COMPLEXITES

Dans les fichiers qui travaillent directement sur les graphes, un graphe G est créé au début du fichier. Sa matrice d'adjacence et sa liste d'adjacence aussi. Elles sont créées avec une complexité en O(m) où m est le nombre d'arêtes du graphe G. Par la suite l'utilisation de ces outils ne sera plus qu'en O(1).
Les graphes sur lesquels nous travaillons sont des graphes simples, c'est-à-dire sans boucles ni arêtes multiples. Le degré maximum d'un sommet et donc n-1 où **n est le nombre de sommet de G**.

FICHIER FONCTIONS_GENERALES:

- ### FONCTION CAPITALES_FRONTALIERES :

Une seule instruction qui retourne la valeur d'un dictionnaire (liste d'adjacence) pour une clé donnée.
Complexité : O(1)

- ### FONCTION IS_CAPITALES_FRONTALIERES :

Instruction conditionnelle qui est en O(1) (vérification d'une valeur à un emplacement précis dans une liste de liste)
Les instructions dans le if et dans le else sont en O(1) (affectation)
Complexité : O(1)

- ### FONCTION NB_CHAINE_LONGUEUR_DONNEE :

Fonction en $O(n^3)$ puis instruction en O(1)
Complexité : $O(n^3)$

- ### FONCTION CHAINE_LONGUEUR2 :

Plusieurs instructions élémentaires sont en O(1). On boucle sur les sommets adjacents au sommet de départ puis sur les sommets adjacents à chaque sommet adjacent au sommet de départ. Soit d(départ), le degré du sommet de départ et d(départ[i]) le degré du sommet i adjacents au sommet de départ. Donc la complexité de l'algorithme est d(départ)*moy(d(départ[i])) où peut s'écrire aussi ∑(d(départ[i])).
Complexité : O(∑(d(départ[i])))

- ### FONCTION PLUS_COURTE_DISTANCE :

Première instruction en O(1), l'algorithme peut s'arrêter là. Mais s'il continu :
La suite des affectations est en O(1). Première boucle est en O(d(départ)) où d(départ) est le degré du sommet de départ.
Puis, boucle sur le nombre de sommets présents dans une liste. Il y au maximum n sommets qui peuvent être présents dans cette liste donc O(n) . Dans cette boucle il y a un second while qui boucle sur le degré de chaque sommet de la liste. Au max degré = n-1. Dans cette seconde boucle est présente une instruction conditionnelle qui vérifie la présence d'une valeur dans une liste, liste de taille au maximum n. Donc cette instruction est en O(n)
On se trouve alors avec le schéma suivant :
Boucle en O(d(départ))
Boucle en O(n)
    Boucle en O(n)

Instruction en O(n)

Donc dans le pire des cas l'algorithme contient n^3 instructions.
La complexité maximum de l'algorithme semble être O(max(d(départ), n^3)).
Complexité : O(n^3)

FICHIER FONCTIONS_DISTANCES:

- ### FONCTION DISTANCES_CAPITALES_FRONTALIERES :

Création d'un dictionnaire en O(1), création d'une liste à partir de la liste d'adjacence correspondant à un sommet i en O(1), boucle sur cette liste afin d'associer dans le dictionnaire une valeur à chaque élément de la liste. Notons d(i) le degré du sommet i.
Complexité : O(d(i))

- ### FONCTION DISTANCES_MIN_CAPITALES_FRONTALIERES :

Création d'une liste à partir de la fonction distances_capitales_frontalieres en O(d(i)). Trie de la liste en O(d(i)*log(d(i))) puisque la taille de notre liste est d(i).
Complexité : O(d(i) + d(i)*log(d(i))) = O(d(i)*log(d(i)))

- ### FONCTION DISTANCES_INFERIEURES_CAPITALES_FRONTALIERES :

Même principe que distances_capitales_frontalieres avec une condition supplémentaire en O(1)
Complexité : O(d(i))

- ### FONCTION DISTANCE_TRAJET :

Boucle sur le nombre de capitales présentes dans la liste. Soit k le nombre d'éléments présents dans la liste, alors l'algorithme est en O(k) puisque le reste des instructions est élémentaire.
Complexité : O(k)

FICHIER FONCTIONS_DUREES :

- ### ACCESSIBLE_CAPITALS(DEPART,DUREE) :

L'initialisation se fait en O(1).
La seule boucle de l'algorithme s'effectue sur l'ensemble des sommets du graphe, soit en n itérations.
La complexité du corps de boucle est en O(1).
Donc la complexité de la boucle est en O(n*1) = O(n).
Finalement, la complexité globale de l'algorithme est en O(max(1,n)) donc en O(n).

- ### BORDER_CAPITALS(CAPITALE) :

L'initialisation se fait en O(1).
La seule boucle de l'algorithme s'effectue sur l'ensemble des sommets du graphe, soit en n itérations.
La complexité du corps de boucle est en O(1).
Donc la complexité de la boucle est en O(n*1) = O(n).
Finalement, la complexité globale de l'algorithme est en O(max(1,n)) donc en O(n).

