

Algoritmos como receitas

Um algoritmo é uma “receita” para se resolver um determinado problema. Não sendo nosso objetivo aqui analisar a noção de algoritmo sob um ponto de vista matemático, formal, ficaremos com essa caracterização, à qual vamos juntar uma série de qualificações e propriedades, mais adiante. Só seria necessário precisar com mais rigor a noção de algoritmo se desejássemos *provar matematicamente* algumas de suas propriedades, o que não é o caso.

O termo receita está sendo empregado com o sentido de “plano de execução”, ou “lista de atividades que leva ao fim desejado”. Isto é, um algoritmo é uma lista de atividades que, se executadas coerentemente, conforme estipulado na receita, leva sempre à resposta desejada.

Um primeiro exemplo

Um exemplo seria ilustrativo, nesse ponto. Primeiramente, devemos enunciar o problema, ou objetivo a alcançar. Vamos supor que o problema é simplesmente “preparar bifes à milaneza”.

Aqui está uma -- entre muitas -- lista de atividades que poderíamos usar para resolver este problema, ou para alcançar o objetivo traçado:

1. Limpar a peça de carne
2. Fatiar a carne em bifes
3. Colocar farinha de rosca em um prato
4. Juntar 2 ovos e mexer
5. Repetir, para cada bife
 - 5.1) passar o bife na mistura de farinha, nos 2 lados
 - 5.2) levar bife à frigideira até dourar
 - 5.3) aguardar dourar, virando ambas as faces
 - 5.4) retirar bife e colocar sobre papel toalha até secar
 - 5.5) retirar do papel toalha e juntar numa travessa
6. Decorar a travessa com folhas de alface
7. Servir

Temos uma seqüência de passos simples. Qualquer pessoa é capaz de entender essa seqüência e deve ser capaz de executá-la, dados os ingredientes e os meios necessários (fogão, panelas, etc).

A receita é composta por tarefas bem simples, tais como “Limpar a peça de carne”, ou “retirar do papel toalha e juntar numa travessa”. A intenção é que se comece executando a tarefa 1, passando à tarefa 2, e assim sucessivamente, segundo a ordem da listagem de tarefas.

A tarefa 5, entretanto, é de natureza diferente. Ela não diz o que fazer com os bifes, diretamente, como as anteriores. Ela estipula algo sobre a própria seqüência de tarefas, ou seja, determina que devemos *repetir* as tarefas 5.1 a 5.5 para cada um dos bifes. Assim, se estivermos lidando com 6 bifes, teríamos na verdade $6 \times 5 = 30$ passos aqui.

As tarefas 6 e 7 são usuais. Embora nada seja dito, fica subentendido que após executada a tarefa 7, a receita termina e o processo todo deve parar.

Isso, essencialmente, é um algoritmo. Nada mais que uma receita para se atingir um objetivo.

Partes básicas de um algoritmo

Da receita, ou algoritmo anterior, podemos identificar alguns itens que sempre vão ocorrer, em outros algoritmos:

- Os *dados de entrada*: são os itens que devem ser supridos pelo ambiente externo, cada vez que o algoritmo executa. Sem os dados de entrada, o algoritmo não consegue executar. No caso do exemplo, são os ingredientes, tais como carne, farinha e ovos. Note que esses itens são “consumidos” a cada execução.
- Itens *de apoio*, ou *de suporte*: são os itens onde são executadas as tarefas. No caso do exemplo: o fogão, as panelas, a frigideira, etc. Esses itens não são “consumidos” a cada execução. Poderiam ser os mesmos a serem usados em várias execuções do mesmo algoritmo.
- Os *dados de saída*: são os itens produzidos pela execução do algoritmo, isto é, são realmente o objetivo a ser alcançado em cada execução do algoritmo. No caso do exemplo, são os bifes à milaneza.
- O *próprio algoritmo*: isto é, a própria receita. Ela é imprescindível para guiar todo o processo de gerar os dados de saída a partir dos dados de entrada.

Logo mais adiante descreveremos uma analogia entre esse exemplo simples e programas de computador, onde esses atores todos assumirão papéis específicos no processo de se automatizar a resolução de problemas com o auxílio dessas máquinas. A transparência 6 adianta essa analogia. Inicialmente, somos expostos a um problema que desejamos resolver de forma automática, com o auxílio do computador.

O próximo passo é importante: precisamos gerar uma idéia de como proceder para resolver o problema com o auxílio da máquina. É aqui que começa a nascer um algoritmo para o problema. Esse é um processo criativo, sem regras definidas de como se poderia proceder sistematicamente. Dependerá da sua experiência e criatividade inventar um bom algoritmo para o problema. Nestas notas introdutórias não estaremos preocupados em, criado o algoritmo, analisar sua complexidade de tempo e de espaço de memória, isto é, determinar analiticamente quão rápido ele será ao executar, ou quanto de memória vai necessitar. Basta que criemos um algoritmo que corretamente resolva o problema para todas as instâncias de dados de entrada. Mais tarde, voltaremos a essas questões de correção e complexidade de algoritmos, com alguns detalhes e mais exemplos.

