ALGORITHMES IMPLÉMENTÉ EN C

25 octobre 2020

Kruskal

Cette algorithme a pour but de créer l'abre couvrant minimal (le graphe ou l'arbre qui touche tout les sommets avec le minimum d'arc possible, il est un arbre car il est un graphe acyclique) pour un graphe donné. Pour un graphe $G(S, v)$ et ça fonction de pondération w on a l'algorithme suivant :

```
algorithme Kruskal(G, w)
    debut
        E ← {}
        pour chaque sommet v ϵ S[G]
            faire CRÉER-ENSEMBLE(v)
        trier les arêtes de A par ordre croissant de poids w
        pour chaque arête (u, v) ϵ A faire
            si TROUVER-ENSEMBLE(u) ≠ TROUVER-ENSEMBLE(v) alors
                E ← E ∪ (u, v)
                UNION(u, v)
        retourner E
    fin
```

La fonction CRÉER-ENSEMBLE(v) va créer un ensemble disjoint pour chaque sommet puis pour chaque arret dont les sommets ne sont pas encore du meme ensemble (TROUVER-ENSEMBLE(u) ≠ TROUVER-ENSEMBLE(v)) que l'on ajoute a notre arbre et on lie leur ensembles.

Prim

De meme Prim permet aussi de créer l'arbre couvrant minimal mais eouvre différemment :

```
algorithme Prim(G, w, r)
    debut
        pour chaque u ϵ S[G] faire
            clé[u] ← ∞
            p[u] ← NIL
        clé[r] ← 0
        F ← S[G]
        tantque F ≠ ∅ faire
            u ← EXTRAIRE-MIN(F)
            pour chaque v ϵ Adj[u] faire
                si v ϵ F et w(u, v) < clé[v] alors
                    p[v] ← u
                    clé[v] ← w(u, v)
    fin
```

Il ajoute a chaque fois le sommet avec arret de poids minimal (grace à EXTRAIRE-MIN(F) et w(u, v) < clé[v]) dont la destination est déjà dans l'arbre mais lui meme (la source) non.

Djikstra

Est un algorithme qui permet de trouver le plus court chemin à origine unique c-à-d, le graphe contenant les plus court chemins (suite arets) à partir d'un origine donné :

```
algorithme Djikstra(G, w, r)
    debut
        pour chaque u ϵ S[G] faire
            clé[u] ← ∞
            p[u] ← NIL
        clé[r] ← 0
        F ← S[G]
        E ← ∅
        tantque F ≠ ∅ faire
            u ← EXTRAIRE-MIN(F)
            E ← E ∪ {u}
            pour chaque v ϵ Adj[u] faire
                si d[v] > d[u] + w(u, v) alors
                    d[v] ← d[u] + w(u, v)
                    p[v] ← u

        retourner E
    fin
```

Pour chaque destination à partir de u il rejoint le graphe si il moins couteu que le chemin actuelle