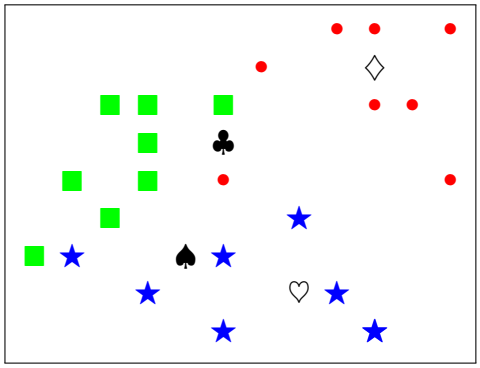
I Les KNN voisins

L'algorithme des KNN voisins (K-Nearest Neighbours) est un algorithme de base pour l'apprentissage automatique. Il prétend classer (catégoriser) une donnée en connaissant ses valeurs et celles des familles possibles.

expl1 : Considérons les familles ronds rouges, carrés verts et étoiles bleues ainsi réparties dans le plan. On ajoute quatre points représentés par chaque couleur d'un jeu de cartes.

Départager les régions selon les trois familles en présence puis déterminer l'appartenance de chaque couleur de cartes.



le diamant semble s'associer avec la famille ...

le cœur semble s'associer avec la famille ...

le pique semble s'associer avec la famille ...

le trèfle semble s'associer avec la famille ...

expl2 : Supposons que nous avons deux élèves A ayant comme notes {15,17,19,15}, l'autre B ayant comme notes {11,12,11,13}. À un nouveau devoir, le professeur corrige une copie sans nom dont la note est 14. À quel élève cette copie a des chances d'appartenir ?

- ◇ Si on prend 1 note, on remarque que 14 est aussi proche des notes A dont la plus basse est 15 que des notes B dont la plus haute est 13.
- ◇ Si on prend 2 notes, 14 est plus proche de {15,15} que {12,13}.
- ◇ Si on prend 3 notes, 14 est plus proche de {15,15,17} que {11,12,13}.
- ◇ Si on prend toutes les notes, 14 est plus loin de {15,15,17,19} que {11,11,12,13}.

Ainsi

— si l'on prend 1 note pour calculer la distance de 14 aux jeux de notes A ou B, on ne peut trancher.

— si l'on en prend 2 ou 3 pour calculer la distance, on conclura que la copie est certainement à l'élève A.

— si l'on prend 4 notes, on penchera pour l'élève B.

Rq : selon le nombre de voisins considérés, la distance calculée, un objet peut appartenir à une famille d'objets A ou une autre famille.

À partir de ces exemples, on peut définir un algorithme d'association à une famille connaissant un objet P et des familles A_i composées d'objets $(P_i^j)_j$.

- ◇ déterminer K le nombre de voisins pris en compte
- ◇ calculer toutes les distances du points P aux points P_i^j
- ◇ trier l'ensemble des distances de la plus petite à la plus grande
- ◇ associer P à la famille A_i qui a le plus de membres parmi les K plus proches voisins de P

Exo 1 : On considère deux familles d'animaux A et B dont on connait la taille (en cm) et le poids (en g). { Famille A : (9,10), (9,8) ; Famille B : (6,11), (7,12), (7,11), (8,13) } et un individu (8; 10,1) pas encore classé.

a) Quels semblent être les caractéristiques de la famille A par rapport à la famille B ?

...

b) On a calculé les carrés de la distance euclidienne ($d_E(X,Y) = \sqrt{(X_0 - Y_0)^2 + (X_1 - Y_1)^2}$) (rq : on peut utiliser une autre distance, comme la distance de Manhattan ($d_M(X,Y) = |X_0 - Y_0| + |X_1 - Y_1|$)) de l'individu inconnu par rapport à chaque individus des familles.

individus	(9,10)	(9,8)	(6,11)	(7,12)	(7,11)	(8,13)
$\mathbf{d_E^2}$	1,00	2,33	2,19	2,15	1,35	2,90

Compléter le tableau ordonnant les couples (famille d'origine, distance) en fonction des distances.

couple (famille, distance)	(A ; 1,00)	(B ; 1,35)	(B, 2,15)	...	...	...

c) À quelle famille l'individu appartient-il si on prend son plus proche voisin ?

...

d) À quelle famille l'individu appartient-il si on prend ses trois plus proches voisins ?

...

e) Que se passe-t-il si on prend ses cinq (ou plus) plus proches voisins ?

...

