DM d'Algorithmique
nov. 2020
Des points dans les arbres.

1 Objectif.

Comment représenter une liste $\mathcal{L}$ de points du plan $\mathcal{P}$ pour faciliter certaines recherches dans cette liste ?

2 Définitions.

Le plan $\mathcal{P}$ est muni d'un repère orthonormé $(O, \vec{i}, \vec{j})$. Chaque point M est défini par ses coordonées dans ce repère : x_M ou $M[0]$ désignent son abcisse et y_M ou $M[1]$ son ordonnée. Nous admettons que x_M et y_M peuvent prendre une valeur infinie ($+\infty$ ou $-\infty$).

Exemple 1. *La liste $\mathcal{L}$, citée dans les exemples du présent document, contient dans l'ordre les points suivants : $A = (7,6)$, $B = (2,10)$, $C = (11,3)$, $D = 10,9$, $E = (4,4)$, $F = (5,8)$, $G = (6,1)$, $H = (8,7)$, $I = (1,5)$, $J = (9,2)$, $K = (8,4)$, $L = (9,5)$, $M = (5,9)$, $N = (6,3)$, $O = (7,8)$, $P = (2,3)$.*

Une droite, demi-droite et segments sont dits horizontaux si ils sont parallèles à $\vec{i}$ et verticaux si parallèles à $\vec{j}$. Une *zone rectangulaire*, ou *zone* pour simplifier, est un ensemble de points de $\mathcal{P}$ défini à l'aide d'un rectangle dont les côtés sont horizontaux ou verticaux, donc pouvant être défini par son point inférieur gauche et son point supérieur droit. La zone z de point inférieur gauche I et de point supérieur droit S est notée $[I,S] = [(x_I, y_I), (x_S, y_S)]$ et est égale à $\{(x,y) \in \mathcal{P} | x_I < x \leq x_S \textit{ et } y_I < y \leq y_S\}$. Notez bien que la zone est 'fermée' sur son côté droit et sur son côté haut. Ex. $\mathcal{P} = [(-\infty, -\infty), (+\infty, +\infty)]$.

Nous notons *Point* le type de données représentant les points. Le code ci-dessous montre quelques manipulations autorisées de points.

```
p, q : Point                 // déclaration de variables de type Point
cs : entier ; cs = 1
p = (3, 7)                   // affectation dans une variable de type Point
q[cs] = p[(cs + 1) % 2]      //  q[1]=p[0] ou q.y = p.x
q.x = p[cs]                  //  q.x=p[1]  ou q.x=p.y
si (p = q) alors:            // comparaison de deux points
   afficher "p et q sont égaux"
```

3 Arbres de partition du plan en zones.

On implémente $\mathcal{L}$ à l'aide d'un arbre binaire de recherche de points *arb* particulier, dit *de partition en zones*. Intuitivement, l'insertion un à un des points de $\mathcal{L}$ dans *arb* partitionne $\mathcal{P}$ en zones rectangulaires. Chacune de ces zones est représentée par un sous-arbre de *arb*.

Définition. Un *arbre binaire de partition en zones*, ou de type *ABRZ* pour simplifier, est défini comme suit :

- **Initialisation**. L'arbre vide *None* et l'arbre racine $(p, cs, None, None)$, où $cs = 0$ et p un point quelconque, sont des arbres de partition de zones.
- **Induction**. $A = (p, cs, A_g, A_d)$ est un arbre de partition de zones ssi :

— p, dit la `valeur` de (la racine de) A, est un point.
— cs, dit la `coordonnée de séparation` de (la racine de) A, appartient à $\{0,1\}$.
— A_g et A_d, dits le `fils gauche` et le `fils droit` de A, sont des arbres de partition de zones.
— si $cs = 0$ alors tout noeud de A_g a une valeur dont l'abcisse est inférieure ou égale à l'abcisse de p et tout noeud de A_d a une valeur dont l'abcisse est strictement supérieure à l'abcisse de p.
— si $cs = 1$ alors tout noeud de A_g a une valeur dont l'ordonnée est inférieure ou égale à l'ordonnée de p et tout noeud de A_d a une valeur dont l'ordonnée est strictement supérieure à l'ordonnée de p.
— les coordonées de séparation de A_g et A_d sont égales à 1 si $cs = 0$ et sont égales à 0 si $cs = 1$ (on parle d'alternance des coordonnées de séparation.)

