17) DESCREVENDO MÁQUINAS DE TURING

Como descrever MTs

cód. máquina
⇓ 001011001
010011001

assembly
⇓ ADD R1,R2,10
LD $45, R1

C/python/java
⇓ i=(i+2)*4;
if (i<5)
printf("oi");

algoritmo
se $S \cap T = \emptyset$
$S = S \cup \{a\}$

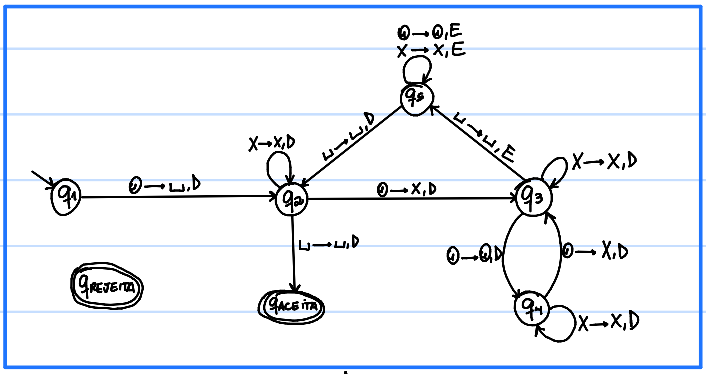
baixo nível
⇓ diagrama
7-upla

intermediário
⇓ detalhes de movimento da cabeça
como os dados são representados

alto nível
especifica o algoritmo
não detalha como a máquina
administra a fita

Como descrever MTs

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



⇓

M = "Sobre a cadeia de entrada w :

(1) Faça uma varredura da esquerda para a direita na fita, marcando um 0 não e outro sim.

(2) Se em (1) a fita tinha um único 0, aceite.

(3) Se em (1) a fita tinha mais de um 0 e o número de zeros era ímpar, rejeite.

(4) Retorne para a extremidade esquerda.

(5) Vá para (1)."

⇓

1. Enquanto houver mais de um 0 não marcado:

   Faça uma varredura marcando um 0 sim e outro não. Se não for possível, rejeite.

2. Se houver apenas um 0 não marcado, aceite.

3. Caso contrário, rejeite.

Mais exemplos

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

M_A = "Sobre a cadeia w de entrada:

(1) Faça uma varredura da esquerda para a direita para determinar se ela é membro de $a^+ b^+ c^+$ e rejeite se não for.

(2) Retorne a cabeça para a extremidade esquerda.

(3) Marque um a e faça uma varredura para a direita até que um b ocorra. Vá e volte entre b's e c's, marcando um de cada, até que todos os b's tenham terminado.
Se todos os c's foram marcados e algum b não foi, rejeite.

(4) Restaure os b's marcados e repita (3) se existir outro a para marcar. Se todos os a's foram marcados, verifique se todos os c's foram marcados.
Se sim, aceite. Caso contrário, rejeite."

M_A = "Sobre a entrada w:

(1) Rejeite se não estiver em $a^+ b^+ c^+$.

(2) Enquanto houver a's não marcados:
   Marque um a.
   Marque um c para cada b.
   Se todos os c's forem marcados e algum b não foi, rejeite.
   Desmarque os b's

(3) Se houver c não marcado, rejeite.
Caso contrário, aceite."

Mais exemplos

$B = \{ \#x_1 \#x_2 \# \dots \# x_k \; : \; x_i \in \{0,1\}^* \text{ e } x_i \neq x_j \; \forall \, i \neq j \}$

M_B = " Sobre a cadeia w de entrada:

(1) Marque o símbolo mais à esquerda na fita. Se era ⊔, aceite. Se era #, continue. Caso contrário, rejeite.

(2) Faça uma varredura procurando o próximo # e marque-o. Se não encontrou # antes de um ⊔, aceite.

(3) Fazendo zigue-zague, compare as duas cadeias à direita dos #'s marcados. Se forem iguais, rejeite.

(4) Mova a marca do # mais à direita para o próximo # à direita. Se nenhum # for encontrado antes de um ⊔, mova a marca mais à esquerda para o próximo # à sua direita e a marca mais à direita para o # depois desse. Dessa vez, se nenhum # estiver disponível, aceite.

(5) Vá para o passo (3). "

M_B = " Sobre a entrada w:

(1) Se houver apenas w_1, aceite.

