UNIVERSIDAD CENTROAMERICANA
“JOSE SIMEON CAÑAS”

Integrantes:

Luis Andree Hernández Maltéz

Edson Enrique Palacios Quinteros

Oscar Ernesto Menjivar Ayala

Asignatura:

Análisis de algoritmos

Actividad:

Taller 01

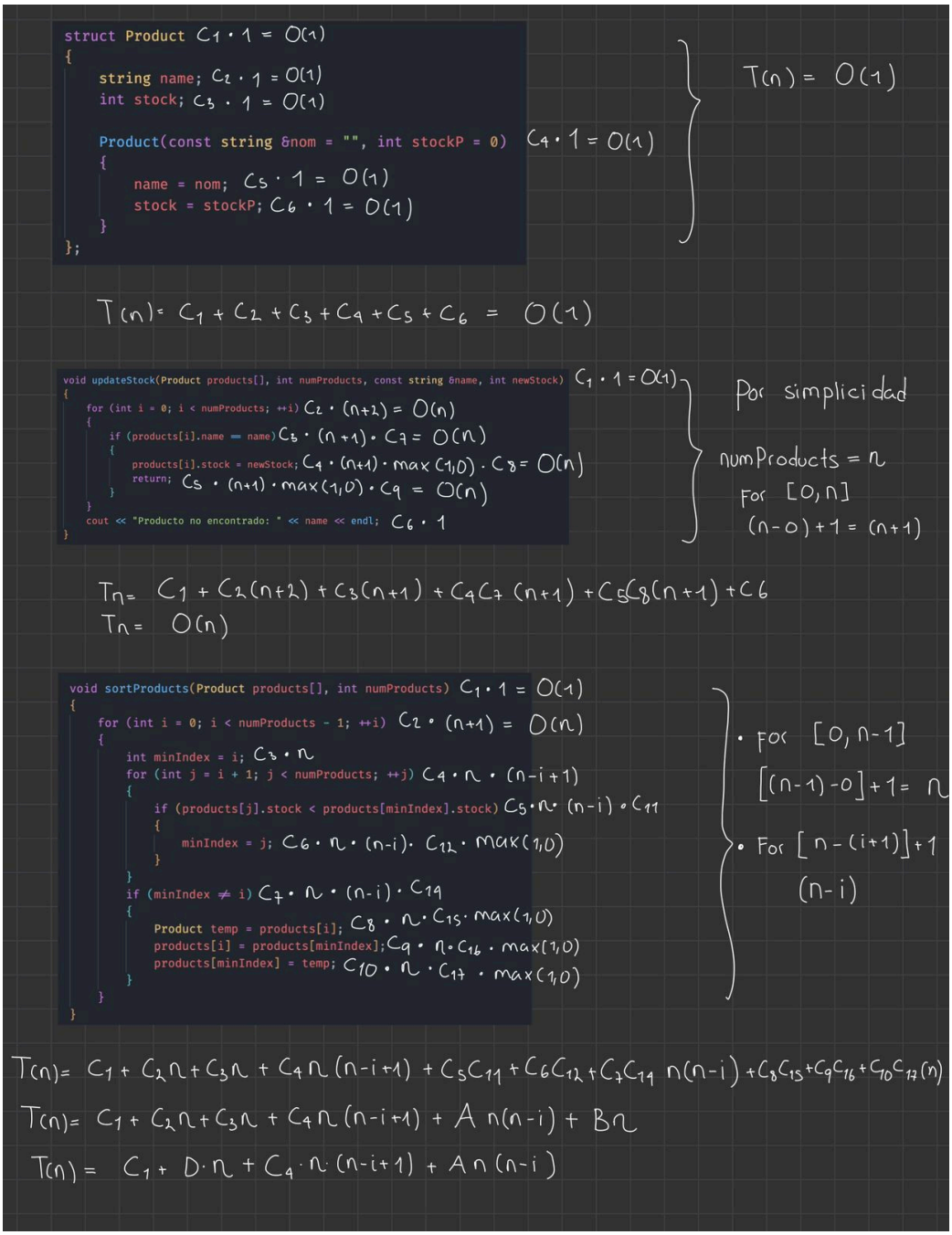
Entrega:

Viernes, 06 de septiembre

Problemática:

En un almacén se busca un sistema capaz de ordenar su productos, dichos productos tienen como datos , su nombre y su cantidad de stock, el sistema tiene como objetivo facilitar la obtención de los n-productos con menor stock.

