ALGORITMOS Y ESTRUCTURAS DE DATOS 2023/1
Proyecto Unidad 3

PROFESOR: Manuel Alejandro Moscoso Domínguez
manuel.moscoso.d@gmail.com

El presente documento entrega la descripción del escenario relacionado con el problema a resolver de la Unidad 3.

Objetivos

- Realizar el análisis de un problema y aplicar el diseño de una solución mediante la implementación de algoritmos de ordenamiento y búsqueda.
- Realizar el análisis y comparación de los algoritmos implementados.
- Desarrollar algoritmos que permitan entregar una solución a los problemas entregados.
- Realizar una argumentación de las ideas desarrolladas que involucran la solución del problema.

Requerimientos generales del Software a desarrollar

El Software desarrollado debe cumplir con los siguientes requerimientos:

- El programa debe estar completamente funcional, con ausencia de errores y alertas (warnings).
- El programa debe estar bien documentado para lo cual se necesita:
    - La inclusión de un archivo README donde se explica la solución desarrollada considerando el formato recomendado para este tipo de archivos. [1]
    - Presencia de comentarios que expliquen el código, considerando la descripción de estructura, clases, métodos y funciones.

[1] Formato README

    - La inclusión de un archivo INSTALL donde se explica el procedimiento para instalar el programa (compilar).
    - El programa debe encontrarse en un repositorio en su cuenta Github y de acceso público.
- El programa debe ser implementado a través del uso del lenguaje de programación C++.

Solamente debe entregar el acceso público al repositorio para su revisión.

Escenario, Sumando aspectos de lógica a los juegos “The Guardians” y “The Guardians Journey”

Existen varios escenarios en los cuales se vuelve fundamental poder optimizar la lógica implementada dentro del juego para lo cual se esta evaluación la utilización de algoritmos ya debidamente documentados con el propósito de optimizar el rendimiento. Actualmente estamos buscando los mejores algoritmos de ordenamiento que permitirán generar las librerías necesarias para ser utilizados a lo largo de todas las versiones de los juegos de los guardianes.

La implementación en donde se busca realizar la inclusión para resolver aspectos vinculados al ordenamiento de datos se encuentra en las siguientes áreas:

- **Tablero de puntaje:** El módulo de tablero de puntaje busca poder ordenar de la mejor manera a los jugadores en un tablero que refleje su avance. El avance se medirá en tiempo real por lo que es necesario contar con un algoritmo que pueda ordenar el tablero posterior a una actualización.
- **Determinación de camino entre aldeas:** En la actualización del viaje de los aprendices se establecieron valores para cada arista con el propósito de sumar dificultad. Se está interesado en poder tener un algoritmo que pueda ordenar y establecer un ranking entre los posibles recorridos.
- **Dibujo o renderizado de objetos:** El siguiente proyecto busca poder entregar un panel de inventario que permita a los jugadores comercializar los tesoros o productos obtenidos. La idea es poder tener un algoritmo que pueda recibir los tesoros o productos de cada jugador y ordenarlos con el propósito de facilitar, en

base a las preferencias del usuario, la selección para transformar, vender o comercializar y que se pueda ver gráficamente.

Por cada área señalada se establece o se estima preliminarmente tener la siguiente cantidad de elementos a considerar:

- **Tablero de puntaje:** Se espera tener hasta un rango de **90.000 a 100.000** de jugadores en la primera versión.
- **Determinación de camino entre aldeas:** Al ser un mundo abierto se considera dentro de las combinaciones posible tener entre **50.000 a 70.000** posibles caminos.
- **Dibujo o renderizado de objetos:** Aquí se considera mantener una variación de objetos totales disponibles entre **500 y 1000** por cada categoría. Al día de hoy se cuenta con **15** categorías en total que conforman la vestimenta completa de los guardianes.

Objetivo

El objetivo es el desarrollo de un programa en C + + que permita realizar una comparación de distintos algoritmos, en base a distintas entradas de datos, con el propósito de determinar qué algoritmo puede ser incluido dentro del juego.

Consideraciones

En una primera instancia se considera realizar esto a través de una carrera de algoritmos. Para generar estas carreras, se debe establecer una manera de generar un **set de pruebas homogéneo** con el propósito que todos los algoritmos trabajen con los mismos para obtener los tiempos de ejecución del ordenamiento.

