Correteude (total)

↳ Garantir que algoritmo retorna resposta correta e que sempre termina de executar

Saber como provar nos ajuda a desenvolver

↳ projetar algoritmo com prova na cabeça

↳ evidencia erros.

Prova formal (matemática) que algoritmo iterativo funciona corretamente: corretude de duas estruturas de repetições

Invariantes de laço: Afirmações verdadeiras no início e no fim de cada iteração de um laço.

Provar a invariante

1) Inicialização: A invariante é válida antes da primeira iteração

2) Manutenção: Supondo que é válida no início da iteração prova que a informação continua verdadeira no fim da mesma iteração

3) Conclusão: Quando o laço termina, a invariante fornece uma propriedade útil que ajuda a provar que o algoritmo é correto.

```
InsertionSort(A[1..n])
  Para i ← 1 até n
    j ← i
    Enquanto j > 1 e A[j] < A[j-1]
      A[j] ↔ A[j-1]
      j ← j - 1
```

Invariante do laço interno:

Para $1 \leq k < k' \leq i$, se $k' \neq j$, então $A[k] < A[k']$.

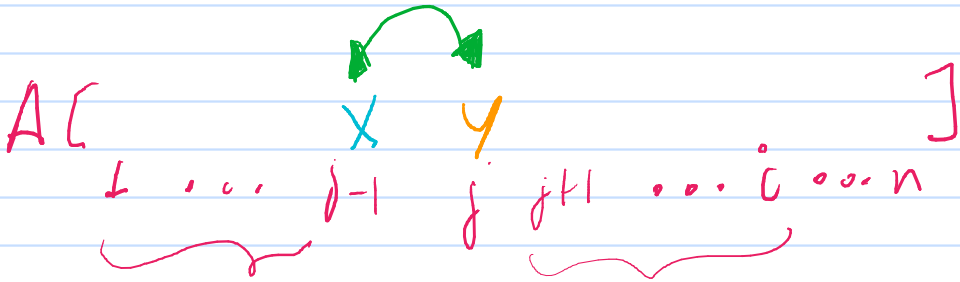
Prova:

1) Inicialização:
   - Recebe vetor ordenado até $i-1$

Como $j = i$, a invariante é válida.

2) Manutenção:
   - Seja x o elemento da posição $A[j-1]$ e y o elemento da pos. j.

Como a invariante é válida antes da iteração do laço, x é maior que todos os elementos que vem antes dele no vetor, e menor que os elementos em $A[j+1..i]$.



Observe que os elementos de $A[1..j-2]$ e $A[j+1..i]$ não mudam de posição dentro do laço. Então a invariante continua válida para k, k' nessas posições.

Dentro do laço, ocorrece apenas a troca de x com o elemento da posição $j(y)$, e portanto x continua sendo maior que todos os elementos que vem antes dele (a troca só ocorrece se $x > y$) e menor que os elementos de $A[j+1..i]$.

Observe que a nova posição de y é $j-1$. Mas, depois da troca, $j \leftarrow j-1$, e, portanto, o elemento da posição j continua sendo o único em $A[1..i]$ que está possivelmente fora de ordem. ▨

Término: Em cada iteração, j é recorrentemente decrementado e em algum momento atingirá valor 1 ou para antes ($A[j] \geq A[j-1]$).

```
InsertionSort(A[1..n])
  Para i ← 1 até n
    j ← i
    Enquanto j > 1 e A[j] < A[j-1]
      A[j] ↔ A[j-1]
      j ← j - 1
```

* Quando o laço interno termina, temos uma propriedade, que vai ajudar na corretude do algoritmo, que passa a ser válida. Qual seria ela?

Conclusão do laço interno:

Existem duas possibilidades para o término do laço:

i) $j = 1$.

Pela invariante, o subvetor $A[1 \ldots i]$ está completamente ordenado.

ii) $A[j] \geq A[j-1]$

Todos os outros elementos estavam ordenados entre si e, se $A[j] \geq A[j-1]$,

Então A[j] é maior que todos os outros elementos anteriores a ele no subvetor. Como A[j] é menor que os elementos que vem depois dele em A[1..i], então A[j] está na posição correta relativa à ordenação de A[1..i], e A[1..i] está ordenado.

```
InsertionSort(A[1..n])
1 Para i ← 1 até n
2     j ← i
3     Enquanto j > 1 e A[j] < A[j-1]
4         A[j] ↔ A[j-1]
5         j ← j - 1
```

Invariante do laço "Para":
A[1..i-1] está ordenado

Prova da Invariante:

Inicialização: O vetor vazio está trivialmente ordenado.

Manutenção: Seja uma iteração em que $i = x$. Agora suponha que a invariante é válida antes da iteração, temos $A[1..x-1]$ ordenado. Pela corretude do laço interno, em seu término, $A[1..x]$ estará ordenado. Quando i é incrementado para $x+1$, verificamos então que a invariante continua válida. Pois $A[1..i-1] = A[1..x]$, que ordenamos pela corretude do laço interno.

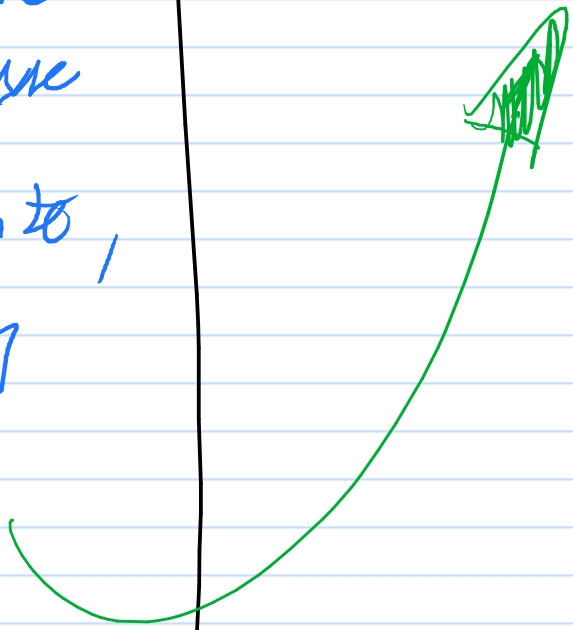
Término do laço externo:

Laço do tipo Para.

Conclusão do laço externo:

O laço para de executar quando $i = n+1$. Nesse momento, pela invariante, temos que $A[1..n]$ está ordenado.

Teorema: O algoritmo Insertion Sort ordena corretamente um vetor de n elementos.



```
InsertionSort(A[1..n])
1 Para i ← 1 até n
2     j ← i
3     Enquanto j > 1 e A[j] < A[j-1]
4         A[j] ↔ A[j-1]
5         j ← j - 1
```

Invariante: $A[1..n]$ é uma permutação dos seus valores originais.

Inicialização: Antes da execução de qualquer iteração do laço externo, temos que $A[1..n]$ é uma perm. dos seus valores originais.

Na linha 2, é feita apenas uma atribuição, portanto essa propriedade não é alterada.

Manutenção: Suponha que, antes de uma certa iteração j, a invariante é válida. Na linha 4, é realizada uma troca*, de forma que os elementos envolvidos permanecem no vetor, mas em posições diferentes. Nenhuma outra alteração é feita.

* A troca é sempre feita entre elementos de posições válidas do vetor, pelos valores que ele assume durante o laço.