PCSI 3 Lycée Sainte-Geneviève
2020/2021 Jeudi 6 mai 2021

DS 2 d'informatique

2 heures

Lisez attentivement les consignes ci-dessous

- *Les documents, calculatrices et autres appareils électroniques sont interdits.*
- *La qualité de la rédaction sera prise en compte dans l'évaluation.*
- *La présentation de la copie sera prise en compte dans l'évaluation :*
  - ▷ *Encadrez les résultats principaux.*
  - ▷ *Soignez votre écriture.*
  - ▷ *Maintenez une marge dans vos copies et aérez vos copies.*
  - ▷ *Numérotez vos copies.*
  - ▷ *Si vous constatez ce qui vous semble être une erreur d'énoncé, signalez-le sur votre copie en expliquant les initiatives que vous avez été amené à prendre.*
- ***Vos programmes doivent être clairs**.*
  - ▷ *Choisissez judicieusement les noms de vos variables ainsi que les noms de vos fonctions auxiliaires.*
  - ▷ *Aérez vos programmes.*
  - ▷ *Faites des indentations suffisamment grandes.*
  - ▷ ***Indiquez les indentations par des traits verticaux**.*
  - ▷ *Si nécessaire, commentez vos programmes en utilisant une couleur secondaire.*

Problème I : Une intégrale classique

Introduction, notations

- *Le but de ce problème est de calculer une valeur approchée de l'intégrale* $\displaystyle\int_0^A e^{-t^2}\,\mathrm{d}t$ *pour tout* $A > 0$ *et de conjecturer la valeur de* $\displaystyle\lim_{A\to+\infty}\int_0^A e^{-t^2}\,\mathrm{d}t$.
- *Dans ce problème, on note* $f : \begin{cases} \mathbb{R} \longrightarrow \mathbb{R} \\ t \longmapsto e^{-t^2}. \end{cases}$
- *Dans ce problème, l'utilisation des fonctions* `exp` *et* `log`*, que ce soient celles du package* `numpy` *ou* `math`*, est interdite.*

1. (a) Donner une équation différentielle simple vérifiée par f.

(b) Donner une condition initiale simple vérifiée par f.

2. Écrire une fonction `fonctionF(a, N)`, où $a > 0$ et $N \in \mathbb{N}^*$, qui renvoie la liste des valeurs approchées de f aux `N+1` points du segment $[0, a]$ découpés en `N` intervalles égaux.
On procèdera en suivant la méthode d'Euler.

3. Écrire une fonction `integrale(A, N)`, où $A > 0$ et $N \in \mathbb{N}^*$, qui renvoie une valeur approchée de $\displaystyle\int_0^A e^{-t^2}\,\mathrm{d}t$.
On procèdera en suivant la méthode des trapèzes.

Résultats admis

- *On admet :*
  (1) *que la limite* $\displaystyle\ell := \lim_{A\to+\infty}\int_0^A e^{-t^2}\,\mathrm{d}t$ *existe et est finie ;*
  (2) *que, si* $A \geqslant 1$*, on a* $\displaystyle\ell - e^{-A} \leqslant \int_0^A e^{-t^2}\,\mathrm{d}t \leqslant \ell$.
- *On admet que la méthode des trapèzes appliquée à* f *sur le segment* $[a, b]$ *découpé en* n *intervalles égaux donne une valeur approchée de* $\displaystyle\int_a^b f(t)\,\mathrm{d}t$ *avec une erreur majorée par* $\dfrac{(b-a)^3}{12n^2} \times M$ *où* M *est une borne de* $|f''|$ *sur* $[a, b]$.

4. (a) Montrer que la fonction $\begin{cases} \mathbb{R}_+ \longrightarrow \mathbb{R} \\ x \longmapsto xe^{-x} \end{cases}$ atteint son maximum en un point qu'on précisera avec une valeur qu'on précisera.

(b) En admettant que $2 < e$, montrer que $\forall x \geqslant 0,\ |xe^{-x}| \leqslant \dfrac{1}{2}$.

(c) Montrer que $\forall x \in \mathbb{R}_+,\ |f''(x)| \leqslant 4$.

