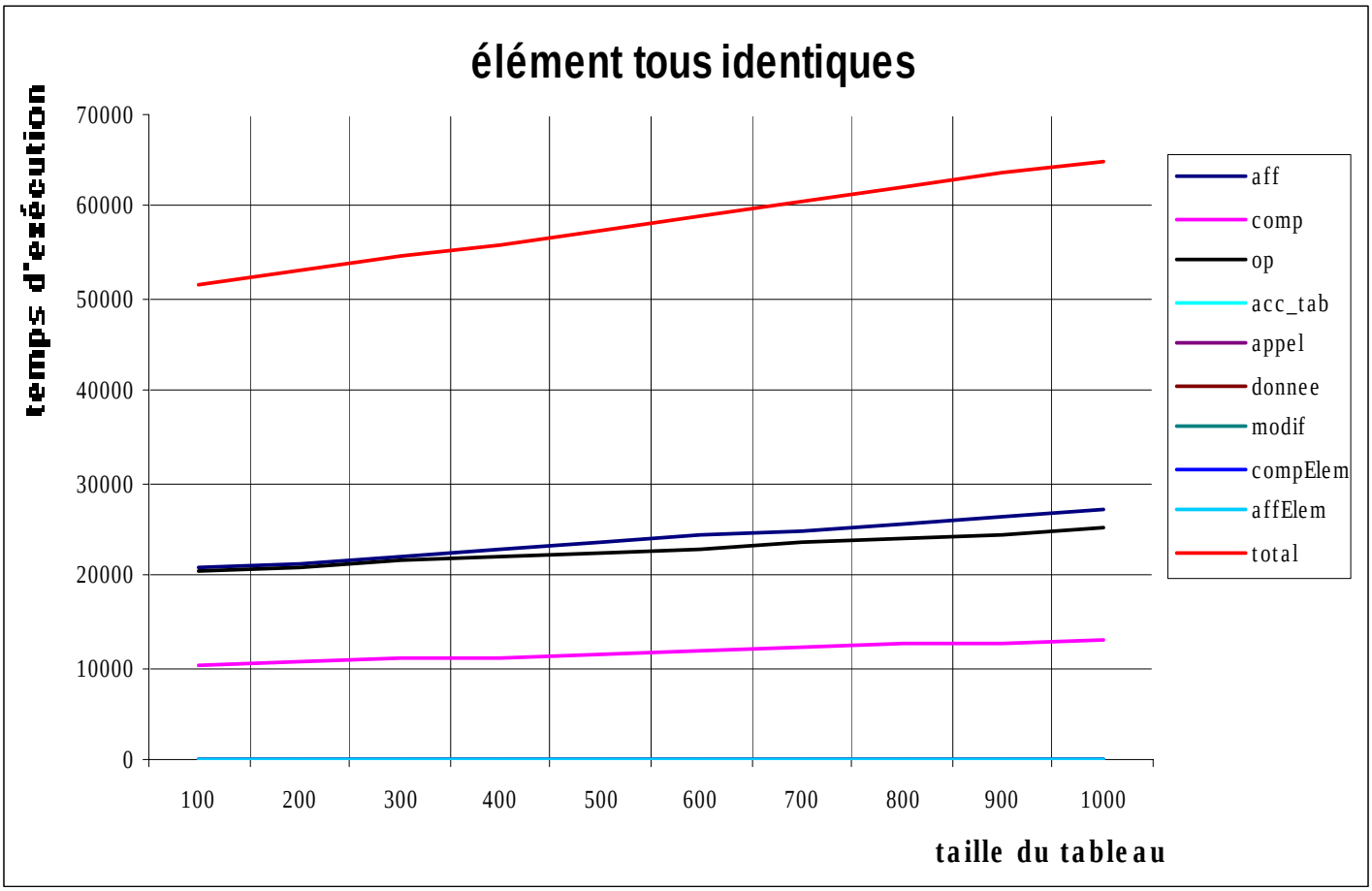
Pour les boucles FOR on a fait intervenir la boucles TANT QUE équivalentes dont on connaît calculer le temps d'exécution .
Toutes le procédures de comptage proposées sont utilisées .
On compile le programme de testes et mesures ,puis on insère les procédures de tri ainsi transformées dans leurs endroits indiqués et on fait les changements nécessaires pour que le programme tient compte des procédures insérées et puisse les mesurer .

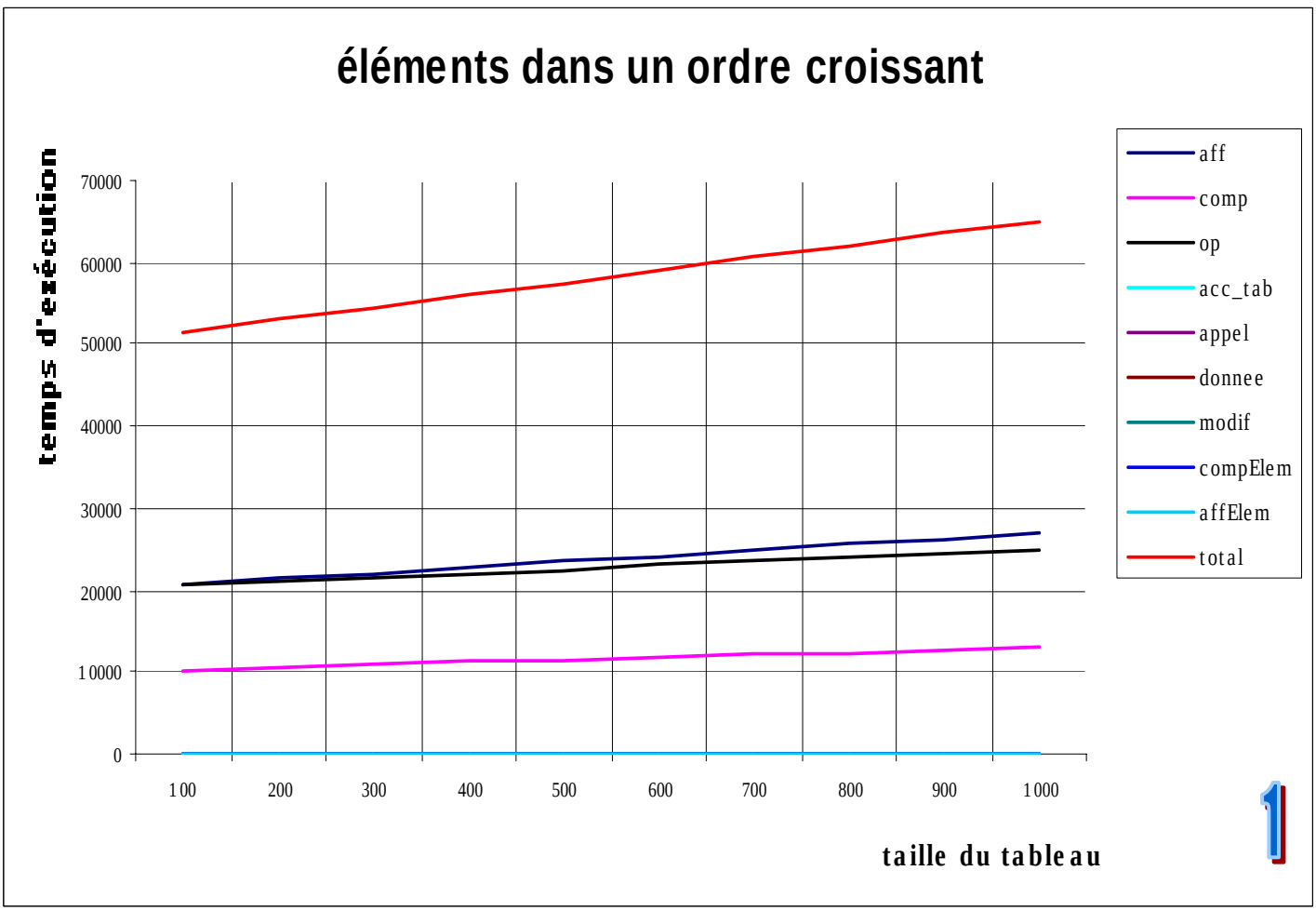
Après l'avoir compiler on exécute le programme ,ce dernier nous permet de choisir la procédure a mesurer et il propose en suite d 'effectuer des mesures selon la taille du tableau ,on choisit alors les tailles 100,200,..,1000, et aussi selon la forme des données , on choisit 1 mesure pour chaque forme possible.
Les résultats sont sauvegarder dans un fichier de notre choit .

Pour pouvoir comparer ces procédures pour chaque procédure et chaque forme de tableau correspondante on trace le graphe comprenant toutes les courbes où chaque courbe représente le nombre d'opération comptabilisées par un compteur en fonction de la taille du tableau ainsi que le nombre total de ses opérations.

