



UNIVERSIDADE ESTADUAL DA PARAÍBA
CENTRO DE CIÊNCIAS E TECNOLOGIA
CIÊNCIA DA COMPUTAÇÃO

LUCAS DE LUCENA SIQUEIRA
DANIEL XAVIER BRITO DE ARAUJO

Programação Lógica: Insertion Sort com Prolog

CAMPINA GRANDE
2022

1. Introdução	3
2. Código em Prolog	3
3. Explicação do Código	4
4. Solução Proposta pelo Código	4
5. Código alternativo em Python	5

1. Introdução

O *Insertion Sort* se trata de um algoritmo de ordenação por comparação *stable* que irá iterar os números pelos índices do array, tendo um custo de custo $\theta(n)$ para o melhor caso e $\theta(n^2)$ para caso médio e pior caso. Assim como todos os algoritmos de ordenação, o *Insertion Sort* tem por objetivo ordenar de forma crescente ou decrescente uma lista de números não ordenada. A seguir, será demonstrada a construção e o funcionamento do algoritmo implementado em um paradigma de programação lógica utilizando o *Prolog* para isso.

2. Código em Prolog

```
% Recebe a lista a ser ordenada em "[H|List]" e armazena o output em "Result"
```

```
% H -> Height, List -> Lista de números
```

```
insertion_sort([H|List], Result) :-
```

```
    insertion_sort(List, Temp), % Produz uma lista temporária em "Temp"
```

```
    print_list(Temp), % Imprime a lista temporária
```

```
    insert_item(H, Temp, Result). % Insere o elemento da posição H da lista
```

```
insertion_sort([], []). % Caso a lista dada for vazia, o output vai ser vazio
```

```
% Recebe um elemento "X" que será inserido na lista "[H|List]"
```

```
% "H" -> Height da lista
```

```
insert_item(X, [H|List], [H|Result]) :-
```

```
    H < X, !, % Caso H seja menor que X
```

```
    insert_item(X, List, Result). % X será inserido em "List" para reduzir o "Result"
```

```
insert_item(X, List, [X|List]). % Caso X seja maior que H, X será o primeiro  
elemento da lista
```

```
% Caso a lista esteja vazia, imprime uma linha em branco
```

```
print_list([]) :- nl.
```

```
% Caso a lista não esteja vazia, vai imprimir, recursivamente, cada elemento da lista
```

```
print_list([X|List]) :-
```

```
    write(X), write(' '),
```

```
    print_list(List).
```

```

1 % Recebe a lista a ser ordenada em "[H|List]" e armazena o output em "Result"
2 % H -> Height, List -> Lista de números
3 insertion_sort([H|List], Result) :-
4     insertion_sort(List, Temp), % Produz uma lista temporária em "Temp"
5     print_list(Temp), % Imprime a lista temporária
6     insert_item(H, Temp, Result). % Insere o elemento da posição H da lista
7
8 insertion_sort([], []). % Caso a lista dada for vazia, o output vai ser vazio
9
10 % Recebe um elemento "X" que será inserido na lista "[H|List]"
11 % "H" -> Height da lista
12 insert_item(X, [H|List], [H|Result]) :-
13     H < X, !, % Caso H seja menor que X
14     insert_item(X, List, Result). % X será inserido em "List" para reduzir o "Result"
15 insert_item(X, List, [X|List]). % Caso X seja maior que H, X será o primeiro elemento da
16
17 % Caso a lista esteja vazia, imprime uma linha em branco
18 print_list([]) :- nl.
19
20 % Caso a lista não esteja vazia, vai imprimir, recursivamente, cada elemento da lista
21 print_list([X|List]) :-
22     write(X), write(' '),
23     print_list(List).

```

Imagem 1 - Insertion Sort em Prolog

3. Explicação do Código

São definidos alguns predicados para que a ordenação possa ser feita, dentre eles o predicado “insertion_sort” que foi definido duas vezes, sendo a primeira para cobrir o caso da lista inserida não ser vazia, e o segundo para caso a lista inserida seja vazia. No primeiro predicado definido como “insertion_sort”, são estabelecidos dois parâmetros, o primeiro referente à lista que será ordenada e o segundo referente à a lista do output.

Além dos predicados citados, foram definidos predicados para a inserção dos itens, chamados de “insert_item”, que recebe um elemento que passará por uma estrutura de decisão que definirá como será a ordenação do elemento inserido.

Por fim, existe mais um último tipo de predicado definido, sendo ele o “print_list”, que tem uma simples função de printar uma lista vazia caso seja inserida uma lista vazia para ordenação, ou caso seja inserida uma lista com números, cada elemento da lista será exibido na mesma linha depois de cada passo da ordenação.

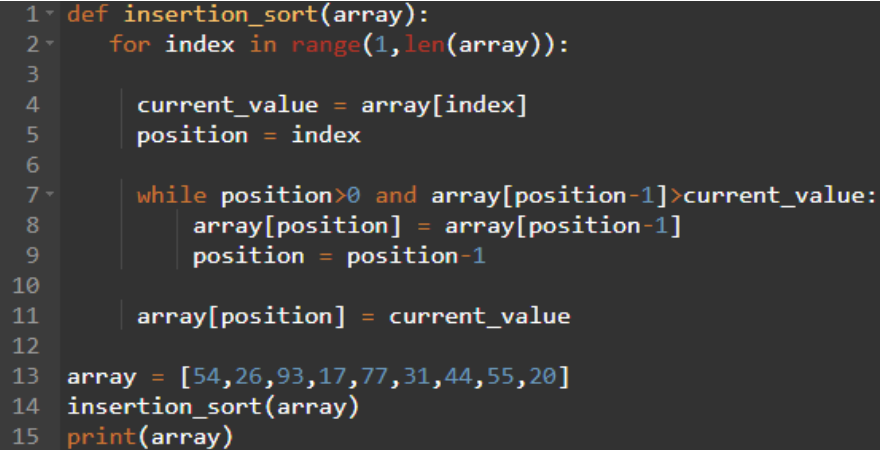
4. Solução Proposta pelo Código

O código apresentado tem por objetivo ordenar uma lista de números que será inserida através de uma query, apesar de não ser o algoritmo de ordenação com

menor custo e maior eficiência já desenvolvido, é um bom algoritmo para compreender a estruturação de dados e de decisões na programação lógica.

5. Código alternativo em Python

```
def insertion_sort(array):  
    for index in range(1,len(array)):  
  
        current_value = array[index]  
        position = index  
  
        while position>0 and array[position-1]>current_value:  
            array[position] = array[position-1]  
            position = position-1  
  
        array[position] = current_value  
  
array = [54,26,93,17,77,31,44,55,20]  
insertion_sort(array)  
print(array)
```



```
1 def insertion_sort(array):  
2     for index in range(1,len(array)):  
3  
4         current_value = array[index]  
5         position = index  
6  
7         while position>0 and array[position-1]>current_value:  
8             array[position] = array[position-1]  
9             position = position-1  
10  
11         array[position] = current_value  
12  
13 array = [54,26,93,17,77,31,44,55,20]  
14 insertion_sort(array)  
15 print(array)
```

Imagem 2 - Insertion Sort em Python