Exo 2 : On souhaite classer informatiquement un nouvel animal dont on connait la taille et le poids. Les familles A et B seront celles définies dans l'exercice 1.

a) Compléter le code suivant qui met en œuvre l'algorithme des KNN voisins .

```
1  familleA = [(9,10), (9,8)]
2  familleB = ...
3  families = {'A': familleA, ...}
4
5  def distance(X: tuple, Y: tuple) -> float:
6      """
7      renvoie la distance euclidienne entre les couples X et Y
8      """
9      return ...
10
11 def distanceFamille(p: tuple, family: list, fnom: str) -> list:
12     """
13     fnom est le nom de la famille family
14     renvoie une liste de couples (fnom, distance) où la distance est celle calculée
   ↪  entre le couple p(taille,poids) et chaque couple de family
15     """
16     n = len(family)
17     distFamily = [0]*n
18     for j in range(n):
19         ...
20     return ...
21
22 def tri(distances: list) -> None:
23     """
24     trie une liste composée de tuple (fnom, distance) selon la distance
25     """
26     n = len(distances)
27     for j ...
28         ...
29
30
31
32
33
34
35
36
37 def algoKNN(K: int, p: object, families: dict) -> int:
38     """
39     families est un dictionnaire de liste de couples dont la clef est le nom de la
   ↪  famille
```

```
    """
    N = len(families) # nombres de familles possibles

    dFamilies = []   # tableau des couples (fnom, distances)
    for fnom in families:
        # pour chaque famille, on calcule la distance de ses membres à l'individu p et
        ↪ on l'ajoute à la liste dFamilies
        # à la fin, dFamilies sera de la forme [('A',1), ('A',2.33), ('B',2.19), ...]
        dFamilies ...

    # on trie le tableau dFamilies selon les distances
    ...

    cptFamilleKVoisins = {fnom: 0 for fnom in families}
    # on compte le nombre de membres d'une famille parmi les K plus proches voisins
    for i in range( K ) :
        cptFamilleKVoisins[    .........    ] += 1

    # détermine le nom de la famille la plus présente parmi les K plus proches
    ↪ voisins
    fmax, vmax = '', -float('inf')
    for ...
        if ...
            ...

    # renvoie la famille la plus probable associée à l'individu p
    return ...
```

b) Proposer un appel qui traite numériquement un animal mesurant 8 cm et pesant 10,1 g.

...

II Algorithmes gloutons

Définition 1

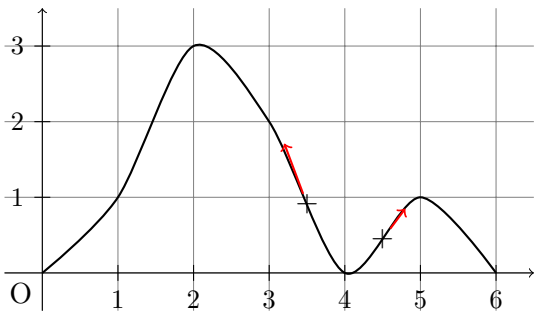
Un algorithme glouton (*greedy algorithm*) est un algorithme qui, à chaque itération, fait un choix local optimal pour résoudre le problème traité.

Propriété 1

L'algorithme glouton ne renvoie pas toujours la meilleure solution (l'optimal global).

expl3 : On suppose que l'on cherche le maximum d'une courbe et qu'à chaque itération notre critère soit de choisir d'aller vers la plus grande pente.
si on part de l'abscisse $x = 3{,}5$, on trouvera pour maximum ... , qui est ...

si on part de l'abscisse $x = 4{,}5$, on trouvera pour maximum ... , qui est ...



1 Les fractions égyptiennes

Pour approfondir le sujet, on pourra se référer à Mathématiques d'École *de D. Perrin (p95) ou les* Olympiades de mathématiques *2016 (exo2).*

2 000 ans avant notre ère, les Égyptiens écrivant tout nombre fractionnaire positif $\frac{p}{q}$ comme somme de fractions unitaires $\frac{1}{q'}$ toutes différentes.

expl4 : $\frac{3}{5} = \frac{1}{2} + \frac{1}{10}$ est une décomposition en fractions égyptiennes mais $\frac{3}{5} = \frac{1}{5} + \frac{1}{5} + \frac{1}{5}$ n'en est pas une.

Exo 3 : Déterminer deux décompositions en fractions égyptiennes de chacun des nombres :

- $\frac{2}{3} =$
- $\frac{8}{7} =$

Propriété 2

La décomposition d'un nombre en fraction égyptienne n'est pas unique.

L'algorithme suivant (dû à Léonard de Pise (Fibonacci) en 1 202) permet de déterminer une décomposition en fractions égyptiennes pour les fractions $0 < \frac{p}{q} < 1$:

k **PREND LA VALEUR** 1
p_k, q_k **PREND LA VALEUR** p, q
TANT-QUE $p_k \neq 0$ **FAIRE**
  n_k **PREND** la plus petite valeur telle que $\frac{1}{n_k} \leqslant \frac{p_k}{q_k}$ $\left(\text{d'où } < \frac{1}{n_k - 1}\right)$
  p_{k+1}**PREND LA VALEUR** $p_k n_k - q_k$
  q_{k+1}**PREND LA VALEUR** $q_k n_k$
  INCRÉMENTER de un k
FIN-TANT-QUE
AFFICHER l'ensemble des $\frac{1}{n_k}$

Exo 4 : Dérouler cet algorithme pour déterminer une décomposition en fractions égyptiennes de $\frac{7}{9}$.