5. Proposez une fonction `valeurApprochee(eps)`, où `eps` > 0, qui renvoie une valeur approchée de ℓ à `eps` près.

6. Démontrer les résultats admis (1) et (2).
Cette question est à traiter après avoir fini le reste du sujet.

Problème II : Enveloppes convexes dans le plan

Ce sujet a pour objectif de calculer des enveloppes convexes de nuages de points dans le plan affine, un grand classique de la géométrie algorithmique. On rappelle qu'un ensemble $C \subset \mathbb{R}^2$ est convexe si et seulement si pour toute paire de points $p, q \in C$ le segment de droite $[p, q]$ est inclus dans C. L'enveloppe convexe d'un ensemble $P \subset \mathbb{R}^2$, notée $\mathrm{Conv}(P)$, est le plus petit convexe contenant P. Dans le cas où P est un ensemble fini (appelé nuage de points), le bord de $\mathrm{Conv}(P)$ est un polygone convexe dont les sommets appartiennent à P, comme illustré dans la figure 1.

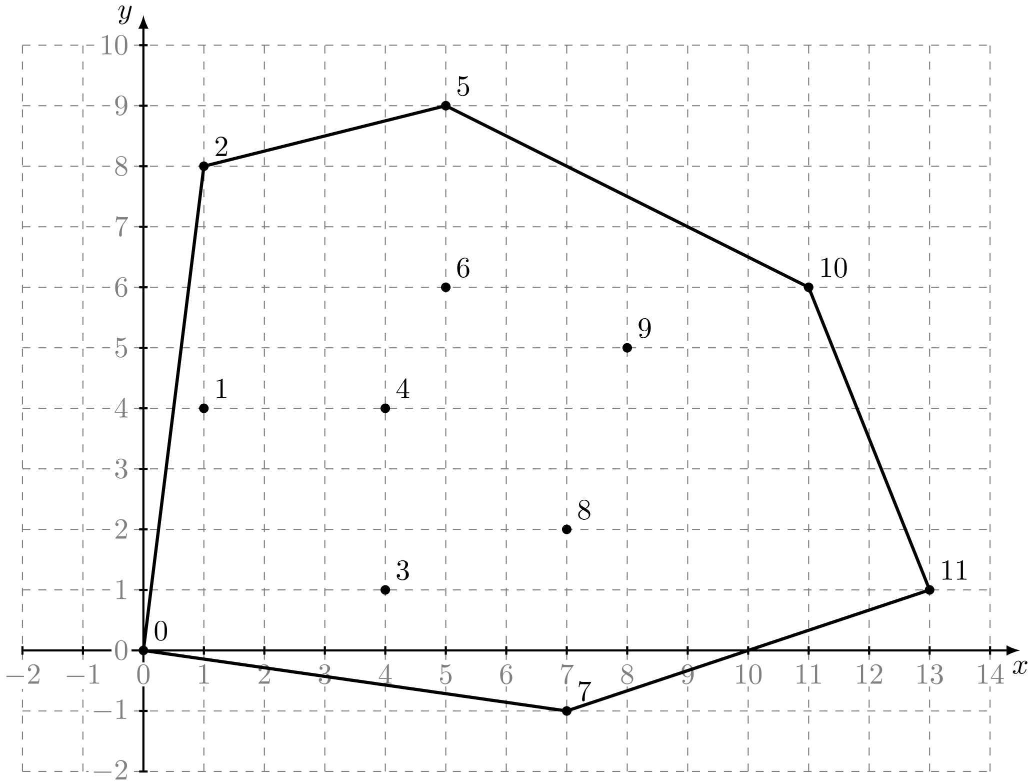

FIGURE 1 – Un nuage de points, numérotés de 0 à 11 et le bord de son enveloppe convexe.

On rappelle que le temps d'exécution d'un programme A (fonction ou procédure) est le nombre d'opérations élémentaires (comparaisons, additions, soustractions, multiplications, divisions, affectations, *etc.*) nécessaires à l'exécution de A. Sauf mention contraire dans l'énoncé du sujet, le candidat n'aura pas à justifier des temps de calculs de ses programmes. Toutefois, il devra veiller à ce que ces derniers ne dépassent pas les bornes prescrites.

Dans toute la suite, on supposera que le nuage de points P est de taille $n \geqslant 3$ et est en position générale, c'est-à-dire qu'il ne contient pas 3 points distincts alignés.

