TNSI DS K plus proches voisins - Recherche dichotomique 2025

1. Algorithme des k plus proches voisins

On se propose de mettre au point un algorithme qui pourrait être utilisé dans une agence immobilière pour aider un propriétaire à estimer le prix de son bien immobilier. On utilisera une base de données qui contient pour les différentes ventes immobilières réalisées dans l'année précédente, les coordonnées Gps des biens déjà vendus et le prix au m² qui a été utilisé pour réaliser la transaction.

Les biens déjà vendus l'année précédente sont regroupés dans une liste "liste_bien"

Cette liste contient elle-même 8 listes qui contiennent chacune :

[x , y , P , None] avec :

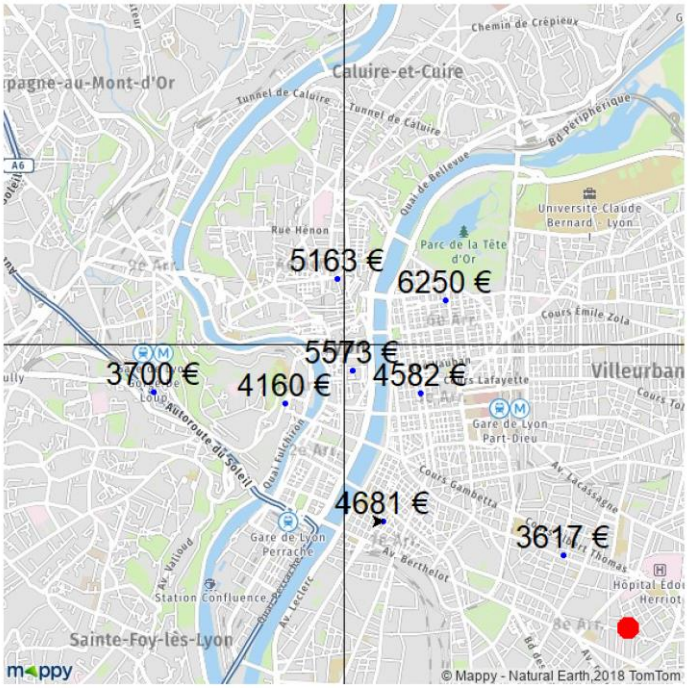
- x = abscisse du bien déjà vendu
- y = ordonnée
- P = prix au m² et en €
- champ vide pour l'instant (None)

```
liste_biens  =  [ [-52 , -52 , 4160 , None],\
                [-168 ,-42 , 3700 , None],\
                [-6 ,58 , 5163 , None],\
                [7 ,-23 , 5573 , None],\
                [89 ,39 , 6250 , None],\
                [67 ,-43 , 4582 , None],\
                [193 ,-186 , 3617 , None],\
                [34 ,-156 , 4681 , None] ]
```

a. Evaluation des distances entre les biens et le bien à vendre

Le bien à vendre se trouve positionné sur la carte aux coordonnées (250,-250)

Ecrire une fonction **distance** prend en compte les 4 paramètres de deux points sur la carte A avec xa,ya et B xb,yb et qui en retourne la distance selon Euclide



Ecrire une fonction **rempli** qui complète la liste_biens avec les **distances** entre chaque biens et le bien à évaluer représenté par ses coordonnées xb,yb

b. Evaluation du prix du bien à vendre

Ecrire une fonction **tri** qui classe les biens de la **liste_biens** complétée dans **un ordre croissant des distances. On utilisera le tri par sélection. Vous pouvez utiliser a,b= b,a.**

Ecrire une fonction **evaluer** qui prend la **liste_biens triée** et qui retourne la moyenne des prix des k biens les plus proches du bien à vendre.

c. Simplification de l'évaluation

A ce stade du programme pour évaluer le bien i est nécessaire dans le programme principal d'écrire (la fonction distance étant appelée par la fonction rempli):

```
# programme principal

rempli(liste_biens,250,-250)
tri(liste_biens)
print(evaluer(liste_biens,3))
```

Ecrire une fonction **evaluer_bis** qui ne prend que deux paramètres la liste de base des biens et un nombre k. Cette fonction appelle autant que nécessaire les autres fonctions : distance, rempli et tri

