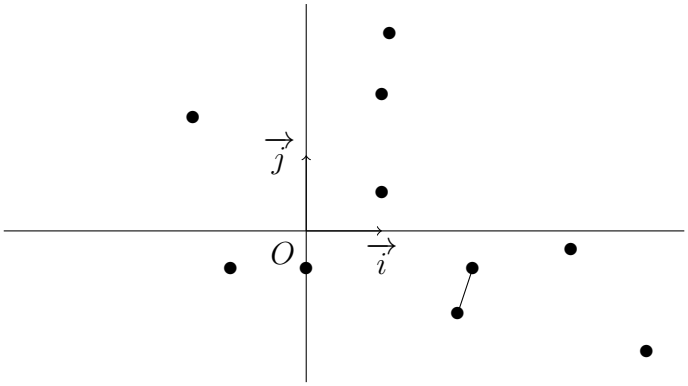
DM1 : Plus proches voisins dans un nuage de points

Nous nous proposons de résoudre un problème de géométrie classique : n étant un entier supérieur ou égal à 2, nous avons n points deux à deux distincts répartis dans le plan et nous voulons trouver la distance minimale entre deux points distincts de ce nuage de points.

Structure de données

1. Représentation d'un point.
   Le plan étant muni d'un repère orthonormé $(O, \vec{i}, \vec{j})$, un point M du plan est parfaitement repéré par ses coordonnées (x, y). En OCaml, un point M du plan sera représenté par le couple `(x,y)` de type `float * float` correspondant donc à ses coordonnées dans le repère $(O, \vec{i}, \vec{j})$.
   (a) Quelles sont les commandes en OCaml qui permettent d'obtenir la première composante d'un couple, la seconde composante d'un couple ?
   (b) Écrire en OCaml une fonction `distance m1 m2` qui détermine le carré (afin d'éviter un laborieux calcul de racine) de la distance euclidienne entre les deux points $M1$ de coordonnées `m1` et $M2$ de coordonnées `m2`.

2. Représentation d'un nuage de points.
   Un nuage de points sera représenté par un tableau donc de type `Array` dont les éléments seront les coordonnées des points du nuage représentés donc par des couples de flottants.




```
let nuage = [| 1.0,2.1 ; 4.5,-1.6 ; .... ; 1.1,2.3 |] ;;
```

   Quelle commande permet d'avoir l'abscisse du deuxième point ?

Algorithme naïf

On considère un nuage de n points `nuage` dont les points sont deux à deux distincts et au moins au nombre de 2.

3. Soit i un entier compris entre 0 et $n-2$.
   Écrire en OCaml une fonction `distance_i nuage i` qui calcule le carré de la distance minimale entre le point M_i de coordonnées `nuage.(i)` et les points M_j de coordonnées `nuage.(j)`, pour tout $i+1 \leq j \leq n-1$.

4. Écrire une fonction en OCaml `distance_mini nuage` qui calcule la distance minimale entre deux points distincts du nuage.
5. Évaluer la complexité de la fonction `distance_mini nuage`, la taille de données étant le nombre de points du nuage et nous cherchons à évaluer le nombre de calculs de distance entre deux points.

Problème en dimension 1

Considérons toujours un nuage de n points `nuage` dont les points sont deux à deux distincts et au moins au nombre de 2.
Nous supposons dans cette question que les points sont alignés et que le repère a été choisi de telle façon que les ordonnées des différents points soient toutes égales. De plus, nous supposons les points rangés par abscisse croissante.

6. Écrire alors une fonction en OCaml `distance_mini_1D nuage` qui détermine la distance minimale entre deux points distincts du nuage en complexité linéaire.
7. Justifier que la complexité de cette fonction est bien linéaire.