Nos programmes prendront en entrée un nuage de points P dont les coordonnées sont stockées dans un tableau `tab` à 2 dimensions, comme dans l'exemple ci-dessous qui contient les coordonnées du nuage de points de la figure 1.

i \ j	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
0	0	1	1	4	4	5	5	7	7	8	11	13
1	0	4	8	1	4	9	6	-1	2	5	6	1

Le tableau `tab` prend en `Python` la forme d'une liste de listes :

```
tab = [ [0, 1, 1, 4, 4, 5, 5, 7, 7, 8, 11, 13],
        [0, 4, 8, 1, 4, 9, 6, -1, 2, 5, 6, 1] ]
```

Partie I – Questions préliminaires

1. Écrire une fonction `plusBas(tab)` qui prend en paramètre un tableau `tab` de taille $2 \times n$ et qui renvoie l'indice j du point le plus bas (c'est-à-dire de plus petite ordonnée) parmi les points du nuage P. En cas d'égalité, votre fonction devra renvoyer l'indice du point de plus petite abscisse parmi les points les plus bas.
Sur le tableau exemple précédent, le résultat de la fonction doit être l'indice 7.

Dans la suite, nous aurons besoin d'effectuer un test de type géométrique : celui de l'orientation.

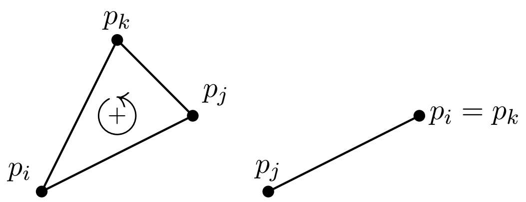

FIGURE 2 – Test d'orientation sur (p_i, p_j, p_k) : positif à gauche, nul au centre, négatif à droite

Définition 1. *Étant donnés trois points p_i, p_j, p_k du nuage P, distincts ou non,* le test d'orientation *renvoie* $+1$ *si la séquence* (p_i, p_j, p_k) *est orientée positivement,* -1 *si elle est orientée négativement, et 0 si les trois points sont alignés (c'est-à-dire si deux au moins sont égaux d'après l'hypothèse de position générale).*

Pour déterminer l'orientation de (p_i, p_j, p_k), il suffit de calculer l'aire signée du triangle, comme illustré sur la figure 2. Cette aire vaut la moitié du déterminant de la matrice 2×2 formée par les coordonnées des vecteurs $\overrightarrow{p_i p_j}$ et $\overrightarrow{p_i p_k}$.

On rappelle que le déterminant d'une matrice $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ vaut

$$\det \begin{pmatrix} a & b \\ c & d \end{pmatrix} = ad - bc.$$

2. Sur le tableau précédent, donner le résultat du test d'orientation pour les choix d'indices suivantes :

(a) $i = 7, j = 3, k = 4$

(b) $i = 8, j = 9, k = 10$

3. Écrire une fonction `orient(tab, i, j, k)` qui prend en paramètres le tableau `tab` et trois indices de colonnes, potentiellement égaux, et qui renvoie le résultat -1, 0, ou $+1$ du test d'orientation sur la séquence (p_i, p_j, p_k) de points de P.

Partie II – L'algorithme du paquet cadeau

Cet algorithme a été proposé par R. Jarvis en 1973. Il consiste à envelopper peu à peu le nuage de points P dans une sorte de paquet cadeau, qui à la fin du processus est exactement le bord de $\mathrm{Conv}(P)$. On commence par insérer le point de plus petite ordonnée (et parmi ceux-ci le plus à gauche, celui d'indice 7 dans l'exemple précédent) dans le paquet cadeau, puis à chaque étape de la procédure on sélectionne le prochain point du nuage P à insérer.

