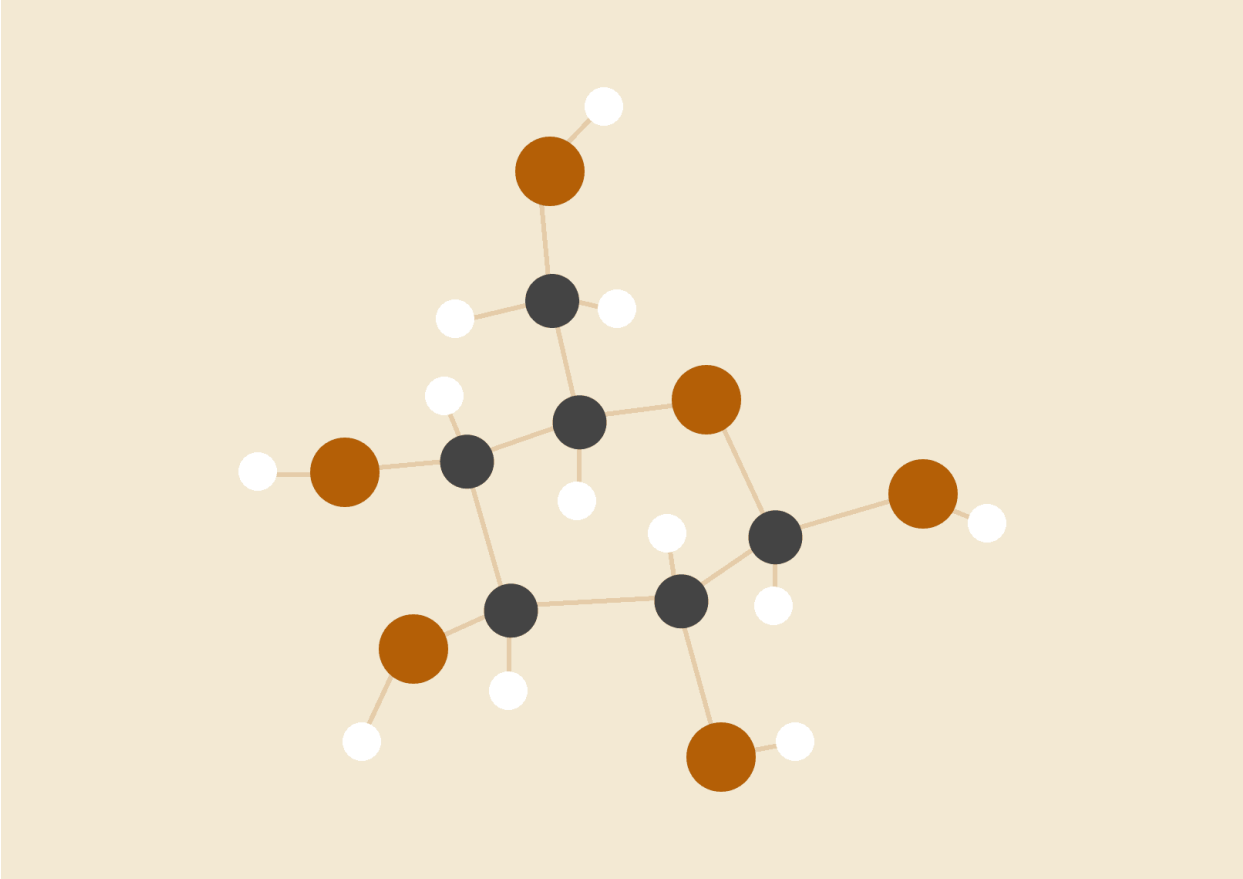
Rapport de TP
Calculs de temps d'exécution



Kamanda Aubin
Placé Louka

17/02/2023
UFR Sciences et Technique - L3 MIAGE
Conception & Analyse d'Algorithmes

Kamanda Aubin

Placé Louka

Faculté des sciences et des techniques

Sommaire

Kamanda Aubin

Placé Louka

Faculté des sciences et des techniques

I) Préambule

L'objectif de ce TP est de vérifier que les résultats théoriques obtenus en TD correspondent (du moins en ordre de grandeur) aux résultats dans la pratique. Le langage utilisé pour implémenter les algorithmes est Python. Ici on va s'intéresser à l'algorithme mystère de l'exercice 1.2 du TD 1 et les algorithmes Max-Somme-Consécutive de l'exercice 1.8 du TD 1.

