UNMSM

Facultad de Ingeniería de Sistemas e Informática

2024

Verificación y Validación del Software

Práctica 4: Reporte Sonarlint de análisis estático

Alumno:

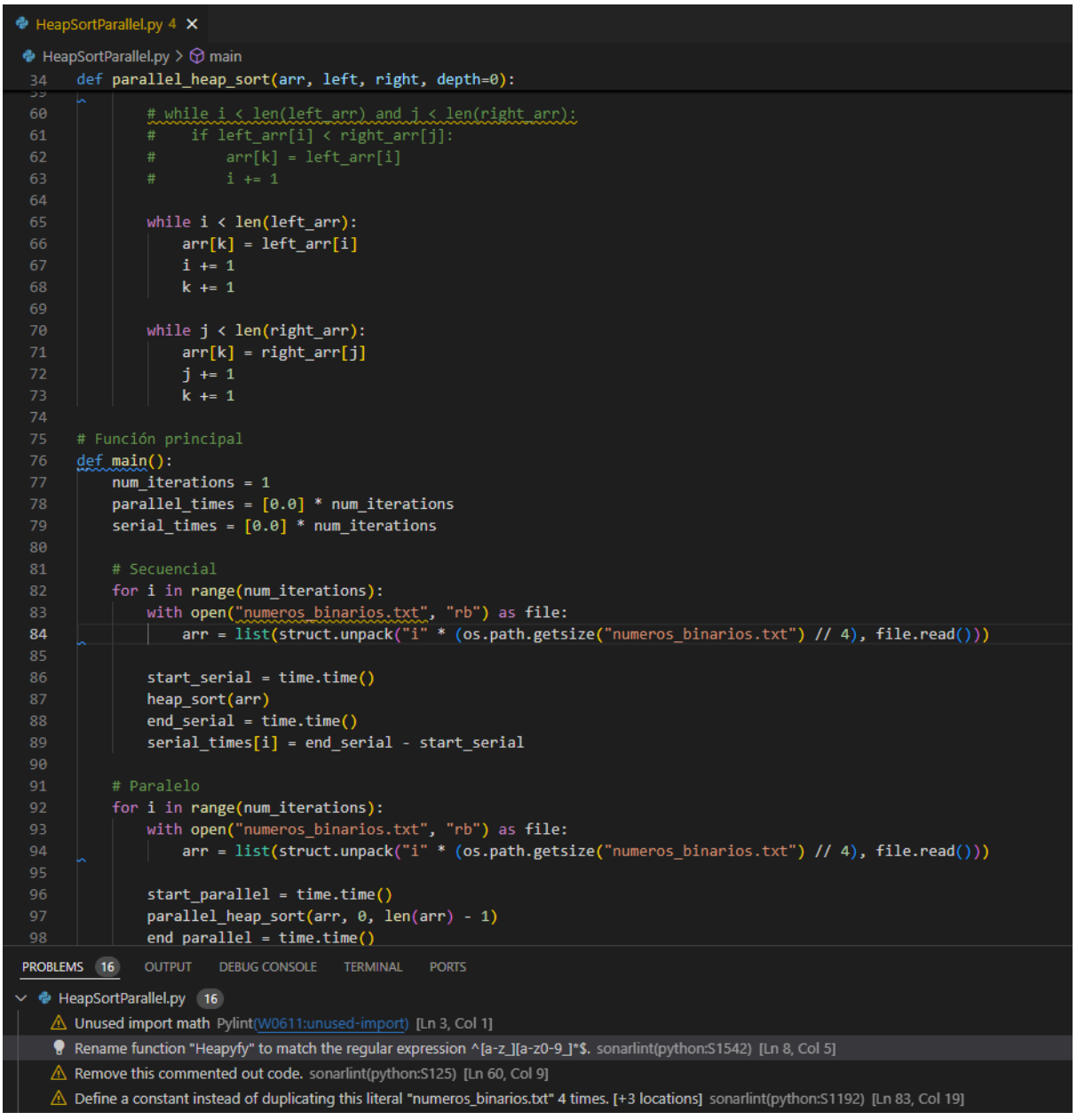
- Coronel Vilca, Brisa Valeria.

Práctica 4

1. Módulo

El presente módulo, desarrollado en el marco de un proyecto sobre programación paralela y concurrente, presenta un estudio comparativo del rendimiento del algoritmo Heapsort, tanto en su versión secuencial como paralela, al ordenar un conjunto de un millón de elementos. Se procede a la evaluación del código una vez implementadas todas las correcciones detalladas en el archivo “README"

Antes de la Refactorización



Después de la Refactorización

heap_sort_parallel.py

heap_sort_parallel.py > parallel_heap_sort

```python
# Función para realizar heap sort
def heap_sort(arr):
    n = len(arr)
    for i in range(n // 2 - 1, -1, -1):
        heapyfy(arr, n, i)

    for i in range(n - 1, 0, -1):
        arr[i], arr[0] = arr[0], arr[i]
        heapyfy(arr, i, 0)

# Función para hacer el heap sort en paralelo
def parallel_heap_sort(arr, left, right, depth=0):
    max_depth = 9
    if depth >= max_depth:
        heap_sort(arr[left:right + 1])
    else:
        mid = left + (right - left) // 2
        left_arr = arr[left:mid + 1]
        right_arr = arr[mid + 1:right + 1]

        left_thread = threading.Thread(
        target=parallel_heap_sort,
        args=(left_arr, 0, len(left_arr) - 1, depth + 1)
)
        left_thread.start()
        parallel_heap_sort(right_arr, 0, len(right_arr) - 1, depth + 1)

        left_thread.join()

        i = j = 0
        k = left
        while i < len(left_arr) and j < len(right_arr):
            if left_arr[i] < right_arr[j]:
                arr[k] = left_arr[i]
                i += 1
            else:
                arr[k] = right_arr[j]
                j += 1
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

No problems have been detected in the workspace.