Um Inglês Esperto: TURING

22/01/2021

Autor: Pedro Luis Kantek Garcia Navarro- GAC

...a palavra algoritmo vem do nome do matemático persa do século IX Abu Ja'far Mohammed ibn Mûsâ al-Khowârizm que escreveu um manual em 825 dC chamado Kitab al jabr w'al-muqabala. O certo seria escrever algorismo, mas por analogia com aritmética ficou algoritmo.

Algoritmos já eram conhecidos antes disso. Um deles é o algoritmo de Euclides para encontrar o máximo divisor comum de 2 números. Chamemos ao primeiro (maior) de A, e ao segundo de B. (Se B for maior, inverta-os).

Algoritmo Euclides (acha o mdc)
// Recebe A e B e devolve o mdc
Repita
    Divida A por B e coloque o resto em C (se B³ A, inverta)
    se C diferente de zero, coloque B em A e C em B
    até C ser igual a zero
Devolva B

Exemplo:

Imaginemos 147 e 102. 147/102 = 1 e resto = 45. A nova dupla agora é 102, 45. Dividindo, 102/45 = 2 e o resto é 12. A nova dupla é 45,12. Dividindo 45/12 = 3 e o resto = 9. De novo a dupla é 12,9. Dividindo temos 1 e o resto = 3. Nova dupla é (9,3). Dividindo, dá 3 e o resto é zero. Logo o mdc é 3.

A definição precisa do que seja um algoritmo geral é de 1930, e a mais vigorosa descrição é devida a Turing, através do seu conceito de Máquina de Turing. Esta não é uma máquina real, sendo apenas matemática.

Em 37, Turing, respondendo ao 10º problema de Hilbert (formulado em 1900 e 1928), que dizia: *haverá algum procedimento mecânico geral que possa em princípio resolver todos os problemas da matemática*? Propôs sua máquina. Ela é composta por um número finito de estados (ainda que possa ser muito grande). Deve tratar um input infinito. Tem um espaço exterior de armazenagem também infinito. Turing imaginou o espaço exterior para dados e armazenamento como

sendo uma fita, que pode mover-se para frente e para trás. Além disso, a máquina pode ler e escrever nesta fita. A fita está composta por quadrados (ou seja, o nosso ambiente é discreto). Apenas 2 símbolos podem ser escritos nos quadrados da fita, digamos 0 (quadrado em branco) e 1 (quadrado preenchido), ou vice-versa.

Isto posto, lembremos que:

1. os estados internos do aparelho são finitos em número;

2. o comportamento da máquina é totalmente determinado pelo seu estado interior e pelo input.

Ou seja, dado o estado inicial, e um símbolo de input, o aparelho deve ser determinista. Logo ele:

a. muda seu estado interno para outro, e/ou

b. muda o símbolo que foi lido pelo mesmo ou por outro, e/ou

c. movimenta-se para a direita ou para a esquerda, 1 quadrado, e/ou

d. decide se continua ou interrompe e pára.

Definição de uma máquina de Turing

Para especificarmos nossa máquina de Turing, teríamos que escrever algo do

Estado em que está	Sinal lido (coluna 2)	faz o que	Vai para o estado	Número que é escrito na saída (sobre col.2)	Anda para onde
0	0	→	0	0	D
0	1	→	13	1	E
1	0	→	65	1	D
1	1	→	1	0	D
2	0	→	0	1	D.PARE
2	1	→	66	1	E

Em lugar de usar 0, 1, 2... para rotular os estados internos, seria melhor usar símbolos formados por 0 e 1s. Usaremos para tanto, o sistema de numeração binário:

$0 \rightarrow 0$,
$1 \rightarrow 1$,
$2 \rightarrow 10$,
$3 \rightarrow 11$,
$4 \rightarrow 100$, etc. etc.

ing:

Estado atual	Sinal lido	faz o que	Vai para estado	Escreve	Anda para onde
0	0	→	0	0	D
0	1	→	1101	1	E
1	0	→	1000001	1	D
1	1	→	1	0	D
10	0	→	0	1	D.PARE
10	1	→	1000010	1	E

Como pretendemos que os números possam ser fornecidos como input, precisamos de um sistema para escrevê-los na fita de entrada. Começaremos escrevendo os números em um sistema *unário* (1 é 1, 11 é 2, 111 é 3, 1111 é 4, e assim por diante). O símbolo 0 passa a ser o separador.