(2) Para cada $i = 1$ até $k-1$

Para cada $j = i+1$ até k

Verifique se $w_i = w_j$.

Se for, rejeite.

(3) Aceite. "

Mais exemplos

$C = \{a^n b^n : n \geq 1\}$

M_C = "Sobre a cadeia w da entrada:

(1) Vá e volte entre os a's e b's, marcando um de cada.

(2) Se todos os a's forem marcados e os b's não forem ou se todos os b's forem marcados e os a's não, rejeite. Caso contrário, aceite."

Mais exemplos

$D = \{a^n b^n c^n : n \geq 1\}$

M_D = "Sobre a cadeia w de entrada:

(1) Use M_C para verificar se a's e b's estão na mesma quantidade. Rejeite se ela rejeitar.

(2) Use M_C para verificar se b's e c's estão na mesma quantidade. Rejeite se ela rejeitar.

(3) Aceite."

Tudo é cadeia

→ Cadeias podem representar qualquer objeto.

NÚMEROS

10 14

10 14

1010 1110

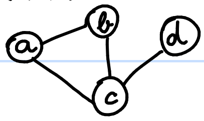
POLINÔMIOS

$5x^2+7y-9$

5x^2+7y-9

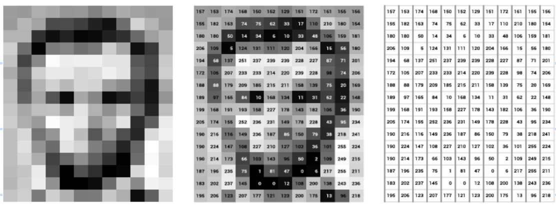
#5#x#2#+#7#y#1#-#9

GRAFOS



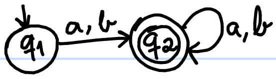
(a,b,c,d)((a,b),(a,c),(b,c),(c,d))

IMAGENS



((157,153,174,168,...,156)(156,....))

AFDs



$((q_1,q_2),(a,b),((q_1,a,q_2),(q_1,b,q_2),(q_2,a,q_2),(q_2,b,q_2)),q_1,(q_2))$

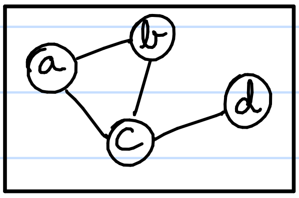
Representação

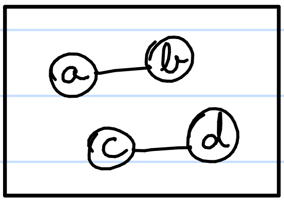
→ Máquinas de Turing recebem cadeias.
- Qualquer outro objeto deve ser convertido primeiro.
- E a máquina pode ser programada para decodificar a representação e interpretá-la como queremos.

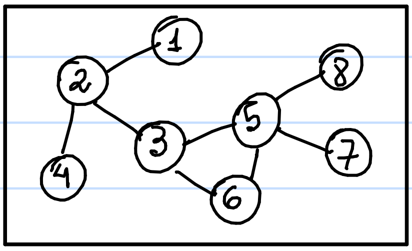
→ Descrições de alto nível não detalham a codificação.
- Se a entrada for w, assumiremos que já é uma cadeia.
- Se for ⟨A⟩, é um objeto A codificado em cadeia. Assumiremos que a máquina implicitamente testa se a codificação está apropriada.

