UNIVERSIDAD TECNOLÓGICA NACIONAL
TECNICATURA UNIVERSITARIA EN PROGRAMACIÓN
PROGRAMACIÓN I

Trabajo Práctico Integrador

Algoritmos de Búsqueda y Ordenamiento en Python

👨‍🏫 Profesor Coordinador: Alberto Cortez

👩‍🏫 Profesora comisión 1: Cinthia Rigoni

👩‍🎓 Tutor: Martín A. García

Estudiantes:

* Hernán E. Bula - hernanbula@gmail.com

* Rodrigo Arias - roarias299@gmail.com

Algoritmos de

Búsqueda

Permiten localizar un elemento específico dentro de un conjunto de datos. El tamaño del conjunto de datos impacta significativamente el tiempo de búsqueda. Su eficiencia se mide O(n).

Lineal / Binaria / Interpolación / Hash

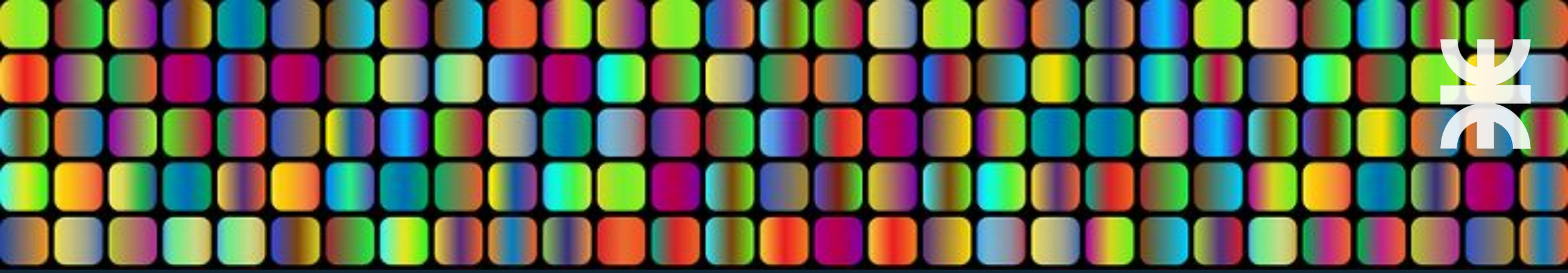
Ordenamiento

Permiten organizar datos según un criterio y estructurarlos de manera eficiente. Posibilitan una búsqueda mucho más eficiente, especialmente utilizando la búsqueda binaria en datos ya ordenados.

Bubble / Insertion / Selection / Quick Sort

Importante

Los algoritmos de búsqueda y ordenamiento son pilares fundamentales en el desarrollo de software, especialmente en la gestión eficiente de datos. La elección del algoritmo de búsqueda y ordenamiento adecuado es crucial y depende del contexto, el tamaño y el estado de los datos (ordenados o no).



Estructuras de datos: listas simples y anidadas

Descripción

Para gestionar grandes volúmenes de información relacionada, se utilizan estructuras de datos (Cimino, 2024)

Arrays o listas

Permiten almacenar múltiples elementos bajo una misma variable. El acceso a los elementos es inmediato mediante índices numéricos, comenzando en 0.

Listas anidadas

Estructuras complejas, como lista de listas o tablas, donde cada sublista funciona como una fila con múltiples atributos. Para acceder a sus datos se utilizan índices múltiples, uno por cada nivel de anidación.

Caso práctico:
Administración de lista de productos y precios

Problema

Se dispone de una lista de productos de supermercado estructurada como una lista anidada, donde cada sub-lista representa un artículo (ID, producto y precio). Debe desarrollarse un sistema que permita gestionar, buscar y ordenar productos.

Objetivo

Desarrollar un sistema interactivo que permita:

- Gestionar artículos CRUD (crear, leer, modificar, eliminar).
- Buscar artículos por ID, nombre o precio.
- Ordenar la lista por distintos criterios (ascendente/descendente).

Metodología

1

Investigación

Revisión apuntes, videos y documentación Python.

2

Proyecto

Elección del tema, definición del problema, planteo del objetivo, desarrollo del abordaje y propuesta del enfoque.