Notation. Le type de données des arbres de partition en zones, noté $ABRZ$, est défini comme suit.

```
NoeudZ = structure {
                 valeur: Point
                 cs : entier
                 gauche, droit : pointeur sur NoeudZ
                 }
ABRZ = pointeur sur NoeudZ
```

Exemple 2. *Le code ci-dessous crée un arbre racine arb de valeur* $(5,6)$ *et de coordonnée de séparation* 0.

```
arb : ABRZ  ; tmp = pointeur sur NoeudZ ; p : Point
p=(5,6)
tmp = nouveau(NoeudZ) ;
tmp->valeur = p ; tmp->cs = 0 ; tmp->gauche = None ; tmp->droit=None
arb = tmp
```

Pour simplifier, nous confondons parfois le noeud *noe* d'un arbre avec le point p qui est sa valeur : *'insérer un point à gauche de p'* veut aussi dire *'dans le fils gauche de noe'* et '*insérer à droite de p*' veut aussi dire '*dans le fils droit de noe*'.

Exemple 3. *Construisons l'arbre arb de type ABRZ obtenu à partir de l'arbre vide par insertion un à un des points de* $\mathcal{L}$*, dans l'ordre donné par* $\mathcal{L}$*.*
— *Le premier élément,* $A = (7,6)$*, de* $\mathcal{L}$ *définit un arbre racine arb de valeur* A *et de coordonée de séparation* 0.
— *Le point suivant,* $B = (2,10)$*, est inséré dans le fils gauche de arb car* $x_B \leq x_A$.
— *Le point suivant,* $C = (11,3)$*, est inséré dans le fils droit de arb car* $x_C > x_A$.
— $D = (10,9)$ *est inséré à droite de* A *et à droite de* C*, c-à-d. dans arb* $\rightarrow$ *droit* $\rightarrow$ *droit, car* $x_D > x_A$ *et* $y_D > y_C$.
— $E = (4,4)$ *est inséré à gauche de* A*, à gauche de* B*, c-à-d. dans arb* $\rightarrow$ *gauche* $\rightarrow$ *gauche car* $x_E \leq x_A$ *et* $y_E \leq y_B$.
— $F = (5,8)$ *est inséré à gauche de* A*, à gauche de* B *et à droite de* E*, c-à-d. dans arb* $\rightarrow$ *gauche* $\rightarrow$ *gauche* $\rightarrow$ *droite, car* $x_F \leq x_A$, $y_F \leq y_B$ *et* $x_F > x_E$.
— $G = (6,1)$ *est inséré à gauche de* A*, à gauche de* B*, à droite de* E *et à gauche de* F *car* $x_G \leq x_A$, $y_G \leq y_B$, $x_G > x_E$ *et* $y_G \leq y_F$
— $H = (8,7)$ *est inséré à droite de* A*, à droite de* C *et à gauche de* D *car* $x_H > x_A$, $y_H > y_C$ *et* $x_H \leq x_D$.
—

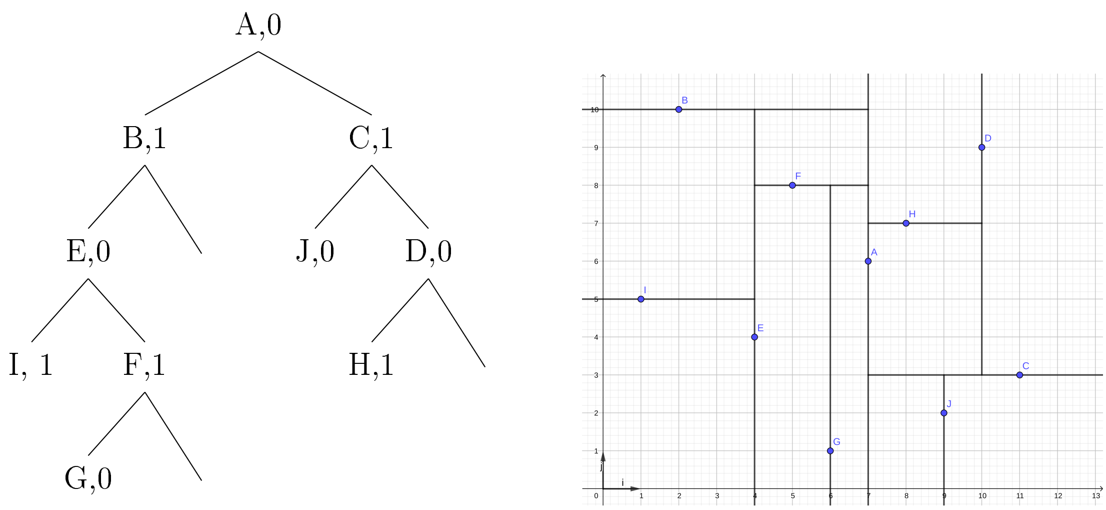

Fig. 1 – Un arbre de type ABRZ et son interprétation en partition en zones de $\mathcal{P}$.