Dans le premier exercice la fonction mystère renvoie la première position d'un élément 'TRUE' dans le tableau ou la taille du tableau +1 si pas de 'TRUE' trouvé dans le tableau.

Dans le second on va s'intéresser à 3 fonctions aux complexités différentes qui vont nous permettre de calculer la somme consécutive la plus grande dans un tableau. On s'intéressera au coût de toutes ces fonctions, notamment leur coût au pire, au mieux et en moyenne.

Kamanda Aubin

Placé Louka

Faculté des sciences et des techniques

I) Algorithme Mystères

a) Explications de l'algorithme de génération des instances

Tout d'abord l'utilisateur rentre son nombre n de taille de tableau qu'il souhaite. Ensuite on créer un tableau 'array**s**' qui va contenir toutes les combinaisons différentes de tableau de booléen possible. Avec une boucle qui va jusqu'à 2^n (qui est le nombre de possibilité de combinaisons différentes), la librairie *numpy* va nous générer tous les tableaux 'array' avec des combinaisons différentes contenant 'FALSE' et/ou 'TRUE' qu'on va ensuite insérer dans le tableau de tableaux 'arrays'.

Toutes les instances de l'algorithme mystère se trouvent donc dans le tableau de tableaux 'arrays'.

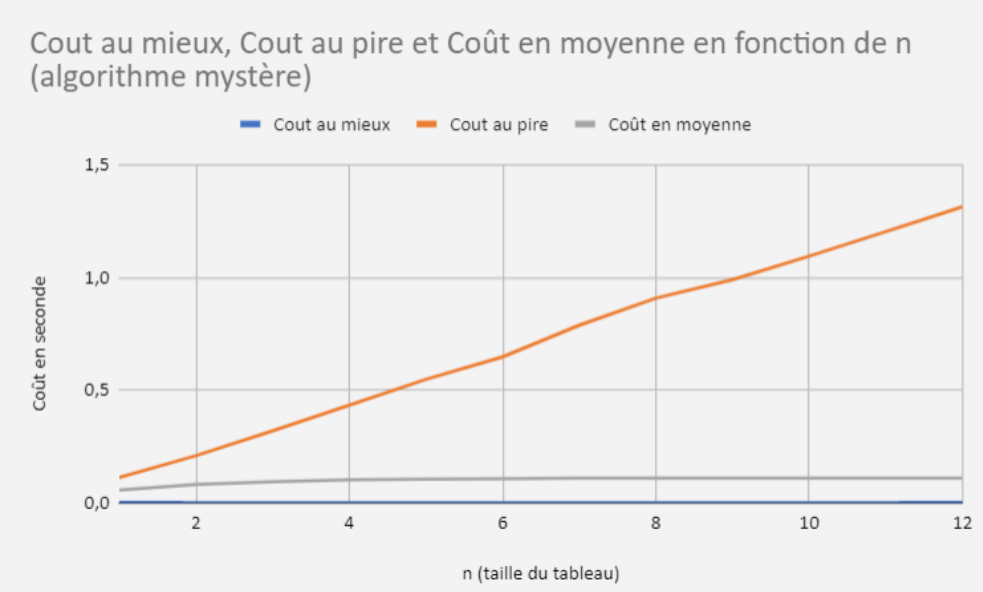
b) Visualisation des temps d'exécutions

Nous avons effectué des tests en faisant varier les tailles des tableaux n. À préciser que nous avons ajouté un délai de 0.1 secondes pour chaque tour de boucle de la fonction mystère. C'est-à-dire qu'à chaque fois que dans le tableau la valeur est égale à 'FALSE' l'algorithme va être ralenti de 0.1 seconde. Cela nous a été utile afin de mettre en évidence les différents coûts de l'algorithme. À savoir que la fonction mystère parcourt les tableaux jusqu'au premier 'TRUE' dans celui-ci. On verra ci-contre que pour chaque taille de tableau n il existera toujours une possibilité que 'TRUE' soit le premier élément du tableau, donc le coût au mieux sera toujours égale à 0 au vu de la rapidité d'exécution.

