MP2I 2021/2022

Informatique
Correction - Devoir Maison 1

CIV

1 Préliminaires

Q01

```
int plus_bas(point *tab, int n)
{
    int i = 0;

    for (int j = 1; j < n; j++)
    {
        if (tab[j].y < tab[i].y)
            i = j;
    }

    return i;
}
```

Q02

On note y_i l'ordonnée de `tab[i]`.
On a l'invariant de boucle $P(i,j) = \forall k \in [\![1,0]\!] j-1, y_k \geq y_i$.

Initialisation En début de boucle, j vaut 1 et $P(0,1)$ vérifié directement.

Hérédité Si l'invariant $P(i,j)$ est vérifié en début de boucle, alors
- Soit $y_i \leq y_j$
- Soit $y_i > y_j$ et alors on passe à $P(j,j+1)$ en fin de boucle et on a pour $k \in [\![1,0]\!] j$, soit $k < j$ et ainsi $y_k \geq y_i > y_j$ soit $k = j$ et $y_k = y_j$. Dans tous les cas $y_k \geq y_j$.

L'invariant est donc valide et ainsi en sortie de boucle on a

$$P(i,n) = \forall k \in [\![1,0]\!] n-1, y_i \leq y_k$$

Q03

1. Positif.
2. Négatif.

Q04

```
int orient(point *tab, int i, int j, int k)
{
    point pi = tab[i];
    point pj = tab[j];
    point pk = tab[k];

    int a = (pk.y-pi.y)*(pj.x-pi.x) - (pk.x-pi.x)*(pj.y-pi.y);

    if (a < 0)
        return -1;
    if (a > 0)
        return 1;
    return 0;
}
```

2 Algorithme du paquet cadeau

Q05

C'est une question de maths ça...
Si p, q, r sont des points, on note ici $A(p,q,r)$ l'aire signée du triangle donnée par la formule du sujet.
La relation $\preceq$ est

réflexive soit q un point, on a directement $A(p_i, q, q) = 0$ donc $q \preceq q$.

antisymétrique soit $q = (x, y), r = (x', y')$ des points tels que $q \preceq r$ et $r \preceq q$. On a $A(p_i, q, r) = \frac{1}{2}((y' - y_i)(x - x_i) - (y - y_i)(x' - x_i)) = -\frac{1}{2}((y - y_i)(x' - x_i) - (y' - y_i)(x - x_i)) = -A(p_i, r, q)$.
Donc $A(p_i, q, r) \geq 0$ car $q \preceq r$ et $A(p_i, q, r) = -A(p_i, r, q) \leq 0$ car $r \preceq q$, donc $A(p_i, q, r) = 0$. Ainsi le triangle est d'aire nulle, donc il est plat, comme les points ne peuvent être alignés par hypothèse sur P, c'est forcément que $q = r$.

totale soit q, r des points, si on n'a pas $q \preceq r$, on a $A(p_i, q, r) < 0$ et donc $A(p_i, r, q) = -A(p_i, q, r) > 0$ donc on a bien $r \preceq q$. Dans tous les cas, soit $q \preceq r$, soit $r \preceq q$: la relation est bien totale.

transitive *Assez compliqué en fait.* Ici, on peut supposer par construction que p_i est dans l'enveloppe convexe de P.
On considère, par l'absurde, trois points p_j, p_k, p_l de $P \backslash \{p_i\}$ distincts tels que $p_j \preceq p_k$, $p_k \preceq p_l$ et $p_l \preceq p_j$. Comme les trois points ne sont pas alignés, il est possible d'exprimer p_i comme un barycentre de ceux-ci : il existe a, b, c tels que $a + b + c = 1$ et $x_i = ax_j + bx_k + cx_l$, $y_i = ay_j + by_k + cy_l$.
On montre alors par le calcul que $A(p_i, p_j, p_k) = cA(p_l, p_j, p_k)$, $A(p_i, p_k, p_l) = aA(p_l, p_j, p_k)$ et $A(p_i, p_l, p_j) = bA(p_l, p_j, p_k)$. Donc $a, b, c \geq 0$ et ainsi p_i est un point strictement intérieur au triangle de sommets p_j, p_k, p_l : il ne peut appartenir à l'enveloppe convexe. Contradiction.

