05. Funções e Sequências

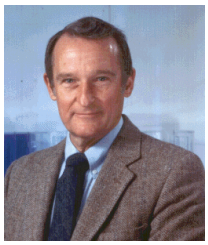
Princípios de Engenharia de Software (Texto em Elaboração)

Italo S. Vega
italo@pucsp.br

Faculdade de Estudos Interdisciplinares (FACEI)

Pontifícia Universidade Católica de São Paulo

Sumário


Seymour Cray — The trouble with programmers is that you can never tell what a programmer is doing until it's too late.

5 Funções e Sequências

 — Sequências são funções, portanto, são conjuntos.

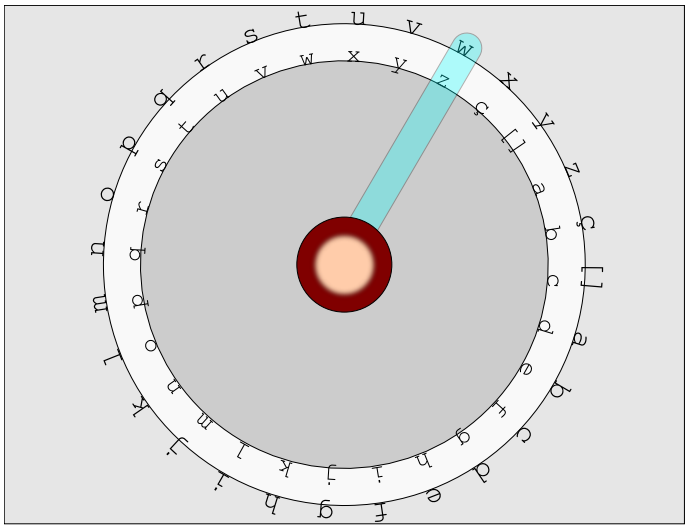
Durante uma conversa a respeito de segurança da informação Fubã descobre que a área de criptografia é essencial.

 — Ouvi dizer que a Cifra de César introduz diversos fundamentos de criptografia...

A Cifra de César é um caso ilustrativo simples da área de criptografia com base em funções. A ideia é substituir um símbolo por outro, calculado a partir de um deslocamento fixo no alfabeto (*Nova Reforma Ortográfica*, 1/1/2009). Um exemplo de disco para decifragem de uma Cifra de César encontra-se na figura a seguir [1] e considera um deslocamento igual a 2. Assim, decodifica-se a letra c por a.

 — Farei um desenho para clarear meus pensamentos usando um deslocamento igual a 3.

[1] Anel de decodificação da Cifra de César. URL: https://images-na.ssl-images-amazon.com/images/I/81JK7V3EeeL._SL1500_.jpg. Acesso em 04/03/2021.



No esquema, o símbolo a encontra-se associado ao d, um deslocamento de cifragem igual a 3. Então a Cifra de César de a é d. Um raciocínio semelhante codifica w como z.

5.0.1 Questão: Cifragem de texto

Contexto Considere a entrada `"PUC ALGORITMO"` a ser codificada conforme uma Cifra de César supondo um deslocamento igual a 3, segundo o esquema de disco de cifragem desenhado pelo Fubã.

Enunciado Assinale a alternativa contendo uma afirmação corresponde à cifragem da entrada indicada no contexto:

1. `"SXFCDOJULWPR"`
2. `"SXFDCOJULWPR"`
3. `"SXFCDOJLUWPR"`
4. `"SXFCDPJULWPR"`

Fubã se questiona a respeito de um apropriado modelo lógico para raciocinar sobre a codificação de símbolos.

"Talvez eu possa montar um conjunto mapeando símbolos em símbolos... a eu mapeio em d...

Funções definidas por enumeração. Fubã percebe que o domínio dos símbolos a serem codificados é finito e talvez pudesse criar uma lógica com base na função "cesar" definida por enumeração dos seus elementos:

$$\text{cesar} \triangleq \{\square \mapsto c, a \mapsto d, b \mapsto e, \cdots, \text{ç} \mapsto b\}$$

Ele utiliza o símbolo "$\square$" para representar um espaço entre duas palavras.

5.1 Aplicação de Funções

O conceito de função como um conjunto de pares nos quais o primeiro elemento ocorre, exatamente, uma única vez, direciona os pensamentos do Fubã.

 — Eis a regra de uma função f que retorna o dobro do valor de uma entrada pertencente ao conjunto $\mathbb{Z}$, desde que seja positivo:

$$f \triangleq \{\ x : \mathbb{Z} \mid x > 0 \bullet x \mapsto 2x\ \}$$

Neste caso, observa-se a declaração de um elemento x de $\mathbb{Z}$, um predicado sobre x que, se verdadeiro, irá associá-lo ao valor-imagem $2x$, quando a função f for aplicada. A aplicação desta regra no valor 5 desencadeará o seguinte cálculo:

$$f(5) = \{\ x : \mathbb{Z} \mid x > 0 \bullet x \mapsto 2x\ \} = \{\ 5 \mapsto 2 \cdot 5\ \}(5) = \{\ 5 \mapsto 10\ \}(5) = 10$$

A intenção de se aplicar uma função é de buscar a imagem de um elemento do domínio (chamado argumento) conforme a definição da função.

