L3 Info Mont-Perrin

MAN
Dossier projet

Thomas MEDARD
Rémi SEGRETAIN

Table des matières

Algorithmes

Les algorithmes utilisés :

Méthodes de base :

- tri par insertion séquentielle
- tri par insertion séquentielle avec liste chaînée
- tri par insertion dichotomique
- tri par sélection-permutation
- tri à bulles

Méthodes évoluées :

- tri pas fusion
- tri rapide
- tri à l'aide d'arbres binaires de recherche
- tri par tas

Études de complexité

Tri par insertion séquentielle avec liste chaînée

On ne considère que les opérations sur les accès à value (->value).

Prenons i l'index de currentNode dans la List. Σ[i = 1, n - 1](i) = somme des i allant de 1 à n - 1.

On a donc pour les comparaisons :

Pour un i donné on a (i + 1) tests (2e while).

=> Σ[i = 1, n - 1](i + 1) <=> Σ[i = 2, n - 1](i)

<=> Σ[i = 1, n](i - 1) = (n(n + 1)) / (2) - 1 ∈ θ(n^2) car on parcoure tous sans sortie préalable.

Pour les transfères :

Pour un i donné on (2 + i) transfères (tmp + for + currentSortedNode).

=> Σ[i = 1, n - 1](2 + i) = Σ[i = 2, n - 1](i) = Σ[i = 2, n](i – 1)

= (n(n + 1)) / (2) - 2 - 1 + (n + 1) ∈ θ(n^2)

Donc en théorie, cet algorithme ∈ θ(n^2). En pratique, il est tout de même plus lent que l'insertion séquentielle normale car l'accès en RAM de la List est plus coûteuse.

Tri à l'aide d'arbres binaires de recherche

Étant donné les récursivités de parcours, on peut en déduire que la complexité est de l'ordre de $\theta(n^2)$. En effet, pour le tri par arbre binaires, le tri que nous avons implémenté est lent notamment lorsqu'il est déjà ordonné. On peut l'améliorer en passant par des arbres binaires de recherche équilibrés (AVL).

Tableaux et courbes

Tri	« 100 »	« 500 »	« 5000 »	« 10000 »
Tri par fusion	0,0292	0,1677	1,5318	2,8387
Tri rapide	0,0094	0,0485	0,4875	0,9575
Tri par tas	0,0216	0,1198	1,5287	3,2073
Tri à bulles	0,1080	1,5153	91,2756	425,3800
Insertion séquentielle	0,0211	0,4515	43,2995	172,7840
Ins. séq. (listes chaînées)	0,0350	0,8004	73,6965	294,6678
Insertion Dichotomique	0,0254	0,3776	22,7849	88,9881
Tri par Arbre Binaire	0,05865	0,2635	3,63875	11,80005
Sélection-Permutation	0,0293	0,5938	55,1440	219,7499
Tri	« 50000 »	« 100000 »	« 200000 »	« 300000 »
Tri par fusion	15,5629	32,1178	66,3263	101,6671
Tri rapide	4,8053	9,2220	17,2935	26,4868
Tri par tas	17,5744	36,6743	76,0928	117,3055
Tri à bulles	12557,9152	63930,9893		
Insertion séquentielle	4316,4249	18344,3054		
Ins. séq. (listes chaînées)	7358,4600	33221,5579		
Insertion Dichotomique	2201,5408	8732,1737	39805,0219	
Tri par Arbre Binaire	360,42985	2074,5627	12413,95165	36203,295375
Sélection-Permutation	5481,2900	23757,0533		
Tri	« 400000 »	« 500000 »	« 600000 »	« 700000 »
Tri par fusion	137,2140	174,4969	212,6654	247,9493
Tri rapide	35,5530	44,7987	52,6732	67,7698
Tri par tas	159,5036	200,9750	243,8198	287,9870
Tri à bulles				
Insertion séquentielle				
Ins. séq. (listes chaînées)				
Insertion Dichotomique				
Tri par Arbre Binaire				
Sélection-Permutation				
Tri	« 800000 »	« 900000 »	« 1000000 »	
Tri par fusion	284,1719	324,4772	361,7700	
Tri rapide	78,9474	82,5152	94,6268	
Tri par tas	331,8450	375,4287	422,6467	
Tri à bulles				
Insertion séquentielle				
Ins. séq. (listes chaînées)				
Insertion Dichotomique				
Tri par Arbre Binaire				
Sélection-Permutation				