Interprétation dans $\mathcal{P}$. Donnons une interprétation intuitive de arb dans $\mathcal{P}$. A chaque sous-arbre $sarb$ non vide de arb est associée une zone de $\mathcal{P}$ notée $zone(sarb)$ de la manière suivante :

- **Initialisation**. $zone(arb) = [(-\infty, -\infty), (+\infty, +\infty)]$
- **Induction**. Soit sn un noeud quelconque de arb, S le point valeur de sn, $sarb$ le sous arbre de arb de racine sn et $sz = zone(sarb)$. Alors :
  - si $sarb \rightarrow cs = 0$ alors la droite verticale passant par sn partitionne sz en une zone $zone(sarb \rightarrow gauche) = \{M \in z | x_M \leq x_S\}$, dite *gauche*, et une autre $zone(sarb \rightarrow droit) = \{M \in sz | x_M > x_S\}$, dite *droite.*
  - si $sarb \rightarrow cs = 1$ alors la droite horizontale passant par sn partitionne sz en une zone $zone(sarb \rightarrow gauche) = \{M \in sz | y_M \leq y_S\}$, dite *gauche* ou *basse*, et une autre $zone(sarb \rightarrow droit) = \{M \in sz | y_M > y_S\}$, dite *droite* ou *haute.*

Exemple 4. — *Soit* $\mathcal{P} = z = [(-\infty, -\infty), (+\infty, +\infty)]$

- $A = (7, 6)$ *sépare* z *selon la droite verticale d'équation* $\{x = 7\}$ *en* $zg = [(-\infty, -\infty), (7, +\infty)]$ *et* $zd = [(7, -\infty), (+\infty, +\infty)]$.
- $B = (2, 10)$ *sépare* zg *selon la droite horizontale d'équation* $\{y = 10\}$ *en* $zgb = [(-\infty, -\infty), (7, 10)]$ *et* $zgh = [(-\infty, 10), (7, +\infty)]$.
- $C = (11, 3)$ *sépare* zd *selon* $\{y = 3\}$ *en* $zdb = [(7, -\infty), (+\infty, 3)]$ *et* $zdh = [(7, 3), (+\infty, +\infty)]$.
- $D = (10, 9)$ *sépare* zdh *selon* $\{x = 9\}$ *en* $zdhg = [(7, 3), (10, +\infty)]$ *et* $zdhd = [(10, 3), (+\infty, +\infty)]$
- $E = (4, 4)$ *sépare* zgb *en* $zgbg = [(-\infty, -\infty), (4, 10)]$ *et* $zgbd = [(4, -\infty), (7, 10)]$
- $F = (5, 8)$ *sépare* $zgbd$ *en* $zgbdb = [(4, -\infty), (7, 8)]$ *et* $zgbdh = [(4, 8), (7, 10)]$
- *etc.*

Question 1. *Compléter la figure 1, en insérant dans l'arbre les autres points de* $\mathcal{L}$*, dans l'ordre donné par* $\mathcal{L}$*.*

Question 2. — *Ecrire une fonction creerArbre(p : Point, g : ABRZ, d : ABRZ, cs : entier) : ABRZ qui crée un arbre de valeur p, de fils gauche g, de fils droit d et de coordonnée de séparation cs.*

- *Ecrire une fonction inserer(arb : ABRZ, p : Point) : ABRZ qui insère un point p donné dans un arbre arb de type ABRZ donné. On suppose que tous les points de arb sont distincts : il ne faut pas re-insérer un point qui est déjà dans l'arbre.*

Indication. Voir l'exemple 2 et compléter le code ci-dessous ou donner votre propre code.

