Actividad 1.2 Algoritmos de decrementa-y-venceras

Victor Misael Escalante Alvarado, A01741176

Ordenamiento Topológico

Foto del codigo

```python
def topological_sort(graph):
    """
    Ordena un grafo dirigido (sin ciclos) topológicamente.
    """
    # Inicialización
    in_degree = {node: 0 for node in graph}  # Grado de entrada de cada nodo
    for node in graph:
        for neighbor in graph[node]:
            in_degree[neighbor] += 1

    # Cola de nodos con grado de entrada cero
    queue = deque([node for node in graph if in_degree[node] == 0])
    top_order = []

    while queue:
        node = queue.popleft()
        top_order.append(node)

        for neighbor in graph[node]:
            in_degree[neighbor] -= 1
            if in_degree[neighbor] == 0:
                queue.append(neighbor)

    return top_order

# Ejemplo
graph = {
    'A': ['B', 'C'],
    'B': ['D'],
    'C': ['D'],
    'D': []
}
print(topological_sort(graph))  # Deberia salir: ['A', 'B', 'C', 'D']
```

Foto del resultado

```
(.venv) misa_v@MacBook-Pro-de-Victor-7 Algoritmos_ % "/Users/misa_v/Libra
Monterrey/Sem 5/Algoritmos_/Scripts/Python/Act1.2Decreaseandconquer.py"
['A', 'B', 'C', 'D']
```

Fake Coin

Foto del codigo

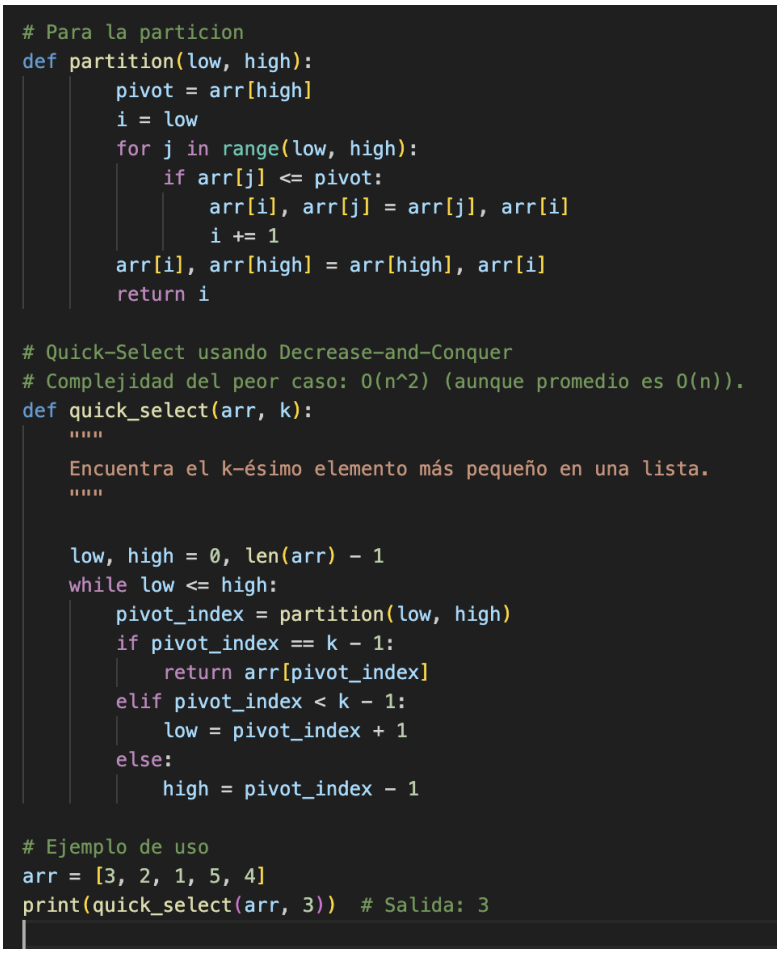
```python
# Algoritmo para Fake-Coin usando Decrease-and-Conquer
# Complejidad del peor caso: O(log n), donde n es el número de monedas.
def find_fake_coin(coins):
    """
    Encuentra la moneda falsa usando una balanza de comparación.
    """

    start, end = 0, len(coins) - 1
    while start < end:
        mid = (end - start + 1) // 2
        left = coins[start:start + mid]
        right = coins[start + mid:end + 1]

        result = compare(left, right)
        if result == -1:  # Izquierda es más ligera
            end = start + mid - 1
        elif result == 1:  # Derecha es más ligera
            start += mid
        else:  # Moneda falsa está en el lado sobrante
            return end

    return start

# Ejemplo de uso
coins = [1, 1, 1, 0.5, 1, 1, 1, 1, 1, 1, 1, 1, 1]
print(find_fake_coin(coins))  # Deberia salir: 3
```

