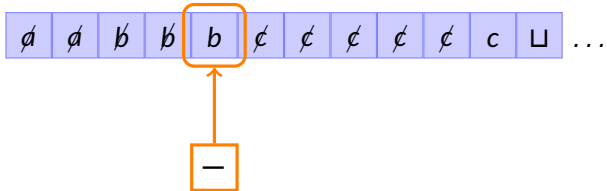
# Máquinas de Turing



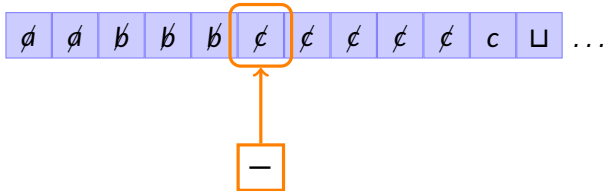
# Máquinas de Turing



# Máquinas de Turing

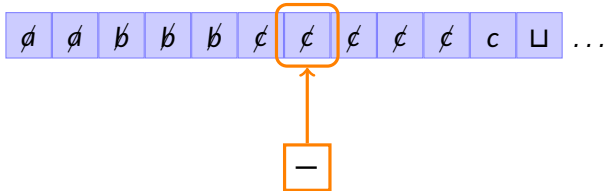


# Máquinas de Turing

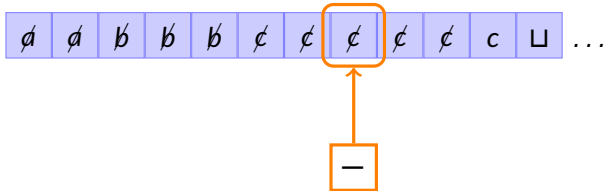




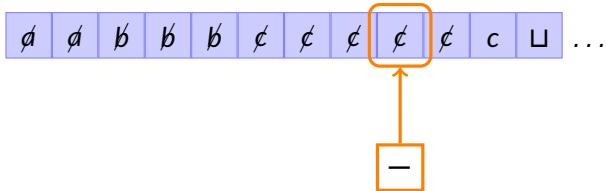
# Máquinas de Turing



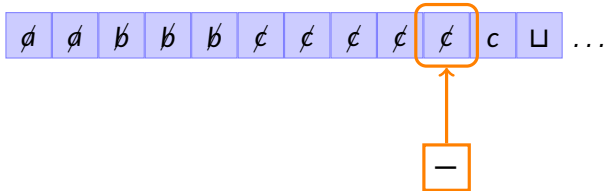
# Máquinas de Turing



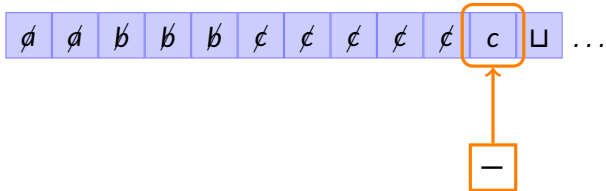
# Máquinas de Turing



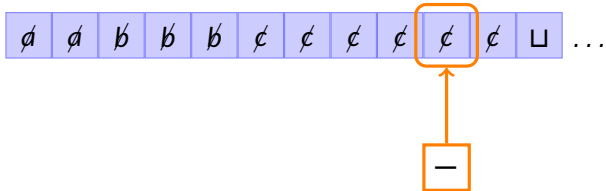
# Máquinas de Turing



# Máquinas de Turing



# Máquinas de Turing



# Máquinas de Turing

## Exercício

Faça uma **descrição de alto nível** de uma Máquina de Turing que decida a linguagem

$$E = \{ \#x_1\#x_2\#\cdots\#x_l \mid \text{cada } x_i \in \{0,1\}^* \text{ e } x_i \neq x_j \text{ para cada } i \neq j \}$$

# Máquinas de Turing

## Exercício

Faça uma **descrição de alto nível** de uma Máquina de Turing que decida a linguagem

$$E = \{ \#x_1\#x_2\#\cdots\#x_l \mid \text{cada } x_i \in \{0, 1\}^* \text{ e } x_i \neq x_j \text{ para cada } i \neq j \}$$

Em palavras: cada  $x_i$  é uma palavra de zeros e uns e todas as palavras devem ser diferentes entre si.