 "E o que acontece se eu aplicar esta função em um argumento que torna o predicado falso?"

Trata-se de um caso onde a função não associa um valor ao argumento x. Representaremos este caso pelo valor $\perp$ (absurdo lógico), representando a ideia de aplicação indefinida de função. Assim:

$$f(-3) = \{\ x : \mathbb{Z} \mid x > 0 \bullet x \mapsto 2x\ \} = \perp$$

5.1.1 Questão: Aplicação de funções

Contexto Considere a função cesar:

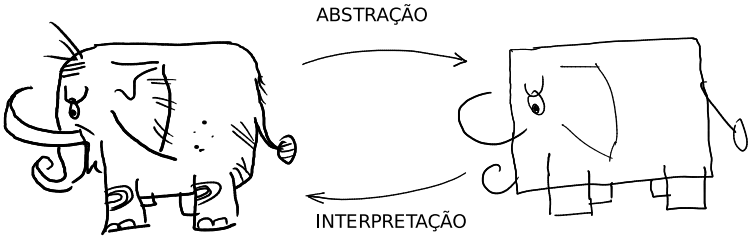
$$\text{cesar} \triangleq \{\square \mapsto c, a \mapsto d, b \mapsto e, \cdots, z \mapsto a, \text{ç} \mapsto b\}$$

Enunciado Assinale a alternativa contendo uma afirmação **falsa**:

1. $\text{cesar}(\square) = c$
2. $\text{cesar}(a) = d$
3. $\text{cesar}(c) = e$
4. $\text{cesar}(z) = a$

5.2 Modelos Inválidos e Inconsistentes

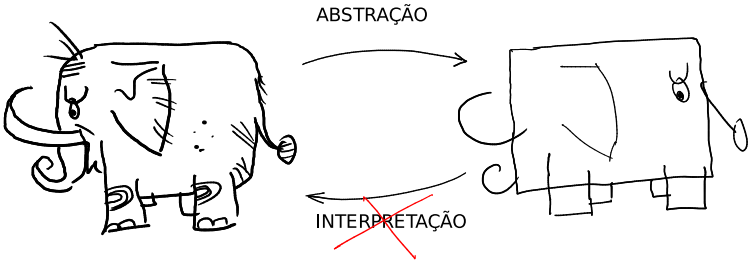
Um modelo representa aspectos relevantes de um sujeito para um certo propósito. Modelos são produzidos por um processo de abstração. Trata-se de um processo que destaca aspectos relevantes em um modelo, omitindo aqueles irrelevantes:



Quando validamos positivamente um modelo, verificamos que ele contém aspectos que suportam raciocínios corretos a respeito do sujeito modelado. O processo de abstração é determinante na programação de computadores visto que nos apoiamos em modelos para o desenvolvimento de programas.

 — Por vezes, construímos um modelo contendo elementos que invalidam o nosso raciocínio.

Erros de abstração podem conduzir a modelos que não representam o sujeito sob estudo, sendo conhecidos por modelos inválidos:



5.2.1 Questão: Modelos inválidos

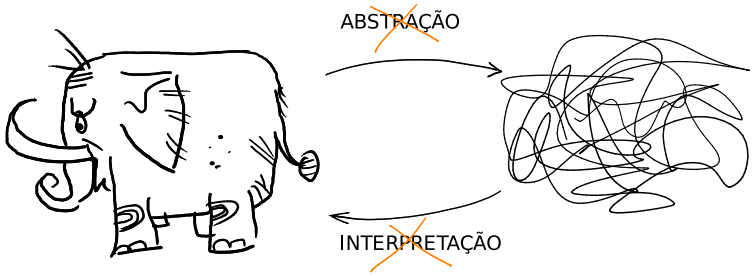
Contexto No problema de cifragem, Fubã precisa elaborar um modelo de alfabeto de acordo com *Nova Reforma Ortográfica*, 1/1/2009, que inclui as letras k, w e y, 26 letras portanto. Ele também decide incorporar os símbolos $\square$ e $ç$ no seu alfabeto modelado pelo conjunto M.

Enunciado Assinale a alternativa contendo uma afirmação verdadeira:

1. O modelo M suporta raciocínios envolvendo 2 e 5.
2. É verdade que a letra a precede a letra b em M.
3. O modelo M possui 26 elementos.
4. Se $\square \notin M$ o modelo é inválido.

— Em outras situações, é óbvio que há algo errado com o próprio modelo, independentemente da consideração do sujeito.

Trata-se de um caso especial de modelo inválido conhecido por inconsistente. Neste caso, o modelo não modela um sujeito possível. Pode ser pensado como um modelo vazio, visto que não existem objetos matemáticos que satisfaçam as condições por ele estabelecidas.