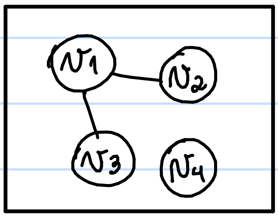
Exemplo

$E = \{ \langle G \rangle : G$ é um grafo conexo $\}$



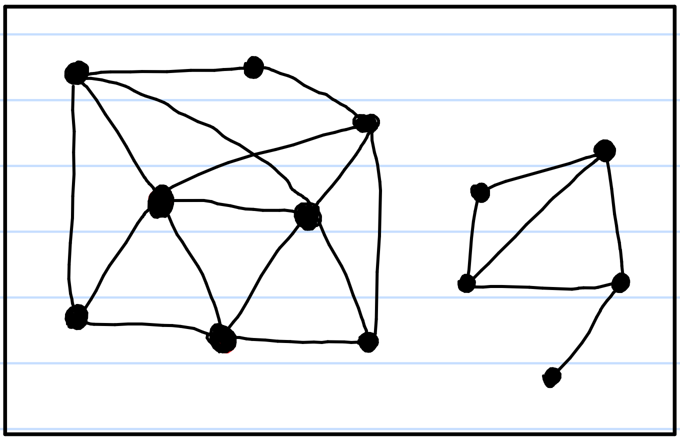


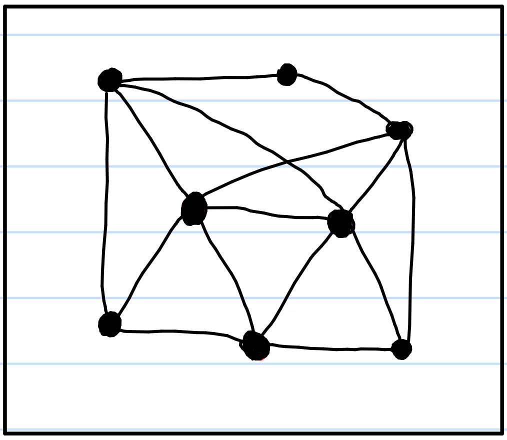




DEFINIÇÃO: Um grafo G é **conexo** se existe caminho entre todo par de vértices de G.

DEFINIÇÃO: Um **caminho** em um grafo G é uma sequência $(v_0, v_1, \dots, v_k)$ de vértices tal que $v_i v_{i+1}$ é aresta, para $0 \leq i < k$.





$E = \{ \langle G \rangle : G$ é um grafo conexo $\}$

M_E = " Sobre a entrada $\langle G \rangle$, onde G é um grafo:

(1) Selecione o primeiro vértice de G e marque-o.

(2) Repita até que nenhum novo vértice seja marcado:

Marque um vértice que tenha aresta para um vértice já marcado.

(3) Se todos os vértices foram marcados, aceite.

Caso contrário, rejeite.

18) A TESE CHURCH-TURING

Senta, que lá vem a história

→ A noção de algoritmo é antiga.
- Suficiente para mostrar o que é computável.
- Inútil para mostrar o que não é.

→ Um polinômio é uma soma de termos, que são produtos entre variáveis e coeficientes:
$$6x^3y^2z^2 + 3xy^2 - x^3 = 10$$
$$9x^4 + 3x^2 - 5x = 7$$

→ 23 problemas matemáticos para o século XX

→ 1900, Hilbert: Crie um processo que possa determinar em um número finito de passos se um polinômio tem raiz inteira.
(Resolubilidade de equação diofantina)

O que pode ser calculado?

→ 1936: Alonzo Church → cálculo-λ e Alan Turing → máquinas
- Provaram que são dispositivos equivalentes

→ Todo outro dispositivo já criado tem poder equivalente às máquinas de Turing.

> **A TESE CHURCH-TURING**
> "Tudo que possa ser calculado é computável."

→ Onde computável é o que pode ser feito com MTs.

Sobre o problema de Hilbert

→ 1900: Crie um processo que possa determinar em um número finito de passos se um polinômio tem raiz inteira.

→ 1936: $A = \{ \langle p \rangle : p$ é um polinômio com raiz inteira $\}$
A é Turing-decidível?

→ 1970: Não

Um problema mais simples

$B = \{ \langle p \rangle : p$ é polinômio sobre x com raiz inteira $\}$

M_B = " Sobre a entrada $\langle p \rangle$, onde p é polinômio em x:
(1) Para cada valor da sequência $0, 1, -1, 2, -2, 3, -3, \ldots$
Calcule p com esse valor em x
Se deu 0, aceite."

→ M_B reconhece B.

M_B = " Sobre a entrada $\langle p \rangle$, onde p é polinômio em x:
(1) Calcule p para cada valor inteiro em x
(2) Se algum faz $p = 0$, aceite.
Caso contrário, rejeite."

M'_B = " Sobre a entrada $\langle p \rangle$, onde p é polinômio em x:
(1) Seja k o nº de termos em p, C_{max} o coeficiente de maior valor absoluto e C_1 o coeficiente do termo de maior ordem.
(2) Para cada valor da sequência $-k\frac{C_{max}}{C_1}, \ldots, 0, \ldots, k\frac{C_{max}}{C_1}$:
Calcule p com esse valor em x
Se deu 0, aceite.
(3) Rejeite"

→ M'_B decide B

→ É impossível calcular tais limitantes para polinômios com mais de uma variável.