Exemplos:

- #0101#100#0
- #11#0001#10101#010
- #000000#111



## Máquinas de Turing

$M_4$  = “Para uma string de entrada  $w$ :

1. Coloque uma marca no símbolo mais à esquerda da fita. Se esse símbolo for vazio, **aceite**. Se esse símbolo for um #, continue para o próximo estágio, caso contrário, **rejeite**
2. Mova para a direita até o próximo # e marque-o. Se nenhum # for encontrado, apenas  $x_1$  foi encontrado, então **aceite**
3. Faça zigue-zague na fita, comparando as duas strings à direita dos #'s marcados. Se forem iguais, **rejeite**
4. Mova a marca mais à direita para o próximo # à direita. Se nenhum # for encontrado, mova a marca mais à esquerda para o próximo # à direita e mova a marca mais à direita para o # seguinte. Dessa vez, se nenhum # for encontrado para a marca à direita, todas as strings foram comparadas, então **aceite**
5. Vá para o estágio 3

# Máquinas de Turing

O objetivo aqui é comparar todas as strings entre si de duas em duas:

- #11#0001#10101#010
- #11#0001#10101#010
- #11#0001#10101#010
- #11#0001#10101#010
- #11#0001#10101#010
- #11#0001#10101#010

# Máquinas de Turing

## *Variantes da Máquina de Turing*

Considere uma variante da Máquina de Turing que, além de poder mover a cabeça de leitura e escrita para a direita e para a esquerda, também possa mantê-la na mesma posição.

Em outras palavras, a função de transição tem a forma:

$$\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{E, D, P\}$$

Essa máquina é mais poderosa que a Máquina de Turing tradicional? Isto é, ela consegue reconhecer linguagens que a máquina tradicional não consegue?

# Máquinas de Turing

## *Variantes da Máquina de Turing*

**Não.** Essa variante pode ser **simulada** pela Máquina de Turing tradicional. Para isso, convertamos cada transição na qual a cabeça de leitura fica parada em duas transições da Máquina de Turing tradicional, uma movendo a cabeça para a esquerda e outra para a direita.

Utilizaremos essa técnica para provar que outras variantes são equivalentes à Máquina de Turing original. Se a variante pode ser simulada pela original, ela não é mais poderosa que a original.

# Máquinas de Turing

## *Variantes da Máquina de Turing*

Considere uma **Máquina de Turing de múltiplas fitas**. Cada uma das fitas tem sua própria cabeça de leitura e escrita. Inicialmente, a entrada está na fita 1 e as demais fitas estão em branco. A função de transição é:

$$\delta : Q \times \Gamma^k \rightarrow Q \times \Gamma^k \times \{E, D, P\}^k$$

onde  $k$  é o número de fitas. A expressão:

$$\delta(q_i, a_1, \dots, a_k) = (q_j, b_1, \dots, b_k, E, D, \dots, E)$$

significa que se a máquina está no estado  $q_i$  e as cabeças de 1 a  $k$  leem os símbolos  $a_1$  a  $a_k$ , a máquina vai para o estado  $q_j$ , escreve os símbolos  $b_1$  a  $b_k$  e move as cabeças nas direções indicadas ou as mantém paradas.

# Máquinas de Turing

## *Variantes da Máquina de Turing*

Essa variante é mais poderosa que a Máquina de Turing original?

# Máquinas de Turing

## *Variantes da Máquina de Turing*

Essa variante é mais poderosa que a Máquina de Turing original?

### Teorema

Toda Máquina de Turing com múltiplas fitas tem uma Máquina de Turing de uma fita equivalente.

# Máquinas de Turing

## *Variantes da Máquina de Turing*

Para simular a Máquina de Turing de múltiplas fitas, precisamos:

- Simular o conteúdo das múltiplas fitas
- Simular as múltiplas cabeças de leitura e escrita

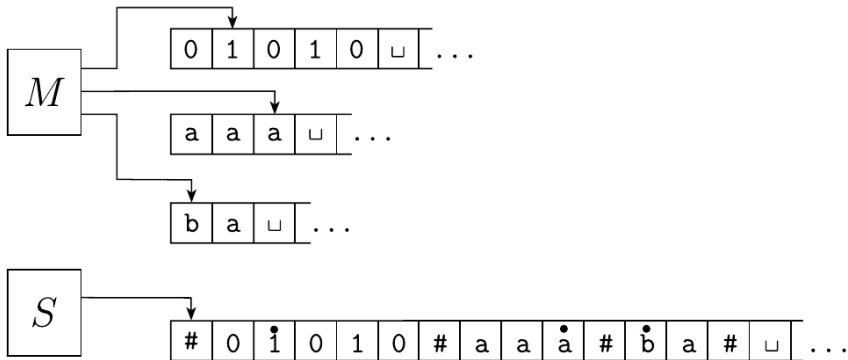
Podemos armazenar o conteúdo das múltiplas fitas na Máquina de Turing original utilizando um separador (por exemplo, #).