Resultados

```
(.venv) misa_v@MacBook-Pro-de-Victor-7 Algoritmos_ % "/Users/misa_v/Libra
Monterrey/Sem 5/Algoritmos_/Scripts/Python/Act1.2Decreaseandconquer.py"
['A', 'B', 'C', 'D']
3
3
```

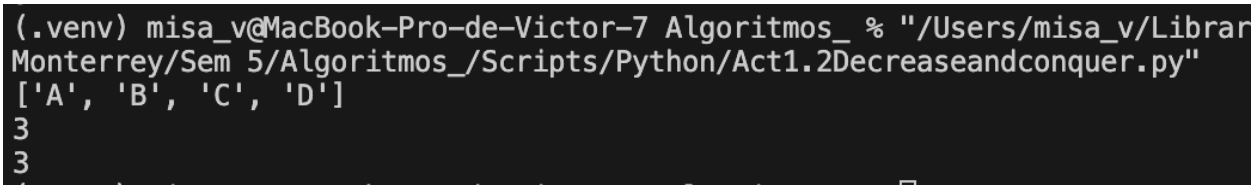
Quick select

Foto del codigo



```python
# Para la particion
def partition(low, high):
        pivot = arr[high]
        i = low
        for j in range(low, high):
            if arr[j] <= pivot:
                arr[i], arr[j] = arr[j], arr[i]
                i += 1
        arr[i], arr[high] = arr[high], arr[i]
        return i

# Quick-Select usando Decrease-and-Conquer
# Complejidad del peor caso: O(n^2) (aunque promedio es O(n)).
def quick_select(arr, k):
    """
    Encuentra el k-ésimo elemento más pequeño en una lista.
    """

    low, high = 0, len(arr) - 1
    while low <= high:
        pivot_index = partition(low, high)
        if pivot_index == k - 1:
            return arr[pivot_index]
        elif pivot_index < k - 1:
            low = pivot_index + 1
        else:
            high = pivot_index - 1

# Ejemplo de uso
arr = [3, 2, 1, 5, 4]
print(quick_select(arr, 3))  # Salida: 3
```

Resultados



```
(.venv) misa_v@MacBook-Pro-de-Victor-7 Algoritmos_ % "/Users/misa_v/Librar
Monterrey/Sem 5/Algoritmos_/Scripts/Python/Act1.2Decreaseandconquer.py"
['A', 'B', 'C', 'D']
3
3
```

Preguntas

¿Los algoritmos para generar todas las permutaciones y subconjuntos de n elementos (vistos en la clase anterior) son o no algoritmos decrease-and-conquer?

No, cuando buscamos generar todas las permutaciones se usan todas las posibilidades con lo que no terminan por decrementar el problema completo

¿Porqué si? ¿Porque no? Si tu respuesta es si, ¿qué factor de decremento tienen?

Entonces ¿estos algoritmos son también búsqueda exhaustiva o no?

No, en estos casos se evita hacer una busqueda exhaustiva gracias a que diferimos nuestra busqueda al momento de decrementar el espacio de busqueda para cada uno de nuestros algoritmos.

Enlace a codespaces:
https://codespaces.new/ElingeMisa/Algoritmos_?quickstart=1