Uma vez obtido o algoritmo, devemos transcrever sua receita em uma linguagem de programação que dominemos. O resultado será o texto do algoritmo armazenado em um arquivo no computador. Este arquivo é então convertido, de forma automática – como veremos – para outro arquivo contendo uma descrição do algoritmo numa forma que o computador consegue interpretar e executar.

Usando o hardware do computador como suporte, e fornecendo os dados de entrada necessários, a máquina segue as instruções do algoritmo armazenado no arquivo convertido e computa as respostas desejadas, produzindo a saída esperada.

Uma breve história do nome

Da antiguidade, cerca de 300-400BC, já se tem notícias de algoritmos sendo desenvolvidos, embora, obviamente, não explicitamente tratados com tal. Em particular, Euclides, em seus famosos tratados, já propunha uma receita, ou um algoritmo, para se resolver o problema de calcular o máximo divisor comum entre dois inteiros positivos.

No entanto, a palavra algoritmo deriva do nome de um matemático persa, que viveu no século IX, cerca de 850 DC. Conhecido como Mohammed al-*Khowârizmî*, ele teve grande influência na solução de problemas algébricos importantes da época. Em particular, sistematizou os métodos de multiplicação de inteiros, na base decimal. Em outras palavras, criou e sedimentou um algoritmo para resolver esse problema.

Traduções de seu nome para o Latim terminaram por gerar a palavra *algorismus* e, daí para a época moderna, surgiu a grafia de *algorithm*, *algoritmo*, etc.

Outro exemplo

Vamos considerar o mesmo problema tratado por Euclides, que pode ser enunciado assim:

Dados dois números inteiros positivos, M e N , obter o máximo divisor comum entre eles.

Não há dúvidas sobre o que seja o máximo divisor comum, nem que este sempre exista, dados dois inteiros positivos como entrada.

A questão é: qual seria uma receita, ou algoritmo, capaz de sempre obter esse máximo divisor comum corretamente?

Nesse ponto entra-se num processo de raciocínio criativo, de tentativa e erros, até que, eventualmente, consigamos criar uma receita satisfatória para resolver o problema.

Avançando sobre esse processo, uma possibilidade seria a receita ou algoritmo abaixo, que nada mais é que a receita básica aprendida no colégio:

1. Se $M = N$, então o MDC é M (ou N); páre.
2. Caso a):
 - se $(M > N)$ então substitua M por $(M-N)$ e repita a partir do passo 1.
3. Caso b):
 - se $(N > M)$ então substitua N por $(N-M)$ e repita a partir do passo 1.

É uma receita simples, de 3 passos apenas. O primeiro passo contém um *teste*. Se M for igual a N , então já temos o MDC desejado: obviamente, ele é igual a M (ou a N). Emitimos essa informação como *dado de saída* e o algoritmo termina.

Se M não for igual a N então só restam duas possibilidades: $M > N$, ou $N > M$. Os dois casos são tratados de forma similar nos passos 2 e 3, respectivamente. Quando $M > N$ – veja o passo 2 – sabemos que o MDC entre M e N é o mesmo que entre $(M-N)$ e N . Nesse caso, o

Algoritmos: o que são?

passo 2 pede para que substituamos *o valor representado por M* pelo *novo valor* dado por $(M-N)$. E devemos voltar a executar o passo 1, com esses novos valores. Quando $N > M$, a situação é similar.

Devemos ressaltar um conceito muito importante aqui: a “variável” M é tratada como se fosse um local onde podemos armazenar valores de números inteiros. O mesmo se dá para a outra “variável”, N. A idéia é essa mesma. Os nomes M e N nada mais são que locais onde se podem armazenar quaisquer números inteiros. Podemos consultar esses valores a qualquer instante. Podemos usar esses valores em computações aritméticas. E podemos armazenar outros valores inteiros – que são resultados de computações ou não – nesses locais. É como se tivéssemos duas caixas de papelão, cada uma contendo uma folha de papel em seus interior, e designadas como M e N, nas respectivas tampas. Podemos abrir a caixa M e inspecionar o valor escrito na folha contida nessa caixa, usando ou não esse valor em cálculos futuros. Observe que cada caixa conterá apenas um número. Podemos também apagar o valor escrito na folha e escrever um outro valor em seu lugar. Mais tarde, veremos que o comportamento de um computador, ao executar um programa, muito se assemelha a esta alegoria de “caixas com valores dentro”. É importante reforçar que podemos *substituir* o valor escrito na folha por um outro de mesmo tipo, por exemplo, substituir um número por outro número. Essa operação é *destrutiva*, ou seja, o valor antigo é completa e irremediavelmente perdido.

