Corrige DS - Paquet Cadeau

*Note finale = 0.4 * Note de Cours + 0.6 * Note de Problème.*

1 Exercices

c.f cours pour le corrigé.

1. Questions de cours coefficients 2
2. Questions algo coefficients 3

2 Problème - Algorithme du Papier Cadeau

2.1 Points

Question 2.1. *Distance*

Coefficient : 1

Attendus :
- Définition d'une fonction (3/9)
  - Syntaxe (1/9)
  - Arguments (2/9)
- Retour du résultat (2/9)
- Expression arithmétique avec puissance (3/9)
- Accès aux coordonnées (1/9)

```
def distance(a,b) :
    return  ( (a[0] - b[0]) ** 2 +  (a[1] - b[1]) ** 2 ) ** (0.5)
```

Question 2.2. *Orientation*

Coefficient : 3

Attendus :
- Arguments de la fonction (2/9)
- Calcul du produit vectoriel (2/9)
  - Accès aux coordonnées (1/9)
  - Expression arithmétique correcte (1/9)
- Calcul du résultat en fonction de condition (5/9)
  - Ecrire des comparaisons (1/9)
  - Faire une disjonction de cas (1/9)
  - Ecrire une conjonction dans une disjonction (2/9)

— Calcul de distance (1/9)

```
def trigo(a,b,c) :
    z_vec = ( a[0] - b[0] ) * ( c[1] - b[1] ) - (a[1] - b[1]) * (c[0] - b[0] )
    return ( a == b
         or z_vec < 0
         or (z_vec == 0 and a != c and distance(a,c) < distance(a,b) ) )
```

Question 2.3. *Ordre*

Coefficient : 2

Attendus :
- Arguments de la fonction (2/9)
- Renvoi d'un booléen (3/9)
- Traduction correcte de la condition (2/9)
- Écriture et calcul correct (2/9)

```
def inferieur(a,b) :
    return a[0] < b[0] or ( a[0] == b[0] and a[1] < b[1] )
```

2.2 Distances

Question 2.4. *Point minimum*

Coefficient : 5

Attendus :
- Faire un parcours d'éléments de la liste (3/9)
- Faire une comparaison en utilisant `inferieur` (2/9)
- Initialisation de la recherche (2/9)
- Algorithme de recherche de minimum correct (2/9)

```
def point_min(l) :
    if not l :
        return None
    m = l[0]
    for p in l :
        if inferieur(p,m) :
            m = p
    return m
```

Question 2.5. *Premier Entier*

Coefficient : 7

Attendus :
- Principe de l'algorithme (5/9) *Exemple pour l'algorithme du corrigé :*
  - Incrémentation d'un entier (2/9)
  - Départ à 0 (1/9)
  - Arrêt lorsque l'entier n'est pas dans la liste (2/9)
- Implémentation (4/9) *Exemple pour l'algorithme du corrigé :*
  - Recherche de l'entier dans la liste (2/9)
  - Arrêt de la boucle correct (1/9)
  - Bon retour (1/9)

```
def min_int_diff(t) :
    i = 0
    while i in t :
        i = i + 1
    return i
```

Question 2.6. *Minimum d'une liste extraite*

Coefficient : 8

Attendus :
- Principe de l'algorithme (4/9) *Exemple pour l'algorithme du corrigé :*
  - Parcours de la liste l (1/9)
  - Faire une comparaison en utilisant `inferieur` (1/9)
  - Vérification que l'indice n'est pas dans t (2/9)
- Implémentation (5/9) *Exemple pour l'algorithme du corrigé :*
  - Initialisations correcte (2/9)
  - Départ du parcours au bon indice (2/9)
  - Retours correct dans tous les cas (1/9)

```
def min_diff(l,t) :
    im = min_int_diff(t)
    if im < len(t) :
        m = l[im]
    for i in range(im+1,len(l)) :
        p = l[i]
        if i not in t and inferieur(p,m) :
            m = p
            im = i
    return im
```

2.3 Algorithme de Jarvis

Question 2.7. *Point le plus à gauche*

Coefficient : 6

Attendus :
- Faire un parcours d'éléments de la liste (2/9)
- Faire une comparaison en utilisant `trigo` (4/9)
  - Utilisation de trigo (1/9)
  - Utilisation des bons points dans le comparaison (3/9)
- Initialisation de la recherche (1/9)
- Algorithme de recherche de minimum correct (2/9)

```
def plus_a_gauche(p,l) :
    x = l[0]
    for x1 in l :
        if trigo(p,x1,x) :
            x = x1
    return x
```

Question 2.8. *Papier Cadeau*

Coefficient : 6

Attendus :
- Initialisation d'une liste vide et ajoute du plus petit point de l'ensemble (2/9)
- Calcul du point le plus à gauche (2/9)
  - appel de la fonction correct (1/9)
  - accès au dernier élément de la liste (1/9)
- Répétition (5/9)
  - Condition d'arrêt correcte (2/9)
  - Premier ajout avant test de la condition d'arrêt (3/9)

```
def enveloppe_convexe_jarvis(l) :
    e = [point_min(l)]
    e.append(point_plus_gauche(e[0],l))
    while e[-1] != e[0] :
        e.append(point_plus_gauche(e[-1],l))
    return e
```

2.4 Algorithme d'Andrew

Question 2.9. *Tri des points*

Coefficient : 9

Attendus :
- Initialisation correcte (1/9)
- Utilisation d'une nouvelle liste pour le résultat (1/9)
- Gestion des indices déjà ajoutés (3/9)
  - Création d'une liste d'indices initialement vide (1/9)

  - Ajout des indices des minimums ajoutés (2/9)
- Répétition correcte (3/9) :
  - Condition d'arrêt (2/9)
  - Utilisation d'une boucle et écriture correcte du corps (1/9)
- Calcul du minimum des indices non ajoutés et ajout dans le liste (1/9)

```
def tri(l) :
    t = []
    l_triee = []
    im = min_diff(l,t)
    while im != -1 :
        t.append(im)
        l_triee.append(l[im])
        im = min_diff(l,t)
    return l_triee
```

Question 2.10. *Enveloppe convexe*

Coefficient : 6

Attendus :
- Initialisation correcte et parcours de la liste (1/9)
- Élimination des points les plus à droite en haut (2/9)
- Élimination des points les plus à droite en bas (2/9)
- Ajout à la fin des listes de p (1/9)
- Concaténation finale correcte (3/9)

```
def enveloppe_convexe_andrew(l) :
    t = tri(l)
    haut = []
    bas = []
    for p in l :
        while len(haut) >= 2 and not trigo(p, haut[-1], haut[-2]) :
            haut.pop()
        haut.append(p)
        while len(bas) >= 2 and not trigo(bas[-2], bas[-1], p) :
            bas.pop()
        bas.append(p)
    return haut[:-1] + bas[:0:-1]
```