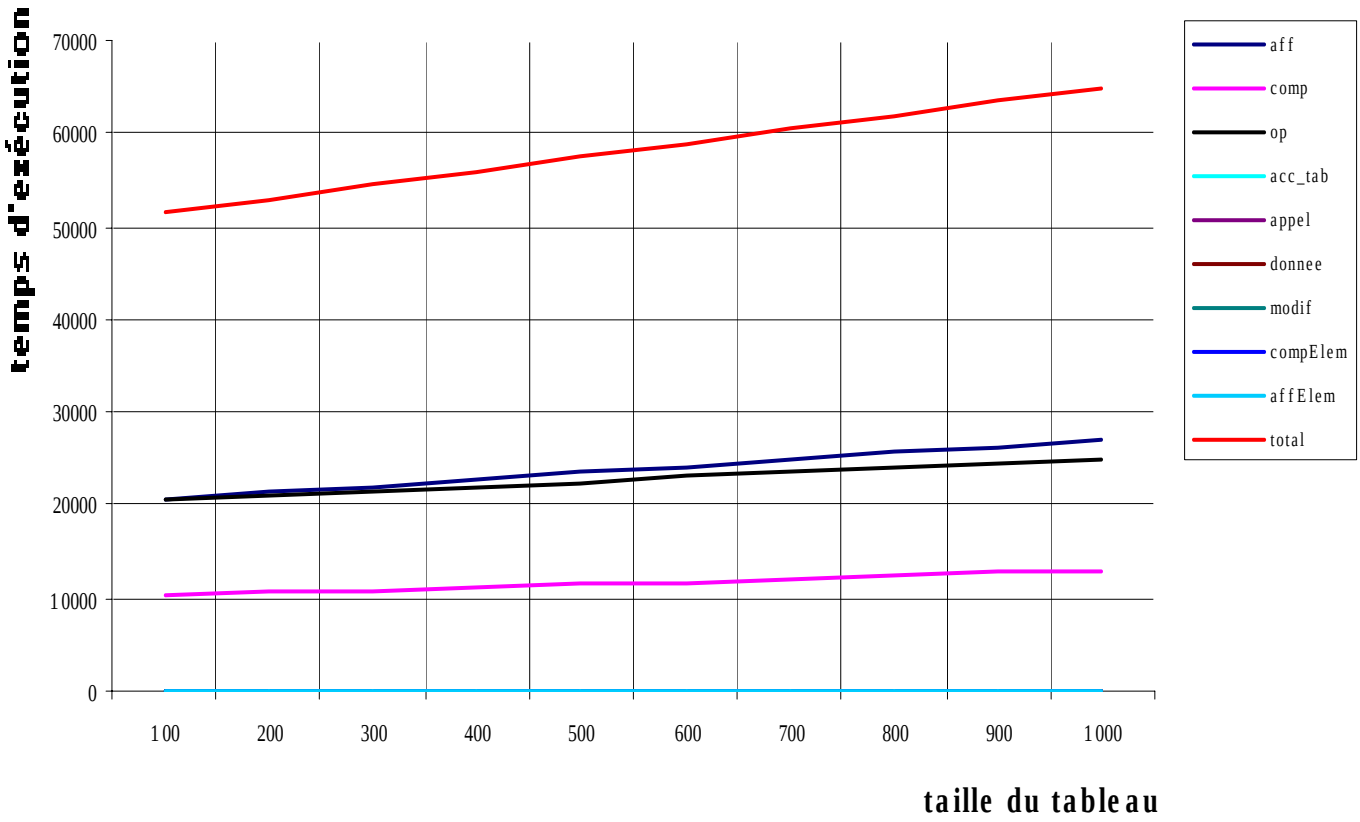
Pour la procédure Tri_comptage certains compteurs ne sont pas insérés, la courbe représentant le total dans ce cas n 'est alors pas signifiante de point de vu avec les autres .
On obtient ainsi les graphes suivants :