Solución:



$$T(n) = C_1 + D \cdot n + C_4 \sum_{i=0}^{n-1} n - i + 1 + A \sum_{i=0}^{n-1} n - i$$

$$T(n) = C_1 + D \cdot n + C_4 \sum_{i=0}^{n-1} n - \sum_{i=0}^{n-1} i + \sum_{i=0}^{n-1} 1 + A \sum_{i=0}^{n-1} n - \sum_{i=0}^{n-1} i$$

$$T(n) = C_1 + D \cdot n + C_4 \left(n + \sum_{i=1}^{n-1} n - 0 + \sum_{i=1}^{n-1} i + 1 + \sum_{i=1}^{n-1} 1 \right) + A \left(n + \sum_{i=1}^{n-1} n - 0 + \sum_{i=1}^{n-1} i \right)$$

$$T(n) = C_1 + D \cdot n + C_4 \left(n + \underbrace{n(n-1)}_{n^2} + \underbrace{\frac{(n-1)(n)}{2}}_{n^2} + 1 + (n-1) \right) + A \left(n + \underbrace{n(n-1)}_{n^2} + \underbrace{\frac{(n+1)(n)}{2}}_{n^2} \right)$$

$$T(n) = O(n^2)$$

```cpp
void getLowStock(Product products[], int numProducts, int n) C1 · 1 = O(1)
{
    sortProducts(products, numProducts); C2 · O(n²) = O(n²)

    int lowStockProducts = 0; C3 · 1 = O(1)
    for (int i = 0; i < numProducts; ++i) C4 · (n + 2) = O(n)
    {
        if (products[i].stock <= 10) C5 · C17 (n+1) = O(n)
        {
            lowStockProducts++; C6 · C18 (n+1) · max(1,0) = O(n)
        }
        else C7 · C19 · (n+1) = O(n)
        {
            break; C8 · C20 (n+1) = O(n)
        }
    }

    if (lowStockProducts == 0) C9 · C21 = O(1)
    {
        cout << "No hay productos con stock igual o menor a 10." << endl; C10 · C21 · max(1,0) = O(1)
        return; C11 · C22 · max(1,0) = O(1)
    }

    if (n > lowStockProducts) C12 · C23 = O(1)
    {
        n = lowStockProducts; C13 · C24 · max(1,0) = O(1)
    }

    cout << "Los " << n << " productos con menor stock (menor o igual a 10) son:\n"; C14 = O(1)
    for (int i = 0; i < n; ++i) C15 · (n+2) = O(n)
    {
        cout << "Nombre: " << products[i].name << ", Stock: " << products[i].stock << endl; C16 · (n+1) = O(n)
    }
}
```

For $[0, n]$ $(n-0)+1 = n+1$

$$T(n) = O(n^2)$$

```
            cout << "Producto no encontrado: " << nameProduct << endl; C54 · 1
        }
        break; C55 · 1
    }
    case 4:
        cout << "Saliendo del programa..." << endl; C56 · 1
        return; C57 · 1
    default:
        cout << "Opción no válida. Inténtelo de nuevo." << endl; C58 · 1
        break; C59 · 1
    }
}
}
```

$= O(1)$

$T(n) = O(n^2)$

```
int main() C1
{
    int numProducts = 0; C2 · 1
    Product* products = nullptr; C3 · 1
    showMenu(products, numProducts); C4 · O(n²)
    delete[] products; C5 · 1
    return 0; C6 · 1
}
```

$T(n) = O(n^2)$

Conclusión:

Para abordar la problemática del almacén es conveniente el uso de algoritmos de ordenamiento, debido a su capacidad de ordenar datos, el algoritmo “Selection Sort” fue crucial para la resolución del problema, su importancia se da debido a su funcionalidad de encontrar el elemento más pequeño de un subarreglo desordenado, Selection Sort itera el arreglo y encuentra el valor mínimo, intercambiando dicho valor con el primer elemento no arreglado y repite el procedimiento para el resto del arreglo, avanzando un elemento en cada iteración.

Debido a la serie de pasos que realiza Selection Sort, la obtención de los n-productos con menor stock tuvo una solución, ordenando los productos de menor a mayor cantidad. La comprensión e implementación del algoritmo es fácil de desarrollar, sin embargo, la eficiencia del algoritmo es de baja calidad, siendo más notorio cuando hay una gran cantidad de datos de por medio.