UNIVERSIDAD PRIVADA BOLIVIANA DIRECCIÓN DE PREGRADO

FACULTAD DE INGENIERÍAS Y ARQUITECTURA

INVESTIGACIÓN

Complejidad Algorítmica Kruskal y PRIM

JUAN CLAUDIO CARRASCO TAPIA

LA PAZ – BOLIVIA

2022

Algoritmo de Kruskal:

```
double kruskal(int nroNodos, int nroAristas) {
    int origen, destino;
    double peso;
    double total = 0;
    numAristasArbol = 0;
    init();
    sort(aristas,aristas + nroAristas);
    for (int i = 0; i < nroAristas; i++)
    {
        origen = aristas[i].origen;
        destino = aristas[i].destino;
        peso = aristas[i].peso;
        if(!sameComponent(origen,destino)) { // est
            total += peso;
            unionRango(origen,destino);  // unimos
            MST[numAristasArbol] = aristas[i]; // G
            numAristasArbol++; // incrementados la
        }
    }
    return total;
}
```

Complejidad O(E log (E)) → Con E= cantidad de aristas(edges)

Algoritmo PRIM:

```
void prim(int start){
    pQueue.insert(make_pair(0, start));
    memset(visited, false, sizeof(visited));
    while (!pQueue.empty()){
        pair<int, int> current = *pQueue.begin();
        pQueue.erase(pQueue.begin());
        int currentV = current.second;
        if (!visited[currentV]){
            visited[currentV] = true;
            for (int i = 0; i < grafo[currentV].size(); i++){
                int neighbor = grafo[currentV][i].first;
                int nbrWght = grafo[currentV][i].second;
                if(!visited[neighbor]){
                    if(parent[neighbor]==-1 || (nbrWght<parentWeight[neighbor])){
                        parent[neighbor]=currentV;
                        parentWeight[neighbor]=nbrWght;
                    }
                    pQueue.insert(make_pair(nbrWght, neighbor));
                }
            }
            if(parent[currentV]!=-1 && parentWeight[currentV]!=-1 ){
                mst[currentV].push_back(
                            make_pair(parent[currentV],parentWeight[currentV]));
                mst[parent[currentV]].push_back(
                            make_pair(currentV,parentWeight[currentV]));
            }


        }
    }
}
```

Complejidad O(E log (V)) → Complejidad de Dijkstra, con E cantidad de aristas y V cantidad de vertices