Estructuras de datos y algoritmos

Valeria Conde

A01281637

Binary search


```
struct Datos{}
int monthInNum(string mes){}
int binSearch(vector<Datos> vDatos, int ind, int mode) {
     while(left<right){
          mid = (left + right) /2;
          if (vDatos[mid].fecha >= ind) right = mid;
          else left = mid+1;
     }
}
```

// Lee el archivo y crea un vector con objetos de tipo Datos

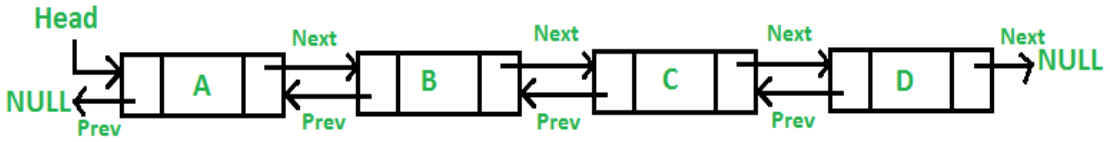
// Ordenar datos por fecha sort()

// Solicitar rango de fechas a buscar

// Desplegar informacion en rango de fechas

// Generar archivo de salida

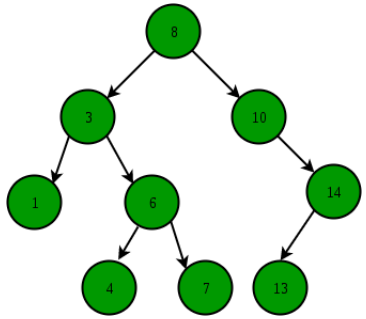
Linked list




```
class LinkedList {
        private:
            Node *head;
            int size;
        Public:
                Datos set(Datos data, int pos) {}
                bool change(int pos1, int pos2) {}
                void sort() {}
                void printRango(string ini, string fin) {}
}
```

```
int main()
    while (bitacora >> mes >> dia >> hora >> direccionIP) {
        getline(bitacora, razon);
        Datos tmp(mes, dia, hora, direccionIP, razon);
        LL.addLast(tmp);
    }

    LL.sort();
    LL.printRango(ipIni, ipFin);
```

Binary search tree

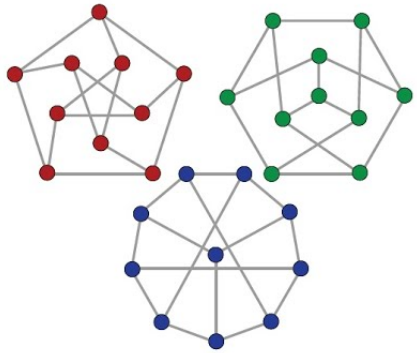



```
while (archivo >> mes >> dia >> hora >> direccionIP) {
    getline(archivo, razon);

    ipSinPuerto = direccionIP.substr(0, direccionIP.find(":", 0));
    ipLong = ipToLong(ipSinPuerto);

    if(ipLong != ipAnterior) {
        Info tmp(contador, ipSinPuertoAnterior, ipAnterior);
        arbol.add(tmp);

        contador = 0;
        ipAnterior = ipLong;
        ipSinPuertoAnterior = ipSinPuerto;
    }
    contador++;

}

if(contador > 0) {
    Info tmp(contador, ipSinPuerto, ipLong);
    arbol.add(tmp);
}

arbol.topNodes();
```

```
void BST::topNodesHelper(NodeT *n) {
    if(n != nullptr) {
        topNodesHelper(n->getLeft());
        if(top > 0) {
            cout << n->getData().accesos << " " << n->getData().ip << endl;
            top--;
        }
        topNodesHelper(n->getRight());
    }
}

void BST::topNodes() {
    topNodesHelper(root);
}
```

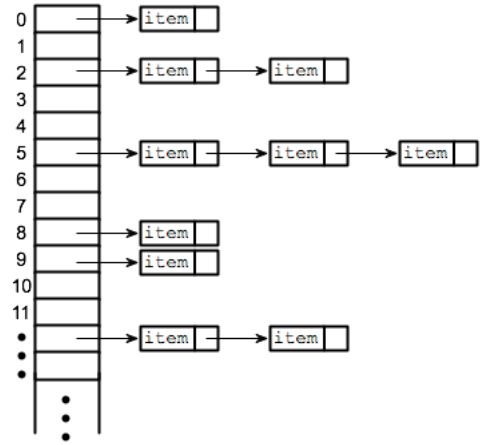
Grafos

```cpp
map<string, vector<string>> lista;
vector<string> nodes;

int main(int argc, const char * argv[]) {
    ifstream arcEnt("bitacora.txt");

    int n, m;
    arcEnt >> n >> m;

    for(int i = 0; i < n; i++) {
        string node;
        arcEnt >> node;
        nodes.push_back(node);
    }

    for(int i = 0; i < m; i++) {
        string mes, dia, hora, source, dest, msg;
        arcEnt >> mes >> dia >> hora >> source >> dest;
        getline(arcEnt, msg);

        string sourceSinPuerto, destSinPuerto;
        sourceSinPuerto = source.substr(0, source.find(":", 0));
        destSinPuerto = dest.substr(0, dest.find(":", 0));

        lista[sourceSinPuerto].push_back(destSinPuerto);
    }

    int maxFanOut = 0;
    for(auto it : lista) {
        if(it.second.size() > maxFanOut) {
            maxFanOut = (int) it.second.size();
        }
    }

    cout << "Botmaster(s):\n";
    for(auto it : lista) {
        if(it.second.size() == maxFanOut) {
            cout << it.first << " " << it.second.size() << endl;
        }
    }
    cout << endl;

    arcEnt.close();
```



Hash

```
// lee el archivo
while (bitacora >> mes >> dia >> hora >> direccionIP) {
    getline(bitacora, razon);
    ipSinPuerto = direccionIP.substr(0, direccionIP.find(":", 0));
    Datos tmp(mes, dia, hora, razon);

    // hash table ip - datos
    hasht[ipSinPuerto].push_back(tmp);
}
```




```
void show(string ip) {
    vector<Datos> tmp = hasht[ip];
    for(int i = 0; i < tmp.size(); i++) {
        cout << "Fecha: " << tmp[i].dia << "/" << tmp[i].mes << " " << tmp[i].hora << endl;
        cout << "Falla: " << tmp[i].razon << endl;
        cout << endl;
    }
}
```