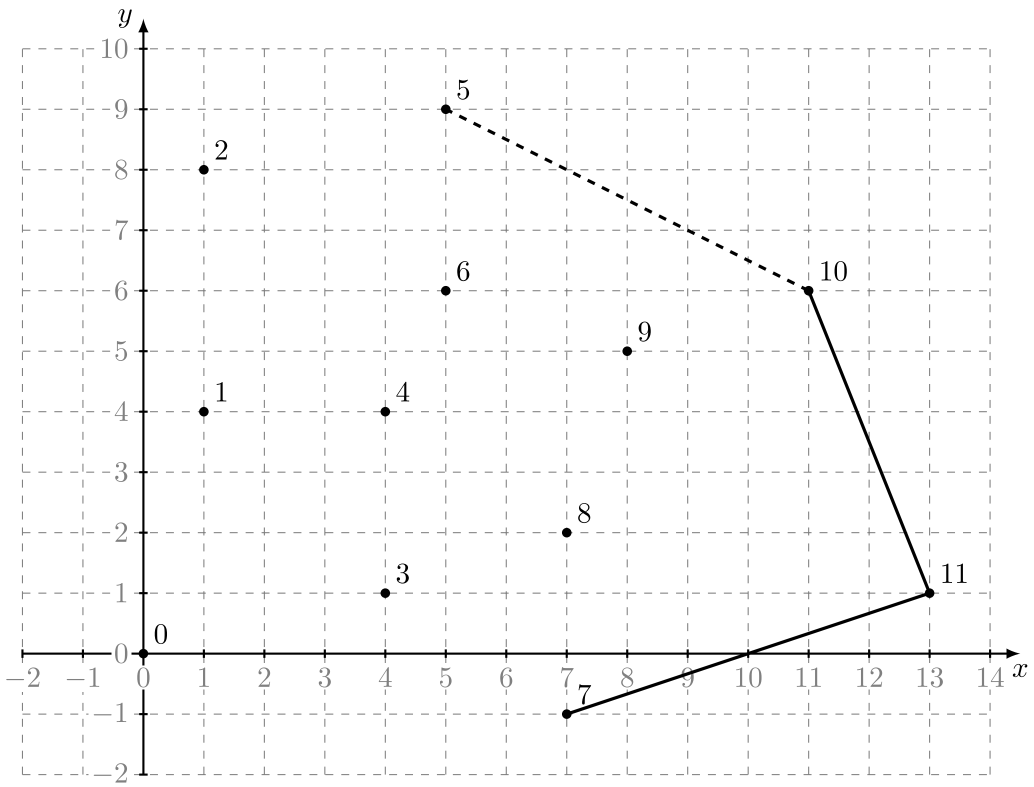

FIGURE 3 – Mise à jour du paquet cadeau après insertion du point p_{10}.

La procédure de sélection fonctionne comme suit. Soit p_i le dernier point inséré dans le paquet cadeau à cet instant. Par exemple $i = 10$ dans l'exemple de la figure 3.

Considérons la relation $\preccurlyeq$ définie sur l'ensemble $P \setminus \{p_i\}$ par :

$$p_j \preccurlyeq p_k \iff \texttt{orient(tab, i, j, k)} \leqslant 0$$

On admet le fait que $\preccurlyeq$ est une relation d'ordre totale sur l'ensemble $P \setminus \{p_i\}$:

- (réflexivité) pour tout $j \neq i$, $p_j \preccurlyeq p_j$
- (antisymétrie) pour tous $j, k \neq i$, $p_j \preccurlyeq p_k$ et $p_k \preccurlyeq p_j$ implique $p_j = p_k$
- (transitivité) pour tous $j, k, l \neq i$, $p_j \preccurlyeq p_k$ et $p_k \preccurlyeq p_l$ implique $p_j \preccurlyeq p_l$
- (totalité) pour tous $j, k \neq i$, $p_j \preccurlyeq p_k$ ou $p_k \preccurlyeq p_j$.

Ainsi, le prochain point à insérer (le point d'indice 5 dans la figure 3) est l'élément maximum de $P \setminus \{p_{10}\}$ pour la relation d'ordre $\preccurlyeq$. Il peut se calculer en temps linéaire (c'est-à-dire majoré par une constante fois n) par une simple itération sur les points de $P \setminus \{p_i\}$.

4. Décrire une réalisation en `Python` de la procédure. Elle prendra la forme d'une fonction `prochainPoint(tab, i)` qui prend en paramètre le tableau `tab` de taille $2 \times n$ ainsi que l'indice `i` du point inséré en dernier dans le paquet cadeau, et qui renvoie l'indice du prochain point à insérer.

Le temps d'exécution de votre fonction doit être majoré par une constante fois `n`*, pour tous* `n` *et* `i`*. La constante doit être indépendante de* `n` *et* `i` *et on ne demande pas de la préciser.*

5. Décrire à la main le déroulement de la procédure `prochainPoint` sur l'exemple de la figure 3. Plus précisément, indiquer la séquence de points de $P \setminus \{p_{10}\}$ considérés et la valeur de l'indice du maximum à chaque itération.

On peut maintenant combiner la fonction `prochainPoint` avec la fonction `plusBas` de la question **1.** pour calculer le bord de l'enveloppe convexe de P. On commence par insérer le point p_i d'ordonnée la plus basse, puis on itère le processus de mise à jour du paquet cadeau jusqu'à ce que le prochain point à insérer soit de nouveau p_i. À ce moment-là, on renvoie le paquet cadeau comme résultat sans insérer p_i une seconde fois.