Tri_comptage

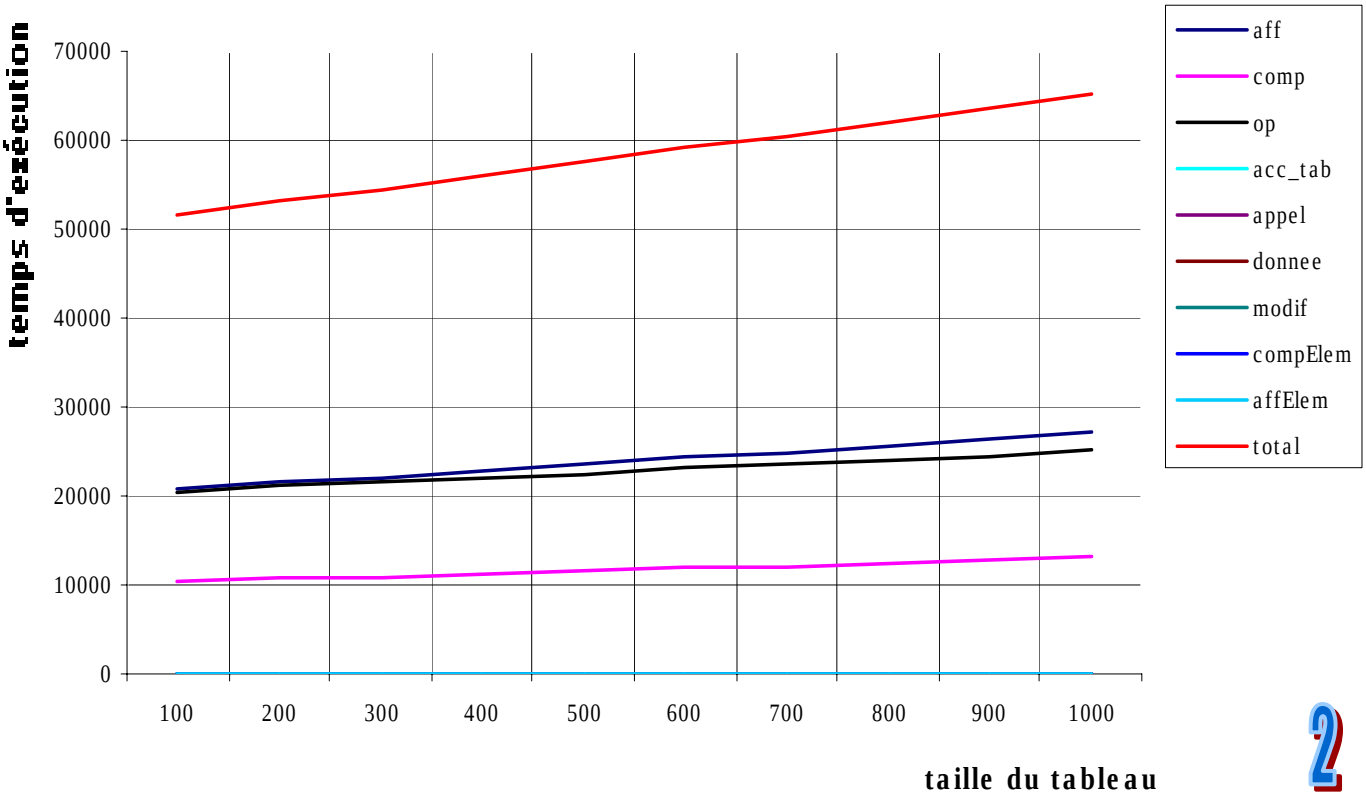




éléments dans un ordre décroissant

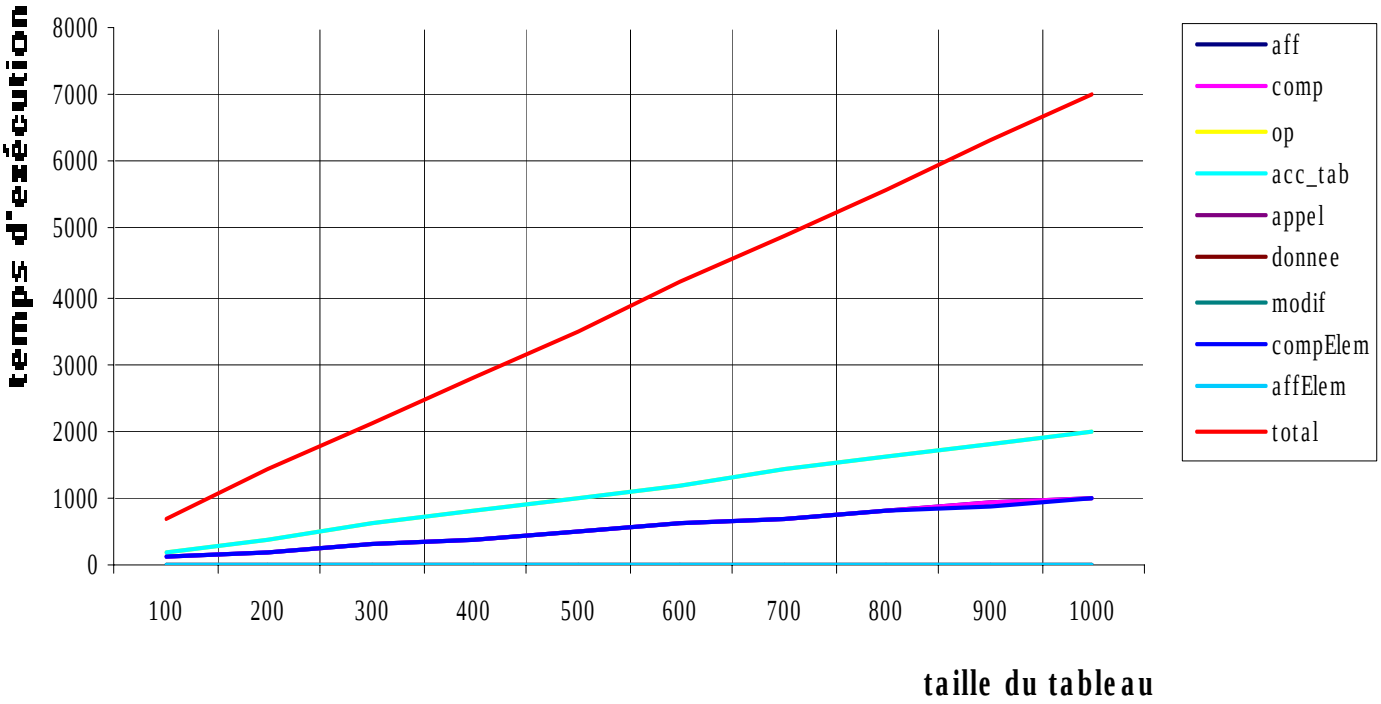


élément dans un ordre quelconque

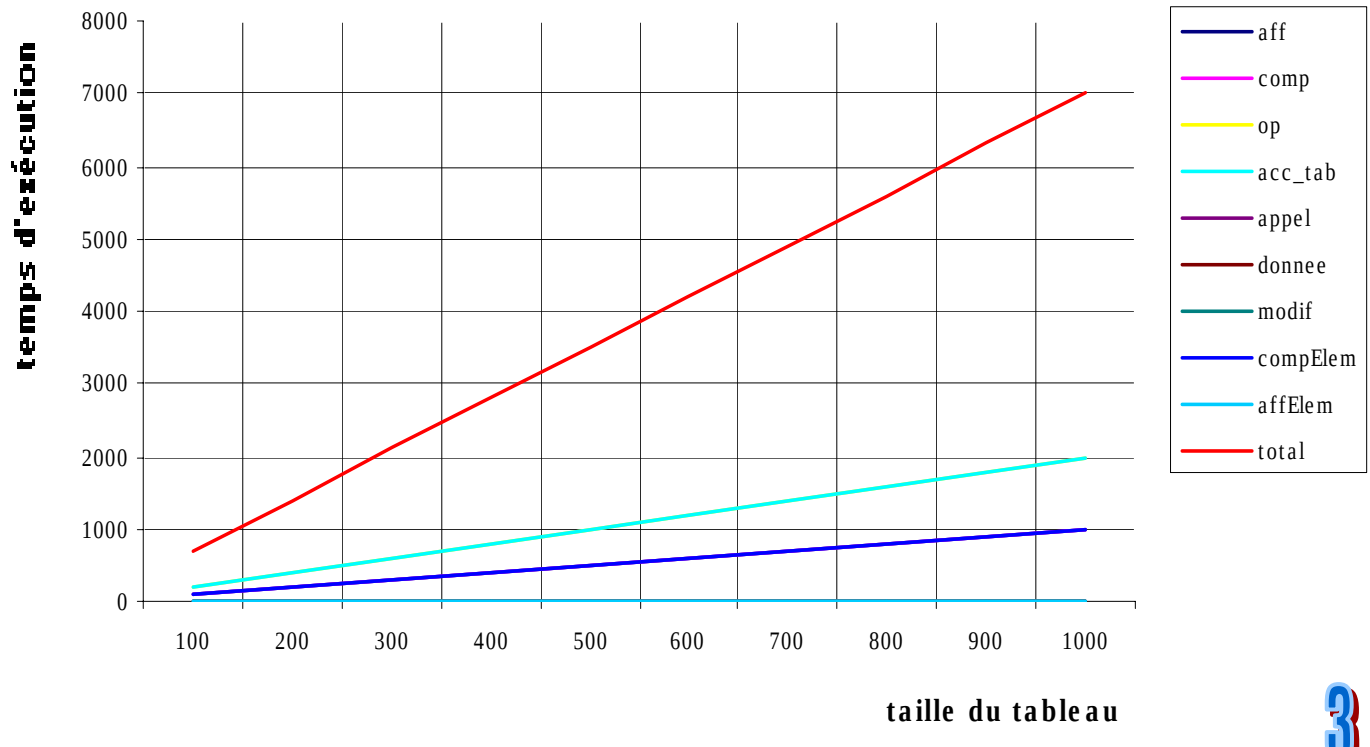


Tri bulle

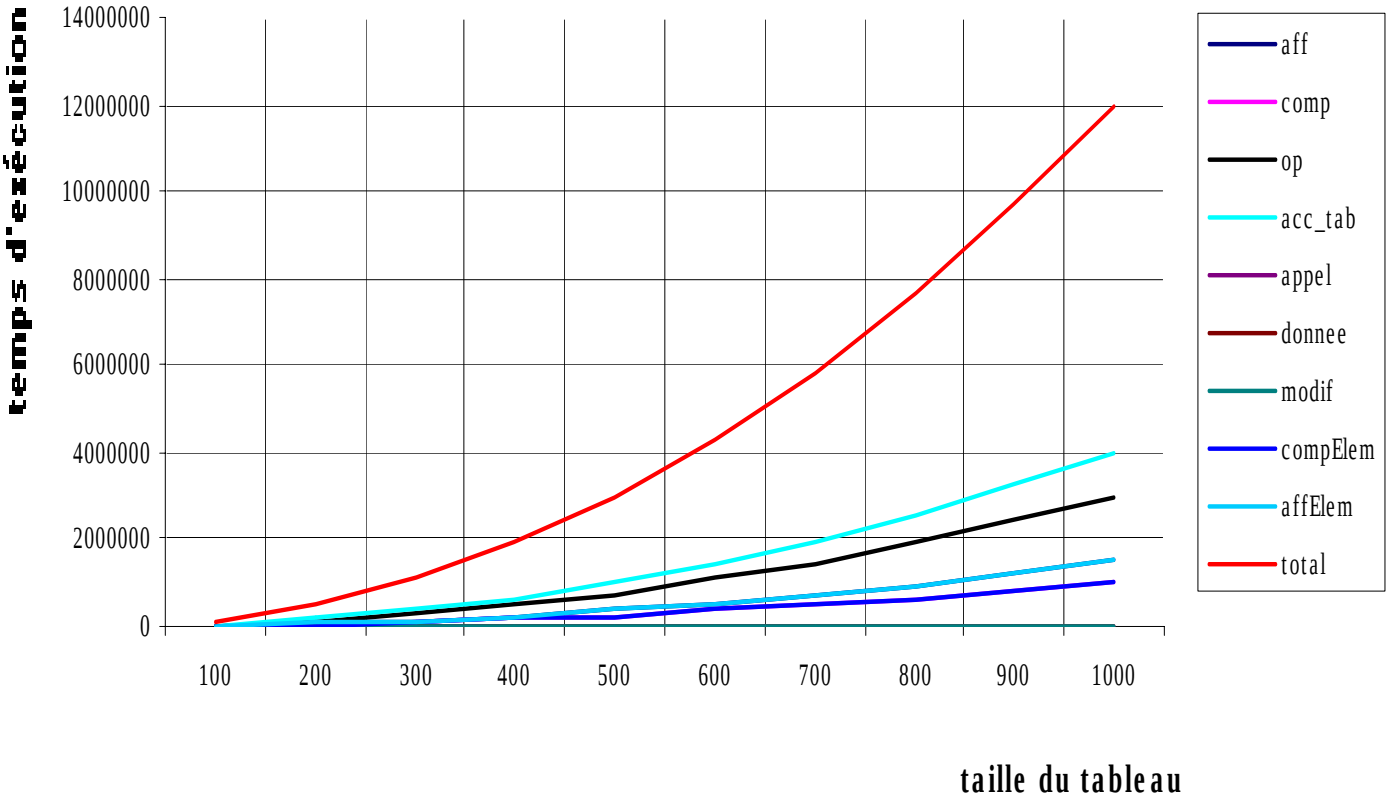
éléments tous identiques



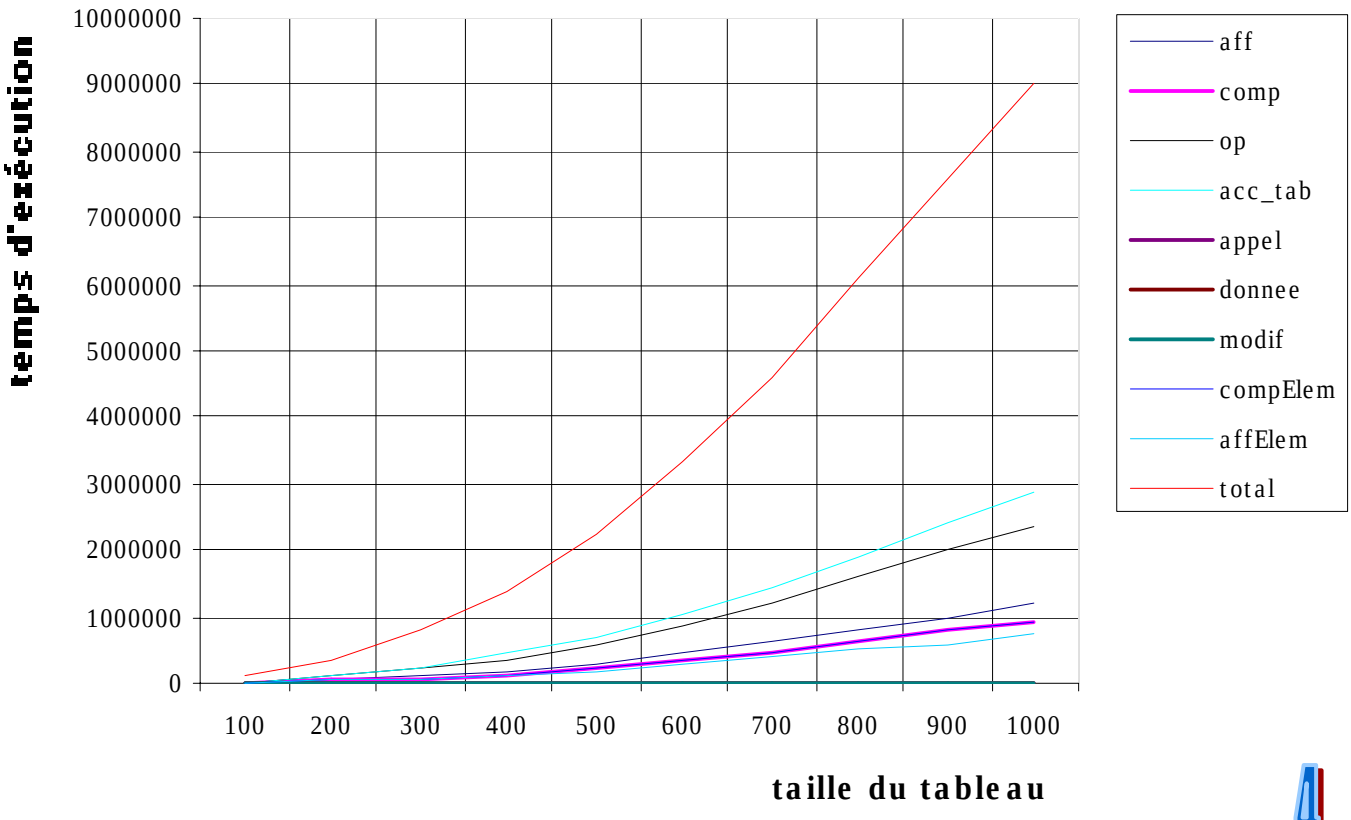
élément dans un ordre croissant



éléments dans un ordre décroissant

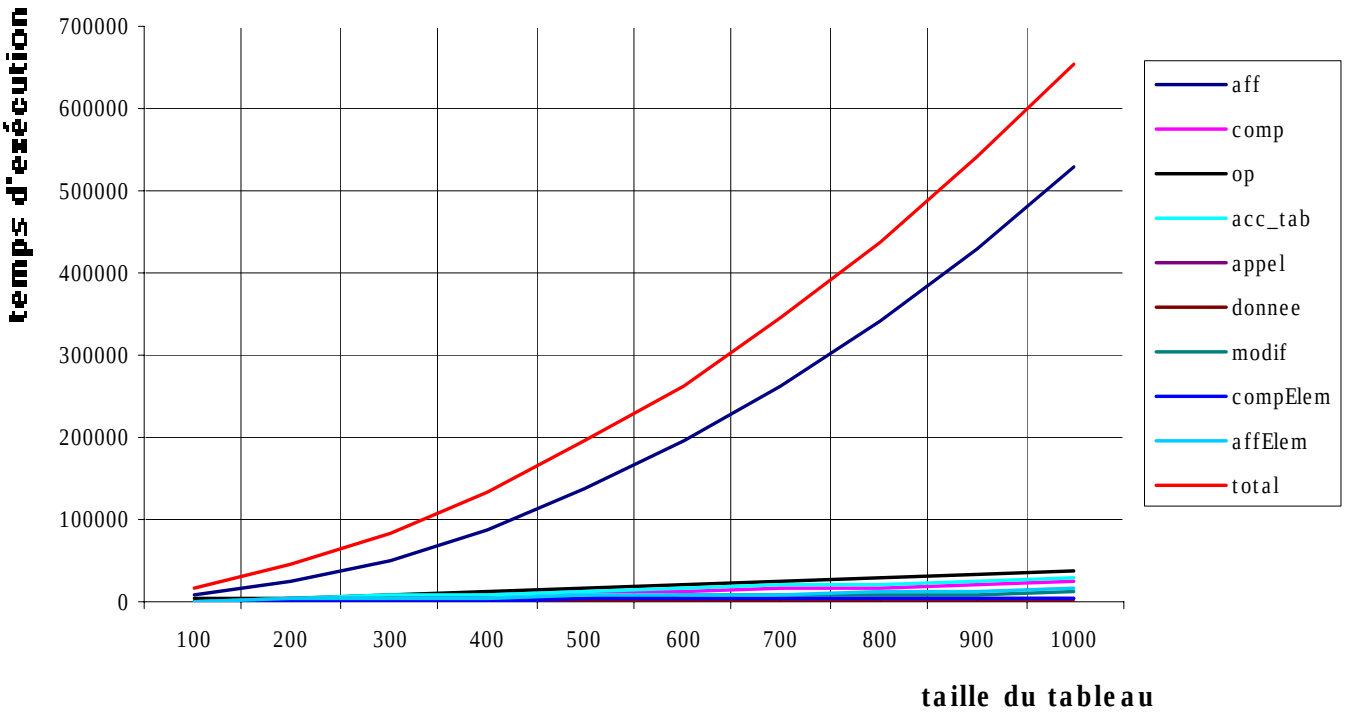


éléments dans un ordre quelconque

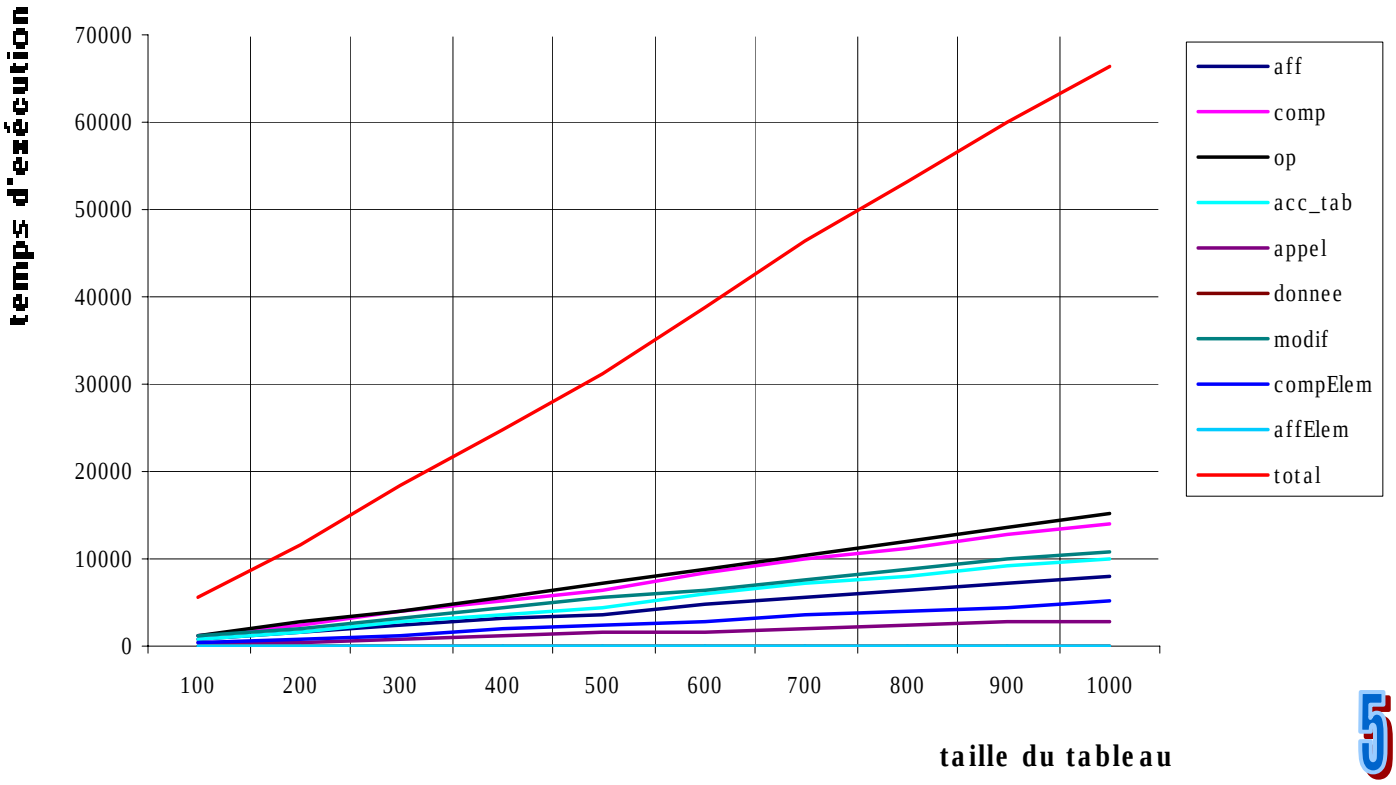


Tri_fusion

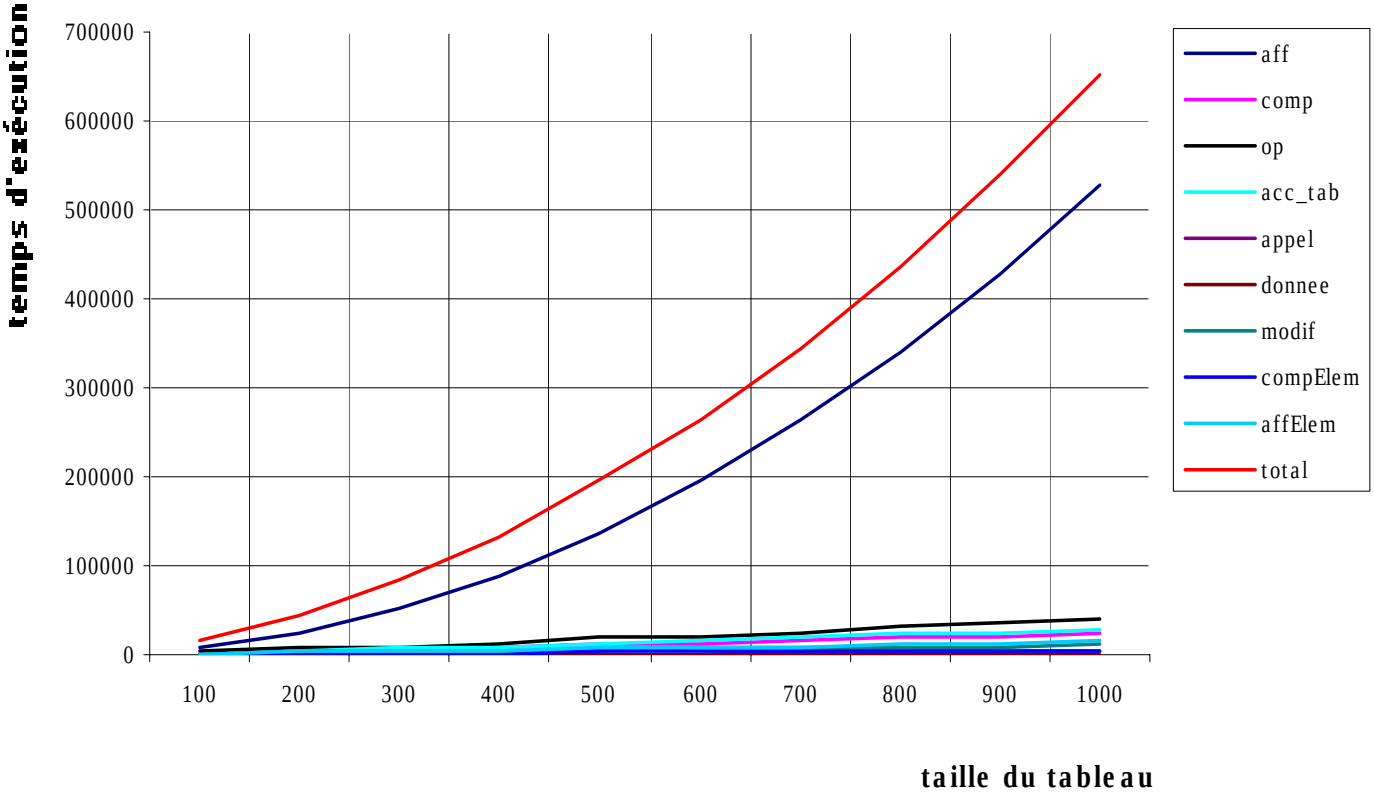
éléments tous identiques



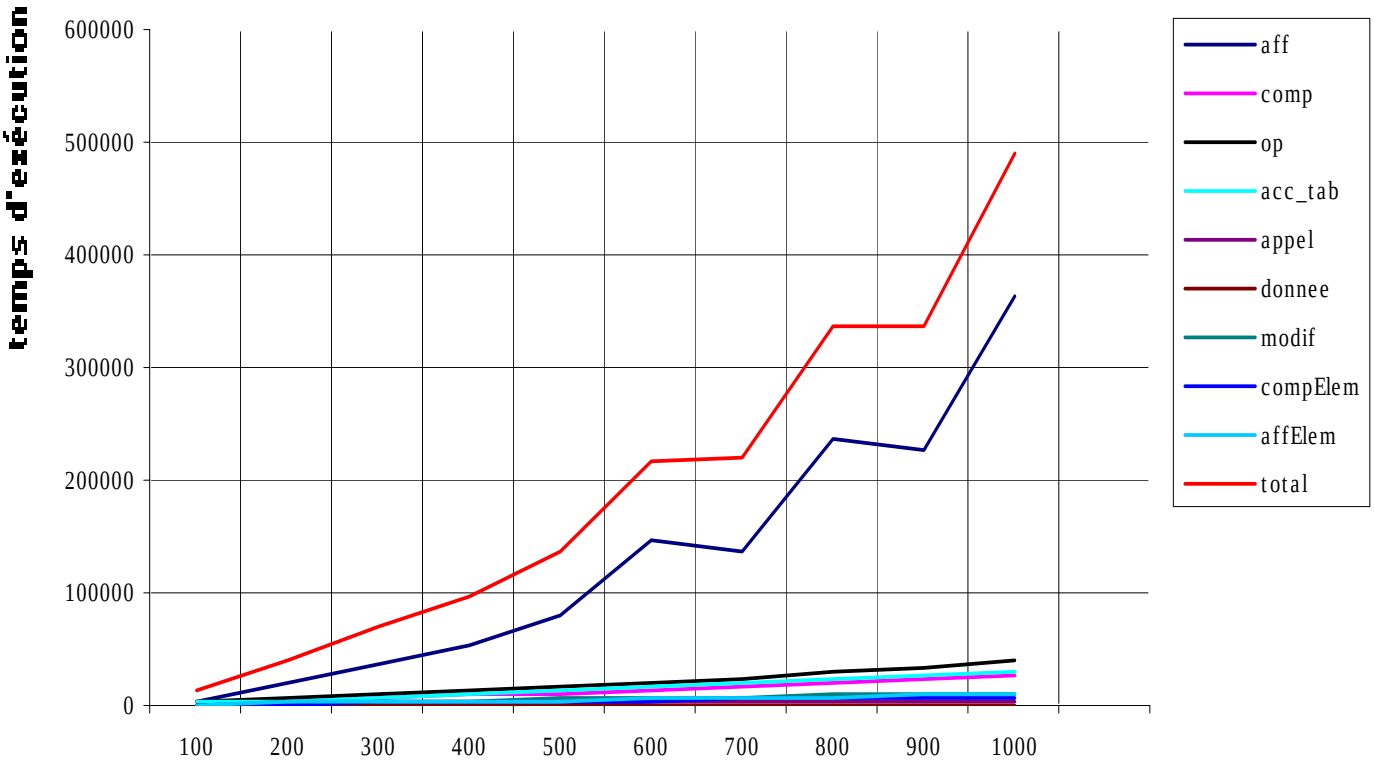
éléments dans un ordre croissant



éléments dans un ordre décroissant

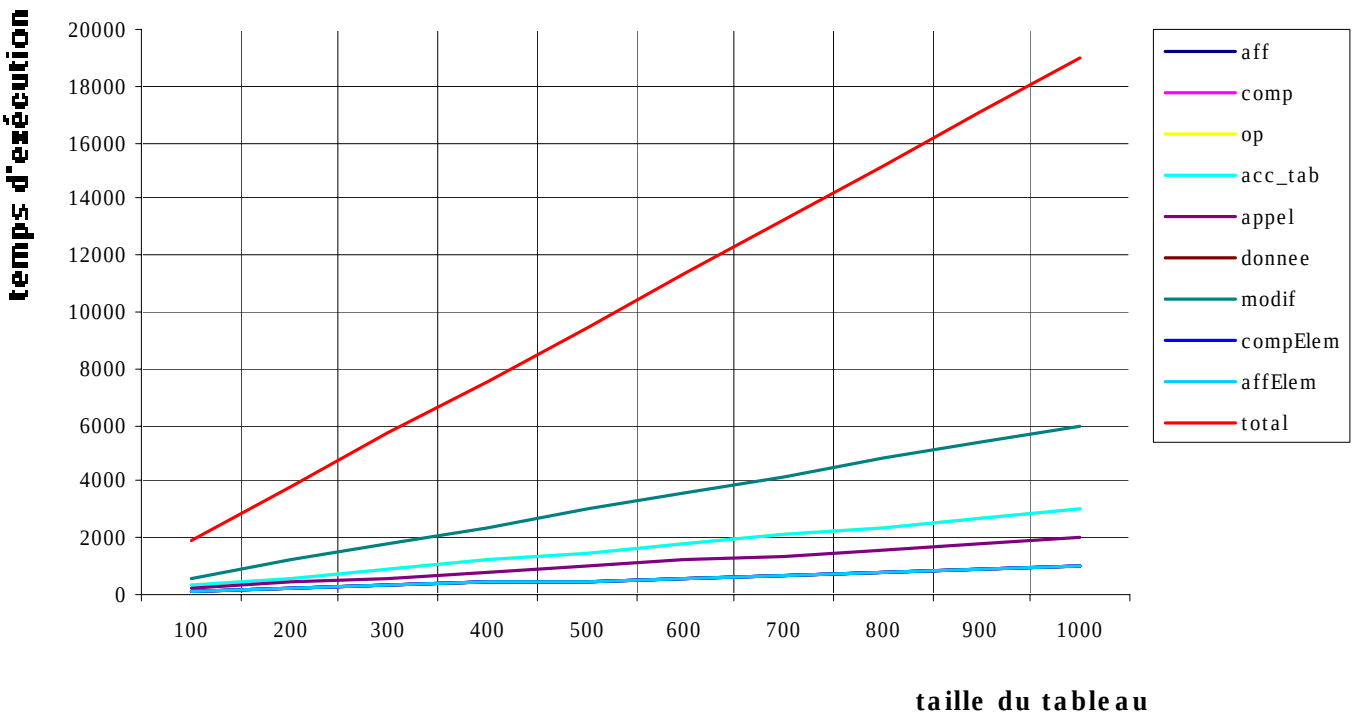


éléments dans un ordre quelconque

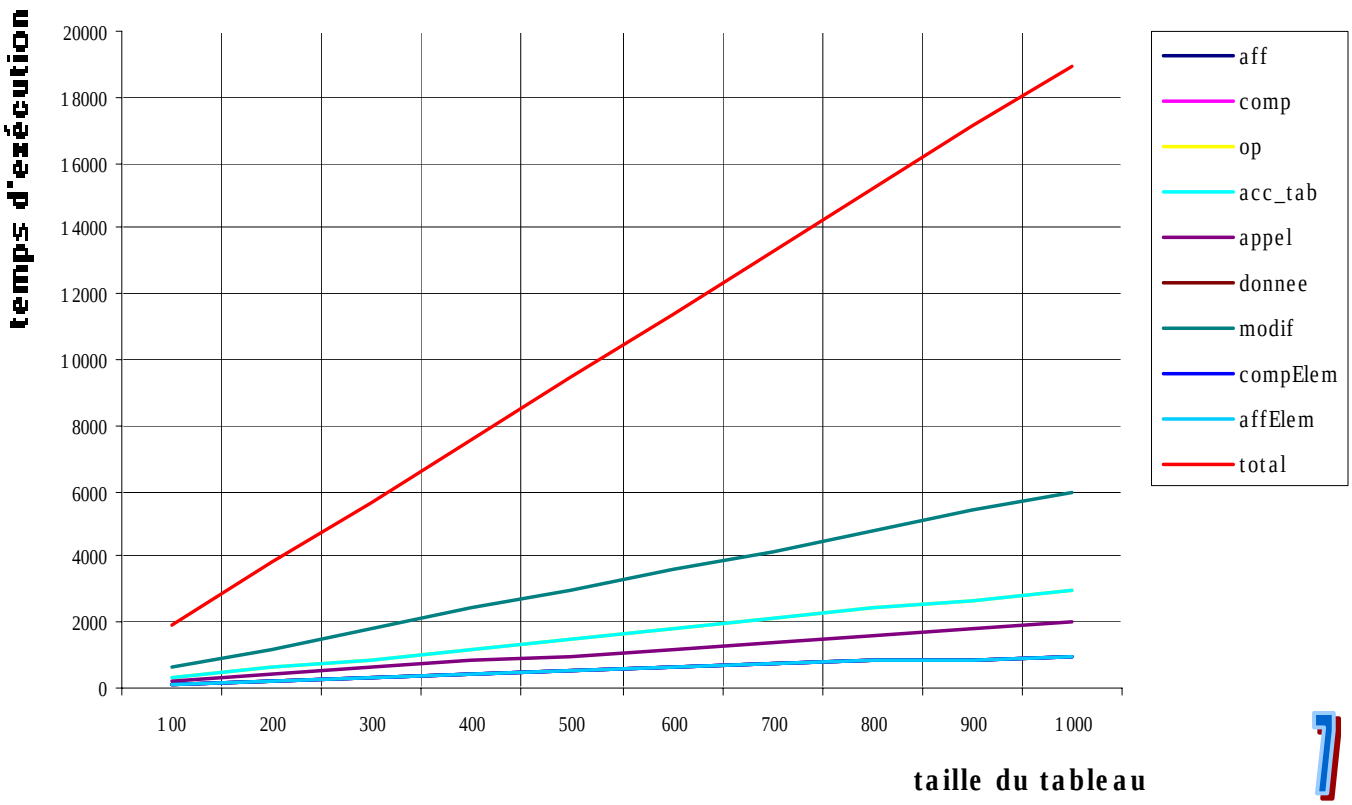


Tri-insertion

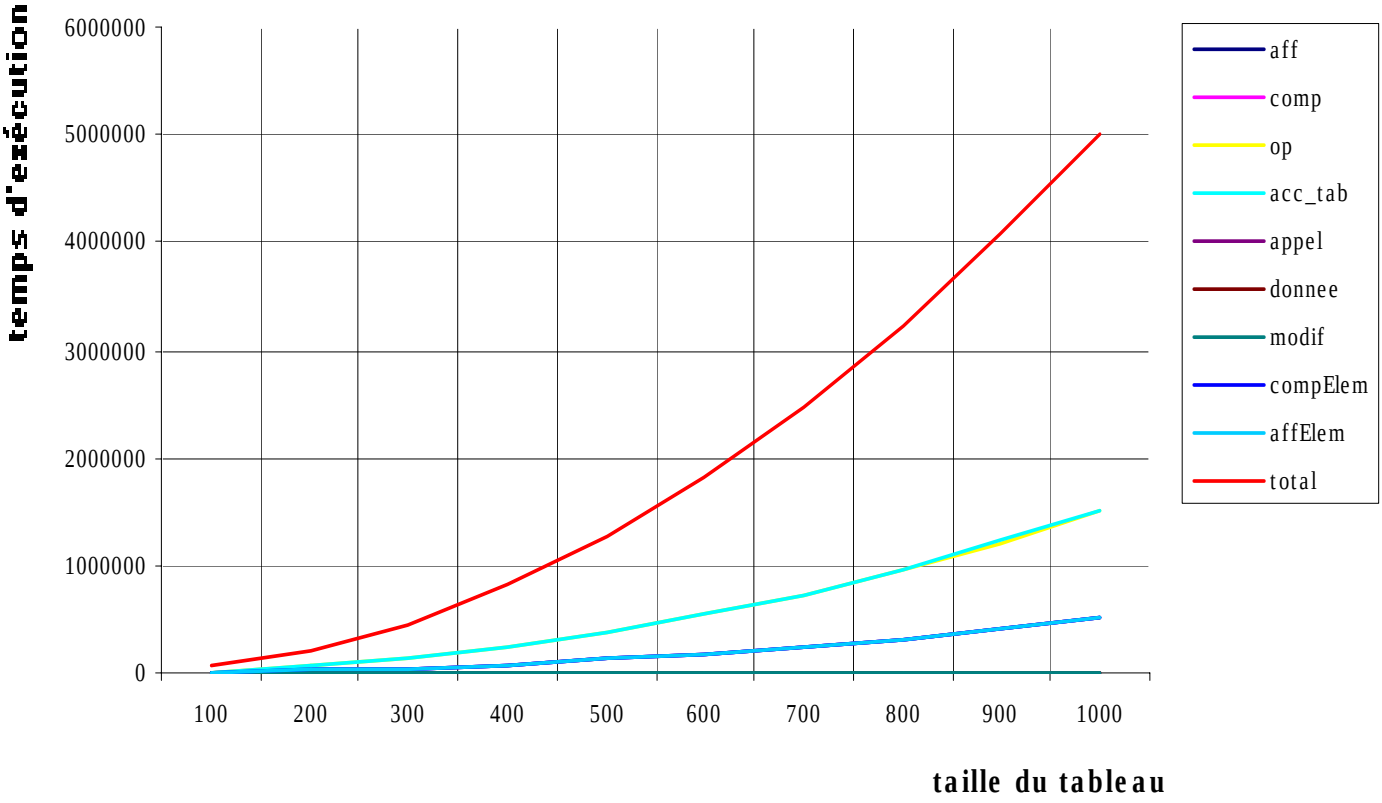
éléments tous identiques