Diviser pour régner

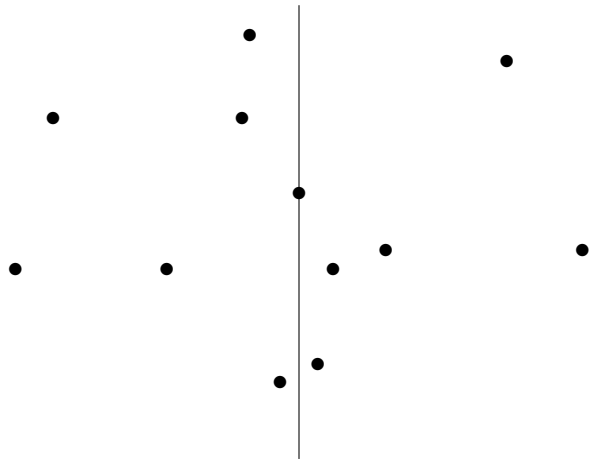
L'algorithme naïf a longtemps été considéré comme le meilleur. L'apparition des méthodes de réduction de complexité comme le paradigme "diviser pour régner", a permis d'améliorer la complexité de notre problème.
Considérons toujours un nuage de n points `nuage` dont les points sont deux à deux distincts et au moins au nombre de 2.
Dans le cas où le nuage contient 2 ou 3 points, nous calculons toutes les distances possibles et nous retenons la distance minimale, méthode basée sur l'algorithme naïf.
Supposons dans la suite que le nuage contient au moins 4 points.
Nous considérons un nuage de points `nuage_h` contenant les mêmes points que `nuage`, seulement les points dans `nuage_h` sont triés par abscisse croissante et en cas d'égalité de la valeur des abscisses, par ordonnée croissante (ordre lexicographique usuel).

8. Écrire une fonction en OCaml `tri_fusion_array_h nuage` qui retourne le nuage trié `nuage_h` en utilisant le principe du tri fusion.
9. Rappeler la complexité du tri fusion.

Nous considérons également un nuage de points `nuage_v` contenant les mêmes points que `nuage`, seulement les points dans `nuage_v` sont triés par ordonnée croissante et en cas d'égalité de la valeur des ordonnées, par abscisse croissante (ordre lexicographique avec les coordonnées dans l'ordre inverse).

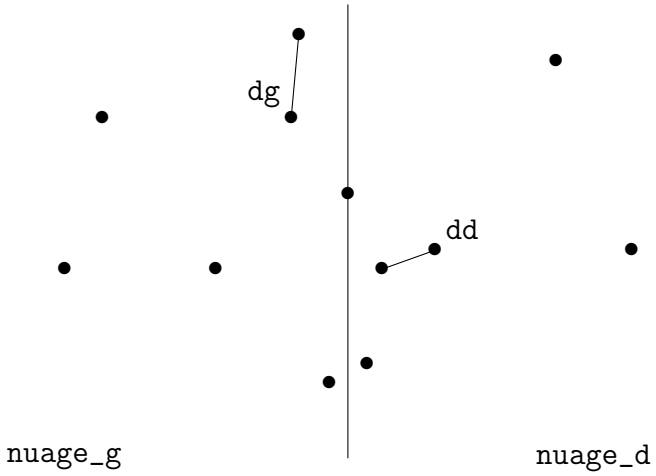
10. Écrire une fonction en OCaml `tri_fusion_array_v nuage` qui retourne le nuage trié `nuage_v` en utilisant le principe du tri fusion.

Nous appliquons maintenant le paradigme "diviser pour régner" en séparant notre nuage en deux nuages comprenant :
- d'une part les points rangés avant `nuage_h.(n/2)` dans `nuage_h`, `nuage_h.(n/2)` exclus, nuage noté `nuage_g` dans la suite,
- d'autre part les points rangés après `nuage_h.(n/2)` dans `nuage_h`, `nuage_h.(n/2)` inclus, nuage noté `nuage_d` dans la suite.



11. Expliquer en quoi ce partitionnement découpe le nuage `nuage_h` en deux sous-nuages de taille approximativement `n/2`.
12. Écrire une fonction en OCaml `separation nuage_h nuage_v` qui effectue ce découpage sur les deux nuages triés `nuage_h` et `nuage_v`, le découpage de `nuage_h` est immédiat, le découpage de `nuage_v` est en complexité linéaire.

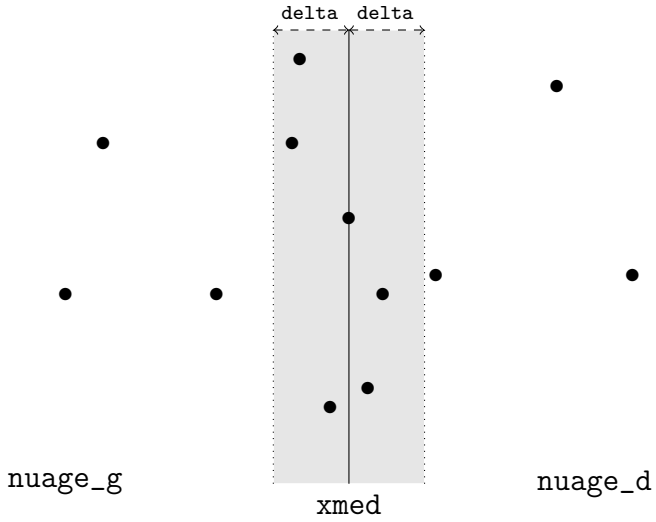
Le problème de recherche de minimum est relancé sur le nuage `nuage_g` et nous fournit une distance minimale `dg`, et sur le nuage `nuage_d` et nous fournit une distance minimale `dd`.