2. Recherche dichotomique

On possède une liste python des différents mots d'un dictionnaire de Français.

liste_mots = ["a","abaissa","abaissable","abaissables",……," zythums"]

On désire vérifier si le mot : "mot_inconnu" y est présent.

a. Recherche naïve

Ecrire une fonction "**recherche_naive**" prenant comme paramètres une liste l et une chaine de caractères mot_inconnu renvoie True ou False et le nombre d'itérations nécessaire pour trouver le mot inconnu (quand il existe)

b. Recherche dichotomique

Quelle nécessité existe poour employer une recherche dichotomique sur la liste liste_mot.

Erire une fonction "**recherche_dichotomique"** qui prend comme paramètres une liste l et mot_inconnu renvoie True ou False et le nombre d'itérations nécessaire pour trouver le mot inconnu (quand il existe).

TNSI DS K plus proches voisins - Recherche dichotomique 2025

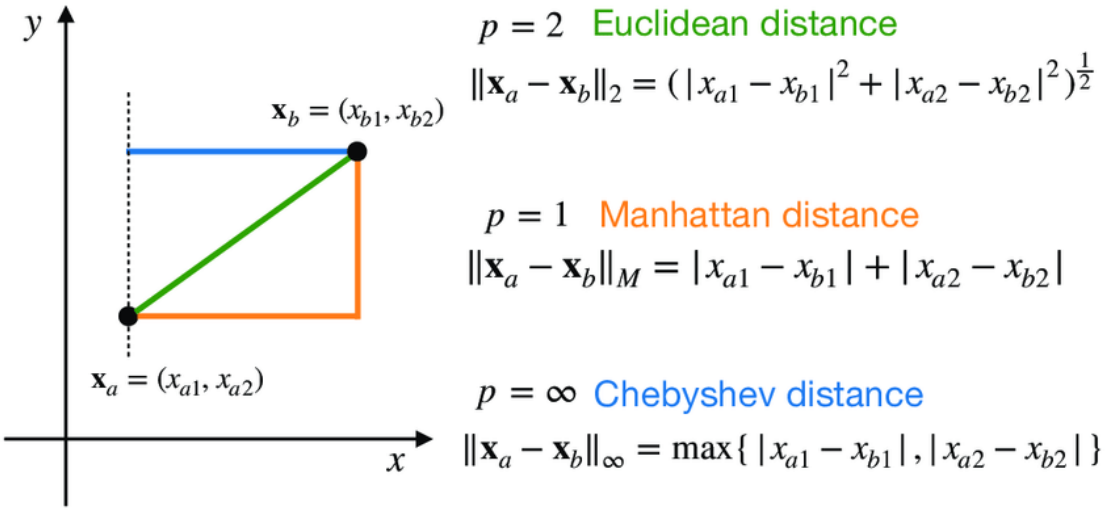
La liste_mot contient 336 531 mots. Quel sera le nombre maximum d'itération possible avant de trouver (ou pas) la présence du mot inconnu dans la liste ?

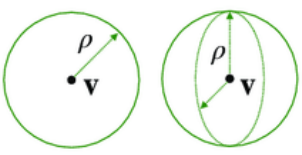
Annexes

Fonctions mathématiques

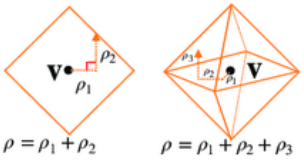
```
import math

nombre = 16


racine_carrée = math.sqrt(nombre)

print("La racine carrée de", nombre, "est", racine_carrée)


puissance_trois =pow(nombre,3)

print("La puissance  trois de nombre est ", puissance_trois)
```

Formules de distances

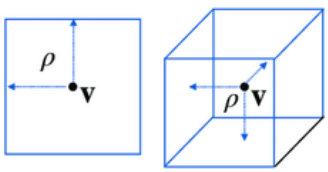




Euclidean distance
(a) $p = 2$



Manhattan distance
(b) $p = 1$



Chebyshev distance
(c) $p = \infty$