Documento de reflexión

Considero que es de vital importancia la implementación de diferentes algoritmos y estructuras de datos al momento de desarrollar software que tiene como objetivo la búsqueda, ordenamiento y acceso de información pero que, más importante, tiene como intención ser eficiente y fácil de manejar y modificar. Es por ello por lo que siento que a lo largo de este curso se nos han entregado todos los recursos y herramientas necesarias para llevar a cabo de manera eficaz y satisfactoria las actividades y evidencias que se nos han presentado.

El uso de los BST se ha vuelto ahora un elemento fundamental en este curso ya que con el Binary Search Tree tenemos la ventaja de poder acceder a los datos de manera más rápida y eficiente debido a que la mayoría de los algoritmos que se emplean en un BST son mayormente de una complejidad de O(log n). Es por esto por lo que para esta solución se decidió emplear un BST en forma de un MaxHeap el cual representa un BST como un vector y hace que sea más fácil su manipulación y la manera en la que se accede a los datos.

La tarea principal de esta actividad integradora fue ordenar un MaxHeap en el cual cargamos y ordenamos los registros con IPs atacadas con el siguiente código:

```
MaxHeap<Log> myMaxHeap(16807);

    myMaxHeap.loadLogs("BitacoraHeap.txt");
    myMaxHeap.heapSort();

template <class T>
  void MaxHeap<T>::loadLogs(std::string txtName){

    std::string month, day, hour, min, sec, ipa, desc;
    std::ifstream in(txtName);

    while(std::getline(in, month, ' ')){
        std::getline(in, day, ' ');
        std::getline(in,hour,':');
        std::getline(in,min,':');
        std::getline(in,sec,' ');
        std::getline(in,ipa,' ');
        std::getline(in,desc);
```

```
        dateTime dt(month, stoi(day), stoi(hour), stoi(min), stoi(sec));
        ipAddress ia(ipa);

        Log tmpLog(month, day, hour, min, sec, ipa, desc, dt, ia);

        push(tmpLog);

    }
  in.close();
}

template <class T>
  void MaxHeap<T>::heapify(int n, int i){

    int biggest = i;

    int l = 2 * i + 1;
    int r = 2 * i + 2;

    if (l < n && data[l].getIp() > data[biggest].getIp()){
      biggest = l;
    }

    if (r < n && data[r].getIp() > data[biggest].getIp()){
      biggest = r;
    }

    if (biggest != i){
      std::swap(data[i], data[biggest]);
      heapify(n, biggest);
    }
  }

  template <class T>
  void MaxHeap<T>::heapSort(){

    int n = size;

    for (int i = (n / 2) - 1; i >= 0; i--){
      heapify(n, i);
    }

    for (int i = n - 1; i > 0; i--){
      std::swap(data[0], data[i]);
      heapify(i, 0);
```

```
    }
  }
```

De este modo, teniendo los registros ordenados por IP, podemos hacer una búsqueda más fácil y rápida para así introducir en nuevo MaxHeap en el que cada espacio de nuestro vector tendrá un nuevo objeto que contiene una IP y las veces que se repite en nuestro primer MaxHeap. De este modo, obteniendo el top del MaxHeap y luego haciendo un .pop() 5 veces podremos obtener las 5 IP más accedidas de nuestro archivo .txt, imprimirlas y escribirlas en un nuevo .txt.

```
--->Creando un MaxHeap: 0x62fd30
--->Creando un MaxHeap: 0x62fd10

Most frequented IPs:

10.15.176.242 : 38
10.15.187.247 : 38
10.15.176.231 : 37
10.15.177.225 : 37
10.15.183.242 : 37

--->Liberando memoria del MaxHeap: 0x62fd10
--->Liberando memoria del MaxHeap: 0x62fd30
```

Finalmente, como reflexión podemos analizar el comportamiento de nuestros registros darnos cuenta de la manera en la que son atacadas todas nuestras IP y cómo podemos llegar a sacar conclusiones a partir de esto. También, puede surgir la interrogante: ¿Cómo podríamos determinar si una red está infectada o no?

Considero que esto depende de muchos factores pero viéndolo desde la perspectiva en la cual se está analizando en este curso, podemos ver que seguramente aquellas IP las cuales han sido más atacadas son aquellas que han sido accedidas con más frecuencia y para poder realizar estos análisis lo más viable puede ser hacer una recopilación de información clave para hacer búsquedas optimizadas las cuales nos ayudarán a detectar accesos maliciosos en nuestra red.