MTH6412B: Projet Voyageur de Commerce

Phase 3: Arbres de recouvrement minimaux (II)

- **Auteur**: El Hadji Abdou Aziz Ndiaye (1879468)
- **Code source**: **Repertoire Github**

Importation du code

```
run_test_prim (generic function with 1 method)
```

```
begin
    using PlutoUI
    using Plots
    include("node.jl")
    include("edge.jl")
    include("read_stsp.jl")
    include("graph.jl")
    include("mst_kruskal.jl")
    include("mst_prim.jl")
    include("tests/test_node.jl")
    include("tests/test_edge.jl")
    include("tests/test_graph.jl")
    include("tests/test_mst_kruskal.jl")
    include("tests/test_mst_prim.jl")
```

Révision du code de la phase 2

La fonction de conversion d'un fichier de type `stsp` en objet `Graph` a été améliorée.

L'ensemble des tests de la phase 2 sont repris dans cette phase 3. De nouvelles lignes de tests sont ajoutées pour tenir compte des changements apportés lors de la phase 3.

Tests des structures de base (**`Node`**, **`Edge`** et **`Graph`**):

```
Test de la structure de données `Node` : -v
Test de la structure de données `Edge` : -v
Test de la structure de données `Graph` : -v
```

Heuristiques d'accélération des opérations sur les composantes connexes

Compression des chemins

À chaque appel de `find_root`, les noeuds traversés sont compressés et deviennent des enfants directs de la racine trouvée.

Union via le rang

Un champ `rank` qui représente le rang d'un noeud a été ajouté dans la structure `Component`. Lors de la fusion de deux composantes connexes, leurs rangs sont comparés pour choisir le parent des deux composantes fusionnées.

Des tests unitaires pemettent de vérifier l'implémentation de ces deux heuristiques.

```
Test de la structure de données `Component` : -v
Test de la structure de données `ConnectedComponents` : -v
Test de l'algorithme de Kruskal : -v
```

Propriétés du rang

- La racine de l'arbre d'une composante connexe est le sommet qui a le rang le plus élevé. Ainsi, au pire cas, on peut avoir un arbre dont les sommets sont reliés un à un:
  $s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow \cdots \rightarrow s_n$ (avec s_n la racine). Dans ce cas, le rang de la racine s_n est n. Par ailleurs, on a $n = |S| - 1$ où $|S|$ est le nombre de sommets du graphe. Ainsi, le rang de tout noeud du graphe est inférieur à $|S| - 1$.
- Lorsqu'on construit une composante connexe, on augmente le rang maximal du graphe seulement lorsque les deux sous-arbres qu'on veut fusionner ont des racines ayant le même rang. Posons $N(i)$ comme étant le nombre minimal de sommets d'un arbre ayant une racine de rang i. On a donc, $N(i) \geq 2N(i-1) \Rightarrow N(i) \geq 2^i N(0)$. Or $N(0) = 1$ (il faut d'un noeud au minimum pour avoir un arbre ayant une racine de rang 0) et $N(i) \leq |S|$ quelques soit la composante connexe. Ainsi, on a donc $2^i \leq |S| \Rightarrow i \leq \log_2(|S|)$. Celà veut dire donc que le rang de tout noeud d'une composante connexe est inférieur à $\lfloor \log_2(|S|) \rfloor$ (le rang est un entier).

Algorithme de Prim

L'algorithme de Prim a été implémenté. L'implémentation utilise les listes d'adjacence noeud-arêtes et une file de priorité pour calculer l'arbre de recouvrement minimal.

L'algorithme a été testé sur l'exemple du cours et il donne un résultat correct.

```
Test de l'algorithme de Prim : -v
```

Programme principal

La fonction *main* permet de lire l'ensemble des fichiers contenus dans le repertoire `instances/stsp/` . Pour chaque fichier, on construit le graphe correspondant et on calcule les arbres de recouvrement minimaux avec les algorithmes de Kruskal et de Prim.

Résultats du programme principal:

```
File: bayg29.tsp
Reading of header : -v
Reading of nodes : -v
Reading of edges : -v
Arbre de recouvrement minimal : 1319 (Kruskal) 1319 (Prim) -v
File: bays29.tsp
Reading of header : -v
Reading of nodes : -v
Reading of edges : -v
Arbre de recouvrement minimal : 1557 (Kruskal) 1557 (Prim) -v
File: brazil58.tsp
Reading of header : -v
Reading of nodes : -v
Reading of edges : -v
Arbre de recouvrement minimal : 17514 (Kruskal) 17514 (Prim) -v
File: brg180.tsp
Reading of header : -v
Reading of nodes : -v
Reading of edges : -v
```