Los métodos para ordenar pueden ser de manera ascendente o descendente y para efectos prácticos de la evaluación se deben establecer los dos escenarios y a través de opciones que seleccione el usuario al inicio del programa. El siguiente apartado entrega el formato de la opción que debe implementar iniciando el programa:

```
Carreras de algoritmos
```

```
1. Ascendente.
2. Descendente.
Opcion elegida:
```

Posterior a esto se establecen las carreras, para lo cual se crearán carreras considerandos los **set de datos**:

- Aleatorio: Un arreglo de datos de orden aleatorio sin la repetición de elementos.
- Aleatorio con duplicados: Un arreglo de datos de orden aleatorio con la posibilidad de tener elementos duplicados.
- Ordenado: Un arreglo de datos de orden según lo establecido (ascendente o descendente).
- Inversamente ordenado: Un arreglo de datos de orden inverso según lo establecido (ascendente o descendente).

> En caso que se determine realizar un **cambio en los rangos** debido a una extensa duración en la **generación de estos datos**, debe quedar estipulado en el README además de señalado en el video.

Las carreras están asociadas a las **áreas en las cuales serán utilizados**, por lo que a continuación se entrega el formato a implementar para identificar cada carrera considerando: Titulo vinculada al área, modo del set de datos y el podio identificando los algoritmos en el orden de competición (aquí solo se consideraron 3 algoritmos de ejemplo) y quien obtuvo el mejor resultado. Esto se debe replicar para los set de datos.

```
Carrera por el Tablero: Modo Ordenado
1. Nombre alg, t
2. Nombre alg, t
3. Nombre alg, t

El ganador es:Nombre alg un tiempo de t segundos
```

El resultado del programa es la salida del resultado de las carreras (ejemplo anterior) relacionadas con las tres áreas y los 4 modo de set de datos.

Para la implementación

Para realizar las tareas de implementación debe considerar:

1. Desarrollar la funcionalidad que permita al usuario determinar el formato de ordenamiento inicial (ascendente o descendente).
2. Desarrollar la funcionalidad que determina de manera aleatoria el total de elementos para cada carrera según el rango entregado dentro del problema.
3. Desarrollar la funcionalidad que permite implementar los elementos a través del modo ordenado e inversamente ordenado dentro del set de datos.
4. Desarrollar la funcionalidad que permite implementar los elementos aleatorios duplicados dentro del set de datos.
5. Desarrollar la funcionalidad que permite implementar los elementos aleatorios únicos dentro del set de datos.
6. Desarrollar la implementación de los algoritmos de ordenamiento cuadrático.
7. Desarrollar la implementación de los algoritmos de ordenamiento logarítmicos.
8. Desarrollar la funcionalidad que permite determinar el tiempo de ejecución de cada algoritmo.
9. Desarrollar la estructura de datos para almacenar los resultados de los algoritmos para entregar el resumen de la carrera.
10. Desarrollar la interfaz de presentación de resultados según las indicaciones entregadas.

Para completar y resolver la actividad se requiere la utilización de estructura de datos lineales, desarrollo de algoritmos y menús interactivos en un contexto práctico relacionado con el desarrollo de videojuegos.

Elaboración de un video de presentación

Junto con lo anteriormente solicitado es necesario la elaboración de un video mediante el cual se realiza una presentación de la solución. El video debe estar enfocado en presentar los aspectos que diferencia a los algoritmos entre sí, los resultados obtenidos además de señalar para cada uno la notación en el peor y mejor caso. Para esto debe considerar:

1. Elaboración de material audiovisual de apoyo (presentación) con la portada (presentación) y la agenda de los temas a tratar como contenido mínimo.

2. Debe utilizar un lenguaje técnico y formal.

3. La duración máxima del video es de 7 minutos. Se recomienda pausar el video o grabación y no mostrar durante la grabación el tiempo que toma esperar los resultados.