- ### NEAREST_BORDER_CAPITALS(CAPITALE) :

La fonction comporte 5 lignes.
Trois d'entre elles sont en O(1).
La seconde ligne utilise la fonction border_capitals donc est en O(n).

La troisième ligne consiste à trier un dictionnaire créé par la fonction border_capitals. La taille de ce dictionnaire correspond au degré du sommet entré en paramètre. Notons-le d(i). Donc ce tri est en O(d(i)log(d(i))).
Donc l'algorithme est en O(max(d(i)*log(d(i)) , n))

- EXIST_PATH_BETWEEN_TWO_CAPITALS(LIST_OF_CAPITALS) :

Cette fonction contient 3 lignes, toutes en O(1). Donc cette fonction est en O(1).

- PATH_DURATION_BETWEEN_TWO_CAPITALS(CAPITALE1, CAPITALES2) :

L'initialisation se fait en O(1).
La seule boucle de l'algorithme s'effectue sur la liste entrée en paramètre. Notons k le nombre d'éléments de la liste, avec k un entier naturel.
Le corps de boucle utilise une fonction en O(1).
Les autres instructions de ce corps de boucle sont en O(1)
Donc la complexité du corps de boucle est en O(1).
Cela implique que la complexité de la boucle est en O(k*1) = O(k).
Finalement, l'algorithme est en O(max(1,k)) donc en O(k).

FICHIER FONCTIONS_DUREES_VISITE :

- FONCTION DUREE_VISITE_INFERIEUR_A :

Initialisation du dico en O(1). Boucle sur les sommets de notre graphe. A l'intérieur de la boucle instruction en O(1).
Complexité : O(n)

- FONCTION DUREE_VISITE_EGALE_A :

Pareil que fonction duree_visite_inferieur_a.
Complexité : O(n)

- FONCTION DUREE_VISITE_INFERIEUR_A :

Pareil que fonction duree_visite_inferieur_a.
Complexité : O(n)

- FONCTION DUREE_SEJOUR :

Boucle sur le nombre de capitale de la liste entrée en paramètre. Soit k la taille de la liste.
Complexité : O(k)

- FONCTION DUREE_VISITE_SEJOUR :

Instruction en O(1)
Complexité : O(1)

FICHIER OUTILS_MATRICES:

- FONCTION PUISSANCE2_MATRICE_ADJACENCE :

L'algorithme contient 3 boucles imbriquées. Chacune est en O(n), où n est le nombre de sommets de notre graphe, puisqu'elles bouclent sur le nombre de sommets de notre matrice d'adjacence.
Complexité : $O(n^3)$

- FONCTION PRODUIT_MATRICE1 :

Même principe que fonction puissance2_matrice_adjacence.
Complexité : $O(n^3)$

- FONCTION PRODUIT_MATRICE2 :

Même principe que fonction puissance2_matrice_adjacence.
Complexité : $O(n^3)$

- FONCTION PUISSANCE_MATRICE_ADJACENCE :

Première instruction en $O(n^3)$. Puis boucle effectuée x fois qui contient une instruction en $O(n^3)$. x représente l'entier entré en paramètre auquel on soustrait 2.
Complexité : $O(x*n^3)$

FICHIER FONCTIONS_INFOS_CAPITALES :

- FONCTION LECTURE_DANS_FICHIER(NOM_FICHIER) :

La fonction est en O(n) où n est le nombre de capitale de notre fichier car toutes les lignes sont constantes, et certaines sont répétées n fois par le « for » qui est un temps linéaire.

- FONCTION INFOS_INITIALISATION() :

Cette fonction reprend la fonction précédente, elle est donc aussi en O(n).

- FONCTION AFFICHE_CAPITALE(CAPITALE) :

Cette fonction est en O(1) car afficher une information est un temps constant.

- FONCTION AFFICHE_LIS_OBJET_CAPITALE(LISTE CAPITALE) :

Cette fonction est en O(n) où n est le nombre de capitale de notre fichier car elle contient un « for » et sinon que des lignes de temps constant.

- FONCTION AFFICHE_LIS_CAPITALE(LISTE) :

Cette fonction contient une boucle en O(k) où k est le nombre d'éléments de la liste prise en paramètre. Dans cette boucle on fait appel à la fonction index_capitale qui est en O(n) donc la complexité de l'algorithme est en O(k*n).

- FONCTION LISTE_CAPITALE() :

Cette fonction est en O(n) où n est le nombre de capitale de notre fichier car elle contient un « for » qui boucle sur notre liste de capitale puis des instructions élémentaires.

- FONCTION INDEX_CAPITALE :

Cette fonction est en O(n) où n est le nombre de capitale de notre fichier car elle utilise la fonction .index(i) qui est en O(n).

- FONCTION CHOIX_INFOS() :

Cette fonction est en O(1) car afficher une information est un temps constant.

- FONCTION CHOIX_CAPITALE() :

Cette fonction est en O(1) car retourne une information en temps constant.