Soma 1 em unário

Vamos escrever uma máquina de Turing para somar 1 em um número unário.

Estado atual	Sinal lido	faz o que	Vai para estado	Escreve	Anda para onde
0	0	→	0	0	D
0	1	→	1	1	D
1	0	→	0	1	pare
1	1	→	1	1	D

Multiplica por 2 em unário

Estado atual	Sinal lido	faz o que	Vai para estado	Escreve	Anda para onde
0	0	→	0	0	D
0	1	→	1	0	D
1	0	→	10	1	E
1	1	→	1	1	D
10	0	→	11	0	D
10	1	→	100	0	D
11	0	→	0	1	pare
11	1	→	11	1	D
100	0	→	101	1	E
100	1	→	100	1	D
101	0	→	10	1	E
101	1	→	101	1	E

por 2:

Estado atual	Sinal lido	faz o que	Vai para estado	Escreve	Anda para onde
0	0	→	0	0	d
0	1	→	1	1	e
1	0	→	10	1	d
1	1	→	1	1	e
10	0	→	1010	0	d
10	1	→	11	0	d
11	0	→	100	0	d
11	1	→	11	1	d
100	0	→	100	0	d
100	1	→	101	0	d
101	0	→	111	0	e
101	1	→	110	1	e
110	0	→	110	0	e
110	1	→	1	1	e
111	0	→	111	0	e
111	1	→	1000	1	e
1000	0	→	1001	0	e
1000	1	→	1000	1	e
1001	0	→	10	0	d
1001	1	→	1	1	e
1010	0	→	0	0	pare
1010	1	→	1010	1	d

itmo mdc de

Sistema contraído.

O sistema unário é ineficiente, logo precisamos usar um sistema binário. Só que não podemos fazê-lo diretamente, já que se assim for, não saberemos quais as marcas de fim e quais os zeros para valer. Para resolver esta questão, vamos fazer uma *contração*. Qualquer grupo de zeros e uns não é lido diretamente como binário, e sim cada quantidade finita de uns é substituida por 2, 3... na segunda seqüência:

Ou seja, na segunda seqüência, cada número representa o número de UNS entre ZEROS.

Antes de prosseguir, vamos entender bem esta passagem. Seja a seqüência:

010.000101101010110100011101010111100110 = 1

010.0.001011010101101000111010101111100110 = 0

0100.0.010110101011010001110101011110011 0 = 0

01000.010.110101011010001110101011110011 0 = 1

0100001.0110.101011010001110101011110011 0 = 2

0100001011.010.101101000111010101111001 10 = 1

010000101101.010.110100011101010111100110 = 1

01000010110101.0110.100011101010111100110 = 2

010000101101010 11.010.001110101011110011 0 = 1

010000101101010 11010.0.011101010111100110 = 0

010000101101010 110100.0.11101010111100110 = 0

010000101101010 110100.0111.01010111100110 = 3

010000101101010 1101000111.010.10111100110 = 1

010000101101010 11010001 11.010.111100110 = 1

010000101101010 11010001 11010.011110.0110 = 4

0100001011010101101000111010101111 0.0.110 = 0

0100001011010101101000111010101111 0.0110. = 2

Note a diferença:

01110111 dá 33; 011100111 dá 303

A seqüência acima ficou sendo: 1 0 0 1 2 1 1 2 1 0 0 3 1 1 4 0 2

Note-se que agora podemos designar aos números códigos de acordo com o que quisermos fazer. Assim, por exemplo, o número 2 pode passar a ser uma vírgula.,. o 3 pode ser menos, o 4 mais, o 5 vezes, ...

A linha original seria lida como:

9 , 3, 4, instrução-3 3 instrução-4 0,

Para quem está perdido,

acompanhe: 1001 (09), 2 (vírgula), 11 (3), 2 (vírgula), 100 (quatro), 3 (Instrução3), 11 (3), 4 (instrução4), 0 (0), 2(vírgula).

Este procedimento nos permite terminar um número, colocando uma vírgula ao final.

Vamos fazer o procedimento contrário, o da expansão:

Seja escrever a seqüência de números inteiros: 5, 13, 0, 1, 1, 4. Em notação binária, tais números ficam: 101, 1101, 0, 1, 1, 100, que expandido na fita, dará:

...0000...1001011010100101100110101101011010001 10..000...