Nous nous sommes arrêtés à la valeur 12 de n car celle-ci dépassait déjà les 3 minutes, 3 minutes 46 précisément.

n	Cout au mieux	Cout au pire	Coût en moyenne	instances
1	0	0,1130096912	0,05650484562	2
2	0	0,2094099522	0,08137345314	4
3	0	0,3192913532	0,09398823977	8
4	0	0,4332048893	0,1028315872	16
5	0	0,5476620197	0,1058706865	32
6	0	0,6471869946	0,1067406423	64
7	0	0,7876737118	0,1090113185	128
8	0	0,9078555107	0,1092677442	256
9	0	0,9900512695	0,1102293013	512
10	0	1,094008684	0,1101159712	1024
11	0	1,203361034	0,1098555343	2048
12	0	1,312713146	0,1098146649	4096

c) Discussions des résultats TP/TD



En TD nous avons déterminé que la complexité au mieux, $C_{mieux}(n)$ était constante donc O(1) et nous constatons que c'est en cohérence avec nos résultats car la courbe du coût au mieux est toujours égale à 0.

De plus le coût au pire, $C_{pire}(n)$ était lui linéaire donc O(n), et nous constatons encore une fois que nos résultats sont toujours cohérent car la courbe du coût au pire grandit linéairement en fonction de la taille n à quelque exception près car les performances de l'ordinateur varient.

Enfin pour le coût moyen $C_{moyen}(n)$, nous avons vu en TD que pour n tendant vers $+\infty$ la complexité était constante donc O(1). Ici le coût moyen tend vers $\simeq$ 0.109 secondes, donc la complexité est bien constante dans nos résultats.

II) Algorithmes Max-Somme-Consécutive

a) Explications de l'algorithme de génération des instances

Tout d'abord l'utilisateur rentre son nombre n de taille de tableau qu'il souhaite, le nombre d'instance à générer est fixe et est égale à 200. On va donc boucler 200 fois sur une création d'un tableau avec les conditions demandées. Premièrement pour s'assurer que chaque tableaux contiennent un entier positif et négatif on va les ajouter directement à l'aide de la librairie *numpy* et de sa commande 'append'. Ensuite, on va remplir le reste du tableau avec des entiers générés aléatoirement entre [-n/2;n/2] si la valeur n renseigner est pair. Si elle est impair, la valeur de -n/2 sera arrondie à l'entier supérieur avec la fonction 'floor' et arrondie à l'entier inférieur pour n/2 avec la fonction 'ceil' de la librairie *math* afin que les nombres soit des entiers et dans l'intervalle demandé. De plus, si la valeur 0 est générée, elle n'est pas ajoutée au tableau. La fonction 'random.shuffle' est effectuée sur chaque tableau pour lui donner un côté plus aléatoire en mélangeant les éléments de celui-ci.

Toutes les instances de tableaux pour les algorithmes *Max-Somme-Consécutive* sont créées à la volée dans cet algorithme.

b) Visualisation des temps d'exécutions

1) MaxSomme1

Nous avons effectué des tests en faisant varier les tailles des tableaux n de 500 en 500. À préciser que nous n'avons ajouté aucun délai pour calculer le délai de la fonction MaxSomme1.

Nous nous sommes arrêtés à la valeur 4 000 de n car celle-ci excédait les 3 minutes.

	Maxsomme1			
n	Cout au mieux	Cout au pire	Coût en moyenne	dépassement
500	0	0,04681921005	0,01570019245	
1000	0,04679751396	0,09373140335	0,06201666236	
1500	0,109275341	0,3436703682	0,1643219411	
2000	0,2030780315	0,5101613998	0,2555558944	
2500	0,3280453682	0,8435087204	0,407950629	
3000	0,5946617126	2,013379812	0,793775785	
3500	0,8928854465	2,504295826	1,129620709	
4000	1,072060585	4,355797291	1,588667359	X

2) MaxSomme2

Nous avons effectué des tests en faisant varier les tailles des tableaux n de 500 en 500. À préciser que nous n'avons ajouté aucun délai pour calculer le délai de la fonction MaxSomme2.

