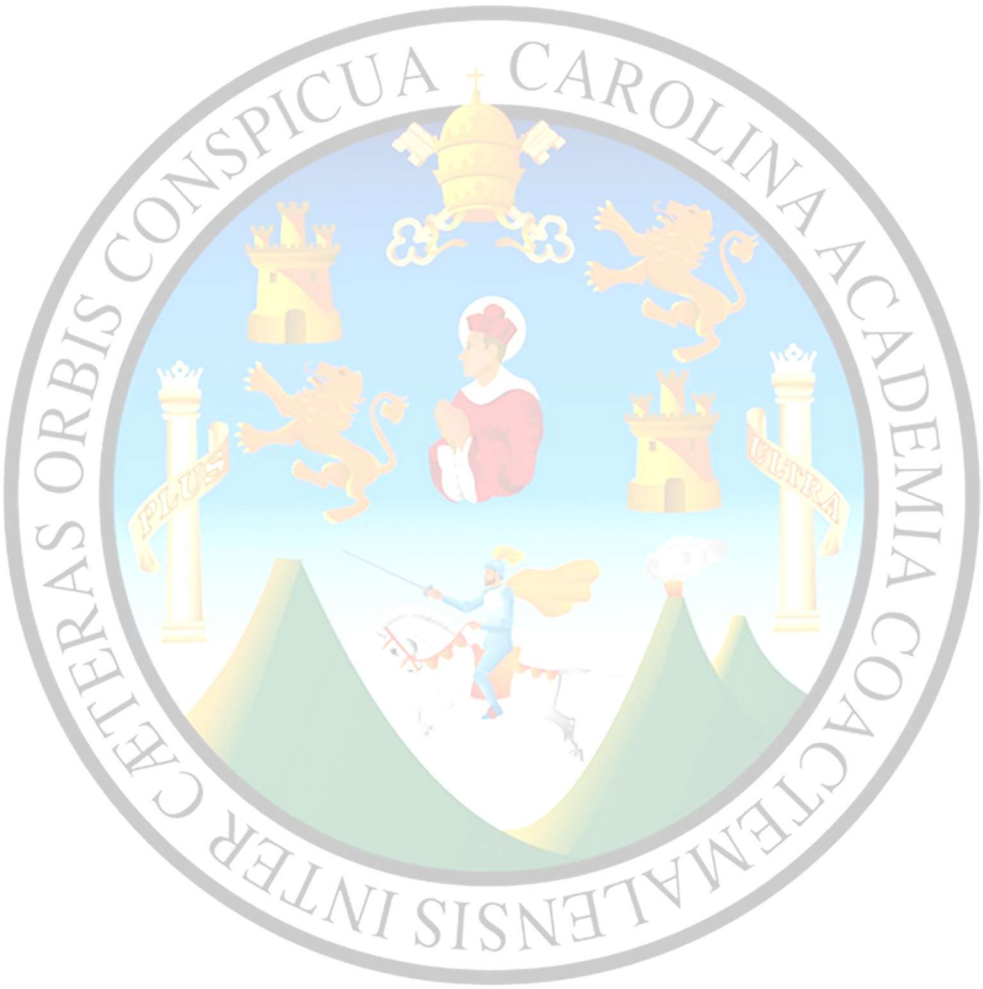
Facultad de Ingeniería

Escuela de Ciencias y Sistemas

Estructuras de datos - Sección A

Catedrático: Ing. Alvaro Obrayan Hernandez Garcia

Tutor académico: Alex René Lopez Rosa



Manual Técnico Battleship

Indice

Objetivos

Objetivo general

- Aplicar los conocimientos del curso Estructuras de Datos en el desarrollo de una aplicación que permita manipular la información de forma óptima

Objetivos específicos

- Demostrar los conocimientos adquiridos sobre estructuras de datos no lineales: tablas hash y grafos, poniéndolos en práctica en el desarrollo del juego batalla naval.
- Utilizar lenguaje C++ para implementar estructuras de datos no lineales.
- Utilizar la herramienta graphviz para graficar estructuras de datos no lineales.
- Definir e implementar algoritmos de búsqueda, recorrido y eliminación.

Requisitos mínimos

Para el funcionamiento óptimo del programa se requiere:

- Espacio en disco mínimo de 200 Mb
- Memoria RAM de 2 GB
- Poseer el compilador MinGw instalado en el sistema.
- IDE VScode instalado.

Librerías utilizadas

Para el desarrollo de este programa se utilizaron las siguientes librerías externas:

- hashlib
- json
- tkinter
- PIL

Métodos utilizados

Los métodos void no tendrán el tipo de método al lado izquierdo.

Método	Ubicación	Descripción breve
Creatienda()	Login.py	Método creador de la tienda de cada usuario.
Merk()	Login.py	Método que crea la ventana emergente con reporte merkle y tabla hash
Crearbotones()	Login.py	Método que agrega los ítems a la tienda
comprar(lista)	Login.py	Método que gestiona las compras del usuario
actualizar_tokens():	Login.py	Método que actualiza los tokens disponibles
logueado(nombre,contrasena)	Login.py	Método que ingresa a la pantalla de usuario
cerrar()	Login.py	Método que cierra la pantalla de logueo
cerraradmin():	Login.py	Método que cierra la pantalla del administrador
Versus()	Login.py	Método gestionador del 1 contra 1
Iniciar()	Login.py	Método que inicia el 1 vs 1
actualizar()	Login.py	Método que actualiza los tableros
Vic()	Login.py	Método que se ejecuta al terminar una partida
Repetir()	Login.py	Método para reiniciar una partida con los parámetros utilizados
Reporte1vs1()	Login.py	Método que genera el reporte tras la partida 1vs1
Registrar_usuario()	Login.py	Método que registra un nuevo usuario
Registrar()	Login.py	Método que crea la ventana de registro
Login()	Login.py	Métoodo que lee el logueo de un jugador
Verificar()	Login.py	Método que valida el logueo de un usuario
Ventana_principal()	Login.py	Método que crea la ventana de inicio.

ListaAdyacencia()	Matriz.py	Método que grafica la lista de adyacencia
Grafo()	Matriz.py	Método que grafica el grafo no dirigido
tablero(tamano,nombre)	Matriz.py	Método que grafica el tablero
Agregarbarcos(tamaño)	Matriz.py	Método que agrega los barcos al tablero
Haybarco()	Matriz.py	Método que revisa las posiciones de los barcos
Agregar()	Doble_items.py, Doble_usuarios.py, Doble.py	Método que permite agregar objetos a las listas
Destruidos()	Doble.py	Método que genera el reporte de barcos destruidos
Gettokens(nombre,contra)	Doble_usuarios.py	Método que actualiza los tokens actuales.
Settienda(tienda)	Doble_usuarios.py	Método que agrega la tienda personal a cada usuario
Comprados()	Doble_items.py	Método que retorna los ítems comprados
Comprar()	Doble_items.py	Método utilizado para comprar ítems.
graficar(nombre)	Merkle.py	Método para graficar el árbol de merkle
MerkleTree(Lista)	Merkle.py	Método para añadir las compras al arbol
Calculatelength(Values)	Merkle.py	Método que mantiene la cantidad de ítems en 2^n
Agregar(key,value)	Tabla_hash.py	Método para añadir ítems a la tabla hash
Drawhashtable(nomber)	Tabla_hash.py	Método para graficar la tabla hash