5.2.2 Questão: Modelos inconsistentes

Contexto Fubã elaborou o seguinte modelo do quebra-cabeça Batalha Naval. Existem 5 navios distribuídos em um campo de batalha, posicionados em coordenadas pertencentes ao conjunto $1..5 \times 1..5$. Em uma partida, há um navio em $(2, 5)$ e, se o jogador disparar nessas coordenadas, o contador de navios destruídos n será incrementado de uma unidade. No estado inicial n tem valor 0, com a seguinte ação de disparo:

$$
\begin{aligned}
\text{disparar}[x, y] \triangleq & \vee && \wedge\ \text{tem\_navio\_em}[x, y] \\
& && \wedge\ n < 5 \\
& && \wedge\ n' = n + 1 \\
& \vee && \wedge\ \text{tem\_navio\_em}[x, y] \\
& && \wedge\ n = 5 \\
& && \wedge\ n' = \bot
\end{aligned}
$$

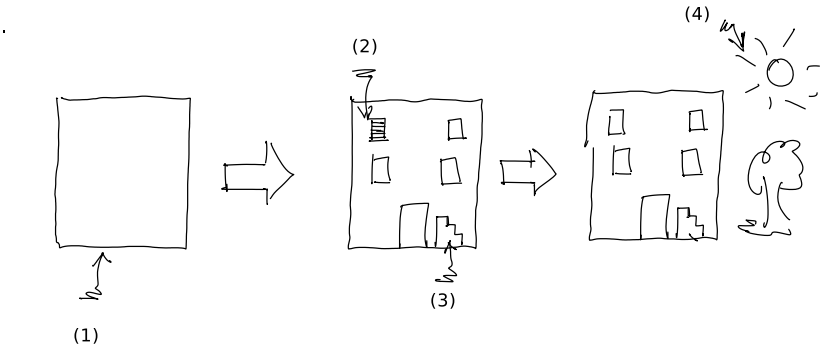
Enunciado Assinale a alternativa contendo uma afirmação **falsa**:

1. O modelo é inválido se não houver um navio em $(2, 5)$.
2. O campo de batalha possui 25 coordenadas.
3. A ação disparar é inconsistente.
4. n nunca será 5.

 — Já ouvi falar nos seguintes tipos comuns de erros de modelagem:

1. elemento omisso
2. elemento detalhe
3. elemento estranho
4. elemento alienígena

Esta classificação ajuda a **evitar erros** de modelagem básicos



Retornando ao problema da cifragem de César, Fubã pediu a ajuda a um colega[2] para desenvolver um código que implementa a função cesar.

5.2.3 Questão: Código Python inválido

Contexto Fubã desenvolveu o seguinte código em Python para implementar a função:

$$\text{cesar} \triangleq \{\square \mapsto c, a \mapsto d, b \mapsto e, \cdots, z \mapsto a, \text{ç} \mapsto b\}$$

```
# CENÁRIO de implementação da Cifra de César
# Definição do método "cesar"
def cesar() :
  entrada         = str ( input("Digite: ") )
  alfabetoCesar   = [ ' ', 'a', 'b' ]
  alfabetoCifrado = [ 'b', 'c', 'd' ]
  palavraCifrada  = []
  for c in range (0, len (entrada)) :
    i = alfabetoCesar.index (entrada[c])
    letraCifrada = alfabetoCifrado [i]
    palavraCifrada.append (letraCifrada)
  for c in palavraCifrada :
    print ( f"{c}", end="")
# Lógica de CONTROLE e INTERAÇÃO
cesar()
```

[2] Código modificado, mas inspirado no raciocínio de Gabriel Arai em 2021.

Enunciado Assinale a alternativa contendo um **absurdo** lógico:

1. `entrada == ' '` $\wedge$ `cesar() == 'b'`
2. `entrada == 'a'` $\wedge$ `cesar() == 'c'`
3. `entrada == 'b'` $\wedge$ `cesar() == 'd'`
4. `entrada == 'c'` $\wedge$ `cesar() == 'e'`

Modelar a Cifra de César por enumeração dos elementos da função cesar é um caminho possível. Entretanto, Fubã decide por uma alternativa, envolvendo funções definidas por uma expressão-λ. Para começar, ele reconhece que as palavras a serem codificadas são formadas pelas letras de um alfabeto e associa um índice a cada uma delas. Arbitrariamente, $\square$ tem índice 1, a tem índice 2 e assim por diante, até ç.

5.2.4 Questão: Implementação de funções por métodos em Python

Contexto Considere um alfabeto modelado pela função A:

$$A \triangleq \{1 \mapsto \square, 2 \mapsto a, 3 \mapsto b, ...27 \mapsto z, 28 \mapsto \text{ç}\}$$

O método `f` se propõe implementar a função A:

```
// CENÁRIO de implementação de "f"
def f (x) :
    return ord(x)
```

É comum utilizarmos a tabela ASCII na implementação de valores do tipo `char` em Python ("ASCII Table", 2020).

Enunciado Assinale a alternativa contendo uma afirmação verdadeira:

1. `f (' ') == 32`.
2. `f ('a') == 2`.
3. `f ('b') == 3`.
4. `f ('c') == 'c'`.

 — Ops! O meu código não implementa a função A... Ele é inválido!

Fubã percebe que o seu conhecimento a respeito de funções precisa melhorar. Também precisa encontrar uma maneira de realizar o mapeamento entre a e d de modo que o deslocamento seja parametrizado. Espec sugere que estude os conceitos de sequência e função inversa, para depois continuar com a modelagem da Cifra de César.