3

Desarrollo Iterativo

Codificado y prototipado de funciones básicas (CRUD).
Integración de algoritmos de búsqueda y ordenamiento.

4

Validación

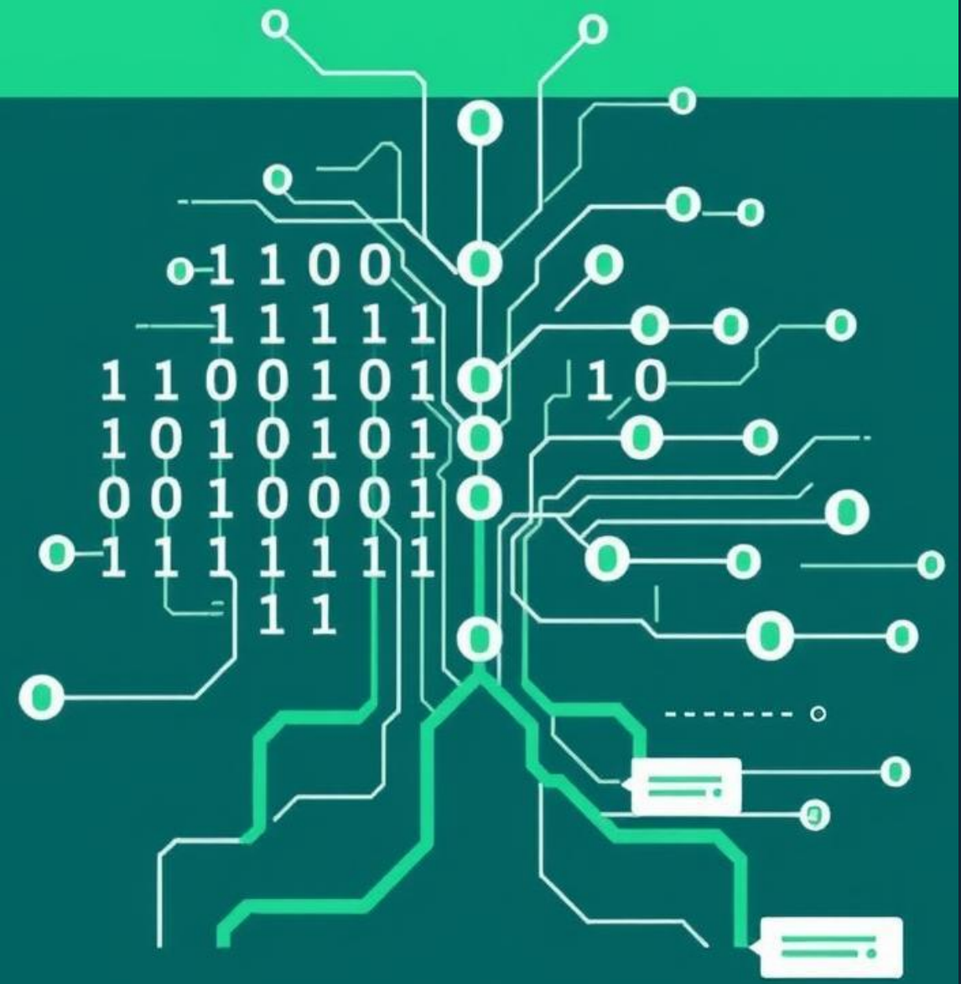
Pruebas manuales con casos límite (ej: búsquedas fallidas, validación de datos, recursividad, eficiencia).

```python
# Función principal para ejecutar funciones del menú de la aplicación y realizar acciones sobre la lista según la opción seleccionada. Controla el flujo principal de la aplicación.

def manipular_lista(opcion):
    match opcion: # Switch para que el usuario elija la opción:
        case "A": # Imprime lista de precios
            imprimir_lista(lista_precios)
            manipular_lista(menu())
        case "B": # Agrega artículo
            agregar_articulo()
            imprimir_lista(lista_precios)
            manipular_lista(menu())
        case "C": # Elimina artículo
            imprimir_lista(lista_precios)
            eliminar_articulo()
            manipular_lista(menu())
        case "D": # Modifica artículo existente
            imprimir_lista(lista_precios)
            modificar_articulo()
            manipular_lista(menu())
        case "E": # Busca un artículo
            buscar_articulo()
            manipular_lista(menu())
        case "F": # Ordena lista por elemento: id_prod, articulo, precio
            # Buscar y elegir el mejor método de ordenamiento para lista de listas
            ordenar_lista()
            manipular_lista(menu())
        case _:
            print("Saliste de la aplicación.")
```

```python
# Función para buscar un artículo de la lista, solicitando el dato al usuario para la búsqueda (id, nombre o precio)

def buscar_articulo():
    print("\nIngrese la opción que necesite\n 1: para buscar por ID del producto\n 2: para buscar por nombre del producto\n 3:
    para buscar por precio.\n")
    opcion = leer_opcion_menu()
    match opcion:
      case 1:
          id_prod = leer_numero_validado("\nIngrese ID del producto", 1, (len(lista_precios)))
          imprimir_producto(lista_precios[id_prod-1])
      case 2:
          prod = solicitar_dato("nombre del producto") # Falta mejorar la función para validar dato (upper, solo letra, etc.)
          encontrado = False # Bandera para saber si encuentra el producto o no
          for i in range(len(lista_precios)):
              if prod == lista_precios[i][1]: # Falta mejorar para que pueda encontrar coincidencias parciales
                  imprimir_producto(lista_precios[i])
                  encontrado = True
          if not encontrado:
              print(f"\n{"-"*65}\nNO existe un producto con el nombre:{prod}\n{"-"*65}")
              buscar_articulo()
      case 3:
          precio = leer_float_validado("\nIngrese precio del producto", 0)
          encontrado = False # Bandera para saber si encuentra el producto o no
          for i in range(len(lista_precios)):
              if precio == lista_precios[i][2]: # Falta mejorar para que pueda encontrar coincidencias entre un rango de precios