Q06

```
int prochain_point(point *tab, int n, int i)
{
    int m = 0;

    if (i == 0) m = 1;
    for (int j = 1; j < n; j++)
    {
        if (j != i && orient(tab, i, m, j) <= 0)
            m = j;
    }

    return m;
}
```

Q07

- Au départ $m = 0$.
- Pour $j = 1$, on a $orient(10, 0, 1) \leq 0$, donc $m = j = 1$.
- Pour $j = 2$, on a $orient(10, 1, 2) \leq 0$, donc $m = j = 2$.
- Pour $j = 3$ et 4, le test est > 0 donc rien ne change.
- Pour $j = 4$, on a $orient(10, 2, 4) \leq 0$, donc $m = j = 4$.
- Pour les valeurs suivantes de j, le test est > 0 donc rien ne change.

Q08

```
int conv_jarvis(point *tab, int n, int *env)
{
    int taille = 1;
    env[0] = plus_bas(tab, n);

    int next = prochain_point(tab, n, env[0]);

    while(next != env[0])
    {
        env[taille] = next;
        taille += 1;
        next = prochain_point(tab, n, next);
    }

    return taille;
}
```

Q09 Si l'enveloppe convexe des n points en contient m, on effectue de l'ordre de m appels à `prochain_point` qui fait de l'ordre de n opérations élémentaires. Donc le cout de la boucle est de l'ordre de mn.

Le coût de l'initialisation avec `plus_bas` est de l'ordre de n.
Donc `conv_jarvis` est de coût temporel de l'ordre de mn, on notera qu'il est en $O(mn)$.

3 Intermède : piles d'entiers

4 Algorithme de balayage

Q10 Le tri fusion.

Q11

```
void maj_es(point *tab, stack es, int i)
{
    int j = pop(es);
    while(!is_empty(es) && orient(tab, i, j, top(es)) <= 0)
    {
        j = pop(es);
    }
    push(j, es);
    push(i, es);
}
```

Q12

```
void maj_ei(point *tab, stack ei, int i)
{
    int j = pop(ei);
    while(!is_empty(ei) && orient(tab, i, j, top(ei)) >= 0)
    {
        j = pop(ei);
    }
    push(j, ei);
    push(i, ei);
}
```

Q13

```
stack conv_graham(point *tab, int n)
{
    stack es = new_stack();
    stack ei = new_stack();

    push(0, es);
    push(0, ei);

    for(int i = 1; i < n; i++)
    {
        maj_es(tab, es, i);
        maj_ei(tab, ei, i);
    }

    print_stack(es);
    print_stack(ei);

    pop(es); // dernier sommet en commun
    while(!is_empty(es))
        push(pop(es), ei);
    pop(ei); // premier sommet en commun

    destroy_stack(es);
    return ei;
}
```

Q14

Les seules opérations signifiantes ici sont les opérations manipulant les piles. L'algorithme effectue de l'ordre n opérations élémentaires car chaque point est ajouté et retiré au plus une fois dans chaque pile.
Si on considère le tri initial, on effectue donc de l'ordre de $n \log n$ opérations élémentaires. On dira que l'algorithme est en $O(n \log n)$.
Remarque pour un nuage de points proche de son enveloppe convexe, cet algorithme est bien plus efficace que celui de Jarvis. Cas dégénéré pour Jarvis : n point sur un cercle, donc qui sont tous dans leur enveloppe convexe. Jarvis sera quadratique (en n^2) alors que Graham est quasi-linéaire (en $n \log n$).