Nous nous sommes arrêtés à la valeur 16 000 de n car celle-ci outrepassait les 3 minutes.

	MaxSomme 2			
n	Cout au mieux	Cout au pire	Coût en moyenne	Dépassement
500	0	0,02496576309	0,003139314651	
1000	0	0,02693128586	0,008860555887	
1500	0	0,03241705894	0,01077268958	
2000	0	0,04787015915	0,01559933543	
2500	0	0,04015922546	0,01739808917	
3000	0,0009958744049	0,04961228371	0,02271049857	
3500	0,004964590073	0,05006814003	0,02565024257	
4000	0,01558804512	0,05877876282	0,0301170361	
4500	0,01671624184	0,06290721893	0,03528851151	
5000	0,01567029953	0,06711101532	0,03809979796	
5500	0,01676654816	0,08759689331	0,04138406038	
6000	0,03236675262	0,09312057495	0,05333081722	
6500	0,03250193596	0,09833669662	0,05132737994	
7000	0,03124213219	0,2367238998	0,07002454042	
7500	0,03129911423	0,1060323715	0,06433831096	
8000	0,0337677002	0,1043617725	0,06152983189	
8500	0,04687190056	0,1510367393	0,07005914092	
9000	0,04815936089	0,1584694386	0,07442971706	
9500	0,04920887947	0,1312043667	0,076796242	
10000	0,04925251007	0,1532962322	0,08017014027	
10500	0,04912352562	0,2001924515	0,08766469121	
11000	0,05124926567	0,1818737984	0,09136802673	
11500	0,06243634224	0,1718342304	0,08089987755	
12000	0,0624320507	0,1405923367	0,07435993195	
12500	0,06242847443	0,1562139988	0,07701139927	
13000	0,06243634224	0,1717913151	0,08182826519	
13500	0,06242966652	0,1718394756	0,08671717763	
14000	0,06242752075	0,1405904293	0,0866665113	
14500	0,0780582428	0,1562111378	0,09455894947	
15000	0,07805657387	0,2186496258	0,09950929761	
15500	0,0780570507	0,1718361378	0,09802909732	
16000	0,07806015015	0,2968041897	0,1151315057	X

3) MaxSomme3

Nous avons effectué des tests en faisant varier les tailles des tableaux n de 500 en 500. À préciser que nous n'avons ajouté aucun délai pour calculer le délai de la fonction MaxSomme3.

Nous nous sommes arrêtés à la valeur 14 500 de n car celle-ci transgressait les 3 minutes. À préciser que ces tests-là ont été effectués sur un PC du CIE.

