Três técnicas de solução de problemas

MC102OP-2018s1-Aula05-180313

Arthur J. Catto, PhD
ajcatto@g.unicamp.br

13 de março de 2018

1 Simplificar o problema resolvendo casos especiais

Muitas vezes, mesmo num problema complexo, alguns casos especiais têm solução trivial.

A técnica consiste em identificar e tratar separadamente esses casos especiais, reduzindo assim progressivamente o tamanho do problema.

1.1 Exemplo: *Avaliar se um dado ano é bissexto*

Problema. Avaliar se um dado ano é bissexto ou não.

Entrada. Um inteiro positivo representando o ano.

Saída. `"XXXX é um ano bissexto."` ou `"XXXX não é um ano bissexto."`

Propriedades. Um ano é bissexto se ele for divisível por 4 mas não por 100, exceto se for divisível por 400.

1.1.1 Desenvolvimento do algoritmo

Examinando as **Propriedades** vemos que este problema tem solução trivial em alguns casos especiais.

Por exemplo, se um ano não for divisível por 4, ele não é bissexto.

Um recurso útil num caso como este é associar um nome a cada uma das condições que serão verificadas, pois isto contribui muito para o entendimento e para a documentação da solução.

Por exemplo, vamos calcular a condição `ano não é divisível por 4` e dar um nome para ela.

```
ano_nao_eh_div_4 = (ano % 4 != 0)
```

Agora é possível começar a desenhar uma estrutura de comandos condicionais que nos levem à resposta.

```
ano_nao_eh_div_4 = (ano % 4 != 0)
if ano_nao_eh_div_4:
    print(ano, "não é bissexto.")
else:
    ...
```

Quais são os casos que ainda precisam ser tratados?

— Ainda precisam ser tratados os anos divisíveis por 4.

Voltando às **Propriedades**, temos que um ano divisível por 4, mas não por 100, é bissexto e também pode ser isolado facilmente.

Vamos calcular esta nova condição e também associar um nome a ela...

```
ano_nao_eh_div_100 = (ano % 100 != 0)
```

Nosso esboço de solução agora ficará assim...

```
ano_nao_eh_div_4 = (ano % 4 != 0)
ano_nao_eh_div_100 = (ano % 100 != 0)

if ano_nao_eh_div_4:
    print(ano, "não é bissexto.")
elif ano_nao_eh_div_100:
    print(ano, "é bissexto.")
elif ...:
    ...
```

O que deve ainda ser tratado numa nova cláusula?

- Devem ser tratados os anos que não foram "pegos" pelas cláusulas anteriores, isto é, aqueles divisíveis por 4 e também divisíveis por 100.
  - Estes, se forem divisíveis por 400, serão bissextos. Caso contrário, não.

Vamos repetir o raciocínio anterior, calculando uma nova condição, dando um nome a ela e depois completando nosso comando condicional...

```
ano_nao_eh_div_4 = (ano % 4 != 0)
ano_nao_eh_div_100 = (ano % 100 != 0)
ano_eh_div_400 = (ano % 400 == 0)

if ano_nao_eh_div_4:
    print(ano, "não é bissexto.")
elif ano_nao_eh_div_100:
    print(ano, "é bissexto.")
elif ano_eh_div_400:
    print(ano, "é bissexto.")
else:
    print(ano, "não é bissexto.")
```

O que deve ainda ser tratado numa nova cláusula?

- Não há mais nada para ser feito...

Como não restam mais casos, o desenvolvimento está concluído...

1.1.2 Solução 1: *Avaliar se um dado ano é bissexto*

```
ano = int(input("Qual o ano a ser testado? "))

ano_nao_eh_div_4 = (ano % 4 != 0)
ano_nao_eh_div_100 = (ano % 100 != 0)
ano_eh_div_400 = (ano % 400 == 0)

if ano_nao_eh_div_4:
    print(ano, "não é bissexto.")
elif ano_nao_eh_div_100:
    print(ano, " é bissexto.")
elif ano_eh_div_400:
    print(ano, " é bissexto.")
else:
    print(ano, "não é bissexto.")
```

Neste exemplo, a complexidade da solução foi reduzida passo a passo, retirando-se do problema um a um casos especiais que admitiam soluções triviais.

Examinando a solução final, vemos que há apenas quatro casos, que convergem para duas soluções distintas.

Isso sugere que talvez haja uma outra maneira de resolver este problema: criar uma condição que reconheça se o ano é bissexto de uma vez só.

Se isso for possível, nosso programa poderá reduzir-se a...

```
ano = int(input("Qual o ano a ser testado? "))
ano_eh_bissexto = ...
if ano_eh_bissexto:
    print(ano, "é bissexto.")
else:
    print(ano, "não é bissexto.")
```

Para completar o comando de atribuição podemos voltar às **Propriedades**: *Um ano é bissexto se ele for divisível por 4 mas não por 100, exceto se for divisível por 400.*

Reescrevendo essa condição de um jeito um pouco mais formal...

```
ano é bissexto se
    (ano é divisível por 4 e ano não é divisível por 100) ou
    (ano é divisível por 400)
```

O que leva a ...

```
ano_eh_bissexto =
    (ano_eh_div_4 and ano_nao_eh_div_100) or
    (ano_eh_div_400)
```

... e nos permite concluir o desenvolvimento do nosso programa.