Tris	Complexité théorique	Complexité pratique
Insertion séquentielle	$\in \theta(n^2)$	$\in \theta(n^2)$
Ins. Séq. (listes chaînées)	$\in \theta(n^2)$	$\in \theta(n^2)$
Insertion dichotomique	$\in O(n^2)$	$\in O(n^2)$
Sélection-permutation	$\in \theta(n^2)$	$\in \theta(n^2)$
Tri à bulles	$\in \theta(n^2)$	$\in \theta(n^2)$
Tri par fusion	$\in \theta(n\log_2 n)$	$\in \theta(n\log_2(n))$
Tri rapide	$\in O(n^2)$ *	$\in O(n\log(n))$
Tri par arbre binaires	$\in O(n^2)$ / $O(n\log(n))$ **	$\in O(n^2)$ / $O(n\log(n))$
Tri par tas	$\in \theta(n\log(n))$	$\in \theta(n\log(n))$

Les tris qui ne finissent pas les tests sont :

- Tri à bulles : 63931 ms à la taille 100 000
- Insertion séquentielle : 18344 ms à la taille 100 000
- Ins. Séq. (listes chaînées) : 33222 ms à la taille 100 000
- Insertion dichotomique : 39805 ms à la taille 200 000
- Tri par arbre binaires : 36203 ms à la taille 300 000
- Sélection-permutation : 23757 ms à la taille 100 000

* Pour le tri rapide, le pire des cas est lorsque le tableau est déjà équilibré ce qui arrive rarement et peut carrément être éviter, c'est pourquoi, en pratique, il est très efficace.

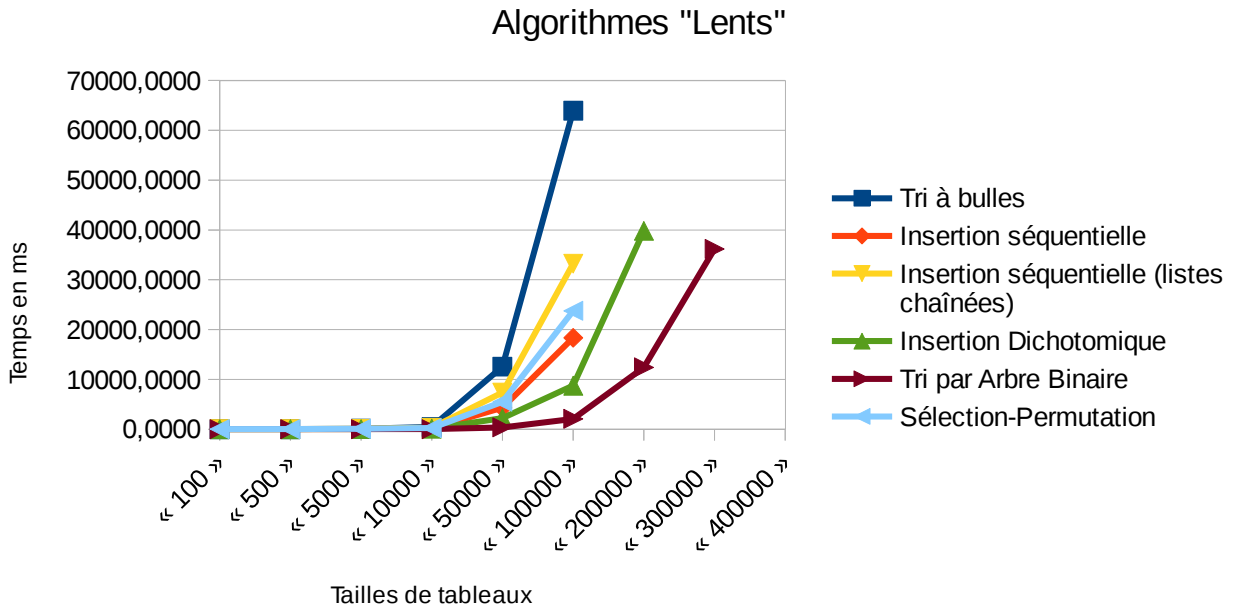
** Pour le tri par arbre binaires, le tri que nous avons implémenté est lent notamment lorsqu'il est déjà ordonné. On peut l'améliorer en passant par des arbres binaires de recherche équilibrés (AVL). En revanche l'équilibrage lui même ramera la complexité au même ordre.