Para representar as múltiplas cabeças, adicionamos caracteres “marcados” ao alfabeto da fita. Esses caracteres indicam a posição de cada uma das cabeças virtuais.



# Máquinas de Turing

## *Variantes da Máquina de Turing*



# Máquinas de Turing

## *Variantes da Máquina de Turing*

$S =$  "Para uma string de entrada  $w = w_1, w_2, \dots, w_n$ :

1. Coloque a fita num formato que represente todas as  $k$  fitas de  $M$ :

$$\# \overset{\bullet}{w_1} w_2 \dots w_n \# \overset{\bullet}{\sqcup} \# \overset{\bullet}{\sqcup} \# \dots \#$$

2. Para simular cada movimento, escaneie a fita da esquerda para a direita para determinar os símbolos que estão sob as cabeças virtuais. Então, faça uma segunda passagem para atualizar as fitas de acordo com a função de transição de  $M$
3. Se, em algum momento,  $S$  mover uma das cabeças virtuais para cima de um  $\#$ , significa que  $M$  moveu a cabeça correspondente para uma porção não lida da fita. Então,  $S$  escreve um símbolo vazio nessa posição da fita e desloca todo o restante do conteúdo para a direita.

# Máquinas de Turing

## *Variantes da Máquina de Turing*

Uma **Máquina de Turing não-determinística** é capaz de estar em mais de uma configuração ao mesmo tempo. A função de transição tem a forma:

$$\delta : Q \times \Gamma \rightarrow \mathcal{P}(Q \times \Gamma \times \{E, D\})$$

# Máquinas de Turing

## *Variantes da Máquina de Turing*

Uma **Máquina de Turing não-determinística** é capaz de estar em mais de uma configuração ao mesmo tempo. A função de transição tem a forma:

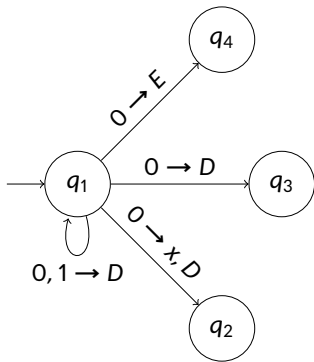
$$\delta : Q \times \Gamma \rightarrow \mathcal{P}(Q \times \Gamma \times \{E, D\})$$

$\mathcal{P}(a)$  representa o **conjunto potência**, isto é, o conjunto de todos os subconjuntos de  $a$ :

$$\mathcal{P}(\{x, y\}) = \{\{\}, \{x\}, \{y\}, \{x, y\}\}$$

Em outras palavras, nossa função de transição agora retorna um **conjunto** de próximos estados.

## Máquinas de Turing



# Máquinas de Turing

## *Variantes da Máquina de Turing*

Essa variante é mais poderosa que a Máquina de Turing original?

# Máquinas de Turing

## *Variantes da Máquina de Turing*

Essa variante é mais poderosa que a Máquina de Turing original?

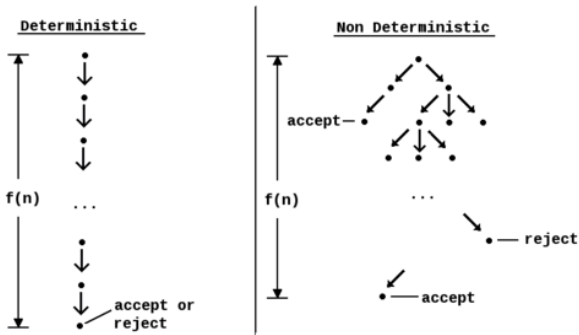
### Teorema

Toda Máquina de Turing não-determinística tem uma Máquina de Turing determinística equivalente.

# Máquinas de Turing

## *Variantes da Máquina de Turing*

Podemos imaginar as configurações possíveis da Máquina de Turing em uma árvore. Nossa Máquina de Turing determinística pode simular a Máquina de Turing não-determinística buscando nessa árvore por um estado aceita.

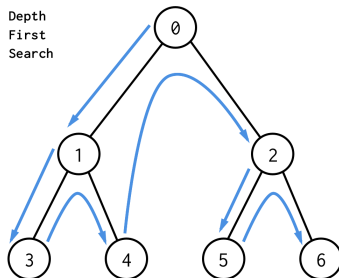
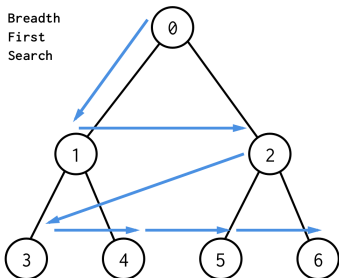




