Introduction à la complexité feuille réponse

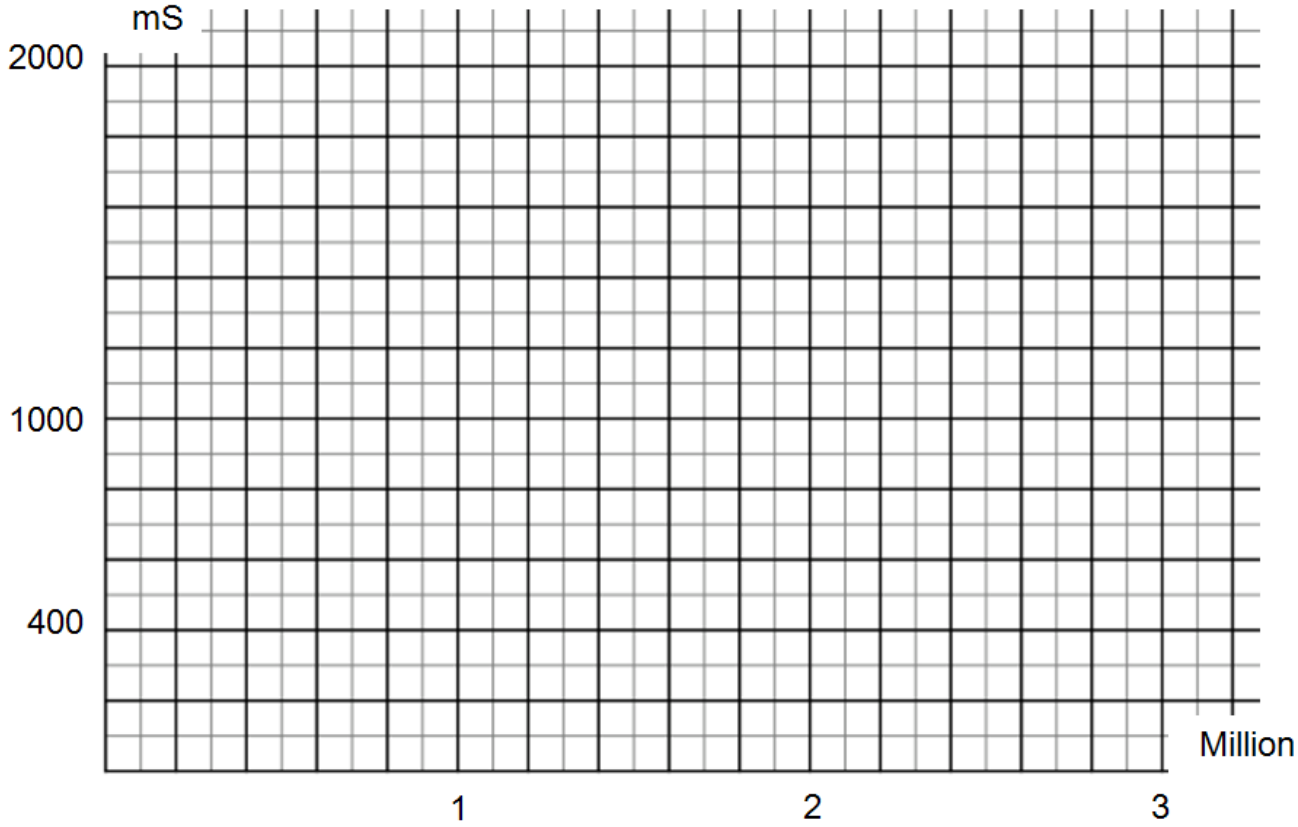
Nom :	**Note : / 20**
	Classe :

Premier exemple de complexité

Q1. Vérifier le coût linéaire de l'algorithme.

Longueur de la liste	Durée de traitement mesurée en mS
200000	
400000	
1000000	
2000000	
3000000	

Durée de traitement en fonction du nombre d'éléments dans la liste



Étude comparative de trois algorithmes

Q2. Déduire de la même manière le coût de l'algorithme A2 en fonction de N.

Q3. Déduire de la même manière le coût de l'algorithme A3 en fonction de N.

Q4. Conclure quant à la rapidité des trois algorithmes quand N est très grand.

Comparaison des trois algorithmes étude expérimentale

Nombre premier à tester	Algorithme A1 Script_complexité_1	Algorithme A2 Script_complexité_2	Algorithme A3 Script_complexité_3
17509			
35023			
70061			
140143			
280001			
560017			
1120001			

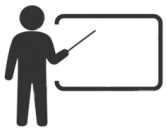
Q5. Les résultats expérimentaux sont-ils conforme à la théorie ?

Étude théorique de la complexité

Q6. Tester l'algorithme comment justifieriez-vous l'appellation linearithmique au vu du tracé des 4 courbes de complexité $O(n)$, $O(n^2)$, $O(n \cdot \log n)$, $O(\log n)$?

Recherche dichotomique de nombres dans une liste

Script_complexité_4. Compléter le script en python pour montrer le nombre de dichotomie pour diviser successivement un nombre N.



Q7. Effectuez la recherche sur les deux exemples ci-dessous :

TABLE[N]

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
7	11	23	41	45	66	71	79	82	94	101	111	124	138	142	157	168	190

VALEUR A TROUVER 157

INITIALISATION

DEB								MIL									FIN

TABLE[N]

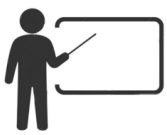
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
7	11	23	41	45	66	71	79	82	94	101	111	124	138	142	157	168	190

VALEUR A TROUVER 200

INITIALISATION

DEB								MIL									FIN

Script_complexité_5. Codez la recherche dichotomique en Python en complétant le script ci-dessous.



Q8. A partir de votre script précédent vérifiez que le doublement du nombre des entrées dans la liste ne rajoute qu'une seule étape à la recherche d'une valeur.