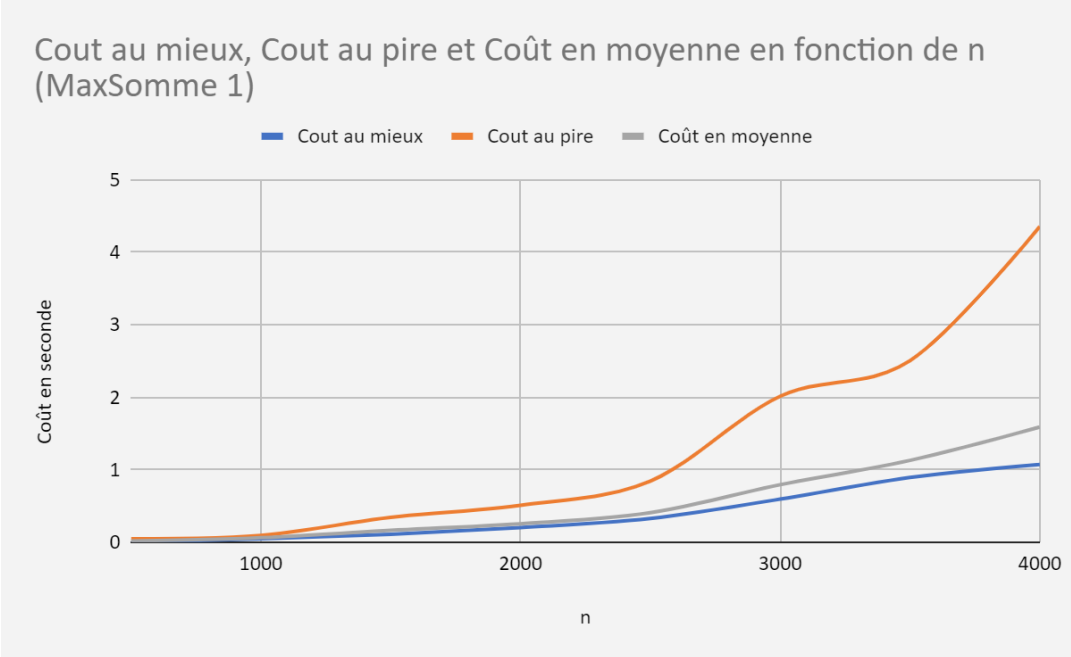
	Max somme3			
n	Cout au mieux	Cout au pire	Coût en moyenne	Dépassement
500	0,007488489151	0,01346111298	0,009032180309	
1000	0,00560593605	0,01819658279	0,01065340519	
1500	0,006049633026	0,01441764832	0,01044534683	
2000	0,005931854248	0,03354358673	0,02265189648	
2500	0,009936571121	0,03627634048	0,02218513727	
3000	0,009723901749	0,03359699249	0,0226641047	
3500	0,009189128876	0,03730177879	0,02413204789	
4000	0,01014924049	0,03975129128	0,02313806415	
4500	0,008969783783	0,04754257202	0,02361319661	
5000	0,008090496063	0,03711366653	0,0238406384	
5500	0,006836414337	0,04279732704	0,02310452223	
6000	0,01160478592	0,03502249718	0,02461559772	
6500	0,01027059555	0,0372428894	0,02502384305	
7000	0,01057887077	0,04039931297	0,02294796824	
7500	0,01116418839	0,0405189991	0,02534853697	
8000	0,009073495865	0,03738760948	0,02403950453	
8500	0,008400440216	0,03786301613	0,0230738616	
9000	0,01100730896	0,03994202614	0,02345863938	
9500	0,01307034492	0,03529691696	0,02368149042	
10000	0,01258969307	0,03199601173	0,01836662412	
10500	0,01384091377	0,05974555016	0,02433291554	
11000	0,01045012474	0,03580188751	0,02362783551	
11500	0,01352620125	0,03446292877	0,02417132139	
12000	0,006069421768	0,03249502182	0,02361393929	
12500	0,01090478897	0,03388881683	0,02425680518	
13000	0,01257157326	0,03758311272	0,02412302017	
13500	0,01440358162	0,03537583351	0,02532954693	
14000	0,009285926819	0,03874588013	0,02514721751	
14500	0,015666008	0,0444726944	0,02554201126	X