1.1.3 Solução 2: *Avaliar se um dado ano é bissexto*

```
ano = int(input("Qual o ano a ser testado? "))
ano_eh_div_4 = (ano % 4 == 0)
ano_nao_eh_div_100 = (ano % 100 != 0)
```

```
ano_eh_div_400 = (ano % 400 == 0)

ano_eh_bissexto = (ano_eh_div_4 and ano_nao_eh_div_100) or \
                  (ano_eh_div_400)
if ano_eh_bissexto:
    print(ano, "é bissexto.")
else:
    print(ano, "não é bissexto.")
```

2 Supor o problema resolvido e caminhar para trás

Uma boa estratégia para resolver um problema complexo é supor o problema resolvido e tentar imaginar qual terá sido a última ação executada.

Essa é a estratégia que as crianças adotam para acelerar a solução de problemas de labirinto... e algumas empresas para acelerar a solução de problemas de engenharia (*reverse engineering*)...

2.1 Exemplo: *Cálculo do Imposto de Renda Retido na Fonte*

Em 2018, o imposto de renda retido na fonte das pessoas físicas empregadas no Brasil, está sendo calculado conforme a tabela de alíquotas progressivas ao lado. Dada a renda mensal de uma pessoa, deseja-se calcular o valor do imposto de renda retido.

Renda	Alíquota	Dedução
Até 1.903,98	0,0%	0,00
De 1.903,99 até 2.826,65	7,5%	142,80
De 2.826,66 até 3.751,05	15,0%	354,80
De 3.751,06 até 4.664,68	22,5%	636,13
Acima de 4.664,68	27,5%	869,36

Neste caso, supondo o problema resolvido, a última ação deve ter sido...

```
print(imposto)
```

Agora coloque-se nessa posição e repita o raciocínio... - Qual terá sido a última ação executada antes dessa?

Para poder mostrar o `imposto`, é preciso calculá-lo e esta provavelmente foi a ação anterior.

O problema sugere que `imposto` é calculado como

```
imposto = renda * aliquota - deducao
print(imposto)
```

E agora? Para calcular `imposto` precisamos de `renda`, `aliquota` e `deducao`.

`renda` é um dado que pode ser pedido para o usuário.

Vamos acrescentar essa ação ao nosso algoritmo, desta vez no início.

```
renda = float(input("Qual a renda? "))
...
imposto = renda * aliquota - deducao;
print(imposto)
```

Conhecida a `renda`, só falta determinar a `aliquota` e a `deducao` correspondentes.

Você consegue criar um comando que resolva completamente esse problema?

— Não... e agora?

Você consegue criar um comando que resolva pelo menos um caso especial do problema?

— Sim. Quando `renda <= 1903.98`, `aliquota` e `deducao` são iguais a zero.

Como isso se aplica a apenas alguns casos, essa ação precisa aparecer num comando condicional.

```
if renda <= 1903.98:
    aliquota, deducao = 0, 0
```

Agora é preciso tratar os casos não cobertos por esse comando. Como fazer isso?

Examinando a tabela, vemos que quando `1903.99 <= renda <= 2826.65`, a `aliquota` é de `7.5%` e a `deducao` é de `142.80`.

É fácil incluir esse caso no nosso comando...

```
if renda <= 1903.98:
    aliquota, deducao = 0, 0
elif 1903.99 <= renda <= 2826.65:
    aliquota, deducao = 0.075, 142.80
```

Será possível simplificar um pouco esse comando?

— Sim.

A condição do `elif` é `1903.99 <= renda <= 2826.65` que se desdobra em `1903.99 <= rendaandrenda <= 2826.65`.

Mas, como todos os casos em que `renda <= 1903.98` foram apanhados pela condição anterior, o termo `1903.99 <= renda` é sempre `True`.

Como sabemos que o resultado de

`True and` *expressão*

é o mesmo que o de

expressão,

o primeiro termo da condição do **elif** pode ser eliminado.