5.3 Sequências

Observando o seu modelo de alfabeto, Fubã conclui que se trata de uma função. Por algum motivo, ele se lembrou de uma estrutura conhecida por sequência:

> **Sequência**: um conjunto ordenado de valores do mesmo tipo.

Representaremos sequências de símbolos conforme a seguinte notação. Dois parênteses angulados delimitarão os elementos de uma sequência, separados por vírgulas, como em $\langle a, b, c\rangle$. Fubã, então, utilizou uma sequência para modelar o alfabeto da Cifra de César.

 — Construirei uma lógica partindo de um alfabeto de símbolos modelado como a sequência A:

$$A \triangleq \langle \square\ , a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z, ç\ \rangle$$

Pela notação, o par de parênteses angulados $\langle\cdots\rangle$ indica uma sequência de valores. Por conseguinte, A corresponde ao conjunto contendo elementos como $1 \mapsto \square$ e $2 \mapsto a$. A abreviação possibilita usarmos o nome A no lugar do próprio conjunto. Fubã sabe que esta é apenas uma forma reduzida de representar a função:

$$A \triangleq \{1 \mapsto \square, 2 \mapsto a, 3 \mapsto b, 4 \mapsto c, ..., 28 \mapsto ç\}$$

 — Sequências são conjuntos!

Devido às características da Cifra de César, a ordem dos símbolos no alfabeto é importante e a noção de sequência captura esta ideia. O primeiro elemento está na posição de número 1, o segundo, na posição 2 e assim por diante. É possível que um elemento ocupe outras posições, pois sequências admitem repetições. No problema do Fubã, entretanto, isso não ocorre. A presença do símbolo $A(1) = A\ 1 = A_1 = \square$ é única na sequência A.

Espec aproveita para lembrar que a quantidade de elementos em uma sequência será representada por "#" seguido do nome da sequência.

5.3.1 Questão: Modelagem com sequências

Contexto Considere as sequências:

I) $n = \langle 2, 4, 6, 6\rangle$
II) $m = \langle a, b, d, b, d\rangle$
III) $t = \langle \mathsf{FUBA}, \mathsf{ESPEC}, \mathsf{PROFE}\rangle$

Enunciado Assinale a alternativa contendo uma afirmação verdadeira:

1. $m_3 = b$.
2. $t_{n_1} = \mathsf{FUBA}$.
3. $\#t = \#m = 3$.
4. $(n_2 = 4) \wedge (\#n = 4)$.

A partir da modelagem do alfabeto como a sequência A, as seguintes afirmações tornam-se verdadeiras na lógica do Fubã:

1. **dom** $A = 1..28$
2. $\{a, c.m\} \subset$ **ran** A
3. $\#A = 28$
4. $A(2) = A\ 2 = A_2 = a$

A notação $1..28$ apenas abrevia o conjunto $\{1, 2, 3, \cdots, 28\}$.

Visualização de Sequências Para visualizar uma sequência na forma de um diagrama, Fubã decide destacar dois elementos:

- um retângulo para a sequência e
- diversas ovais para destacar elementos de interesse, desenhadas na parte interna do retângulo.

Como um exemplo, ele retoma a representação do alfabeto, modelado como a sequência A:

$$A \triangleq \langle \square\ , a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z, ç\ \rangle$$

Em A, encontramos 28 elementos, mas Fubã resolve destacar apenas três no diagrama que, ao ser inspecionado, revela que $A(5) = d$, por exemplo.

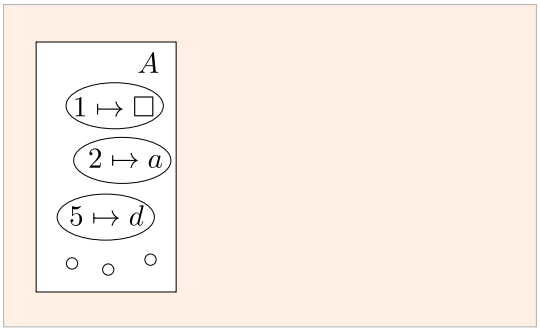


Figura 5.1: Diagrama de um alfabeto visto como a sequência A

Domínio e imagem de uma sequência Como sequências também são funções, as projeções **dom** e **ran** podem ser aplicados sobre elas. A projeção **dom** A produz o conjunto dos índices de A:

$$\begin{aligned}\textbf{dom}\ A =& \{\ i : \mathbb{N} \mid (i \mapsto x) \in A\ \bullet i\ \} \\ =& \{1, 2, 3, \cdots, 28\}\end{aligned}$$

Os elementos de **ran** A formam o conjunto-imagem da sequência A, constituído pelos segundos elementos de cada par:

$$\begin{aligned}\textbf{ran}\ A =& \{\ i : \mathbb{N} \mid (i \mapsto x) \in A\ \bullet x\ \} \\ =& \{\ \square, a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, x, ç\ \}\end{aligned}$$

Para visualizar as projeções **dom** A e **ran** A Fubã elabora o diagrama mostrado a seguir.