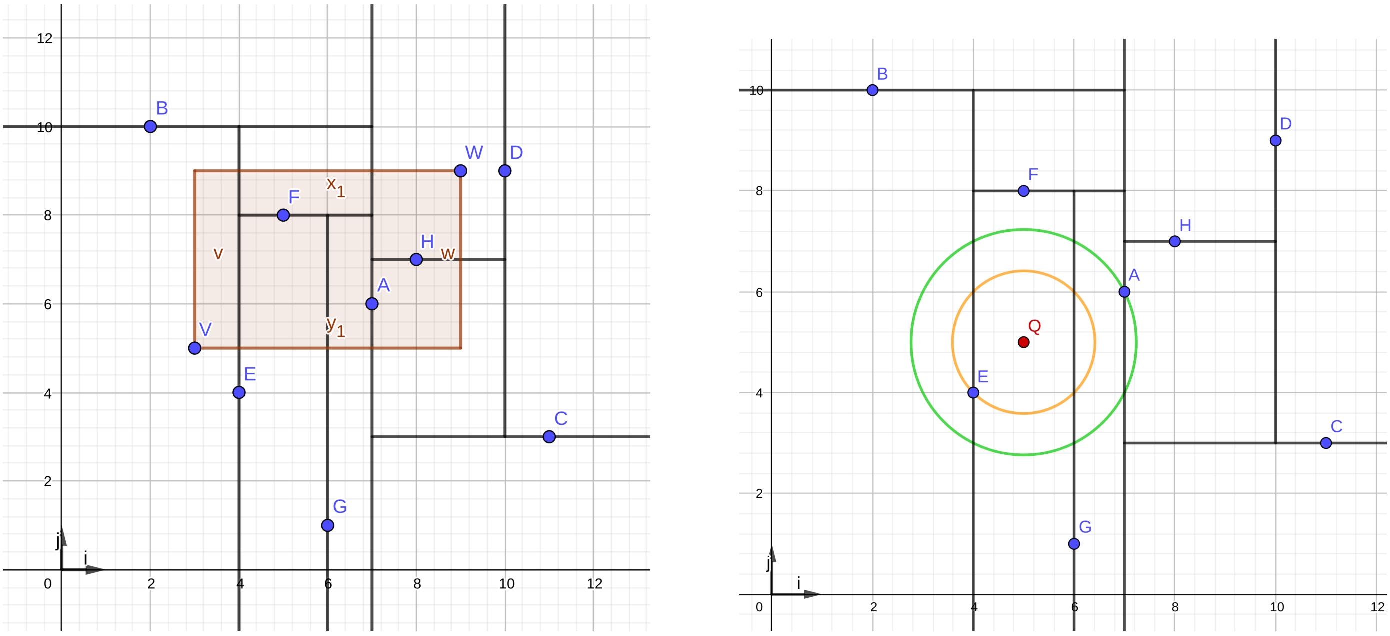

Fig. 2 – A gauche : l'intersection de `[v, w]` *avec l'arbre représentant cette partition contient* `A`, `F` *et* `H`. *A droite :* `Q` *est le point à approcher, le point le plus proche courant* `ppc` *prend la valeur* `A` *, puis* `E`

```
creerArbre(p:Point, g:ABRZ, d:ABRZ, cs:entier): ABRZ
...

inserer(arb: ABRZ, p: Point): ABRZ
    si (arb = None) alors retourner insererAux(arb, p, 0)
    sinon retourner insererAux(arb, p, arb->cs)

insererAux(arb: ABRZ, p: Point, cs: entier): ABRZ
    ptr : Point  //point à la racine
    si (arb = None) alors retourner ...
    sinon:
        ptr = arb->val
        si (ptr = p) alors ...
        si (cs=0) alors:
            si (p.x <= ptr.x) alors  arb->gauche = insererAux( ...)
            sinon arb->droit = ...
        sinon:     // cs=1
            ...

        retourner arb
```

Remarque. Noter que le code peut être raccourci si on accède aux coordonées d'un point p avec $p[0]$ et $p[1]$ et si on incrémente cs à *modulo* 2 près : $cs = 0 \Rightarrow (cs+1)\%2 = 1$ et $cs = 1 \Rightarrow (cs+1)\%2 = 0$.

3.1 Recherches dans un arbre de type $ABRZ$.

Sauf mention contraire, arb représente un arbre de type $ABRZ$ dans toute la suite.