c) Discussions des résultats TP/TD

1) MaxSomme1

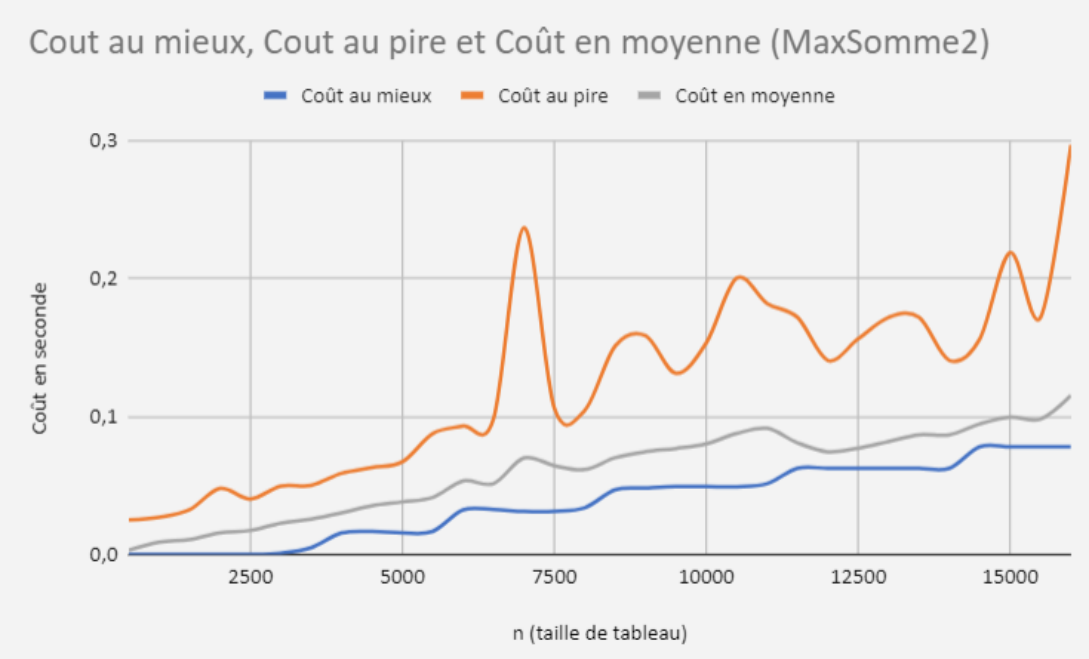
En TD nous avons déterminé que la complexité C_{pire} et C_{mieux} était $\Theta(n^2)$. De plus la courbe représentative de O(n * log(n)) semble elle très linéaire. Avec nos valeurs entre 500 et 3 000 on ne le voit pas forcément bien, mais si on est attentif à ce qui se passe vers la fin donc pour n égale 3500 et 4000 on constate que l'évolution commence à se faire de manière quadratique pour le coût au mieux et l'est pour le coût au pire.

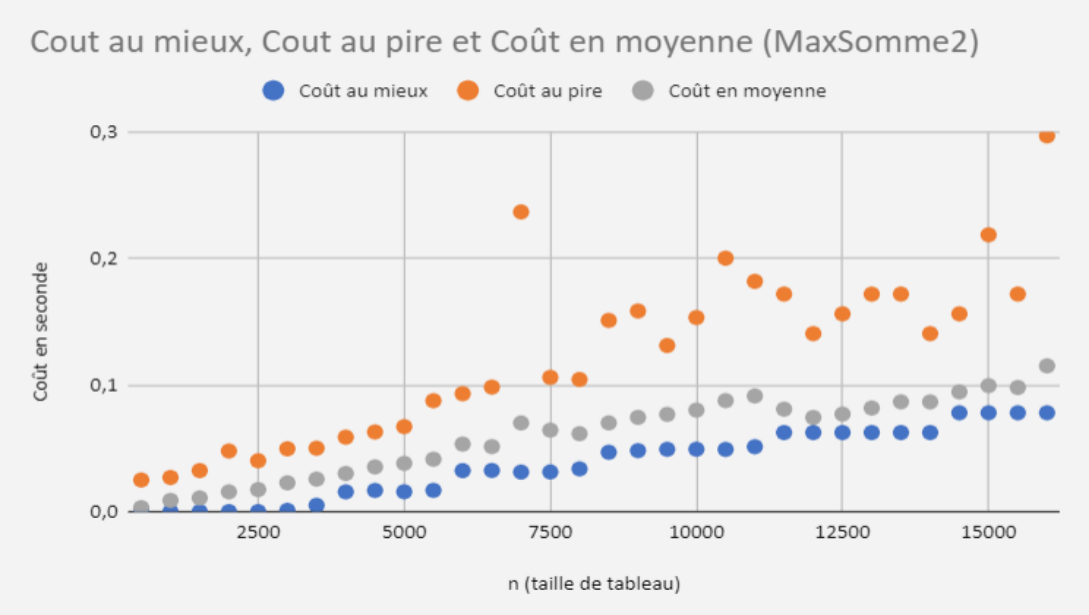
Cela illustre donc la cohérence des résultats du TD ainsi que celle de ce TP.