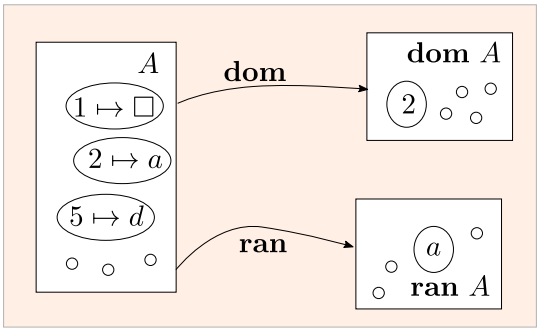


Figura 5.2: Domínio e imagem da sequência A

Aplicação de sequência Quando aplicamos uma sequência a um argumento do domínio, obtemos a sua imagem. Qual a imagem de 3 sob a função $\langle 2, 4, 6, 8, 10\rangle$? "Devo me lembrar do conjunto representado pela sequência e encontrar o par cujo primeiro elemento é 3", pensa Fubã. Ele escreve:

$$\langle 2, 4, 6, 8, 10\rangle\ (3) = \{1 \mapsto 2, 2 \mapsto 4, 3 \mapsto 6, 4 \mapsto 8, 5 \mapsto 10\}(3) = 6$$

O seguinte diagrama ilustra o caso da aplicação da sequência A no argumento 2: $A(2) = A\ 2 = A_2 = a$.

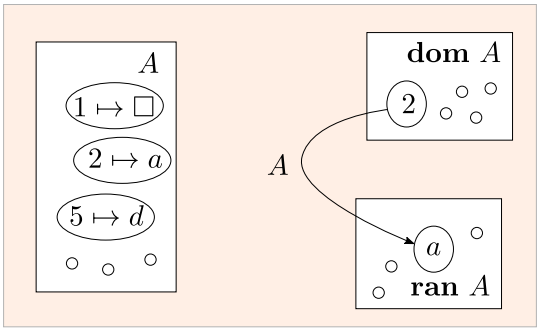


Figura 5.3: Aplicação de A em 2 produz a

5.4 Função Inversa

Ao modelar o alfabeto como uma sequência, Fubã tem acesso aos símbolos a partir dos seus índices. E como recuperar o índice a partir de um símbolo?

 — Você precisa do conceito de função inversa!

Fubã já havia esquecido das funções inversas.

> Função inversa: o conjunto de pares ordenados obtido quando se reverte cada par na função.

Se o elemento $(x, y) \in f$, sendo f uma função, a sua inversa, indicada por $f^{\sim}$, contém o elemento (y, x):

$$(y, x) \in f^{\sim}$$

O conjunto resultante pode ou não ser uma função. No particular caso de uma função f cuja inversa também é uma função, indicaremos esta última por f^{-1}.

 — A função inversa troca os componentes de um par ordenado

No específico caso da sequência A, temos uma função bijetora. A sua inversa, neste caso, também é uma função, como podemos observar no seguinte diagrama:

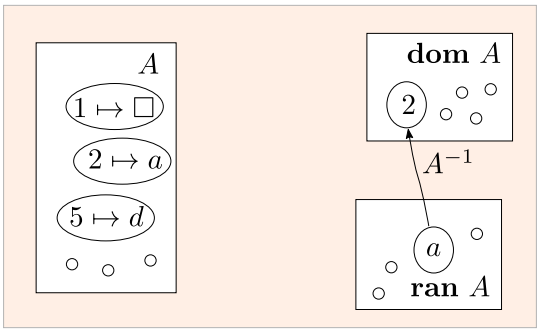


Figura 5.4: Aplicação da inversa de A em a

$$A \triangleq \{\ 1 \mapsto \square, 2 \mapsto a, ..., 28 \mapsto \text{ç}\ \}$$

$$A^{-1} = \{\ \square \mapsto 1, a \mapsto 2, ..., \text{ç} \mapsto 28\ \}$$

5.4.1 Questão: Modelagem com sequências

Contexto Considere uma sequência de atividades de avaliação em um curso de computação:

$$\text{atv} \triangleq \{\ 1 \mapsto A_{11}, 2 \mapsto A_{12}, 3 \mapsto A_{21}, 4 \mapsto A_{22}\ \}$$

Enunciado Assinale a alternativa contendo uma afirmação verdadeira:

1. $\text{atv}(2) = A_{21}$
2. $\text{atv}^{-1}(2) = A_{12}$
3. $\#\text{atv} = \text{atv}^{-1}(A_{22})$
4. $\text{atv}^{-1}(A_{11}) + \text{atv}^{-1}(A_{12}) = \bot$

5.4.2 Questão: Inversas em Python

Contexto Considere a sequência:

$$s \triangleq \langle a, b, c, d\ \rangle$$

Considerando-se as apropriadas restrições em tempo de execução, s pode ser implementada pelo seguinte método:

```
# CENÁRIO de implementação da sequência
#   s = <a,b,c,d>
def s (i) :
    return chr (i+96)
```

e a sua inversa, s^{-1}:

```
# CENÁRIO de implementação da inversa da sequência
#   s = <a,b,c,d>
def s_inv (x) :
    return ord(x) - 96
```

Enunciado Assinale a alternativa contendo uma afirmação verdadeira:

