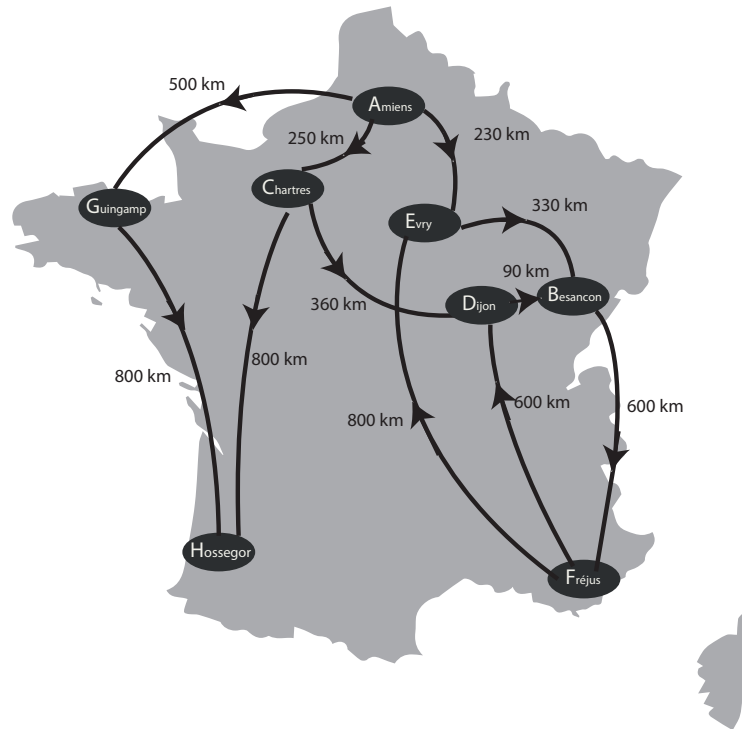


TP graphes (II)

1 - Graphe non orienté non pondéré

Dans ce TP, on se base sur le graphe suivant où chaque lettre fait référence à une ville :



2 - Représentation en Python

La variable `M0` contient la matrice d'adjacence du graphe non orienté non pondéré :

```
M0 = [ [0,0,1,0,1,0,1,0], [0,0,0,0,1,1,0,0], [1,0,0,1,0,0,0,1], [0,0,1,0,0,1,0,0],  
[1,1,0,0,0,1,0,0], [0,1,0,1,1,0,0,0], [1,0,0,0,0,0,0,1], [0,0,1,0,0,0,1,0] ]
```

Les villes sont indicés par des entiers de 0 à 7 (0 pour A, 1 pour B, etc...) On testera bien chaque fonction suivante avec la matrice d'adjacence `M0`.

1. Ecrire une matrice `M` correspondant au graphe ci-contre.
2. Définir une fonction `successeurs(M, i)` retournant la liste des successeurs du sommet `i`.
3. Définir une fonction `predecesseurs(M, i)` retournant la liste des predecesseurs du sommet `i`.

3 - Introduction aux parcours de graphe

Le parcours de graphe s'appuie sur une boucle `while` validée par la présence de successeurs à un sommet donné.

1. définir une fonction `chemin(M,i=6)` qui renvoie le chemin contenant le plus de villes possibles partant de `i = 6` (soit Guingamp).
2. Que ce passe-t'il pour `i = 4`? Proposer une modification de votre code permettant d'éviter les cycles.

4 - Algorithme glouton

L'algorithme glouton consiste à choisir le sommet le plus proche à chaque itération.

1. Proposer une fonction `plus_proche_sommet(M,i)` renvoyant le successeur le plus proche de `i`. Pour `i=0` (Amiens), la fonction doit renvoyer 4 (Evry).
2. Proposer une fonction `chemin_glouton(M,i)` qui renvoie le chemin contenant le plus de villes à partir de `i`, chaque ville étant choisie la plus proche de la précédente. Valider votre fonction pour `i = 0`.