Question 3. — *Ecrire une fonction* $recherche(arb : ABRZ,\ p : Point) : ABRZ$ *qui vérifie*

si un point donné p appartient ou non à arb. La fonction retourne le sous-arbre dont la racine est p ou l'arbre vide.
— *Complexité. Quel est au maximum le nombre de noeuds visités dans arb ?*

Question 4. 1. *Ecrire une fonction récursive* $minX(arb : ABRZ) : ABRZ$ *qui, étant donné un arbre non vide arb, recherche un point de arb dont l'abcisse est minimum. Retourner le sous arbre dont la racine contient ce point.*

2. *Appliquer cette fonction avec l'arbre obtenu dans la question 1. Donner alors la liste des points ainsi visités : A, ...*
3. **Option.** *Ecrire une fonction* $min(arb : ABRZ,\ dim : entier) : ABRZ$ *qui, étant donné un arbre non vide arb, recherche un point dont l'abcisse est minimum si* $dim = 0$ *et dont l'ordonnée est minimum si* $dim = 1$*. Retourner le sous-arbre dont la racine contient ce point.*

La question 3 en option peut remplacer la première, elle permet de gagner en plus la moitié de la note attribuée à la première.

Indications. *arb* est non vide donc arrêter les appels récursifs avant que *arb* soit vide. Distinguer les cas où
— *arb* est une feuille
— $arb \to cs = 0$
— $arb \to cs = 1$

Il est parfois utile de connaitre les points de $\mathcal{P}$ qui appartiennent à une droite donnée (ex. dans un jeu, quels sont les points traversés par un rayon laser ?)

Question 5. 1. *Ecrire une fonction* $dansDroiteV(arb : ABRZ,\ a : entier)$ *qui affiche les points de arb qui appartiennent à la droite verticale d'équation* $\{x = a\}$*.*

2. *Appliquer cette fonction avec l'arbre obtenu dans la question 1 et avec la droite verticale d'équation* $\{x = 9\}$*. Donner alors la liste des points ainsi visités : A, ...*
3. **Option.** *Ecrire une fonction* $dansDroite(arb : ABRZ,\ a : entier,\ dim : entier)$ *qui affiche les points de arb qui appartiennent à la droite verticale d'équation* $\{x = a\}$ *si* $dim = 0$ *et à la droite horizontale d'équation* $\{y = a\}$ *si* $dim = 1$*.*

La question 3 en option peut remplacer la première, elle permet de gagner en plus la moitié de la note attribuée à la première.

Question 6. *Ecrire une fonction* $intersection(arb : ABRZ,\ v : Point,\ w : Point)$ *qui affiche les points de arb qui appartiennent à une zone* $z = [v, w]$ *donnée.*

Indication . Visiter un à un les points de *arb*. Soit *pc* le point courant et supposons la coordonnée de séparation de ce point égale à 0. Vérifier si ce point appartient à z ou non. Puis, considérer la droite verticale (V_{pc}) passant par *pc*. Etudier les cas suivants (voir figure 3) :
— (V_{pc}) coupe z
— (V_{pc}) est à droite de z
— (V_{pc}) est à gauche de z

3.2 Recherche du point le plus proche.

On suppose que $dist(M, N)$ calcule la distance entre deux points M et N.