1. `s_inv ('d') == s_inv ('c') - 1`
2. $[\![\#s]\!]_{Python} =$ `s_inv ('d')`
3. `s (1) + s (2) == 3`
4. `s (3) == 'd'`

Com a ajuda dos colegas, Fubã chega na seguinte proposta para a implementação de A^{-1}:

```
def A_inv(x) :
  dom = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
    11, 12, 13, 14, 15, 16, 17, 18, 19, 20,
    21, 22, 23, 24, 25, 26, 27]
  ran = [' ','a','b','c','d','e','f',
    'g','h','i','j','k','l','m','n','o',
    'p','q','r','s','t','u','v','w','x','y','z']
  return { ran[i]: dom[i] for i in range (len (dom)) } [x]
```

5.4.3 Questão: Propriedades da inversa

Contexto Fubã modeloou o alfabeto do seu estudo da Cifra de César pela sequência A:

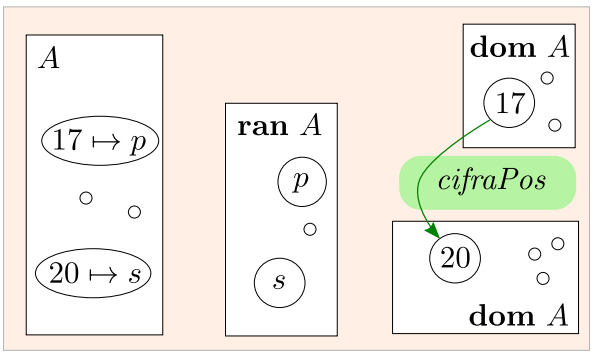
$$A \triangleq \langle \square\ , a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z, ç\ \rangle$$

Enunciado Assinale a alternativa contendo uma propriedade **falsa** da inversa de A:

1. $\mathbf{dom}\ A^{-1} = \mathbf{ran}\ A$
2. $\mathbf{ran}\ A^{-1} = 1..28$
3. $A^{-1}(a) = A(a)$
4. $A^{-1}\ \square = 1$

5.5 Funções e Expressões-λ

Retornando ao seu problema, Fubã começa a elaborar uma regra para a função cifraPos, que deve mapear um índice de símbolo em outro, considerando um deslocamento de cifragem:



Um tanto apressado, Fubã propõe uma definição para a função que calcula a posição cifrada de um símbolo. Espec sugere, no entanto, que Fubã dê o seguinte nome para a sua função: "cifraPos$_{Erro}$".

 "Não sei porque ele sugeriu isso, mas vamos lá..."

$$\text{cifraPos}_{Erro} \triangleq \lambda i : \mathbf{ran}\ A \bullet i + 3$$

E, rapidamente, Fubã produz uma implementação da nova função em Python:

```
# CENÁRIO de implementação da função "cifraPos"
def cifraPosErro (x) :
  return x+3
```

5.5.1 Questão: Modelos Inválidos em Python

Contexto O código de implementação da lógica de cifragem de um símbolo envolve os seguintes métodos:

```
# CENÁRIO de cifragem logicamente ERRADA de um símbolo
def A (x) :
  if x == ' ' :
    return 1
  return ord(x) - 95

def cifraPosErro (x) :
  return x+3

def A_inv (x):
  if (x == 1) :
    return ' '
  return chr (x + 95)
```

Enunciado Assinale a alternativa contendo uma afirmação **logicamente absurda** na interpretação em Python:

1. `A ('a') == 2`
2. `A_inc (5) == 'd'`
3. `cifraPosErro (1) == 5`
4. `cifraPosErro (28) == 31`

Casos de uma Função Após refletir por algum tempo, Fubã entendeu o seu erro de lógica na definição da "cifraPos$_{Erro}$". Usando a notação para regras de funções definidas por partes, Fubã escreve o domínio e a imagem da "cifraPos" corrigida:

$$\text{cifraPos} : \textbf{dom}\ A \rightarrow \textbf{dom}\ A$$

Se a posição de cifragem x estiver entre 1..25, a regra deverá mapeá-la em $x + 3$:

$$\text{cifraPos}_1 \triangleq \{\ x : \textbf{dom}\ A \mid p = x + 3 \land p \leq \#A \bullet x \mapsto p\ \}$$

Caso contrário, a imagem será calculada pela expressão $(x + 3) - \#A$:

$$\text{cifraPos}_2 \triangleq \{\ x : \textbf{dom}\ A \mid x + 3 > \#A \bullet x \mapsto (x + 3) - \#A\ \}$$

 — Sensacional! A definição da "cifraPos" envolve dois casos:

$$\text{cifraPos} \triangleq \text{cifraPos}_1 \cup \text{cifraPos}_2$$

Fubã decide aplicar essa função no valor 25 para recapitular a computação que será feita pelo interpretador Python:

$$\begin{aligned}
\text{cifraPos}(25) &= \text{cifraPos}_1(25) \\
&= \{\ x : \textbf{dom}\ A \mid p \leq \#A \bullet x \mapsto p\ \}(25) \\
&= \{\ x : \textbf{dom}\ A \mid x + 3 \leq \#A \bullet x \mapsto x + 3\ \}(25) \\
&= \{\ x : \textbf{dom}\ A \mid x + 3 \leq 28 \bullet x \mapsto x + 3\ \}(25) \\
&= \{\ x : \textbf{dom}\ A \mid 25 + 3 \leq 28 \bullet 25 \mapsto 25 + 3\ \}(25) \\
&= \{\ x : \textbf{dom}\ A \mid 28 \leq 28 \bullet 25 \mapsto 28\ \}(25) \\
&= \{\ 25 \mapsto 28\ \}(25) = 28
\end{aligned}$$