13. Notons `delta` la valeur minimale entre `dg` et `dd`. Cette valeur `delta` est-elle la valeur minimale attendue, c'est-à-dire la distance minimale entre deux points distincts du nuage ?
14. Il nous reste donc à étudier les distances entre deux points M_1 et M_2, l'un situé dans `nuage_g`, l'autre dans `nuage_d` et à retenir la distance minimale.

    Nous notons : `xmed = fst (nuage_h.(n/2))`.

    Montrer que si le point M_1 situé dans `nuage_g` a une abscisse strictement inférieure à `xmed-delta` ou si le point M_2 situé dans `nuage_d` a une abscisse strictement supérieure à `xmed+delta`, alors la distance de M_1 à M_2 est strictement supérieure à `delta`.



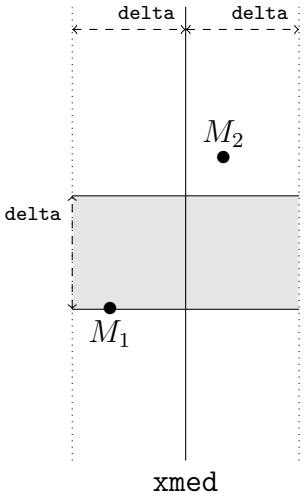
15. Nous pouvons alors limiter notre recherche aux points appartenant à `nuage` dont les abscisses sont comprises entre `xmed-delta` et `xmed+delta`, ce qui nous délimite une bande B du plan.

    Écrire une fonction en OCaml `bande_du_plan nuage_h nuage_v delta` qui détermine la liste `b` des points contenus dans la bande B triés par d'ordonnée croissante.

16. δ désigne la variable mathématique associée à la variable informatique `delta`, x_{med} désigne la variable mathématique associée à la variable informatique `xmed`.

    Considérons alors un point $M_1(x_1, y_1)$ appartenant à la bande B.

    Nous étudions alors la distance de M_1 à tout autre point de la bande B situé au-dessus de M_1, c'est-à-dire ayant une ordonnée supérieure ou égale à celle de M_1.

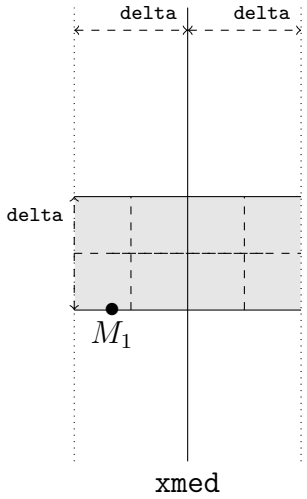


    Justifier que pour tout point $M_2(x_2, y_2)$ tel que : $y_2 > y_1 + \delta$, la distance de $M1$ à $M2$ est strictement supérieure à δ.

17. Nous pouvons donc limiter notre recherche à des points de la bande B d'ordonnée comprise en y_1 et $y_1 + \delta$.

    Justifier que dans un carré de coté de longueur $\delta/2$ situé dans le demi-plan : $x < x_{med}$, il existe au plus un point du nuage, et justifier que dans un carré de coté de longueur $\delta/2$ situé dans le demi-plan : $x \geq x_{med}$, il existe au plus un point du nuage.

18. En déduire qu'il existe au plus 7 points du nuage situé dans la bande B et au-dessus de M_1 avec une ordonnée inférieure ou égale à $y_1 + \delta$.



19. Nous pouvons ainsi limiter notre étude des distances des points du nuage à M_1 aux 7 points de la bande B situés immédiatement au dessus de M_1, et une distance strictement inférieure à δ nous amènera alors à actualiser la valeur de δ.

    Écrire une fonction en OCaml `minimum_bande b delta` qui, `b` et `delta` étant connus, détermine maintenant la distance minimale entre deux points du nuage.

20. Écrire une fonction en OCaml `distance_mini_dpr nuage` qui calcule la distance minimale entre deux points du nuage.

21. Évaluer la complexité de la fonction `distance_mini_dpr`.