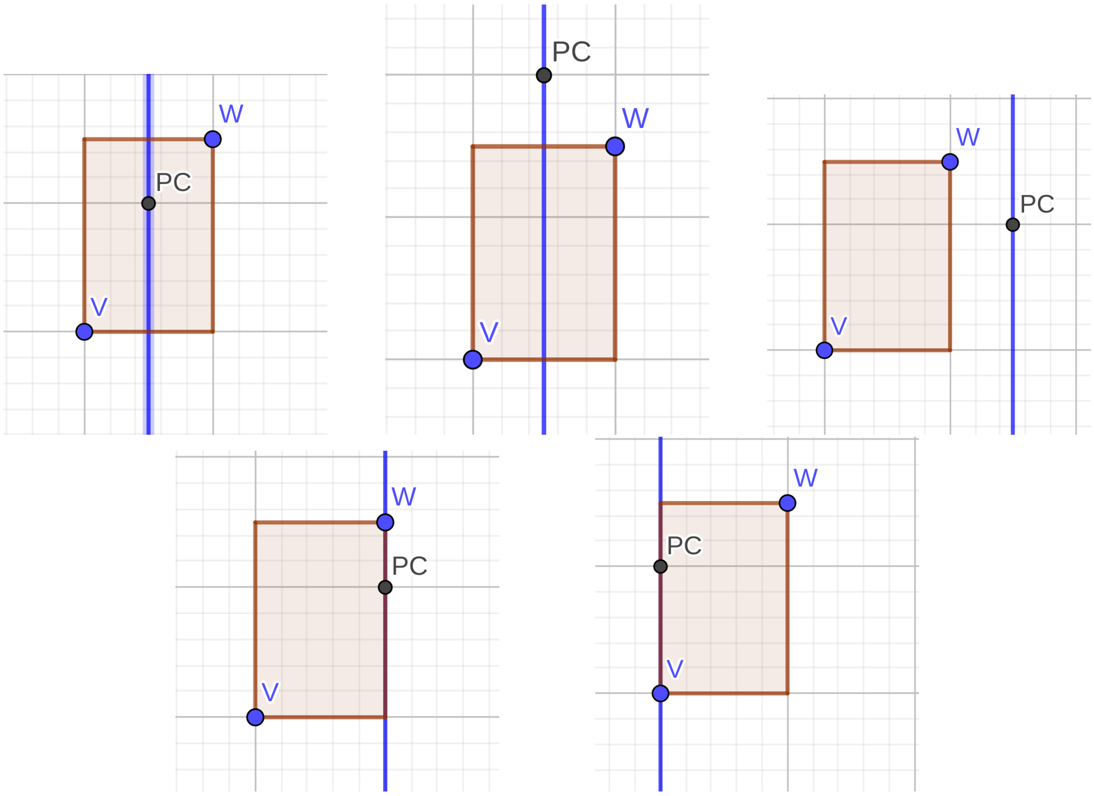

Fig. 3 – `[v, w]` *est la zone d'intersection,* `pc` *est le point courant,* `pc` $\rightarrow$ `cs = 0`

Question 7. — *Ecrire un fonction $plusproche(arb : ABRZ, q : Point) : ABRZ$ qui, étant donnée un arbre de points non vide arb, recherche le point de arb le plus proche d'un point donné q (en choisir un si il y en a plusieurs). Retourner le sous arbre de arb dont la racine est ce point.*

— *Appliquer cette fonction à l'arbre obtenu dans la question 1 et le point $q = (5, 5)$. Donner alors dans l'ordre la liste des points visités : A, ...*

Indication. Utiliser une fonction auxiliaire $plusprocheAux()$ qui possède deux paramètres en plus : ppc pour stocker le point le plus proche courant (initialisé avec la valeur de la racine de arb) et $dminc$ pour stocker $dist(p, ppc)$.

la figure 4 montre divers cas possibles où on est en train de visiter un point courant PC, où PPC est le point le plus proche courant, et où on cherche à voir si, oui ou non, Q est plus proche de PC que de PPC.

Exemple 5. *Soit $Q = (5, 5)$ le point à approcher. Soit pp le point le plus proche recherché, pc le point courant visité, ppc le point le plus proche courant, $dminc = dist(Q, ppc)$ la distance minimale courante et Cminc le cercle $Cercle(Q, dminc)$ de centre Q et de rayon dminc.*

— *Initialisation. $p = Q$, $pc = A$, $ppc = A$ et $dminc = dist(p, ppc)$.*

— *On recherche pp dans Cminc.*

— *La droite verticale (V_A) passant par A coupe Cminc donc pp peut se trouver dans la zone zg à gauche de (V_A) ou dans la zone zd à droite de (V_A).*

— *Recherchons d'abord pp dans zg. Le fils gauche de arb, qui est B, n'est pas dans Cminc. ppc change-t-il ?*

— *Soit (H_B) la droite horizontale passant par B. (H_B) partitionne zg en une zone zgb en bas de (H_B) et une zone zgh en haut de (H_B). Dans lesquelles de ces deux zones peut se trouver pp si H_B coupe Cminc ? si H_B ne coupe pas Cminc ?*