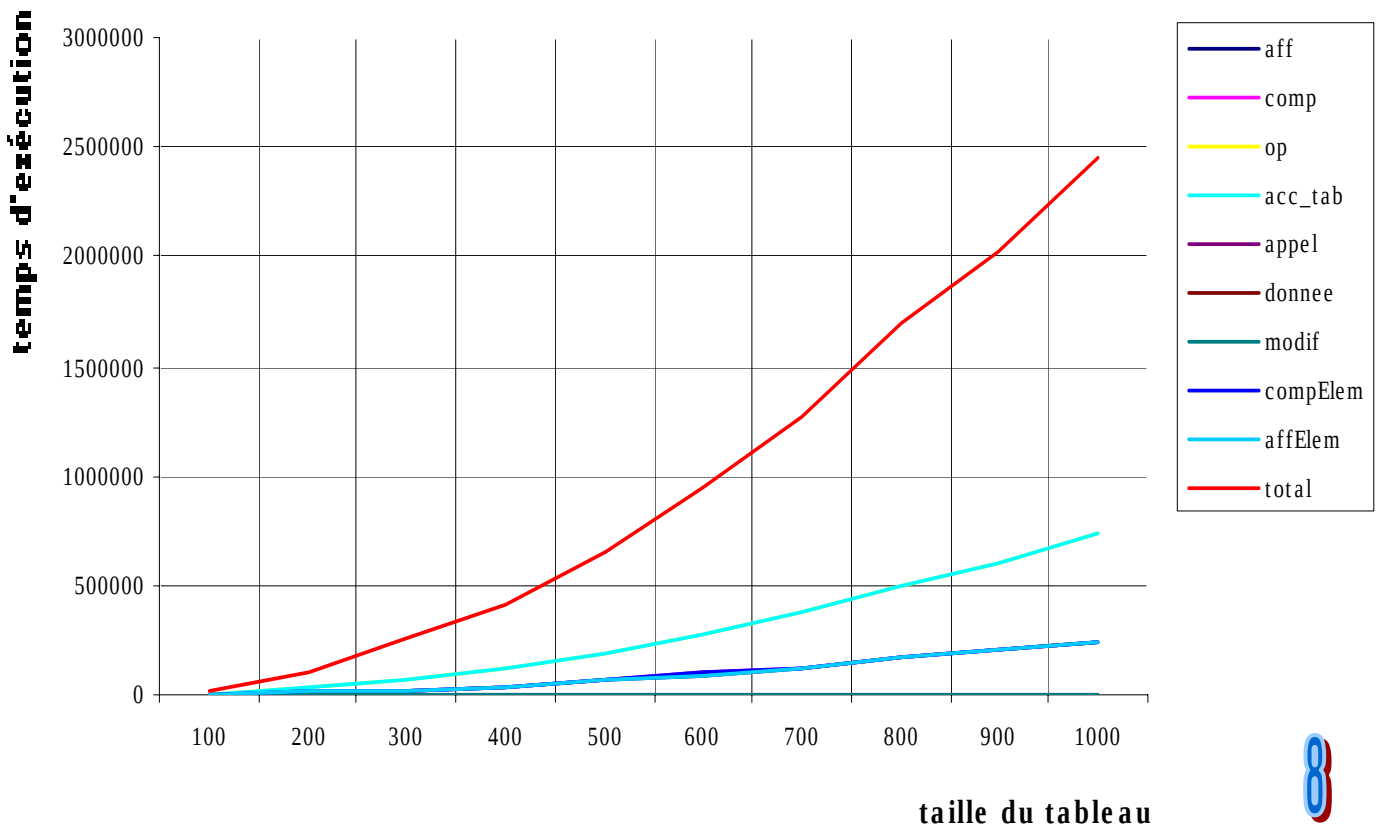
éléments dans un ordre croissant



éléments dans un ordre décroissant

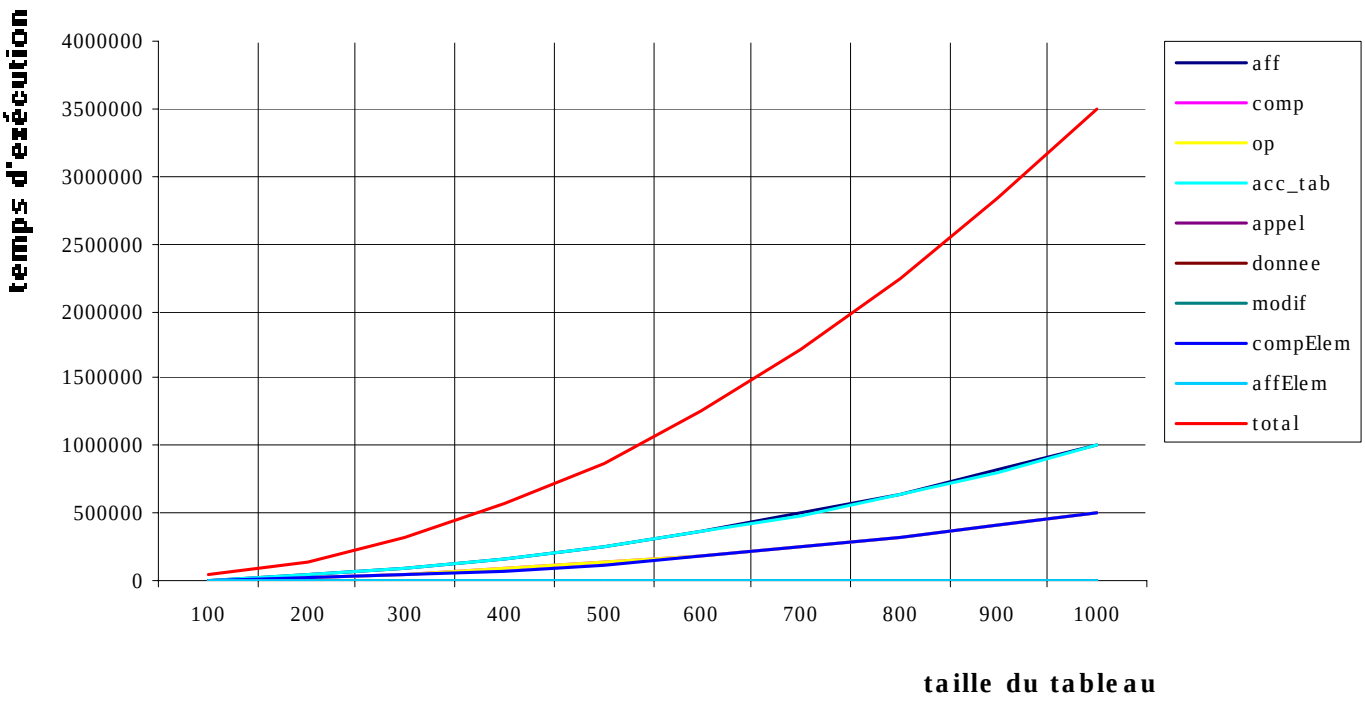


éléments dans un ordre quelconque

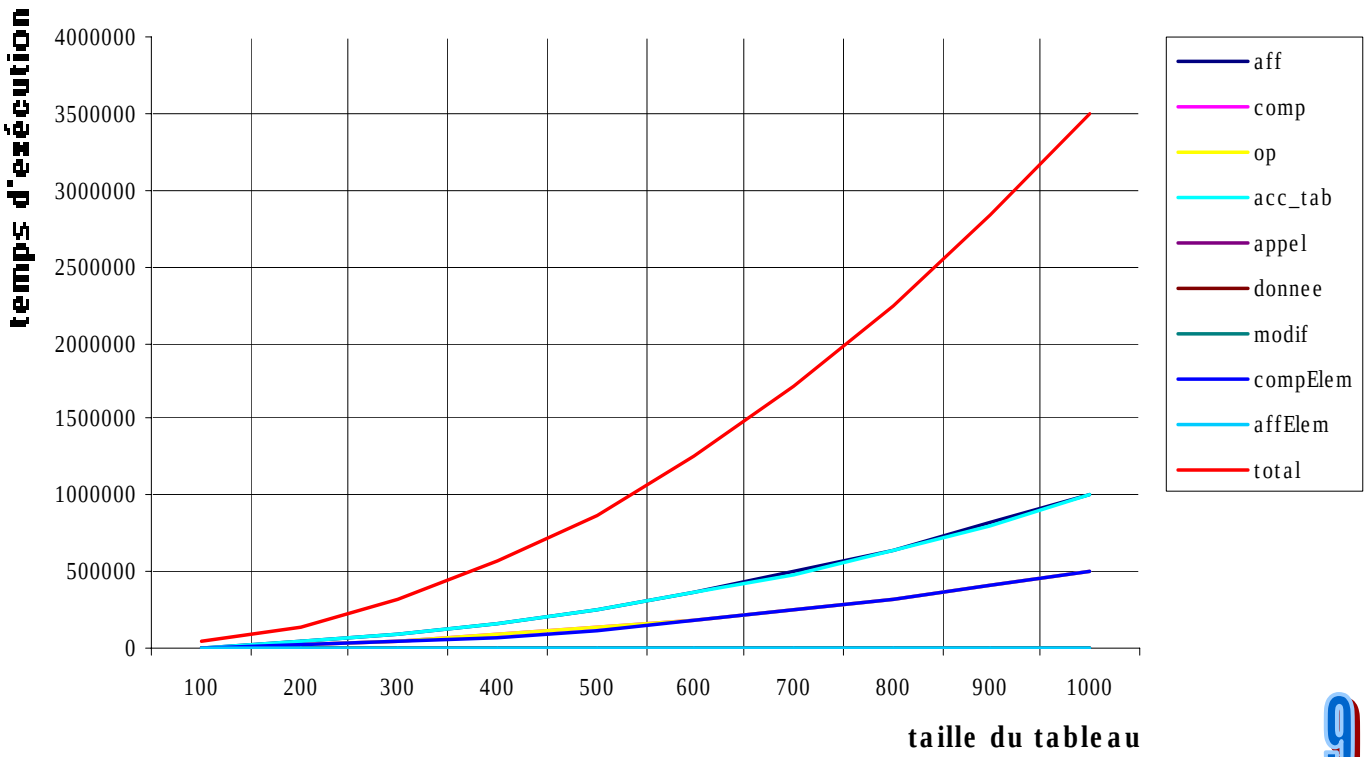


Tri extraction

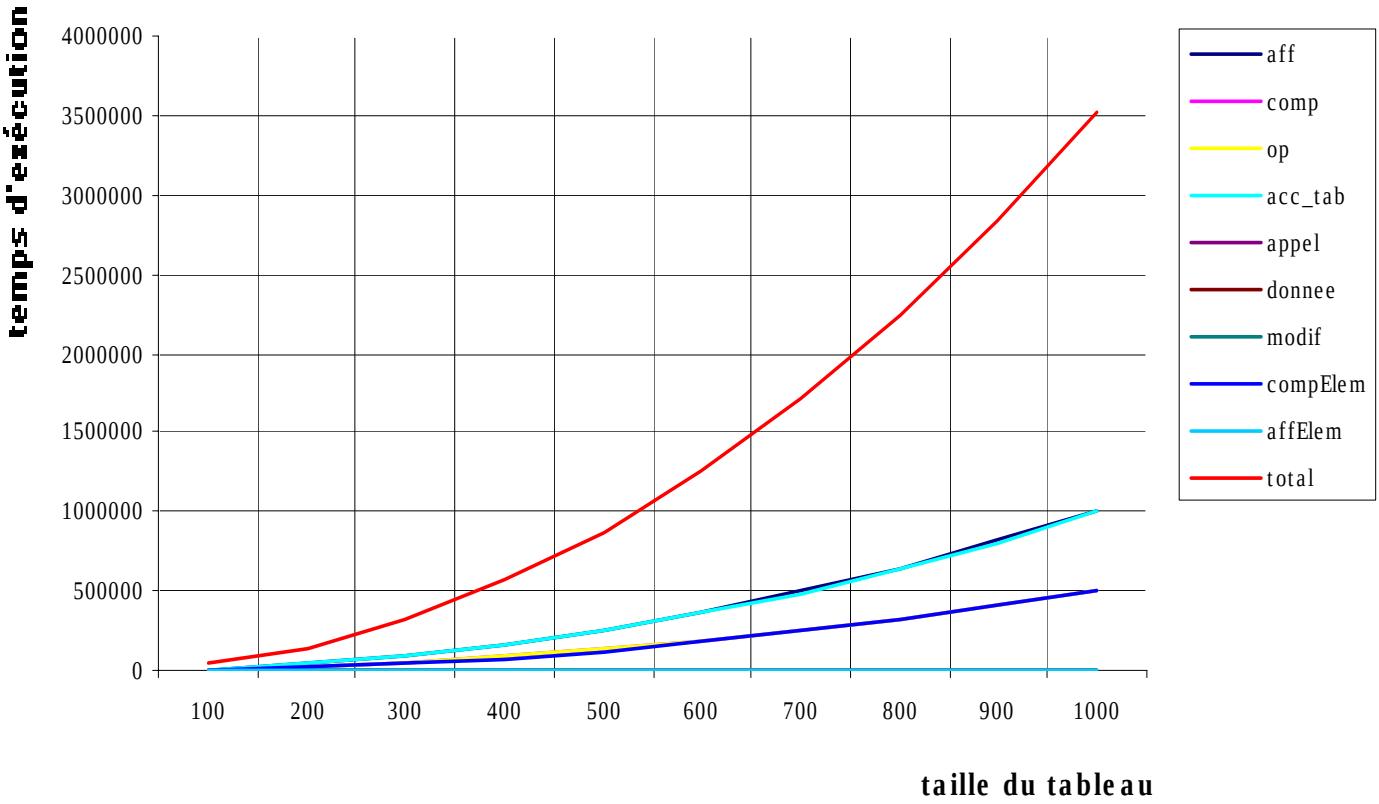
éléments tous identiques



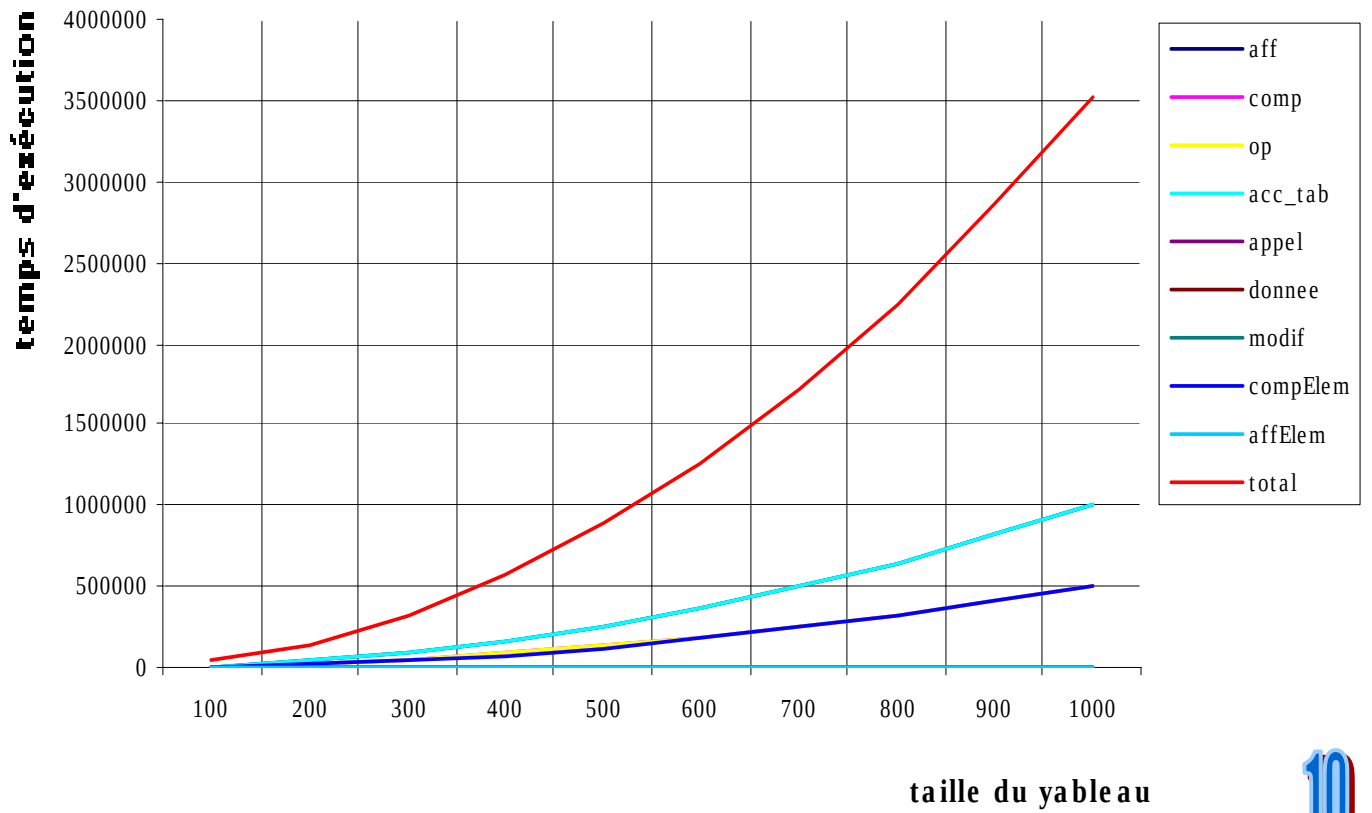
éléments dans un ordre croissant



élément dans un ordre décroissant



élément dans un ordre quelconque



PRESENTATION DE L'ETUDE

On ne s’intéressera qu’aux deux tri suivant :

-Tri_bulle

-Tri_extraction

qui semblent intéressants par leurs façon de traiter les données et surtout par leurs temps d ’exécution qui peuvent varier de façon proportionnelle ou pas par rapport à la taille des éléments à trier .

D ’après les courbes visualisés, pour le Tri_bulle ,dans le cas ou tous les éléments sont identiques ou en ordre croissant ,les courbes sont linéaires avec un temps d ’exécution (7 000 pour une taille de 1000) négligeable devant le cas où les éléments sont en ordre décroissant ou quelconque (12 000 000 pour une taille de 1000 dans le cas décroissant et 9 000 000 pour une taille de 1000 dans le cas quelconque) où les courbes sont plutôt de forme exponentielle .
Par contre pour le Tri_extraction, les courbes sont de forme exponentielle pour chaque forme des données et le temps d ’exécution et le même pour chaque cas.

ANALISE

Pour le Tri_bulle, quant les élément sont identiques ou en ordre croissant le tableau est trié, la procédure compare chaque deux éléments successifs qui sont tous en ordre ,donc aucun changement n ’est fait, la variable echange reste de valeur faux, par conséquent le corps de la boucle est exécuté une seule fois, ce qui explique le faible temps d’exécution .
On a pour un tableau à n élément :

coût Tri_bulle = **coût** repeter + tappel

=1(taff + top + **coût** for) + tappel

=taff + top + tappel +2 taff +tcomp (n-1) *

(taff + 2top + tcomp + tcompelem + 2 tacc_tab + **coût** si)

coût si =0 car le tableau est trié .

Donc **coût** tri_bulle < 7 t +(n-1)(7 t)

<7 t n

(on retrouve bien l ’équation de la droite du total visualisée dans le graphe)
Cette fonction appartient a O(n) ce qui explique sa linéarité .

Tandis que pour un tableau non trié ,on passe par la boucle REPEAT plusieurs fois car à chaque fois le tableau est juste trié partiellement , le nombre de fois que REPEAT est exécuté nb varie proportionnellement avec le nombre d ’éléments n.

donc **coût** Tri_bulle = (6 t + (7t +c))+t

avec c dans {0,10} .
Cette formule contient une somme d’une somme , elle permet donc d ’expliquer la forme exponentielle des courbes et le temps d ’exécution très élevé.
On peut donc dire que Tri_bulle est efficace seulement quant le tableau est suffisamment arrangé.

