1 Un nouveau tri : le tri par sélection

On considère une liste, L, de nombres que l'on souhaite trier dans l'ordre croissant. Le principe du tri par sélection est le suivant :

- On parcourt la liste L pour trouver le minimum.
- On échange ce minimum avec le premier élément de la liste.
- On recommence le parcours de la liste à partir du second élément, puisque le premier est bien placé, en trouvant le nouveau minimum, que l'on placera à la seconde place.
- On recommence ce procédé jusqu'à trier l'avant dernier élément de la liste, le dernier élément sera alors automatiquement le plus grand et la liste sera triée.

On va implémenter ce tri, en procédant par étapes et en créant plusieurs fonctions.

1. Écrire une fonction *indmin* qui prend en paramètre une liste L et un entier i et renvoie un indice correspondant au minimum de la liste L que l'on parcourt à partir de l'indice i. Par exemple si $L = [1,\ 2,\ 3,\ 7,\ 5,\ 9]$ et que l'on utilise $indmin(L,\ 3)$ on doit obtenir 4 puisque l'indice du minimum de la liste que l'on parcourt à partir de l'indice 3 est 4 (5 étant le plus petit élément parmi ceux restants).
2. Écrire une fonction *tri_sel* qui prend en paramètre une liste L et renvoie la liste triée dans l'ordre croissant avec le tri sélection. On utilisera la fonction *indmin*.
3. Combien de comparaisons effectue cet algorithme ? Quelle est sa complexité ?

2 Recherche d'un motif dans un texte

La question est de déterminer si un motif est présent ou absent dans un texte. Le texte et le motif sont représentés par des chaînes de caractères.

1. **Un cas particulier.** Dans le cas particulier où le motif recherché ne comporte qu'un seul caractère, écrire une fonction $recherche\_lettre(texte,\ lettre)$ qui prend en paramètre une chaîne de caractères et une lettre et renvoie un indice où se trouve la lettre si elle est présente dans la chaîne de caractères et $False$ sinon.
2. **Recherche naïve.** Le principe est le suivant :

   - on cherche la présence du premier caractère du motif dans le texte,
   - si on le trouve, on vérifie si les caractères suivants du motif coïncident avec ceux du texte,
   - si tous les caractères du motif coïncident, le texte est trouvé, sinon on reprend la recherche de la présence du premier caractère.

   (a) Écrire une fonction $recherche(texte,\ motif)$ qui réalise cet algorithme, on pourra utiliser une boucle for pour parcourir le texte à la recherche du premier caractère et une boucle *while* pour tester la coïncidence des caractères une fois le premier trouvé.

   (b) Pour évaluer la complexité de cet algorithme, on s'intéresse au nombre de comparaisons que l'on fait entre les lettres du texte et les lettres du motif. On note n la longueur du texte et m la longueur du motif. Montrer que le nombre total de comparaisons est majoré par $m(n-m+1)$. Justifier que si $m = \dfrac{n}{4}$ alors la complexité est quadratique.

3 Trier des points

On dispose de points dans le plan muni d'un repère orthonormé. Chaque point possède un couple de coordonnées, représenté par le tuple (x, y). Nous allons nous intéresser à trois problèmes :

- trouver le point le plus éloigné de l'origine,
- trouver les deux points les plus proches,
- trier les points selon leur distance à l'origine.

Plus précisément, nous étudierons la distance au carré pour éviter l'usage de la racine carrée.

1. (a) Écrire une fonction *distance* qui prend en paramètre un couple de coordonnées représentant un point du plan et renvoie le carré de la distance du point à l'origine. Par exemple *distance*((3, 4)) doit renvoyer 25.
   (b) Écrire une fonction qui prend en paramètre une liste de points, toujours représentés sous la forme de tuples, et renvoie les coordonnées du point le plus éloigné de l'origine du repère.
2. (a) Écrire une fonction *distance*2 qui prend en paramètre deux points et renvoie la distance au carré entre ces deux points.
   (b) Écrire une fonction qui prend en paramètre une liste de points et renvoie les coordonnées des deux points les plus proches.
3. Écrire une fonction *tri_points*(L) qui prend en paramètre une liste de points et renvoie la liste triée dans l'ordre croissant suivant les distances des points à l'origine. On utilisera le tri bulles.