Descriptif code TIPE :

Contexte global :

Mon sujet concerne l'« Interception de mobiles dans un graphe ». Les fonctions dans le fichier joint sont donc des générateurs des graphes aléatoires, des implémentations d'algorithmes d'interception, ou bien fonctions permettant d'afficher des résultats.

I. Partie chemins connus :

Dans cette partie, on veut intercepter un mobile dont on connait le chemin dans un graphe censé représenter une ville.

1. Dessin

Les fonctions servent à afficher les graphes sur lesquels on travaille.
generateDicoPosition crée un dictionnaire pour donner la position des nœuds à afficher.
dessine affiche le graphe, avec les noms sur les nœuds.
dessine_noname affiche le graphe, sans les noms sur les nœuds.

2. Génération chemin connu

generate sert à créer un graphe de taille *longueur* * *largeur*, où deux nœuds (physiquement) proches sont reliés avec une probabilité de *probarete*. Cette fonction met des poids sur les arêtes déterminées par la fonction *fonctionpoids* qui prend pour paramètres les nœuds de la potentielle arête et *parampoids*.

generateConnexe répète la fonction précédente jusqu'à générer un graphe connexe ou dépasser la *limite* d'itération (si on veut un graphe connexe obligatoirement, on peut fournir pour limite math.inf)

Les fonctions portant le nom *xxxxSensUnique* ont le même but que la fonction *xxxx* mais proposent une possibilité pour les arêtes d'être à sens unique.

3. Résultats connexité

Ces fonctions servent à tester la proportion des graphes connexes parmi les graphes générés par les fonctions de la partie précédente en fonction de la taille et de probArete de ces graphes.

La fonction *resultats_connexite* affiche aussi un graphique représentant ces résultats.

4. Proba

Ces fonctions servent à l'attribution des poids.
La fonction *constante* donne un poids constant égal au paramètre
La fonction *uniforme* donne un poids compris entre les deux entiers (large) contenus dans le tupple fourni en paramètre.

5. Interception chemins connus

La fonction *temoin_connus* trouve un point d'interception en emmenant l'intercepteur à la fin du chemin, puis en le remontant.

La fonction *dijkstraAdapte* utilise l'algorithme de Dijkstra jusqu'à trouver tous les points d'interception possibles puis détermine celui qui intercepte en premier.

6. Résultats interception connu

generateurChemin génère un chemin de point de départ aléatoire et dont le nœud suivant est choisi équiprobablement parmi les possibilités, de taille fournie en paramètre.
generateurPosition choisit un nœud aléatoire du graphe (équiprobabilité entre les noeuds).
temps_intercpt_moyen, *resultats_interception_connu_taille* et *resultat_interception_connu* itèrent un nombre de fois (fourni en paramètre) les algorithmes témoins et de Dijkstra adapté, puis font la moyenne des temps d'interception et en tracent les courbes (en fonction de la taille du chemin, de celle du graphe et de la probabilité d'arête).

II Partie chemins inconnus :

1. Dessin

racinesNieme donne un dictionnaire pour placer les nœuds sur un cercle.
dessineGPE et *dessineGPE_noname* tracent des GPE avec et sans les noms des noeuds.

2. Générateur chemin inconnu

La fonction *generateurGPEC* génère aléatoirement un graphe planaire extérieur dont chaque nœud est voisin des nœuds adjacents.
Pour le faire, elle crée les nœuds, puis les arêtes entre eux et leur successeur et leur prédécesseur sur le cercle.
Puis, elle crée les arêtes « centrales » en stockant dans une liste de liste tous les voisins avec lesquels chaque nœud peut « créer » une arête. En effet, chaque nœud peut potentiellement faire une arête avec les suivants, sans qu'une telle arête saute plus de n/2 nœuds.
Lors de chaque essai, si une arête est créée on retire les liaisons impossibles entre les nœuds « à l'intérieur » de l'arête et les nœuds « à l'extérieur ».