Tris "lents"

Tri	« 100 »	« 500 »	« 5000 »	« 10000 »
Tri à bulles	0,1080	1,5153	91,2756	425,3800
Insertion séquentielle	0,0211	0,4515	43,2995	172,7840
Ins. séq. (listes chaînées)	0,0350	0,8004	73,6965	294,6678
Insertion Dichotomique	0,0254	0,3776	22,7849	88,9881
Tri par Arbre Binaire	0,05865	0,2635	3,63875	11,80005
Sélection-Permutation	0,0293	0,5938	55,1440	219,7499
Tri	« 50000 »	« 100000 »	« 200000 »	« 300000 »
Tri à bulles	12557,9152	63930,9893		
Insertion séquentielle	4316,4249	18344,3054		
Ins. séq. (listes chaînées)	7358,4600	33221,5579		
Insertion Dichotomique	2201,5408	8732,1737	39805,0219	
Tri par Arbre Binaire	360,42985	2074,5627	12413,9517	36203,2954
Sélection-Permutation	5481,2900	23757,0533		

Tri	Complexité théorique	Complexité pratique
Tri à bulles	$\in \theta(n^2)$	$\in \theta(n^2)$
Insertion séquentielle	$\in \theta(n^2)$	$\in \theta(n^2)$
Ins. Séq. (listes chaînées)	$\in \theta(n^2)$	$\in \theta(n^2)$
Insertion dichotomique	$\in O(n^2)$	$\in O(n^2)$
Tri par arbre binaire	$\in O(n^2)$ / $O(n\log(n))$	$\in O(n^2)$ / $O(n\log(n))$
Sélection-permutation	$\in O(n^2)$	$\in O(n^2)$

Courbes



Commentaire

Le tri par Arbre binaire se trouve avec les algorithmes lents car il est bien plus lent que les algorithmes rapide. En effet, nous avons implémenté une version sans équilibrage.

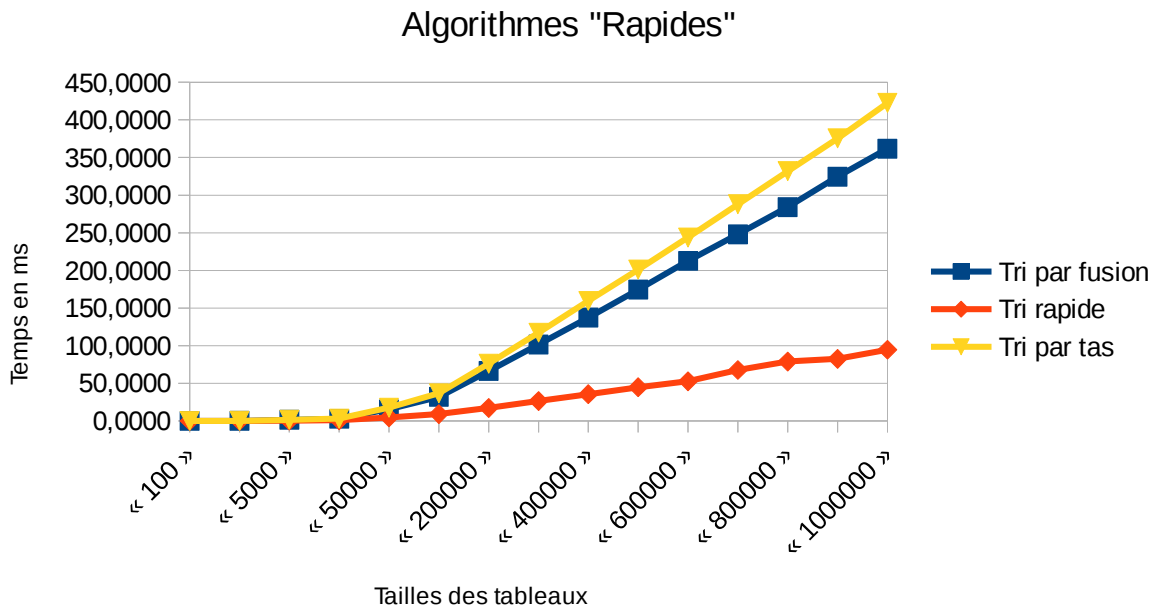
On peut voir tout de suite que le tri à bulle n'est pas du tout efficace.