4. El acceso al vídeo debe estar presente en el README de la solución a través de un link de acceso.

Indicadores y ponderaciones

#	Indicadores	Ponderación
1	Desarrolla la funcionalidad que permita al usuario determinar el formato de ordenamiento inicial (ascendente o descendente).	5%
2	Desarrolla la funcionalidad que determina de manera aleatoria el total de elementos para cada carrera según el rango entregado dentro del problema.	3%
3	Desarrolla la funcionalidad que permite implementar los elementos a través del modo ordenado e inversamente ordenado dentro del set de datos	2%
4	Desarrolla la funcionalidad que permite implementar los elementos aleatorios duplicados dentro del set de datos	3%
5	Desarrolla la funcionalidad que permite implementar los elementos aleatorios únicos dentro del set de datos.	5%
6	Desarrolla la funcionalidad que permite el desarrollo de las carreras por cada set o modo de datos.	5%
7	Desarrolla la implementación del algoritmo selection sort	5%
8	Desarrolla la implementación del algoritmo bubble sort	5%
9	Desarrolla la implementación del algoritmo insertion sort	5%
10	Desarrolla la implementación del algoritmo shell sort	6%
11	Desarrolla la implementación del algoritmo merge sort	6%
12	Desarrolla la implementación del algoritmo quick sort	6%
13	Desarrolla la implementación del algoritmo heap sort	6%
14	Desarrolla la funcionalidad que permite determinar el tiempo de ejecución de cada algoritmo.	5%
15	Desarrolla la estructura de datos para almacenar los resultados de los algoritmos para entregar el resumen de la carrera e implementa la interfaz solicitada.	5%
16	Desarrolla el correcto uso de un repositorio para la mantención del proyecto con la debido incorporación del código (sin archivos adicionales), archivo README e INSTALL.	3%

17	Realizar la construcción de material multimedia (video) para ser entregado según los requerimientos solicitados (presentación).	10%
18	Desarrolla material de apoyo para ser expuesto en el video.	5%
19	Desarrolla una exposición exposición de la solución desarrollada utilizando un lenguaje técnico y formal	10%
	TOTAL	**100%**

Ayuda

```
#ifdef _WIN32
#include <Windows.h>
#else
#include <unistd.h>
#endif
#include <cstdlib>
#include <iostream>
#include <vector>
#include <bits/stdc++.h>
#include <random>

using namespace std;
//https://www.geeksforgeeks.org/measure-execution-time-with-high-precision-in-c-c/

void SelectionSort(vector<int>& arr) {
    int n = arr.size();
    for (int i = 0; i < n ; ++i) {
        for (int j = i; j < n ; ++j) {
            if (arr[j] < arr[i]) {
                int temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
            }
        }
    }
}

double getResultFromAlg(vector<int>& arr,int option) {
    time_t start, end;
    double time_taken;
    time(&start);
    ios_base::sync_with_stdio(false);
```

```
    SelectionSort(arr);
    time(&end);
    time_taken = double(end - start);
    return time_taken;
}

int main(int argc, char* argv[]) {

    cout << "Generando set de datos: " << endl;

    for (int i = 0; i < amount ; ++i) {
        arrSorted.push_back(i+1);
        arrReverse.push_back(amount-i);
        if ( i == 0 ) {
            random_value = 1 + rand() % (amount);
            arr.push_back(random_value);
        }
        else {
            random_value = 1 + rand() % (amount);
            arr.push_back(random_value);
        }
    }

    unordered_map<string, double> results;
    results["SeleciontSort"] = getResultFromAlg(arr);

    vector<int> arr1,arr2;
    arr1.assign(arr.begin(), arr.end());
    arr2.assign(arr.begin(), arr.end());
    int id = 1;
    for (const auto& pair : results) {
        const string& key = pair.first;
        double value = pair.second;
        cout << id << ". " << key << ", " << fixed << value << setprecision(5)
<< endl;
        id++;

    }
    return 0;
}
```

```
#include <iostream>
```

```cpp
#include <chrono>

using std::cout;
using std::endl;
using std::chrono::high_resolution_clock;
using std::chrono::duration;
using std::chrono::duration_cast;

auto myFunction() {
    // Perform some computation
    auto start = high_resolution_clock::now();
    for (int i = 0; i < 100000; ++i) {
        // Do something
    }
    auto end = high_resolution_clock::now();
    return duration_cast<duration<double>>(end - start);
}

int main() {
    auto time_taken = myFunction();
    cout << "Execution time: " << time_taken.count() << " seconds" << endl;
    return 0;
}
```