UNMSM

Facultad de Ingeniería de Sistemas e Informática

2024

Verificación y Validación del Software

Práctica 4: Revisión de Código

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.

Caso 1

- **Violación (python:S1542):**
  Los nombres de funciones deben cumplir con una convención de nomenclatura.

  Issue: Consistencia

- **Fragmento de Código:**
  El nombre de la función no respeta la convención de nomenclatura.

```
# Función para aplicar heapify
def Heapify(arr, n, i):
```

  heap_sort_parallel.py

- **Corrección/Refactorización:**
  Según PEP 8 – Style Guide for Python Code, el nombre de la función debe estar en minúsculas.

```
# Función para aplicar heapify
def heapyfy(arr, n, i):
```

  heap_sort_parallel.py

Caso 2

- **Violación (python:S125):**
  Las secciones de código no deben comentarse.

  Issue: Intencionalidad

- **Fragmento de Código:**
  El código comentado distrae la atención del código ejecutado en sí.

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

heap_sort_parallel.py

- **Corrección/Refactorización:**
  El código comentado debe eliminarse.

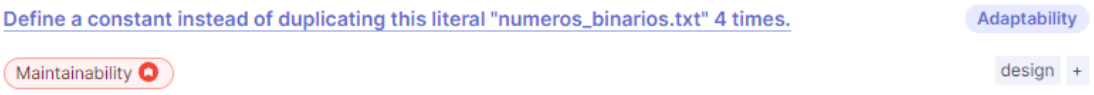
```
# Eliminado del código
```

heap_sort_parallel.py

Caso 3

- **Violación (python:S1192):**
  Las cadenas no deben duplicarse.

  Issue: Adaptabilidad

- **Fragmento de Código:**
  Las cadenas duplicadas hacen que el proceso de refactorización sea complejo y propenso a errores, ya que cualquier cambio debería propagarse en todas las ocurrencias.

```
# Secuencial
   for i in range(num_iterations):
      with open("numeros_binarios.txt", "rb") as file:
         arr = list(struct.unpack("i" * (os.path.getsize("numeros_binarios.txt") // 4),
file.read()))

 # Paralelo
   for i in range(num_iterations):
      with open("numeros_binarios.txt", "rb") as file:
         arr = list(struct.unpack("i" * (os.path.getsize("numeros_binarios.txt") // 4),
file.read()))
```

heap_sort_parallel.py

- **Corrección/Refactorización:**
  Utilice constantes para reemplazar las de cadenas duplicadas.

```
# Constante para el nombre del archivo
FILENAME = "numeros_binarios.txt"

# Secuencial
   for i in range(num_iterations):
      with open(FILENAME, "rb") as file:
         arr = list(struct.unpack("i" * (os.path.getsize(FILENAME) // 4), file.read()))
#Paralelo
 for i in range(num_iterations):
      with open(FILENAME, "rb") as file:
         arr = list(struct.unpack("i" * (os.path.getsize(FILENAME) // 4), file.read()))
```

heap_sort_parallel.py