Tris "rapides"

Tri	« 100 »	« 500 »	« 5000 »	« 10000 »
Tri par fusion	0,0292	0,1677	1,5318	2,8387
Tri rapide	0,0094	0,0485	0,4875	0,9575
Tri par tas	0,0216	0,1198	1,5287	3,2073
Tri	« 50000 »	« 100000 »	« 200000 »	« 300000 »
Tri par fusion	15,5629	32,1178	66,3263	101,6671
Tri rapide	4,8053	9,2220	17,2935	26,4868
Tri par tas	17,5744	36,6743	76,0928	117,3055
Tri	« 400000 »	« 500000 »	« 600000 »	« 700000 »
Tri par fusion	137,2140	174,4969	212,6654	247,9493
Tri rapide	35,5530	44,7987	52,6732	67,7698
Tri par tas	159,5036	200,9750	243,8198	287,9870
Tri	« 800000 »	« 900000 »	« 1000000 »	
Tri par fusion	284,1719	324,4772	361,7700	
Tri rapide	78,9474	82,5152	94,6268	
Tri par tas	331,8450	375,4287	422,6467	

Tri	Complexité théorique	Complexité pratique
Tri par fusion	Є $\theta(n\log_2 n)$	Є $\theta(n\log_2(n))$
Tri rapide	Є $O(n^2)$	Є $O(n\log(n))$
Tri par tas	Є $\theta(n\log(n))$	Є $\theta(n\log(n))$

Courbes



Commentaire

Comme on peut le voir, le tri rapide est effectivement le plus efficace en pratique.

Le tri fusion, bien que théoriquement très efficace, demande beaucoup de mémoire et cela se répercute sur les tableaux de grandes tailles.