2) MaxSomme2

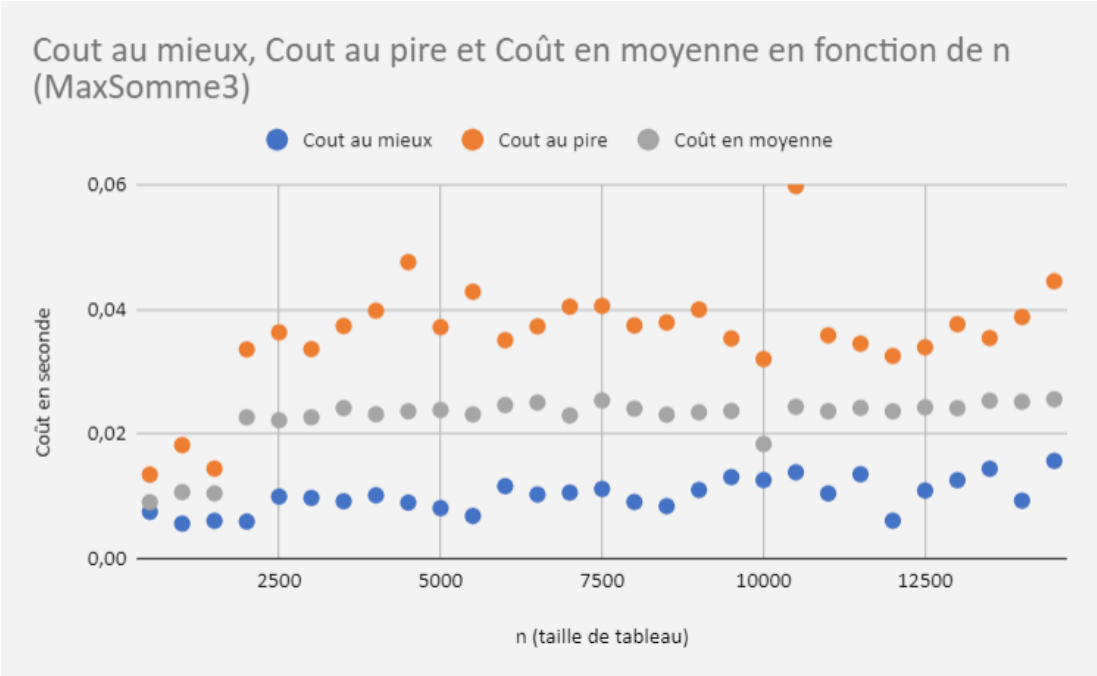
En TD nous avons déterminé que la complexité au pire, $C_{pire}(n)$ était de O(n * log(n)) et que la fonction est polylogarithmique(avec le mélange de polynomial et logarithmique). De plus la courbe représentative de O(n * log(n)) est assez linéaire. Avec nos valeurs sur la courbe on ne le voit pas forcément bien, c'est pour cela que nous avons ajouté le nuage de points associés afin de faciliter le visuel et pour effectivement démontrer la cohérence des résultats du TD ainsi que celle de ce TP.





3) MaxSomme3

En TD nous avons vu que le parcours du tableau est linéaire et que ce que l'on fait à l'intérieur du tableau est constant. Ce qui nous donne $C_{mieux} = \Omega(n)$ et $C_{pire} = O(n)$ et que les deux donne une complexité de $c(n) = \Theta(n)$.
Ici à l'aide de ce nuage de points des coûts au pire, au mieux et en moyenne en fonction de la taille n du tableau, on peut clairement dégager une tendance linéaire plutôt faible pour le coût au mieux et au pire pour la fonction MaxSomme3 comme vu en TD.



CONCLUSION

Nous en concluons que malgré des valeurs pas souvent très précises du fait des différents ordinateurs utilisés et de leurs performances qui peuvent varier, nous arrivons quand même toujours à en dégager au moins une tendance et des ordres de grandeur. Grâce à ce TP nous avons pu confirmer les différents résultats étudiés lors des séances de TD.

Kamanda Aubin 17/02/2023

Placé Louka L3 MIAGE