# Máquinas de Turing

## Variantes da Máquina de Turing

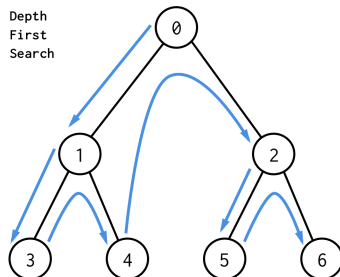
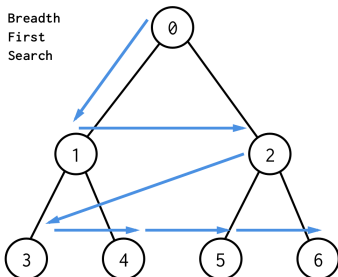
Busca em largura (BFS) ou busca em profundidade (DFS)?



# Máquinas de Turing

## *Variantes da Máquina de Turing*

Busca em largura (BFS) ou busca em profundidade (DFS)?

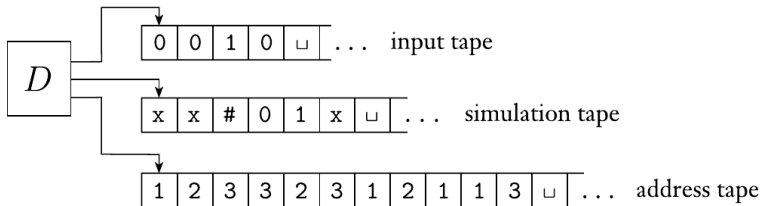


Com a busca em profundidade, corremos o risco de não explorar a árvore inteira.

# Máquinas de Turing

## *Variantes da Máquina de Turing*

Para simular a Máquina de Turing não-determinística, vamos utilizar uma Máquina de Turing com três fitas (equivalente à Máquina de Turing original).



# Máquinas de Turing

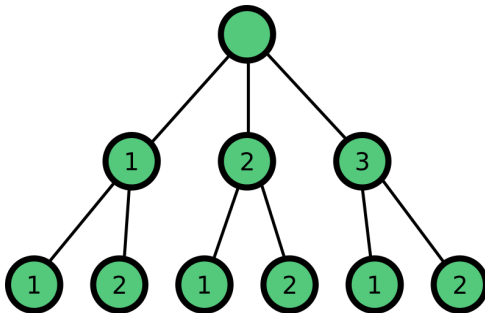
## *Variantes da Máquina de Turing*

- A primeira fita armazena a entrada da máquina e nunca é alterada
- A segunda fita armazena uma cópia da fita da máquina não-determinística em algum ponto da computação
- A terceira fita armazena a posição da máquina na árvore de possibilidades

# Máquinas de Turing

## *Variantes da Máquina de Turing*

Representação dos dados na fita 3:



# Máquinas de Turing

## *Variantes da Máquina de Turing*

Agora, estamos prontos para descrever a nossa simulação:

1. Inicialmente, a fita 1 contém a entrada  $w$  e as fitas 2 e 3 estão vazias
2. Copie a fita 1 para a fita 2 e inicialize a string na fita 3 para  $\epsilon$  (string vazia)

# Máquinas de Turing

## *Variantes da Máquina de Turing*

Agora, estamos prontos para descrever a nossa simulação:

3. Use a fita 2 para simular a máquina não-determinística com entrada  $w$  em um dos ramos da árvore de possíveis configurações. Antes de cada passo da máquina, consulte a fita 3 para saber qual escolha deve ser tomada dentre as possíveis. Se nenhum símbolo resta em 3, ou se essa escolha é inválida, aborte este ramo indo para o estágio 4. Também vá para o estágio 4 se uma configuração **rejeita** é encontrada. Se uma configuração **aceita** é encontrada, **aceite** a entrada
4. Substitua a string na fita 3 pela próxima string seguindo a ordem. Simule o próximo ramo na árvore de possibilidades indo para o estágio 25

# Máquinas de Turing

## *Variantes da Máquina de Turing*

Diversos outros modelos de computação foram propostos, alguns semelhantes e alguns muito diferentes das Máquinas de Turing.

Todos eles têm uma semelhança: **acesso irrestrito a memória ilimitada**

Todos os modelos com essa característica são equivalentes.



# Máquinas de Turing

## *Variantes da Máquina de Turing*

Um exemplo: linguagens de programação

- **Brainf\*ck**

```
+++++++[>++++[>+>+>+>+>+<<<<-]  
>+>+>->+ [<]<-]>>.>---.+++++.+.+  
+.>>.<-.<+.+.-----.->+.>+.
```

- **C**

```
#include <stdio.h>  
  
int main() {  
    printf("Hello world\n");  
    return 0;  
}
```