Comme la taille du paquet cadeau augmente peu à peu lors du processus, et qu'à la fin elle peut être petite par rapport au nombre n de points de P, nous stockerons les indices des points du paquet cadeau dans une liste. Par exemple sur le nuage de la figure 1, le résultat sera la liste

```
[7, 11, 10, 5, 2, 0]
```

6. Écrire une fonction `convJarvis(tab)` qui prend en paramètre le tableau `tab` de taille $2 \times n$ représentant le nuage P, et qui renvoie une liste contenant les indices des sommets du bord de l'enveloppe convexe de P, sans doublon. Le temps d'exécution de votre fonction doit être majoré par une constante fois nm, où m est le nombre de points de P situés sur le bord de $\mathrm{Conv}(P)$.

7. Justifier brièvement le temps d'exécution de l'algorithme du paquet cadeau.

Intermède : piles d'entiers

Dans la suite nous aurons besoin d'utiliser des piles d'entiers, dont on rappelle la définition ci-dessous :

Définition 2. *Une* pile d'entiers *est une structure de données permettant de stocker des entiers et d'effectuer les opérations suivantes en temps constant (indépendant de la taille de la pile) :*

- *créer une nouvelle pile vide,*
- *déterminer si la pile est vide,*
- *insérer un entier au sommet de la pile,*
- *déterminer la valeur de l'entier au sommet de la pile,*
- *retirer l'entier au sommet de la pile.*

Nous supposerons fournies les fonctions suivantes, qui réalisent les opérations ci-dessus et s'exécutent chacune en temps constant :

- `newStack()` : qui ne prend pas d'argument et renvoie une pile vide ;
- `isEmpty(s)` : qui prend une pile `s` en argument et renvoie `True` ou `False` suivant que `s` est vide ou non ;
- `push(i, s)` : qui prend un entier `i` et une pile `s` en argument, insère `i` au sommet de `s` (c'est-à-dire à la fin de la liste), et ne renvoie rien ;
- `top(s)` : qui prend une pile `s` (supposée non vide) en argument et renvoie la valeur de l'entier au sommet de `s` (c'est-à-dire à la fin de la liste) ;
- `pop(s)` : qui prend une pile `s` (supposée non vide) en argument, supprime l'entier au sommet de `s` (c'est-à-dire à la fin de la liste) et renvoie sa valeur.

Dans la suite, il vous est demandé de manipuler les piles uniquement au travers de ces fonctions, sans aucune hypothèse sur la représentation effective des piles en mémoire.

Partie III – Algorithme de balayage

Cet algorithme a été proposé par *R.* Graham en 1972. Nous allons écrire la variante (plus simple) proposée par A. Andrew quelques années plus tard.

À partir de maintenant, on supposera que les points fournis en entrée sont triés par abscisse croissante, comme c'est le cas dans l'exemple du tableau `tab` donné au début du sujet.

L'idée de l'algorithme est de balayer le nuage de points horizontalement de gauche à droite par une droite verticale, tout en mettant à jour l'enveloppe convexe des points de P situés à gauche de cette droite, comme illustré dans la figure 4.

Plus précisément, l'algorithme visite chaque point de P une fois, par ordre croissant d'abscisse (donc par ordre croissant d'indice de colonne dans le tableau `tab` car celui-ci est trié). À chaque nouveau point p_i visité, il met à jour le bord de l'enveloppe convexe du sous-nuage $\{p_0, \ldots, p_i\}$ situé à gauche de p_i. On remarque que les points p_0 et p_i sont sur ce bord, et on appelle enveloppe supérieure la partie du bord de $\mathrm{Conv}\,\{p_0, \ldots, p_i\}$ située au-dessus de la droite passant par p_0 et p_i (p_0 et p_i compris), et enveloppe inférieure la partie du bord de $\mathrm{Conv}\,\{p_0, \ldots, p_i\}$ située au-dessous (p_0 et p_i compris). Le bord de $\mathrm{Conv}\,\{p_0, \ldots, p_i\}$ est donc constitué de l'union de ces deux enveloppes, après suppression des doublons que sont p_0 et p_i.

