Sistema de Restaurante

Bruna Floro, Davi Góes, Emily Oliveira, Gabriel Macedo, Lizandra Soares.

[1]Faculdade de Excelência (UNEX)
Praça José Bastos, 55 –Itabuna – BA,45600-080– Brazil

[2]Data Structure
Lucas Almeida

[3]Estrutura de Dados
Universidade de Excelência (UNEX) – Feira de Santana, BA – Brazil

{ bruna.floro@aluno.unex.edu.br,emilly.santana@ftc.edu.br,
gabrielmacedop98@hotmail.com,lizandrasoaress@outlook.com

Abstract. *This report presents the application of the Insertion Sort algorithm in the development of a restaurant management system implemented in Python. The main objective is to demonstrate how the method contributes to the efficient organization of orders and items, especially in small or partially ordered lists, typical of restaurant management systems. The work describes fundamental concepts of the algorithm, its operating logic, and its integration into the system code, illustrating with examples, pseudocode, and tests performed. The analysis shows that, although not the most efficient algorithm for large volumes of data, Insertion Sort proves to be adequate and effective in the proposed context.*

Resumo. *Este relatório apresenta a aplicação do algoritmo de ordenação Insertion Sort no desenvolvimento de um sistema de restaurante implementado em Python. O objetivo principal é demonstrar como o método contribui para a organização eficiente de pedidos e itens, especialmente em listas pequenas ou parcialmente ordenadas, típico de sistemas de gerenciamento de restaurantes. O trabalho descreve conceitos fundamentais do algoritmo, sua lógica de funcionamento e sua integração no código do sistema, ilustrando com exemplos, pseudocódigos e testes realizados. A análise evidencia que, apesar de não ser o algoritmo mais eficiente para grandes volumes de dados, o Insertion Sort se mostra adequado e eficaz no contexto proposto.*

1. Introdução

Insertion Sort, ou ordenação por inserção, é o terceiro e último método de ordenação simples de vetores. Esse método realiza a comparação a partir do segundo elemento do vetor com os elementos das posições anteriores, para inseri-lo na posição correta até aquele momento (FORBELLONE, 2005). Para inserir determinado elemento na posição correta é preciso reposicionar os

demais elementos do vetor daquela posição em diante. Isso faz o método Insertion Sort não ser eficiente com vetores grandes, assim como os métodos apresentados anteriormente (SCHILDT, 1997). No entanto, quando um vetor se encontra em ordem pré-classificada ou até classificada, sua eficiência é bem superior, proporcionando bons resultados em termos de desempenho, pois realiza poucas comparações e quase nenhuma troca, dependendo da distribuição dos dados (MATTOS;LORENZI;CARVALHO, 2007).

2. Fundamentação Teórica

A ideia da ordenação por inserção é dividir os elementos em duas subestruturas, uma com os elementos já ordenados e outra com elementos ainda por ordenar. O método de ordenação por inserção funciona de forma semelhante à maneira como organizamos as cartas em nossa mão em um jogo de cartas.

Partimos do pressuposto de que a primeira carta já está ordenada e, em seguida, selecionamos uma carta não ordenada. Se a carta não ordenada for maior que a carta na mão, ela é colocada à direita; caso contrário, à esquerda. Da mesma forma, outras cartas não ordenadas são retiradas e colocadas em seus respectivos lugares.

Uma abordagem semelhante é utilizada pelo algoritmo de ordenação por inserção.

Primeiramente, defina a primeira posição da lista como ordenada.

- Em seguida, pegue o primeiro elemento da lista não ordenada e coloque-o na posição correta na lista ordenada.
- Por fim, repita o passo 2 até que todos os elementos da lista não ordenada sejam inseridos na lista ordenada.

O que o algoritmo de ordenação por inserção faz, é percorrer cada elemento de uma lista e ir comparando-os com os elementos já ordenados à esquerda. Dessa forma, para cada iteração, ele pega o próximo elemento, compara com os elementos da esquerda até encontrar a posição correta, insere-o nesta posição e desloca os elementos maiores para a direita. Ao final, ou seja, quando o algoritmo percorrer toda a lista, esta estará ordenada.

3. Metodologia

A implementação de ordenação por inserção no código de sistema de restaurante desenvolvido em python, mostra a relevância do algoritmo para manter a organização dos conjuntos de elementos de acordo com as posições depois de ficarem ordenadas.

Inicialmente, foram estudados os fundamentos teóricos do algoritmo e analisado seu funcionamento passo a passo, considerando sua aplicação em listas de pedidos e itens. Em seguida:

1. **Integração do algoritmo ao sistema:**
   O Insertion Sort foi inserido no módulo responsável pela organização de

pedidos, garantindo que novos elementos fossem posicionados adequadamente conforme sua numeração ou identificação.

2. **Testes com diferentes cenários:**
   Foram realizados testes utilizando listas de tamanhos variados, contendo números de pedidos e itens associados a clientes. Esses testes permitiram observar o comportamento do algoritmo ao inserir novos pedidos e reorganizar listas já existentes.

3. **Construção de pseudocódigos e exemplos ilustrativos:**
   Diagramas e pseudocódigos foram elaborados pela equipe para demonstrar visualmente a lógica de ordenação, permitindo melhor compreensão do processo.

A metodologia focou em validar a aplicabilidade do algoritmo dentro de um ambiente computacional real, assegurando clareza, organização e confiabilidade no funcionamento do sistema.