                  imprimir_producto(lista_precios[i])
                  encontrado = True
          if not encontrado:
              print(f"No exite un producto con el precio:{precio}")
      case _:
          pass
```

```python
# Función con algoritmo de ordenamiento usando el método Selection Sort adaptado a una lista (de productos) de listas.
# Para mejorarlo trabajamos con ayuda de IA Deepseek.

def ordenar_selection_sort(lista, indice_elemento, orden=False): # orden: Si True (mayor a menor), si False (menor a mayor).
    n = len(lista)
    for i in range(n):
        # Inicializamos la posición del elemento extremo (mínimo o máximo)
        extremo = i
        for j in range(i+1, n):
            # Comparamos según el orden deseado
            if orden:  # Orden descendente (mayor a menor)
                if lista[j][indice_elemento] > lista[extremo][indice_elemento]:
                    extremo = j
            else:  # Orden ascendente (menor a mayor)
                if lista[j][indice_elemento] < lista[extremo][indice_elemento]:
                    extremo = j
        # Intercambiamos los elementos
        lista[i], lista[extremo] = lista[extremo], lista[i]

    imprimir_lista(lista)
```

```
------------------------------------------------------------------
------------------------------------------------------------------

 Ingrese la opción de la acción que necesite realizar:

 A: imprimir lista de precios
 B: agregar artículo
 C: eliminar artículo
 D: modificar artículo
 E: buscar un artículo
 F: ordenar lista por elemento

 IMPORTANTE: Si ingresa cualquier otra tecla y enter, sale de la aplicación.


Ingrese opcion: a



------------------------------------------------------------------
ID    | PRODUCTO                                  |      PRECIO
------------------------------------------------------------------
1     | Aceite Natura - 900 cc                    | $     2090.50
2     | Tomate Arco en lata - 400 grs             | $     1190.00
3     | Mayonesa RI-K - 250 grs                   | $     1090.50
4     | Aceite Patito - 1500 cc                   | $     3590.50
5     | Azúcar Chango - 1 kg                      | $      990.90
6     | Ensalada frutas lata - 850 grs            | $     1930.00
7     | Arroz Gallo Oro - 1 kg                    | $     1850.00
8     | Fideos Matarazzo - 500 grs                | $     1680.50
9     | Leche entera La Serenísima - 1 lt         | $     2320.00
10    | Yogur bebible Ilolay - 1 lt               | $     2850.75
11    | Queso crema Casancrem - 290 grs           | $     3890.00
12    | Jabón líquido Ala - 750 ml                | $     6920.30
13    | Papel higiénico Elite - 4 rollos          | $     4120.00
14    | Café La Morenita - 500 grs                | $     9150.00
15    | Galletas Oreo - 117 grs                   | $     1500.00
16    | Chocolate Águila - 100 grs                | $     5010.00

¿Quiere ingresar al menu de opciones? (S/N):
```