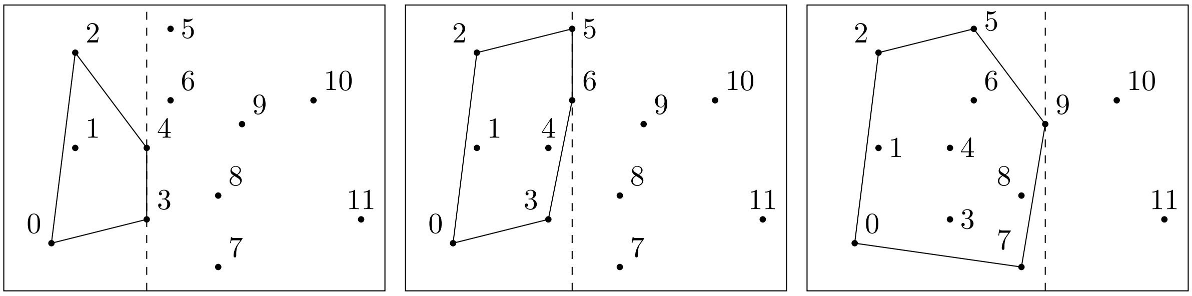

FIGURE 4 – Diverses étapes dans la procédure de balayage. La droite de balayage est en tirets.

Par exemple, dans le cas du nuage P de la figure 4 gauche, le sous-nuage $\{p_0, p_1, p_2, p_3, p_4\}$ a pour enveloppe supérieure la séquence (p_0, p_2, p_4) et pour enveloppe inférieure la séquence (p_0, p_3, p_4), le bord de son enveloppe convexe étant donné par la séquence (p_0, p_3, p_4, p_2)

Informatiquement, les indices des sommets des enveloppes inférieure et supérieure seront stockés dans deux piles d'entiers séparées, `ei` (pour enveloppe inférieure) et `es` (pour enveloppe supérieure).

La mise à jour de l'enveloppe supérieure est illustrée dans la figure 5 : tant que le point visité (p_9 dans ce cas) et les deux points dont les indices sont situés au sommet de la pile `es` (dans l'ordre : p_8 et p_5) forment une séquence (p_9, p_8, p_5) d'orientation négative, on dépile l'indice situé au sommet de `es` (8 dans ce cas). On poursuit ce processus d'élimination jusqu'à ce que l'orientation devienne positive ou qu'il ne reste plus qu'un seul indice dans la pile. L'indice du point visité (p_9 dans ce cas) est alors inséré au sommet de `es`. La mise à jour de l'enveloppe inférieure s'opère de manière symétrique.

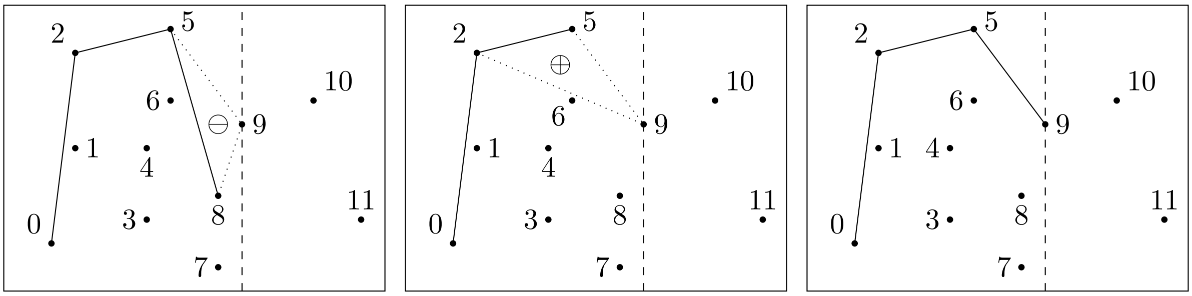

FIGURE 5 – Mise à jour de l'enveloppe supérieure lors de la visite du point p_9.

8. Écrire une fonction `majES(tab, es, i)` qui prend en paramètre le tableau `tab` ainsi que la pile `es` et l'indice `i` du point à visiter, et qui met à jour l'enveloppe supérieure du sous-nuage. Le temps d'exécution de votre fonction doit être majoré par une constante fois `i`.

9. Écrire une fonction `majEI(tab, ei, i)` qui effectue la mise à jour de l'enveloppe inférieure, avec le même temps d'exécution.

10. Écrire maintenant une fonction `convGraham(tab)` qui prend en paramètre le tableau `tab` de taille $2 \times n$ représentant le nuage P, et qui effectue le balayage des points de P comme décrit précédemment. On supposera les colonnes du tableau `tab` déjà triées par ordre croissant d'abscisse. La fonction doit renvoyer une pile `s` contenant les indices des sommets du bord de $\text{Conv}(P)$ triés dans l'ordre positif d'orientation, à commencer par le point p_0.