4. Figuras e Legendas

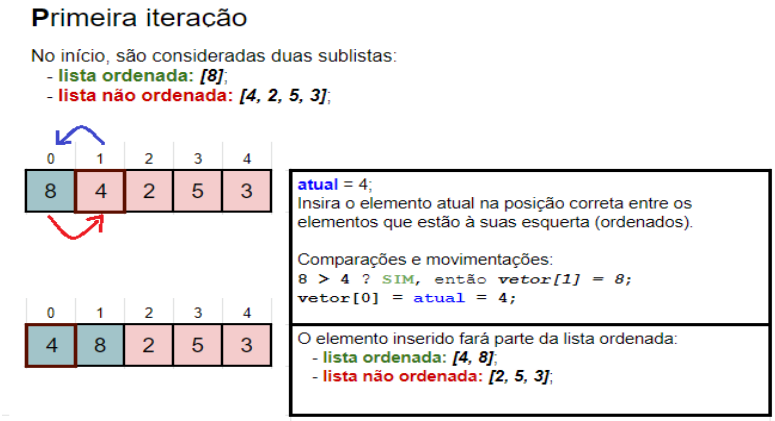

Figure 1. Exemplo de uma etapa de inserção.

```
Lista original: [64, 34, 25, 12, 22, 11, 90]
Lista ordenada: [11, 12, 22, 25, 34, 64, 90]

Pedidos originais: [3, 1, 2]
Pedidos ordenados: [1, 2, 3]
```

Figure 2. Testes com números e números de pedidos.

```python
# Criar uma cópia para não modificar a lista original
sorted_arr = arr.copy()

# Percorrer da posição 1 até o final
for i in range(1, len(sorted_arr)):
    current = sorted_arr[i]
    current_key = key(current)
    j = i - 1

    # Mover elementos maiores que a chave atual para a direita
    while j >= 0 and key(sorted_arr[j]) > current_key:
        sorted_arr[j + 1] = sorted_arr[j]
        j -= 1

    # Inserir o elemento na posição correta
    sorted_arr[j + 1] = current

return sorted_arr
```

Figure 3. Pseudocódigo Ordenação por Inserção. Desenvolvido pela equipe.

```python
# Ordenar pedidos usando Insertion Sort
sorted_orders = insertion_sort(self.orders, key=lambda x: x['order_number'])

for order in sorted_orders:
    print(f"\nPedido #{order['order_number']}")
    print(f"Cliente: {order['cliente']}")
    print(f"Status: {order['status']}")
    print(f"Total: R${order['total']:.2f}")
    print("Itens:")
    for item in order['items']:
        print(f"  - {item['nome']} (x{item['quantidade']}) - R${item['preco'] * item['quantidade']:.2f}")
```

Figure 3.1. Pseudocódigo Ordenação de Pedidos.

```
--- Pedidos (Ordenados por Número) ---

Pedido #1
Cliente: Emily
Status: aceito
Total: R$226.00
Itens:
  - Croquete de cupim (x1) - R$38.00
  - Lasanha della nonna (x1) - R$99.00
  - Petit Gateau (x1) - R$42.00
  - Negroni (x1) - R$47.00

Pedido #2
Cliente: Alec
Status: aceito
Total: R$287.00
Itens:
  - Paleta de Cordeiro (x1) - R$129.00
  - Risotto de limão siciliano com Porchetta (x1) - R$69.00
  - Cannoli ao Caramelo e Nutela (x2) - R$72.00
  - Heineken (x1) - R$17.00
```

Figure 4. Ordenação dos pedidos por números de clientes após serem feitos.

```
Pedido #5
Cliente: Maria Hellena
Status: pendente
Total: R$479.00
Itens:
  - Bruschetta (x2) - R$70.00
  - Paleta de Cordeiro (x2) - R$258.00
  - Bolo de aipim/Sorvete  (x2) - R$76.00
  - Amstel Ultra (x2) - R$28.00
  - Negroni (x1) - R$47.00
```

Figure 4.1. Ordenação dos itens da cliente Maria Hellena ainda pendente. Contendo 5 itens.

```
Pedido #5 encontrado!
Cliente: Maria Hellena
Status: aceito
Total: R$621.00
Itens:
  - Bruschetta (x2) - R$70.00
  - Paleta de Cordeiro (x2) - R$258.00
  - Bolo de aipim/Sorvete  (x2) - R$76.00
  - Amstel Ultra (x2) - R$28.00
  - Negroni (x1) - R$47.00
  - Lulas/Linguine/Pesto (x2) - R$142.00
```

Figure 4.2. Após o pedido ser aceito a cliente decidiu adicionar mais um item.

4. Considerações Finais

A utilização do algoritmo Insertion Sort no sistema de restaurante demonstrou ser uma solução adequada para a organização e manipulação de listas de pedidos em tempo real. Apesar de possuir limitações quando aplicado a grandes conjuntos de dados, o algoritmo apresenta excelente desempenho em listas pequenas ou parcialmente ordenadas — características comuns no contexto de um restaurante.

A implementação permitiu um controle mais preciso da ordem dos pedidos e maior eficiência no fluxo de atendimento. Além disso, o estudo reforçou conceitos importantes de estrutura de dados e evidenciou a utilidade de algoritmos clássicos em situações práticas do cotidiano. O trabalho destaca a relevância do Insertion Sort para cenários onde simplicidade, clareza e boa performance em listas pequenas são suficientes para atender às necessidades do sistema.

Referências

https://ww2.inf.ufg.br/~hebert/disc/aed1/AED1_04_ordenacao1.pdf(slide 09)

https://akiradev.netlify.app/posts/algoritmo-insertion-sort/

https://www.programiz.com/dsa/insertion-sort

https://www.estrategiaconcursos.com.br/blog/algoritmo-ordenacao-insercao/

https://integrada.minhabiblioteca.com.br/reader/books/9788595023932/pageid/37

Código desenvolvido pela equipe.