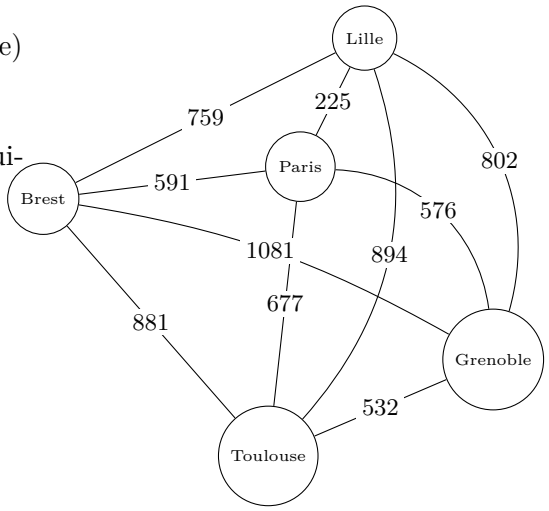
lignes	1-2	3	4	5-7	3	4	5-7	3	4	5-7	3
k											
p_k											
q_k											
$p_k \neq 0$											
n_k											

Exo 5 : Proposer une fonction `fractionEgyptienne(p : int, q : int) -> list`, qui implémente cet algorithme. Déterminer les variants, invariants et la complexité temporelle.

2 Problème du voyageur de commerce

Le problème du voyageur de commerce est un autre problème classique des algorithmes gloutons. Il consiste à minimiser la distance qu'un voyageur de commerce fait en visitant une et une seule fois toutes les villes et en revenant à l'initiale.

expl5 : (les graphes seront vus plus en détails au prochain chapitre)



On peut considérer le tableau des distances entre les villes suivantes :

	Brest	**Grenoble**	**Lille**	**Paris**
Toulouse	881	532	894	677
Paris	591	576	225	
Lille	759	802		
Grenoble	1081			

Un voyageur de commerce démarre sa tournée de Paris.

On peut faire une liste exhaustive des solutions et choisir la plus petite.

En voici une partie des solutions

Parcours	**calcul de distance (km)**	**total (km)**
Paris-Brest-Grenoble-Lille-Toulouse-Paris	591+1081+802+894+677	4045
Paris-Brest-Grenoble-Toulouse-Lille-Paris	591+1081+532+894+225	...
...	...	⋮
Paris-Lille-Brest-Toulouse-Grenoble-Paris	225+759+881+532+576	2973
Paris-Lille-Grenoble-Brest-Toulouse-Paris	225+576+1081+881+677	3666

On trouve alors la meilleure solution : Paris-Lille-Brest-Toulouse-Grenoble-Paris pour un total de 2 973 kms.

Ceci dit, lister l'ensemble des solutions n'est pas envisageable en pratique dès que le nombre de villes (noté N) devient plus grand qu'une dizaine.

Exo 6 : Combien de combinaisons doivent être calculées pour une étude exhaustive avec N villes ?

En supposant que l'ordinateur vérifie 1 million de cas à la seconde, il mettra

- ◇ pour $N = 10$ villes, 181 440 possibilités à vérifier soit près de 0,1 seconde de calculs
- ◇ pour $N = 20$ villes, $60\,822\,550\,204\,416\,000 \approx 6.10^{16}$ possibilités à vérifier soit plus de 1 925 années de calculs (!)

Exo 7 : Préciser en quoi consiste l'algorithme glouton du voyageur de commerce ?

...

L'algorithme serait donc :

```
distanceTotal PREND LA VALEUR 0
villeActuelle PREND LA VALEUR Paris
parcours PREND LA VALEUR Paris
TANT-QUE il reste des villes à visiter FAIRE
    villeActuelle PREND LA VALEUR de la ville la plus proche non-encore visitée
    AJOUTER cette nouvelle distance à disTotale
    AJOUTER villeActuelle au parcours
FIN-TANT-QUE
AFFICHER parcours
AFFICHER distanceTotal
```

expl6 : sur l'exemple précédent, le résultat est l'optimal, à savoir Paris-Lille-Brest-Toulouse-Grenoble-Paris pour 2 973 kms.

Exo 8 : Supposons que le voyageur de commerce parte de Grenoble cette fois. Appliquer l'algorithme glouton et comparer avec le parcours Grenoble-Paris-Lille-Brest-Toulouse-Grenoble.