Para realizar esta codificação diretamente, usemos a equivalência:

$0 \rightarrow 0$
$1 \rightarrow 10$
$, \rightarrow 110$

e acrescentando zeros à vontade à esquerda e à direita, fica:

...000...10 0 10 110 10 10 0 10 110 0 110 10 110 10 110 10 0 0 110...0...

zeros 1 0 1 , 1 1 0 1 , 0 , 1 , 1 , 1 0 0 , zeros

Chamamos a esta última notação de notação binária desdobrada.

Uma observação final, é de que os zeros à esquerda são dispensáveis. Assim, e por via de conseqüência, também o zero isolado é dispensável, podendo-se escrever apenas, significando estas duas vírgulas que há aí um zero entre elas.

Codificando isto da maneira como foi vista, a seqüência ficaria 101,1101,1,1,100 e desdobrada ficaria

...0000...10010110101001011011010110101101000110...000...

note-se que sumiu um zero no meio (o local correto está marcado com um _).

...0000...100101101010010110_11010110101101000110...000...

Examinemos agora uma Máquina de Turing para rodar o algoritmo de Euclides: Por exemplo, para

110,1000, que por expansão é codificado como:

...000...10 10 0 110 10 0 0 0 110...000...

Mas usando as representações vistas acima, o par ficaria 6,.8 em binário 110,1000, que por expansão é codificado como:

...000...10 10 0 110 10 0 0 0 110...000...

Para este par específico não há muito ganho, mas supondo que os números a representar fossem: 1583169,8610 em notação binária dariam 110000010100001000001,10000110100010, e na fita iriam como:

...000...10100000010010000010000001011010000010100100000100110...000... ou

...000... 10 10 0 0 0 0 0 10 0 10 0 0 0 0 10 0 0 0 0 0 10 110 10 0 0 0 0 10 10 0 10 0 0 0 10 0 110 ...000... ou

Na notação unária estes 2 números precisariam 1.6Mcaracteres (umas 800 páginas de livro).

Adiciona 1 em notação expandida

Naturalmente, a MT poderia pegar estes números contraídos e desdobrá-los em

representação unária antes de operar. Mas nada impede que façamos uma MT que já trabalhe direto com números expandidos. Fazê-la para o algoritmo de Euclides seria muito complicado, então vamos a um exemplo mais simples. Seja

Estado atual	Sinal lido	faz o que	Vai para estado	Escreve	Anda para onde
0	0	→	0	0	d
0	1	→	1	1	d
1	0	→	0	0	d
1	1	→	10	1	d
10	0	→	11	0	e
10	1	→	10	1	d
11	0	→	0	1	pare
11	1	→	100	0	e
100	0	→	101	1	e
100	1	→	101	1	e
101	0	→	110	0	d
101	1	→	10	1	d
110	0	→	...	...	...
110	1	→	111	1	d
111	0	→	11	1	d
111	1	→	111	0	d

Para testar, use o número 167, cuja representação binária é 10100111 e expandido é ...0000... 10 0 10 0 0 10 10 10 110 ...0000... Supondo 167+1=168, fica (em binário normal).

10100111 + 1 = 10101000. 168 expandido é: ...0000... 10 0 10 0 10 0 0 0 110 ...00000...

Multiplica por 2 em notação expandida

Uma maquininha simples, onde paradoxalmente trabalhar com números expandidos é mais fácil do que com números unários é a máquina que multiplica um número por 2.

0	0	→	0	0	d
0	1	→	0	0	d

Tese de Church-Turing

Tendo estudado as máquinas de Turing podemos afirmar serem elas capazes de realizar qualquer computação (os adjetivos algoritmável, computável, recursivo e efetivo são equivalentes). Turing, gastou um bocado de tutano provando isso. Veio em seu apoio a idéia de Alonso Church, lógico americano, que com a ajuda de Kleene apresentou um esquema para resolver o 10º problema de Hilber chamado l -cálculo. Outra proposta foi a de Emil Post, lógico polonês-americano. Viu-se logo que todos estes esquemas eram equivalentes. Isto deu origem à tese de Church-Turing, de que a máquina de Turing define o que matematicamente

entendemos por um procedimento algorítmico. (A tese diz que "*as funções recursivas enumeráveis são únicas*".)