Resultados obtenidos

Implementación exitosa del sistema

Desarrollo de las funcionalidades básicas: CRUD de artículos, búsqueda y ordenamiento por múltiples criterios.

Dificultades técnicas

Validación incompleta inputs no numéricos en algunas funciones. Limitación temporal para implementar manejo robusto de errores.

Algoritmo Selection Sort

Algoritmo de ordenamiento óptimo para el caso. Balance adecuado entre simplicidad y rendimiento para listas pequeñas. Evita complejidad de Quick Sort y mejor performance que Bubble Sort.

Mejoras a futuro

Optimización general del código. Mejora en validación de datos de entrada.
Comparativa de rendimiento entre algoritmos.

Conclusiones y Reflexiones finales

Integración exitosa

Integración de conocimientos teóricos-prácticos para resolver un problema concreto: estructuras de algoritmos y datos, modularidad, funciones, búsqueda y ordenamiento.

Búsqueda y ordenamiento

Reconocer el valor fundamental de estos algoritmos para la gestión eficiente de datos y desarrollo de software escalable.

Mejoras futuras identificadas

- ➔ Optimización del manejo de errores y validación de inputs
- ➔ Refactorización para mayor eficiencia y legibilidad del código
- ➔ Implementación de algoritmos más avanzados para comparativa de rendimiento

Bibliografía 📚

- Codecademy Team (2025, 31 de enero). *How to Sort Lists of Lists in Python with Examples.* https://codecademy.com/article/how-to-sort-lists-of-lists-in-python-with-examples
- González, S., & Benítez, M. (2025). *Algoritmos de Búsqueda y Ordenamiento. TP modelo.* TUPaD, UTN.
- Programación I (2025a). *Búsqueda y Ordenamiento en Programación* [Apunte de cátedra]. TUPaD, UTN.
- Programación I (2025b). *Implementación de Algoritmos de Búsqueda Binaria en Python.* TUPaD, UTN.
- Tecnicatura (2025a, 17 de febrero). *BÚSQUEDA* [Video]. TUPaD, UTN. https://youtu.be/gJlQTq80llg
- Tecnicatura (2025b, 17 de febrero). *ORDENAMIENTO* [Video]. TUPaD, UTN. https://youtu.be/xntUhrhtLaw
- Tecnicatura (2025c, 19 de mayo). *Introducción Busqueda y Ordenamiento- Integrador* [Video]. TUPaD, UTN. https://youtu.be/u1QuRbx-_x4

Recursos complementarios

- Charly Cimino (2024, 11 de abril). *ARRAYS en PROGRAMACIÓN. La base necesaria* [Video]. https://youtu.be/OHwl0RO4bac
- DeepSeek. https://chat.deepseek.com/
- Google for Education (2024a, 29 de agosto). *Clase de Python. Orden de Python.* https://developers.google.com/edu/python/sorting?hl=es-419
- Google for Education (2024b, 6 de noviembre). *Clase de Python. Listas de Python.* https://developers.google.com/edu/python/lists?hl=es-419
- La Geekipedia De Ernesto (2022, 21 de octubre). *Curso Python 3 desde cero #62, Listas anidadas* [Video]. https://youtu.be/8tXEia5ZcCE
- Python Software Foundation (2025). *documentación de Python - 3.13.3: Guía para principiantes: Listas.* https://docs.python.org/es/3.13/tutorial/introduction.html#lists