Corrigé — Paquet Cadeau

I Consignes

II Algorithme Classiques

Question 1. *Exponentiation rapide*

Question 2. *Comptage*

III Enveloppe Convexe

Question 3. *Distance*

coeff	syntaxe	accès	calcul
2	3	3	3

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

Question 4. *Orientation*

coeff	syntaxe	calcul	condition
2	2	2	5

```
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 5. *Point minimum*

coeff	cas vide	parcours	maj min
2	2	3	4

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

IV Jarvis

Question 6. *Point le plus à gauche*

coeff	parcours	appel	maj min
2	3	2	4

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

Question 7. *Papier Cadeau*

coeff	initialisation	boucle	ajouts
3	2	4	3

```
def 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
```

V Algorithme d'Andrew

Question 8. *Tri des points*

coeff	effet bord	calcul min	échange	élimination
3	1	2	3	3

```
def tri_points(l) :
    for i in range(len(l)-1):
        imin = i
        for j in range(i+1, len(l)):
            if l[j] < l[imin]:
                imin = j
        l[i], l[imin] = l[imin], l[i]
    return None
```

Question 9. *Algorithme d'Andrew*

coeff	initialisation	parcours	renversement	concatenation
4	2	4	2	1

```
def andrew(l) :
    t = l.copy()
    tri_point(t)
    haut = []
    bas = []
    for p in t :
        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)
    bas_renverse = [bas[i] for i in range(len(bas)-1,-1,-1)]
    return haut[:len(haut)-1] + bas_inverse[:len(bas_renverse)-1]
```