Máquina Universal de Turing

O princípio é simples. Teremos uma máquina U, que antes de mais nada lerá as informações na fita de entrada para que ela passe a se comportar como a máquina T. Depois disso vêm os dados de input que seriam processados pela máquina T, mas que serão pela U, operando como se fosse a T.

Para ver isso funcionando, precisamos numerar as máquinas de Turing. É preciso um pouco de esperteza aqui.

Primeiro, convertamos D, E, PARE, ® e vírgula como os números 2, 3, 4, 5 e 6. Ou nas nossas contrações como 110, 1110, 11110, 111110 e 1111110. Zero será 0, e 1 será 10, como já vimos .

As colunas 4 e 5 nas nossas tabelas não precisam separação, já que o conteúdo da coluna 5 é apenas 1 ou zero e sempre ocupa 1 caractere.

Podemos nos dar ao luxo de não codificar nenhuma seta, nem nada do que as antecede (colunas 1, 2 e 3) desde que organizemos as ordens em rigorosa ordem crescente (e tomando cuidado para preencher ordens que no modo tradicional não eram escritas por nunca ocorrerem na máquina).

Fazendo tudo isso, e escrevendo toda a programação da máquina que soma 1 a um número unário, ela fica:

10101101101001011010100111010010110101111010000111010010101110100010111010100011010010110110101010101101010101101010100.

Convertendo este número binário para decimal chegamos a:

450813704461563958982113775643437908

O que significa que a máquina que soma 1 em unário é a 450. 813. 704. 461. 563. 958. 982. 113. 775. 643. 437.908ª máquina de Turing.

Naturalmente, mudando-se (otimizando-se) alguma coisa na programação, esta máquina muda de número.

A máquina que soma 1 em unário é a 177.642ª máquina. A que multiplica 2 em binário expandido é 10.389.728.107ª máquina e a que multiplica por 2 um número unário é a 1. 492. 923. 420. 919. 872. 026. 917. 547. 669 ª máquina.

Dentro desta perspectiva qual seria a máquina T_0?

0	0	→	0	0	d
0	1	→	0	0	d

Movimenta-se para a direita, apagando tudo o que encontra, sem nunca parar e sem nunca voltar.
E T_1?

0	0	→	0	0	d
0	1	→	0	0	d

Faz a mesma coisa que T_0, só que pula 1 para trás depois de apagar cada marca da fita.
E T_2?

0	0	→	0	0	d
0	1	→	0	0	d

Se movimenta interminavelmente para a direita, mas deixa tudo como está Nem a 0, nem a 1 nem a 2 são máquinas de Turing, já que não param nunca.
T_3

0	0	→	0	0	d
0	1	→	0	0	pare

Esta é a primeira máquina. Ela pára depois de mudar o 1 mais a esquerda em zero.

E assim vai. A T_7, é *não corretamente especificada*, já que existe uma seqüência de 5 uns, e ela emperra ao processar tal seqüência.

Dizendo que quando a n-ésima máquina de Turing atua sobre o número m produz o número p, podemos escrever $T_n(m)=p$. Podemos pensar nesta relação como sendo uma função de 2 parâmetros (n e m) que leva a p. Esta função, pelo que vimos é totalmente algorítmica. Logo, ela pode ser realizada por uma máquina de Turing, a quem chamaremos de U. Então, fornecendo o par (n,m) a U, obtemos como resposta p. Vamos escrever n e m em uma fita, separando-os pela seqüência (nova) 111110.

Por exemplo, para n=11 e m=6, a fita ficaria:

...000...1011 111110 110 ...1000...

Podemos escrever $U(n,m) = T_n(m)$.

A máquina U quando alimentada primeiro com o número n, passa a se comportar como a T_n.

Como U é uma máquina de Turing, ela é uma $T_{\text{alguma-coisa}}$. Quanto é essa alguma coisa?

É mais ou menos 7.24×10^{1688} , ou seja o número 724 seguido de 1686 zeros.

Todos os computadores modernos, desde um humilde Z80 até um Cray multivetorial são máquinas de Turing.

REFERÊNCIA BIBLIOGRÁFICA

PENROSE, Roger. **A mente nova do rei**: computadores, mentes e as leis da física. 2. ed. Rio de Janeiro: Campus, 1993.