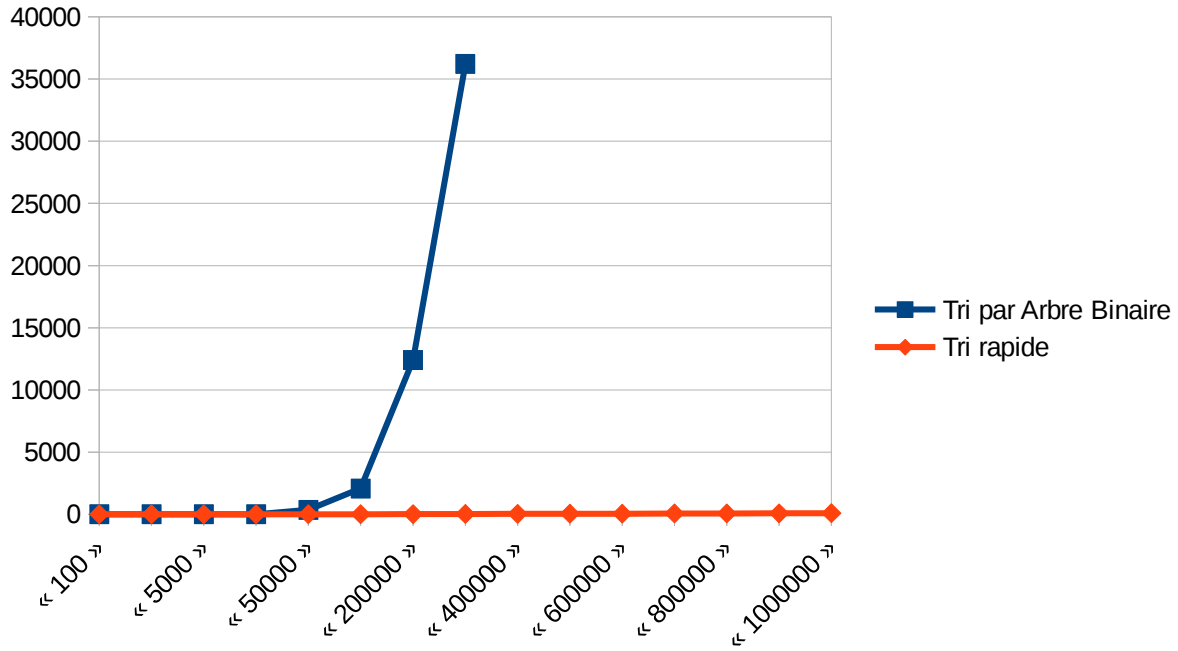
Tableaux et courbes comparatives

Tableau et courbe comparative entre le plus rapide des tris "lents" et le plus rapide des tris "rapides"

Tris	Complexité théorique	Complexité pratique
Tri par Rapide	Є O(n^2)	Є O(nlog(n))
Tri par Arbre Binaire	Є O(n^2) / O(nlog(n))	Є O(n^2) / O(nlog(n))

Tri	« 100 »	« 500 »	« 5000 »	« 10000 »
Tri par Arbre Binaire	0,05865	0,2635	3,63875	11,80005
Tri rapide	0,0094	0,0485	0,4875	0,9575
Tri	**« 50000 »**	**« 100000 »**	**« 200000 »**	**« 300000 »**
Tri par Arbre Binaire	360,42985	2074,5627	12413,952	36203,295
Tri rapide	4,8053	9,2220	17,2935	26,4868
Tri	**« 400000 »**	**« 500000 »**	**« 600000 »**	**« 700000 »**
Tri par Arbre Binaire				
Tri rapide	35,5530	44,7987	52,6732	67,7698
Tri	**« 800000 »**	**« 900000 »**	**« 1000000 »**	
Tri par Arbre Binaire				
Tri rapide	78,9474	82,5152	94,6268	