Cifra da Posição em Python Uma vez especificada a função "cifraPos", Fubã resolve implementá-la, chegando ao código:

```
# CENÁRIO de implementação do método "cifraPos"
def cifraPos(x) :
    global K # comprimento do alfabeto de cifragem
    p = x + 3
    return p if p < K else p-K
# aplicação de "cifraPos"
```

```
K = len([' ','a','b','c','d','e','f',
    'g','h','i','j','k','l','m','n','o',
    'p','q','r','s','t','u','v','w','x','y','z'])
cifraPos(1)
# cifraPos(2)
# cifraPos(27)
# cifraPos(28)
```

5.6 Resumo

1. Sequência é um conjunto ordenado de valores do mesmo tipo.
2. A função inversa é o conjunto de pares ordenados obtido quando se reverte cada par na função.

5.7 Exercício

5.7.1 Acesso Posicional

Implemente as funções-sequência A e A^{-1} em Python.

5.7.2 Sequências em Python

Contexto Os seguintes elementos do modelo Cifra de César correspondem ao alfabeto e ao seu comprimento, respectivamente:

$$A \triangleq \langle \square\ , a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z, ç\ \rangle$$
$$K \triangleq \#A$$

As seguintes interpretações devem ser respeitadas:

$$[\![A]\!]_{Python} = \texttt{def A(s) : ...}$$
$$[\![K]\!]_{Python} = \texttt{K = ...}$$

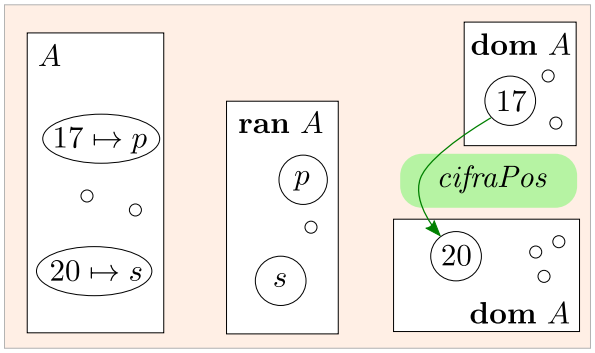
Enunciado

1. Implemente A e K em Python.
2. Escreva os resultado produzido pelas chamadas:

a. `A(1)`
b. `A(15)`
c. `A(28)`

5.7.3 Dominios e Imagens

Contexto Considere o diagrama a seguir:



com

$$\textbf{ran}\ \text{cifraPos} = x$$

Enunciado É correto afirmar-se que:

1. $x = \textbf{dom}\ A$
2. $x = \textbf{ran}\ A$
3. $x = A$
4. $x = A^{-1}$

5.7.4 Função como `dict`

Contexto Fubã decide investigar melhor o modelo de cifragem e implementações alternativas. Uma ideia é usar um tipo dicionário[3] para traduzir a função A:

```
# CENÁRIO de implementação de função por um "dict"
def f (x) :
    return {
        1 : ' ', 2 : 'a', 3 : 'b', 4 : 'c', 5 : 'd', 6 : 'e',
        7 : 'f', 8 : 'g', 9 : 'h', 10: 'i', 11: 'j', 12: 'k',
        13: 'l', 14: 'm', 15: 'n', 16: 'o', 17: 'p', 18: 'q',
        19: 'r', 20: 's', 21: 't', 22: 'u', 23: 'v', 24: 'w',
        25: 'x', 26: 'y', 27: 'z', 28: 'ç'
    } [x]
```

Enunciado A partir desta definição, calcular as imagens:

```
# CENÁRIO de chamada do método "f"
f (2) == ?  # <-- (A)
f (24) == ? # <-- (B)
f (27) == ? # <-- (C)
```

5.7.5 Função como `string`

Contexto Outra possível tradução da função A segue[4]:

```
# CENÁRIO de implementação de função por uma imagem "String"
def A (x) :
    imagem = ' abcdefghijklmnopqrstuvwxyzç'
    return { i + 1: s for i, s in enumerate (imagem) } [x]
```

Como no caso anterior, também obtenho a mesma imagem para o argumento 2:

Enunciado A partir desta definição, calcular as imagens:

[3] Código modificado a partir da sugestão do estudante Vitor Couto.
[4] Código modificado a partir da sugestão do estudante Pedro Lucas.

```
# CENÁRIO de chamada do método "f"
f (2) == ?  # <-- (A)
f (24) == ? # <-- (B)
f (27) == ? # <-- (C)
```