Em princípio, ao inventar algoritmos, podemos usar quantas “variáveis” distintas, ou caixas com valores, quisermos. Em geral, usamos várias delas como “espaço de rascunho”, onde podemos armazenar valores intermediários da computação.

Criado o algoritmo, é sempre uma boa idéia testarmos seu funcionamento em casos simples para nos convenceremos de que o algoritmo vai funcionar sempre a contento. Poderíamos também tentar uma demonstração matemática de que o algoritmo sempre vai funcionar. Mas, como já ressaltado, nessas notas introdutórias não lidaremos com essas questões mais sofisticadas. Vamos, então, executar a receita cuidadosamente, para os valores iniciais $M=36$ e $N = 21$. A seguinte tabela reproduz o comportamento do algoritmo, passo a passo, para esses valores:

| Passo | Linha | M | N | Comentários |
|-------|-------|----|----|--------------------------|
| --- | --- | 36 | 21 | --- |
| 1 | 1 | 36 | 21 | $36 < 21$ |
| 2 | 2 | 15 | 21 | $36 - 21 = 15$ |
| 3 | 1 | 15 | 21 | $15 < 21$ |
| 4 | 2 | 15 | 21 | não executado: $15 < 21$ |
| 5 | 3 | 15 | 6 | $21 - 15 = 6$ |
| 6 | 1 | 15 | 6 | $15 < 6$ |
| 7 | 2 | 9 | 6 | $15 - 6 = 9$ |
| 8 | 1 | 9 | 6 | $9 < 6$ |
| 9 | 2 | 3 | 6 | $9 - 6 = 3$ |
| 10 | 1 | 3 | 6 | $3 < 6$ |
| 11 | 2 | 3 | 6 | $3 < 6$; não executado |
| 12 | 3 | 3 | 3 | $6 - 3 = 3$ |
| 13 | 1 | 3 | 3 | MDC é 3. Pare |

Algoritmos: o que são?

A coluna “Passo” numera sequencialmente os passos que são executados, num total de 13 passos. A coluna “Linha” informa a qual linha *do algoritmo* corresponde cada passo. As colunas M e N informam, respectivamente, os valores armazenados “nas caixas” M e N. A última coluna contém comentários adicionais, onde o símbolo “< >” é usado para indicar “diferente de”.

Seguindo a receita do algoritmo, no passo 1 estaremos executando a linha 1, com os valores $M=36$ e $N=21$. A linha 1 do algoritmo diz:

1. Se $M = N$, então o MDC é M (ou N); páre.

Sabemos que, nesse instante, o valor armazenado em M, i.e. 36, *não é igual* ao valor armazenado em N, i.e. 21. A instrução não se aplica, portanto, e devemos prosseguir com a próxima linha do algoritmo, ou seja, a linha 2. Essa linha diz:

2. Caso a):

se $(M > N)$ então substitua M por $(M-N)$ e repita a partir do passo 1.

Como $36 > 21$, então vale $M > N$ nesse ponto. Logo, devemos executar a substituição de valores preconizada. Como resultado, os valores de M e N passam a $36-21=15$ e 21, como indicado nas colunas correspondentes. E devemos voltar à linha 1 do algoritmo, como indicado no próximo passo, ou seja no passo 3 na tabela.

O processo se repete: como ainda não temos $M=N$, uma vez que temos agora $M=15$ e $N=21$, passamos de novo diretamente para a linha 2 do algoritmo. Isso é indicado na tabela, pois o passo seguinte, i.e., o passo 4, se refere à linha 2 ao algoritmo. Mas agora não vale mais a condição $M > N$, pois $M=15$ e $N=21$ nesse ponto. Então, as ações descritas na linha 2 são ignoradas e vamos para a linha seguinte, ou seja, linha 3 do algoritmo. É o que ocorre no passo 5. A linha 3 diz:

3. Caso b):

se $(N > M)$ então substitua N por $(N-M)$ e repita a partir do passo 1.

Agora, vale $N > M$, pois $21 > 15$. Então a ação associada é executada e substituímos o valor de N por $21-15=6$, e devemos retornar de novo à linha 1 do algoritmo. Como indicado nas colunas de M e N no passo 5.

O processo prossegue desta forma, como pode ser seguido na tabela acima, até que atinjamos a linha 13 da tabela, com os valores de M e N em 3, e vamos executar de novo a linha 1 do algoritmo. Agora, temos que a condição $M=N$ se verifica. Realizamos então a ação prevista na linha 1 do algoritmo: informamos que o MDC é 3 e o processo pára.

Realmente, o MDC correto entre 36 e 21 é 3.

Exercitando o algoritmo para alguns outros valores de M e N, podemos nos convencer que a receita vai produzir, sempre, o valor correto do MDC de quaisquer dois números inteiros positivos dados como entrada.

Como exercício, demonstre que, se $M > N$, então o MDC entre M e N é o mesmo que o MDC entre $(M-N)$ e N.