Assim, nosso comando pode ser simplificado para...

```
if renda <= 1903.98:
    aliquota, deducao = 0, 0
elif renda <= 2826.65:
    aliquota, deducao = 0.075, 142.80
```

Repetindo sucessivamente esse raciocínio, podemos concluir o desenvolvimento...

```
if renda <= 1903.98:
    aliquota, deducao = 0, 0
elif renda <= 2826.65:
    aliquota, deducao = 0.075, 142.80
elif renda <= 3751.05:
    aliquota, deducao = 0.150, 354.80
elif renda <= 4664.68:
    aliquota, deducao = 0.225, 636.13
else:
    aliquota, deducao = 0.275, 869.36
```

2.1.1 Solução: *Cálculo do Imposto de Renda Retido na Fonte*

Reunindo todos os fragmentos da solução temos...

```
renda = float(input("Qual a renda? "))

if renda <= 1903.98:
    aliquota, deducao = 0, 0
elif renda <= 2826.65:
    aliquota, deducao = 0.075, 142.80
elif renda <= 3751.05:
    aliquota, deducao = 0.150, 354.80
elif renda <= 4664.68:
    aliquota, deducao = 0.225, 636.13
else:
    aliquota, deducao = 0.275, 869.36

imposto = renda * aliquota - deducao

print("Imposto a ser retido =", format(imposto,".2f"))
```

3 Aproveitar uma solução (parcial) disponível livremente

Quando o problema tiver uma solução parcial ou aproximada disponível livremente, nossa principal preocupação será aferir a qualidade do código e até que ponto ela realmente atende nossas necessidades.

3.1 Exemplo: Cálculo da data da Páscoa

Problema. Dado um ano, exibir a data (dia, mês, ano) do domingo de Páscoa.

Buscando na rede, encontramos a seguinte solução para este problema, sujeita a algumas restrições:

- O dia da Páscoa num ano entre 1900 e 2099 pode ser calculado por:
  -
    ```
    a = ano % 19
    b = ano % 4
    c = ano % 7
    d = (19 * a + 24) % 30
    e = (2 * b + 4 * c + 6 * d + 5) % 7
    pascoa = 22 + d + e
    ```
- Restrições
  - Para os anos de 1954, 1981, 2049 e 2076, é preciso subtrair 7 do resultado.
  - O dia da Páscoa é calculado a partir do início de março (1 = 01 de março) e pode cair em abril.

Vamos supor que a restrição ao período coberto pela solução seja aceitável. Nesse caso precisaremos apenas criticar a entrada, recusando anos fora do período coberto pelas fórmulas.

O maior problema é como garantir a corretude das fórmulas... E se o cálculo estiver errado?

Dando uma nova busca na rede encontramos tabelas com os dias da Páscoa para vários períodos, incluindo esse que nos interessa.

Como as fórmulas cobrem apenas 200 anos, é possível testar o resultado exaustivamente contra os valores tabelados. (Um programa para fazer uma comparação exaustiva entre os valores calculados e os tabelados será estudado como exemplo numa aula futura.)

Se o valor calculado e o valor tabelado sempre forem iguais, a confiança melhora, e teremos duas possíveis soluções para o nosso problema:

- Se nossa principal preocupação for tempo e não espaço, provavelmente a tabela será a melhor opção.
- Se nossa principal preocupação for espaço e não tempo, provavelmente as fórmulas serão a melhor opção.
- Em todo caso, será bom incluir uma ***asserção protetora*** para prevenir surpresas, como discutido a seguir.

3.1.1 O comando `assert`

Uma *asserção* é uma expressão lógica que se acredita que seja verdadeira num determinado ponto de um programa. Comandos `assert` são uma maneira conveniente de inserir asserções depuradoras num programa em desenvolvimento.

Na sua forma mais simples, o comando tem a forma

`assert` *`expressão`*`.`

Quando executado, `assert` avalia a *`expressão`*.

Neste caso, damos uma busca na Wikipedia e descobrimos que a Páscoa nunca pode cair antes de 22 de março, nem depois de 25 de abril.

Lembrando que as fórmulas calculam o dia da Páscoa contado a partir de 01 de março, 22 de março corresponderá a 22 e 25 de abril a 56 (25 + 31).

Portanto, depois de calcular o valor final da variável `pascoa`, podemos nos proteger incluindo...

```
    assert 22 <= pascoa <= 56
```

3.1.2 Solução calculada

Um primeiro esboço de programa poderia ser...

```
ano = int(input("Qual o ano desejado? (1900-2099) "))
if 1900 <= ano < 2100:
    # calcular o dia da Páscoa
    # corrigir o cálculo para os anos especiais
    # tratar os casos em que a Páscoa cai em abril
else:
    print("Ano fora do intervalo aceitável.")
```

Calcular o dia da Páscoa Expandindo os comentários e reunindo os fragmentos...

```
ano = int(input("Qual o ano desejado? (1900-2099) "))
if 1900 <= ano < 2100:
    # calcular o dia da Páscoa
    a = ano % 19
    b = ano % 4
    c = ano % 7
    d = (19 * a + 24) % 30
    e = (2 * b + 4 * c + 6 * d + 5) % 7
    pascoa = 22 + d + e
    # corrigir o cálculo para os anos especiais
    if ano == 1954 or ano == 1981 or ano == 2049 or ano == 2076:
```

```
        pascoa -= 7
    assert 22 <= pascoa <= 56
    # tratar os casos em que a Páscoa cai em abril
    if pascoa <= 31:
        dia, mes = pascoa, 'março'
    else:
        dia, mes = pascoa - 31, 'abril'
    print(dia, mes, ano)
else:
    print("Ano fora do intervalo aceitável.")
```