Quant à Tri_extraction, elle utilise deux boucles FOR imbriquées , le nombre de fois qu ’on passe par chaque procédure est déjà déterminé donc il reste constant pour chaque tri; d’après l ’instruction‘ For j := 1 to fin ’on déduit qu’on passe par la première n fois (avec n = nombre d ’éléments), à chaque fois j est augmenté de 1 , donc d ’après l ’instruction ‘ For i:=(j+1) to fin ’ on déduit que le coût est de la forme :

coût Tri_extraction = 4t + (9t + c_j t + (5t+c ’$_i$ t))

avec c_j dans {0,7} selon la condition (j<>indmin) et c ’$_i$ dans{0,1} selon la condition (tab[i] < tab[indmin]).
Cette formule montre bien que le temps d ’exécution varie plus vite que n ’importe quelle fonction linéaire car elle contient deux sommes imbriquées; ce qui explique la forme des courbes.

D ’autre part on peut considérer que c_j et c_i sont constantes car chacune d ’elle est égale à deux valeurs possibles relativement proches.Donc le coût de Tri_extraction est le même quelque soit la forme des données pour un même tableau, ce qui explique la presque égalité des courbes dans chaque cas.

Le temps de Tri_extraction (3 500 000 pour 1000 éléments)est largement inférieur au temps que Tri_bulle aurait mis pour des élément en ordre croissant (12 000 000 pour 1000 éléments)ou en ordre quelconque (9 000 000 pour 1000 éléments).

De points de vu de la taille des procédures, Tri_bulle et Tri_extraction occupent à peu près le même espace mémoire, on peut aussi remarquer qu ’ils ont une même complexité car chacune d ’elles utilise deux boucles imbriquées avec des simples opérations sur des entiers, booléens, et les éléments d’un tableau .

CONCLUSION

La procédure Tri_bulle n’est efficace que pour des données arrangées ,tandis que pour des données brutes ou arrangées de façon ‘ vicieuse ’!!,c ’est le Tri_extraction qui se montre le plus intéressant avec son temps d’exécution stable dans chaque cas.