5.7.6 Função como `array`

Contexto Ao caminhar para casa, Fubã lembrou-se de uma conversa que teve com outro colega[5] a respeito da tradução para Python da função A. Nesta versão, ele utilizou-se de dois valores do tipo *array* em Python:

```
# CENÁRIO de implementação de função usando "array"
def f(x) :
  dom = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
    11, 12, 13, 14, 15, 16, 17, 18, 19, 20,
    21, 22, 23, 24, 25, 26, 27, 28]
  ran = [' ','a','b','c','d','e','f',
    'g','h','i','j','k','l','m','n','o',
    'p','q','r','s','t','u','v','w','x','y','z', 'ç']
  return { dom[i]: ran[i] for i in range (len (dom)) } [x]
```

Enunciado A partir desta definição, calcule as imagens:

```
# CENÁRIO de chamada do método "f"
f (2) == ?  # <-- (A)
f (24) == ? # <-- (B)
f (27) == ? # <-- (C)
```

5.7.7 Implementação Monolítica

Contexto Com a ajuda de outro colega[6], Fubã conseguiu o seguinte:

```
# CENÁRIO de implementação da Cifra de César
# Definição do método "cesar"
def cesar(entrada, k) :
  A = 'abcdefghijklmnopqrstuvwyz'
  saida = ''
  for s in entrada :
    p = A.find (s)
    if p == -1 :
      # NÃO cifro o símbolo "s"
      saida += s
    else:
      # cifro o símbolo "s"
      pc = (p + k) % len (A)
```

[5] Código modificado a partir da sugestão do estudante Kevin.
[6] Código modificado mas inspirado no raciocínio de Paulo Restaino em 2021.

```
      saida += A [pc]
  return saida
# Lógica de CONTROLE e INTERAÇÃO
entrada = 'puc cdia es'
saida = cesar (entrada, k=2)
print (saida)
```

Enunciado Assinale a alternativa contendo um absurdo lógico:

1. `cesar(' ') == 'n'`
2. `cesar('a') == 'c'`
3. `cesar('b') == 'd'`
4. `cesar('c') == 'e'`

5.7.8 Expressões Regulares [**]

Contexto Uma expressão regular descreve a estrutura interna de uma cadeia de caracteres (um literal do tipo `string`). Em muitas situações de "limpeza de dados" utilizam-se expressões regulares. A importação do módulo `re` da biblioteca Python disponibiliza os recursos para a programação com expressões regulares.

Considere uma entrada de usuário do tipo `string` contendo dois literais do tipo `int` misturados com outros caracteres: `'Pagamento 2-8 em dia!'`. Em termos lógicos, deseja-se interpretar tal cadeia como um par (x, y) com $x \in 1..4$ e $y \in 7..9$:

$$[\![\texttt{'Pagamento 2-8 em dia!'}]\!]_{\text{lógico}} = (2, 8) \in (\{1, 2, 3, 4\} \times \{7, 8, 9\})$$

Por outro lado, cadeias de entrada sem a presença de dois literais para representar (x, y), devem produzir um absurdo:

$$[\![\texttt{'Pagamento em dia!'}]\!]_{\text{lógico}} = \bot$$

Em Python, a expressão regular `r'([1..4])-([7..9])'` descreve o conjunto de cadeias que possuem dois grupos de subcadeias. Cada grupo encontra-se demilitado por parênteses. O primeiro grupo, `[1..4]`, representa o valor x e o segundo, `[7..9]`, representa y.

O método `search` recebe dois argumentos: uma expressão regular e uma cadeia de caracteres de entrada. Como resultado, ele produz uma estrutura `m`, por exemplo, que pode referenciar `None` (representando uma situação absurda) ou a lista de partes da entrada descritas pelos grupos da expressão regular:

```
# CENÁRIO de implementação com expressões regulares
import re
entrada = 'Pagamento 2-8 em dia!'
m = re.search(r'([1-4])-([7-9])', entrada)
print (m.group(1)) # <-- (A)
print (m.group(2)) # <-- (B)
```

Enunciado

1. Escreva o resultado mostrado pela máquina Python no ponto `(A)`.
2. Escreva o resultado mostrado pela máquina Python no ponto `(B)` se `entrada = 'Data: (3-7)'`
3. Escreva o resultado mostrado pela máquina Python no ponto `(A)` se `entrada = 'Parcelas com erro= 0-9'`.
4. Escreva o resultado mostrado pela máquina Python no ponto `(A)` se `entrada = 'Erro quando 1-'`.
5. Modifique a expressão regular de modo que $(x, y) \in (1..5) \times (1..5)$.
6. De acordo com o item anterior, escreva o resultado mostrado pela máquina Python nos pontos `(A)` e `(B)` se `entrada = 'Disparo de torpedo em 2-4'`.
7. Modifique a expressão regular de modo a ser possível extrair $(x, y) \in (1..5) \times (1..5)$ de entradas como `entrada = 'Disparo de torpedo em (2,4)'`.
8. De acordo com o item anterior, escreva o resultado mostrado pela máquina Python nos pontos `(A)` e `(B)` se `entrada = 'Disparo em (3,5)'`.

Referências

ASCII Table. (2020, fevereiro). Wikipedia, the free encyclopedia; eletronic. Recuperado de https://en.wikipedia.org/wiki/ASCII