Estratégias de invariantes para listas

Mais da entrada

Dada uma lista, o algoritmo pode fazer o tratamento lidando com um elemento de cada vez.

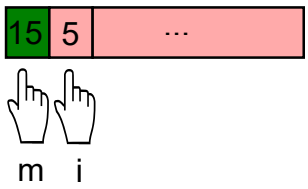
Medida de progresso: A quantidade da entrada considerada até o momento.

Invariante: Considerando a porção da entrada que foi tratada, se ela for a entrada completa, temos a solução completa.

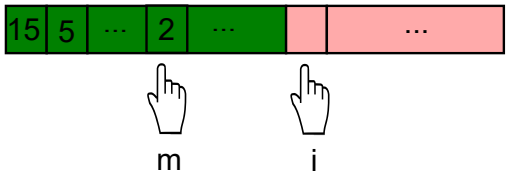
Exemplo: Encontrar o índice do menor elemento da lista

Algoritmo dos dois dedos: um dedo indica o próximo elemento a ser analisado, outro dedo indica o menor encontrado até o momento.

- O dedo que indica o menor elemento até o momento é representado por `m`.
- O dedo que indica o próximo elemento a ser tratado é representado por `i`.
- Começamos com o dedo `m` apontando à primeira posição da lista e o dedo `i` apontando à segunda posição.



Invariante: Depois de `i` iterações, o dedo esquerdo aponta ao maior elemento da lista `a[:i]` sdaf



In [ ]:

```
def indiceDoMenor(a):
    m = 0
    for i in range(1, len(a)):
        if a[i] < a[m]:
            m = i
    return m

indiceDoMenor([2,4,1,9,0,10])
```

Mais da saída:

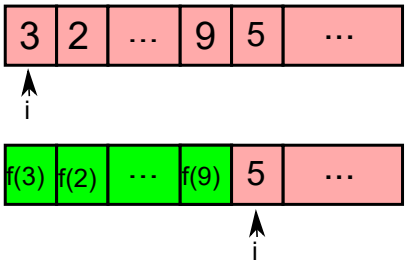
Se a solução é uma lista, uma forma natural é tentar construir a solução um elemento por vez.

Medida de progresso: A quantidade da saída construída.

Invariante: A saída construída até o momento é correta.

Exemplo: mapear uma função a todos os elementos de uma array

Invariante: Todos os elementos até a posição `i-1` estão mapeados (aplicados a função de entrada)



In [3]:

```
def mapear(f, a):
    for i in range(len(a)):
    # invariante: para todo 0<=k<i, a[k] == f(valorAnterior(a[k]))
        a[i] = f(a[i])

xs = ['1','2','3']
mapear(int, xs)
xs
```

Out[3]:

```
[1, 2, 3]
```

Trabalho feito:

A medida de progresso pode ser dada por uma função mais criativa que representa o trabalho feito até o momento.

Exemplo: Ordenação pelo método da bolha

Veremos depois.