Tutoriel : Tri pointu pour une liste

Si vous désirez trier une liste d'objets GeoGebra, il n'est pas improbable que la manière de trier que vous désirez ne soit pas possible avec un simple appel à la commande Trier.

(Si vous désirez simplement inverser le résultat, vous pouvez utiliser Trier() en association avec la commande Retourner()).

Mais dans le cas où le type d'objet n'est pas traité par Trier() ou que vous désirez un critère de tri complètement différent, cette page peut vous aider dans votre recherche.

Démarche :

1. Soit **liste** la liste que nous voulons trier.
2. **liste_a = Trier(Séquence((*nombre basé sur Elément(liste, k)*, k), k, 1, Longueur(liste)))**
   Elle correspond à ***c*** dans l'explication (*la liste d'aide*). La ***section en gras*** est appelée ***f*** dans l'explication.
3. **liste_t = Séquence(Elément(liste, y(Elément(liste_a, k))), k, 1, Longueur(liste))**
   C'est *la liste triée*.

Explication :

Pour clarifier la situation, essayons de formuler la demande mathématiquement :
Soit une liste $(a_1,a_2, ..., a_n) \subset X$ et une application $f : X \to IR$, nous voulons trouver une per mutation $\sigma \in S_n$ telle que

$$f(a_{\sigma(k)}) \leq f(a_{\sigma(l)}) \quad \forall (k,l) \in N^2 \ : \ 1 \leq k \leq l \leq n \ .$$

Vous pouvez interpréter la permutation σ comme une renumérotation telle que $(a_{\sigma(1)}, a_{\sigma(2)}, ..., a_{\sigma(n)})$ soit la liste triée.

Ainsi nous devons d'abord établir une application f en adéquation avec notre tri.

L'astuce pour résoudre ce problème avec GeoGebra est d'utiliser la commande existant pour trier une liste de points.
Au lieu de considérer géométriquement un point, nous le considérons simplement comme un couple de nombres.
GeoGebra nous permet de trier ces points par leur abscisse, ainsi nous créons une liste de points $\tilde{c}=(\tilde{c_1},\ldots,\tilde{c_n})$ avec $\tilde{c_k}:=(f(a_k),k)$ et trions cette liste.

Appelons cette liste triée $c=(c_1,\ldots,c_n)$. Elle nous permet maintenant de construire σ , parce que $y(c_{\sigma(k)})=k$. En utilisant σ nous pouvons ensuite construire la version triée de notre liste originale.

Exemple 1 :
Si nous voulons ordonner une liste de points suivant les valeurs croissantes de leur distance à l'origine, nous avons $X:=\mathbb{R}^2$ et $f((x,y))=\sqrt{(x^2+y^2)}$.
Dans GeoGebra, f pour un point A peut être Longueur(A), ainsi la ***section en gras*** dans l'algorithme peut être lue comme **Longueur(Elément(liste,k))**.

Exemple 2 : Si nous voulons ordonner une liste de cercles suivant les valeurs croissantes de leur rayon, la ***section en gras*** dans l'algorithme peut être lue comme **Rayon(Elément(liste,k))**.

Exemple 3 : Si nous voulons ordonner une liste de nombres complexes dans le sens trigonométrique (antihoraire) en fonction de leur argument.

Il faut se méfier que arg() retourne (en général, cela pouvant être corrigé par certaines options) un élément de $]-\pi\ ,\ \pi]$

arg(Elément(liste, k)) + Si(arg(Elément(liste, k)) < 0, 2 π, 0)

Et pour les noms ?

La démarche précédente peut sembler frustrante, si on veut récupérer la liste ordonnée des **noms** des objets.
Définissez au départ, la liste des noms (en tant que textes) $liste_N$={"nom1","nom2", ... }.

La liste de l'algorithme ci-dessus, sera alors définie par liste=Séquence(Objet(Elément($liste_N$, k)), k, 1, Longueur($liste_N$))

Et maintenant, vous pouvez obtenir la liste ordonnée des noms par la commande :

$liste_R$=Séquence(Elément($liste_N$, y(Elément(liste_a, k)), k, 1, Longueur($liste_N$))

Ainsi l'exemple utilise les listes :

$liste_N$ = {"z_1", "z_2", "z_3", "z_4", "z_5", "z_6", "z_7"}

liste=Séquence(Objet(Elément($liste_N$, k)), k, 1, Longueur($liste_N$))
liste_a = Trier(Séquence((arg(Elément(liste, k)) + Si(arg(Elément(liste, k)) < 0, 6.28319, 0), k), k, 1, Longueur(liste)))

liste_t = Séquence(Elément(liste, y(Elément(liste_a, k))), k, 1, Longueur(liste))

liste_R = Séquence(Elément(liste_N, y(Elément(liste_a, k))), k, 1, Longueur(liste_N))