Type de méthodes	n	Θ	cpt / Θ	limn->∞ f(n) / Θ cste1, cste2, cste3	temps (ms)	Commentaire
Recherche séquentielle	8388608	n	1.0	Dans le pire des cas (calcul approximatif) f(n) = n limn->∞ f(n) / Θ = n / n = 1	18.055713	La recherche séquentielle étant de complexité Θ(n), multiplier n par k doit multiplier le temps d'exécution par k. Le premier est bizarre mais cela est sûrement du au lancement de fonction ensuite le 3 et 4 se ressemble ce qui est bizarre mais juste approximatif car le reste est cohérent si le 4 était 85
	16777216		1.0		19.725632	
	33554432		1.0		42.374911	
	67108864		1.0		53.93729	
	134217728		1.0		173.034402	
	268435456		1.0		339.157738	
Recherche dichotomique	8388608	log2n	1.0	Dans des cas (calcul approximatif) $f(n) = n/2^k$ (k = nb itérations) $n/2^k = 1$ $n = 2^k$ log2n = k = f(n) limn->∞ f(n) / Θ = log2n / log2n = 1	0.040717	Le premier temps est bizarre mais peut s'excuser par le temps d'exécuter pour la première fois, le reste est proche ce qui est normal car multiplier n par 2^k doit augmenter le temps d'exécution de k (en réalité k T secondes où T = 1/f, f étant la fréquence de l'horloge). Il est donc que ça ne change pas beaucoup vu que k n'est pas grand
	16777216		1.0		0.023117	
	33554432		1.0		0.025702	
	67108864		1.0		0.024933	
	134217728		1.0		0.025282	
	268435456		1.0		0.025422	
Tri simple	2048	n^2	0.499	Dans des cas (calcul approximatif) $f(n) = n^2/2 - n/2$ limn->∞ f(n) / Θ = $(n^2/2 - n/2) / n^2$ $= 1/2 – 1/2n = 1/2$	5.832575	Pour le tri simple, multiplier n (nombre d'éléments dans le tableau) par k doit multiplier le temps d'exécution par k^2. Pour les premiers cela est bizarre on peut supposer un temps augmenté à cause du lancement ou problème pc (RAM) car fiable à 100 %. Sinon même si ce n'augmente pas par 2^2 on voit quand même une augmentation significatif donc on peut penser que c'est normal
	4096		0.499		8.592349	
	8192		0.499		4.918634	
	16384		0.499		136.452379	
	32768		0.499		269.489245	
	65536		0.499		1044.141707	
Tri rapide	2097152	nlog2n	1.25	Dans le meilleur des cas (calcul approximatif) f(n) = n x k $2 = n / 2^{(k-1)}$ $2 \times 2^{(k-1)} = n$ $2^k = n$ k = log2n f(n) = n x log2n limn->∞ f(n) / Θ = nlog2n / nlog2n = 1	196.566523	Pour le tri rapide, multiplier n par 2^k doit augmenter le temps d'exécution par une valeur calculable : $(n\ 2^k)\log2(n\ 2^k) = n\ 2^k$ (log2n + k). On voit ici que cela augmente bien mais ne ralentit pas ce que peut être normal car dans le meilleur des cas on est en Θ(nlog2n) mais dans le pire $\Theta(n^2)$ donc normal de retrouver une augmentation croissante mais régulière
	8388608		1.27		865.311647	
	16777216		1.32		1800.224791	
	33554432		1.23		3707.62734	
	67108864		1.27		7865.69209	
Tri comptage de fréquences	8388608	n+k	1.99	Dans des cas (calcul approximatif) Cas où k est petit et donc insignifiant par rapport à n, f(n) = 2n (car dans k, on appelle n tout de même fois) et Θ = n + k donc limn->∞ f(n) / Θ = 2n / n = 2 (k étant insignifiant) Cas où k est plus largement plus grand que n et donc n est insignifiant (k = 268435456) f(n) = n + k = k et Θ = n + k = k (n étant insignifiant) donc limn->∞ f(n) / Θ = k / k = 1	89.852437	Pour le tri comptage de fréquences, il est normal de voir une croissance dans le cas où k est insignifiant car n monte et modifie donc le temps vu qu'il est le plus important
	16777216		1.99		119.02853	
	33554432		1.99		129.373936	
	67108864		1.99		415.349691	
	134217728		1.99		518.054274	
	268435456		1.99		620.655933	
	8192		0.99		367.196676	Pour le tri comptage de fréquences, il est normal de voir une constant dans le cas où k est très grand car le temps sera en fonction de k qu'il est le plus important
	16384		0.99		369.443616	
	32768		0.99		368.188618	
	65537		0.99		368.932548	
	131072		0.99		367.165628	
	262144		0.99		372.164224	
Tri à bulles	1024	n^2	0.498	Dans le pire des cas (calcul approximatif) $f(n) = ((n-1)(n-2))/2$ $f(n) = (n^2-3n+2)/2$ limn->∞ f(n) / Θ = $((n^2-3n+2)/2) / n^2$ $= (n^2/2-3n/2+1) \times 1/n^2$ $= 1/2 – 3/2n + 1/n^2$ limn->∞ f(n) / Θ = 1/2	6.226057	Pour le tri à bulles, on augmente n de k = 2 soit $2^2 = 4$ vu qu'on est en complexité $\Theta(n^2)$ ce qu'on peut donc à peu près voir, ici cela augmente en n*5 ce qui paraît plutôt logique vu la complexité
	2048		0.499		6.688196	
	4096		0.499		10.078353	
	8192		0.499		56.704427	
	16384		0.499		278.643805	
	32768		0.499		1198.588016	