Courbe comparative



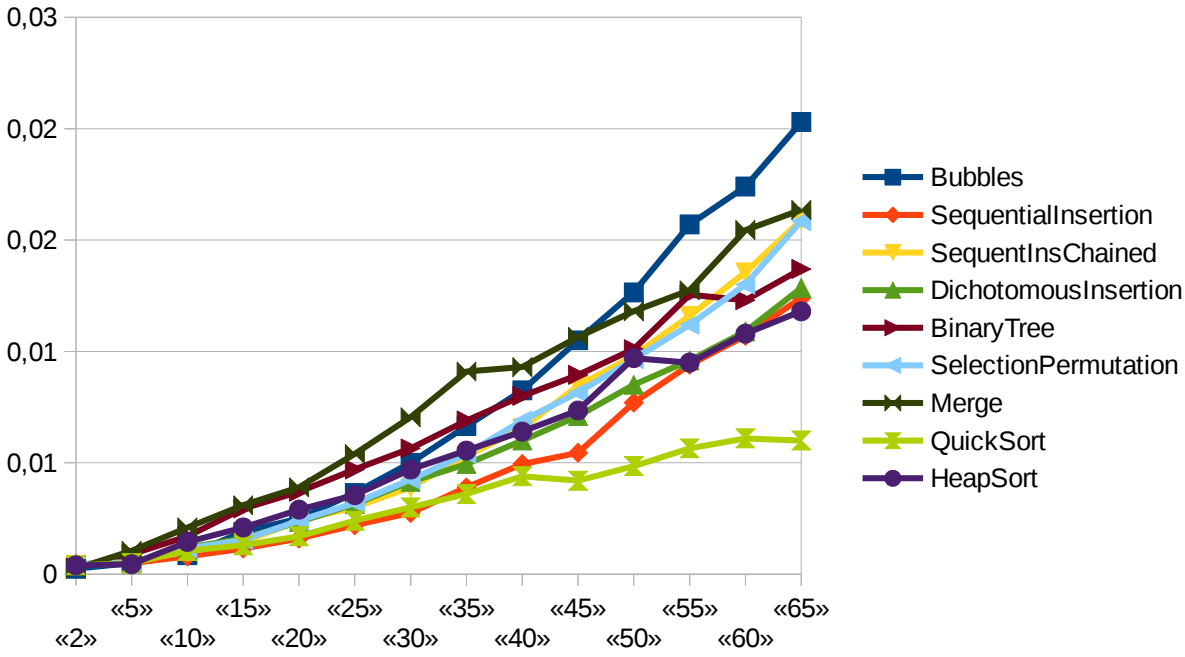
Commentaire

On peut constater que les tris « lents » ne dépassent pas les tests avec des tableaux de tailles supérieurs à 400 000 en moins de 5 minutes. En effet même le plus rapide des tris lent est loin d'être aussi efficace que les tris rapides.

Test sur des tableaux de petites tailles

TRI	«2»	«2»	«5»	«10»
Bubbles	0,00025	0,00025	0,0005	0,00085
SequentialInsertion	0,0003	0,00045	0,0005	0,0008
SequentInsChained	0,0005	0,0004	0,0006	0,001
DichotomousInsertion	0,00045	0,0004	0,0005	0,0012
BinaryTree	0,0025	0,0003	0,0009	0,0017
SelectionPermutation	0,00045	0,00045	0,00055	0,00115
Merge	0,003	0,00025	0,00105	0,0021
QuickSort	0,00045	0,0004	0,0005	0,00105
HeapSort	0,00045	0,0004	0,00045	0,00145
TRI	«15»	«20»	«25»	«30»
Bubbles	0,0019	0,0025	0,00365	0,005
SequentialInsertion	0,00115	0,0016	0,0022	0,00275
SequentInsChained	0,00145	0,0024	0,003	0,0039
DichotomousInsertion	0,0016	0,00235	0,00315	0,00415
BinaryTree	0,00295	0,0037	0,0047	0,00565
SelectionPermutation	0,00155	0,0024	0,0032	0,00425
Merge	0,0031	0,0039	0,0054	0,00705
QuickSort	0,0013	0,0017	0,0024	0,003
HeapSort	0,0021	0,0029	0,00355	0,0047
TRI	«35»	«40»	«45»	«50»
Bubbles	0,00665	0,00825	0,0105	0,01265
SequentialInsertion	0,0039	0,00495	0,00545	0,0077
SequentInsChained	0,0053	0,00655	0,00845	0,0098
DichotomousInsertion	0,00495	0,006	0,0071	0,0085
BinaryTree	0,0069	0,008	0,00895	0,0101
SelectionPermutation	0,0054	0,0069	0,00815	0,00965
Merge	0,0091	0,0093	0,01065	0,0118
QuickSort	0,0036	0,0044	0,0042	0,00485
HeapSort	0,00555	0,0064	0,00735	0,0097
TRI	«55»	«60»	«65»	
Bubbles	0,0157	0,0174	0,0203	
SequentialInsertion	0,0094	0,0107	0,0124	
SequentInsChained	0,0116	0,01355	0,0159	
DichotomousInsertion	0,0096	0,0109	0,01285	
BinaryTree	0,01255	0,0123	0,0137	
SelectionPermutation	0,0112	0,013	0,01585	
Merge	0,01275	0,01545	0,01635	
QuickSort	0,00565	0,0061	0,006	
HeapSort	0,0095	0,0108	0,0118	

Courbe



Commentaire :

On peut constater que les tris dit « rapides » ne sont pas forcément plus efficace pour des tableaux de petites tailles. On peut déduire de ces résultats des possibilités d'optimisation.

Par exemple, le tri par fusion pour être optimisé en utilisant le tri par insertion séquentielle pour les sous-sections de tableaux de taille inférieur à 100.

Conclusion

En conclusion, on peut affirmer que les tris « rapides » surpassent largement les tris « lents », et ce, dès les premières grandes tailles de tableaux testées.

Il faut tout de même prendre en compte que pour de très petites tailles, certains tris dit "lent" ont des performances semblables aux rapides.