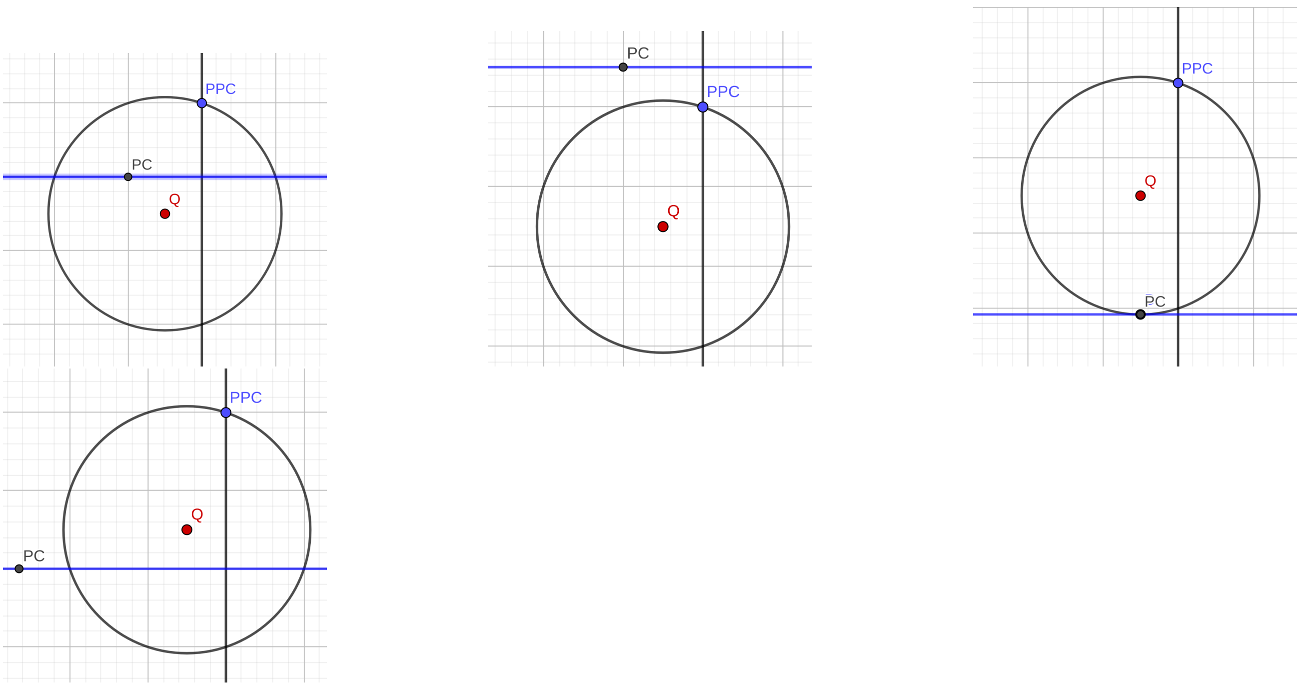

Fig. 4 – Le point `Q` *à approcher est-il plus proche du point courant visité PC que du point le plus proche courant* `PPC` *?*

- ...
- *Recherchons maintenant pp dans zd. Le fils droit de arb, qui est C, n'est pas dans Cminc. ppc change-t-il ?*
- *Soit* (H_C) *la droite horizontale passant par C.* (H_C) *partitionne zd en une zone zdb en bas de* (H_C) *et une zone zdh en haut de* (H_C)*. Dans lesquelles de ces deux zones peut se trouver pp si* H_C *coupe Cminc ? si* H_C *ne coupe pas Cminc ?*
- ...
- *Supposons que* pp_g *est le point le plus proche trouvé dans zg et* pp_d *celui trouvé dans zd. Lequel de ces deux points est le point le plus proche recherché ?*

Intersection d'un cercle et d'une droite Il est facile de tester si une droite horizontale ou verticale donnée coupe un cercle donné ou non. Donc utiliser, dans la question précédente, une fonction $position\_droite\_cercle(p : Point,\ cs : entier,\ cen : Point,\ ray : double) : entier$ qui teste si la droite passant par p, verticale si $cs = 0$ et horizontale si $cs = 1$, coupe ou non le cercle de centre cen et de rayon ray. On pourrait retourner 0 si la droite coupe le cercle, -1 si elle est en bas ou à gauche du cercle et 1 si elle est au dessus ou à droite du cercle.

4 Organisation.

Le DM comporte une partie théorique, et une implémentation en `C` ou en `Python`.

Le devoir se fait en binôme. Chaque binôme ne rend qu'un seul fichier compressé contenant un fichier pour la partie théorique et un autre fichier pour le code d'implémentation. Les noms et numéros d'étudiants du binôme doivent figurer dans ces documents et dans le nom du fichier compressé. Tout plagiat constaté entre binômes différents sera sanctionné sévèrement.

Pour toute information sur le sujet, écrire à `Solomampionona Ranaivoson`