SAE S1.02 – Comparaison d'approches algorithmiques

Recherche d'une sous-chaîne dans un texte

Auteurs : Sanders Leandre – Jolle Bryan
Groupe : TD B2

Introduction

L'objectif de cette SAE est de comparer plusieurs approches algorithmiques permettant de résoudre le problème de la recherche d'une sous-chaîne (motif) dans un texte. Ce problème est central en informatique et intervient dans de nombreux domaines tels que les éditeurs de texte ou l'analyse de séquences.

Dans ce travail, nous avons implémenté et étudié quatre algorithmes : - l'algorithme naïf, - l'algorithme de Knuth-Morris-Pratt (KMP), - l'algorithme de Rabin-Karp, - l'algorithme de Boyer-Moore.

Leur efficacité est comparée à l'aide de mesures empiriques basées sur le nombre d'opérations élémentaires et le temps d'exécution.

---

1. Présentation du problème

Le problème consiste à trouver toutes les occurrences d'un motif (chaîne de caractères) dans un texte donné. Le texte est représenté sous la forme d'un `ArrayList<Character>` et le motif sous la forme d'une chaîne `String`.

Toutes les positions où le motif apparaît dans le texte doivent être détectées.

---

2. Méthodologie d'évaluation

Pour chaque algorithme, nous avons : - vérifié le bon fonctionnement sur des exemples simples, - mesuré le nombre d'opérations élémentaires à l'aide d'un compteur `cpt`, - mesuré le temps d'exécution, - réalisé des tests sur différents types de textes : - textes aléatoires, - textes répétitifs.

Les résultats sont présentés sous forme de tableaux et de graphiques.

---

3. Algorithme naïf

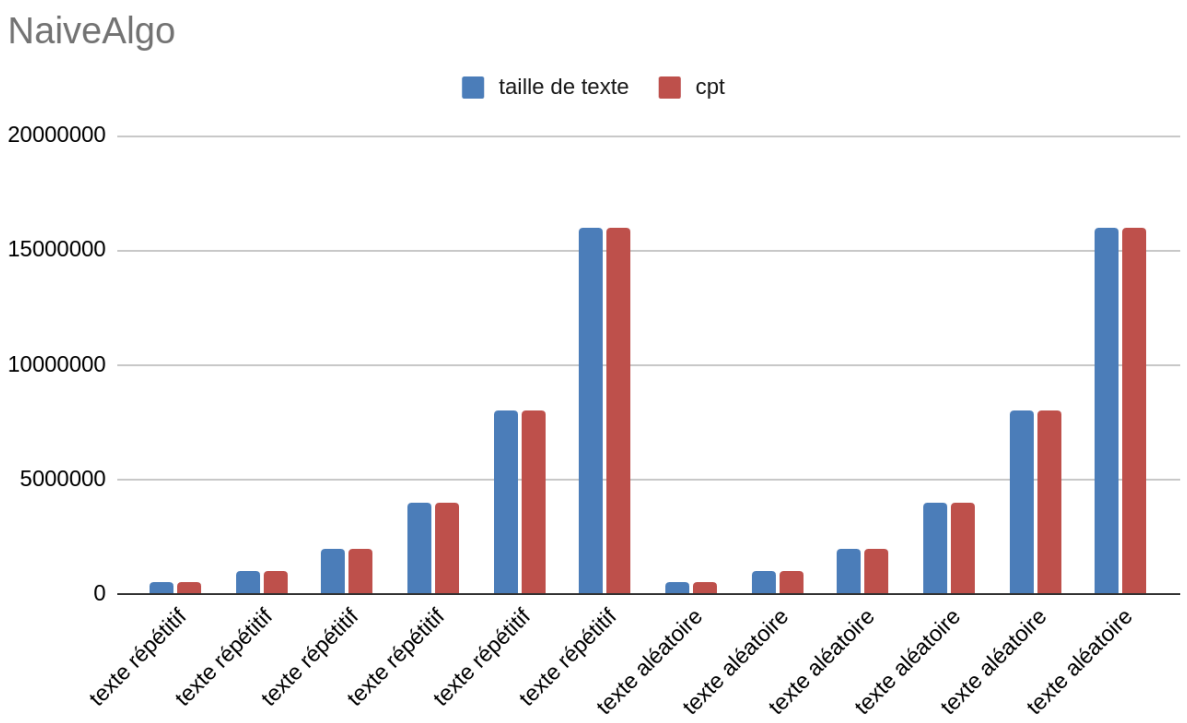
3.1 Principe

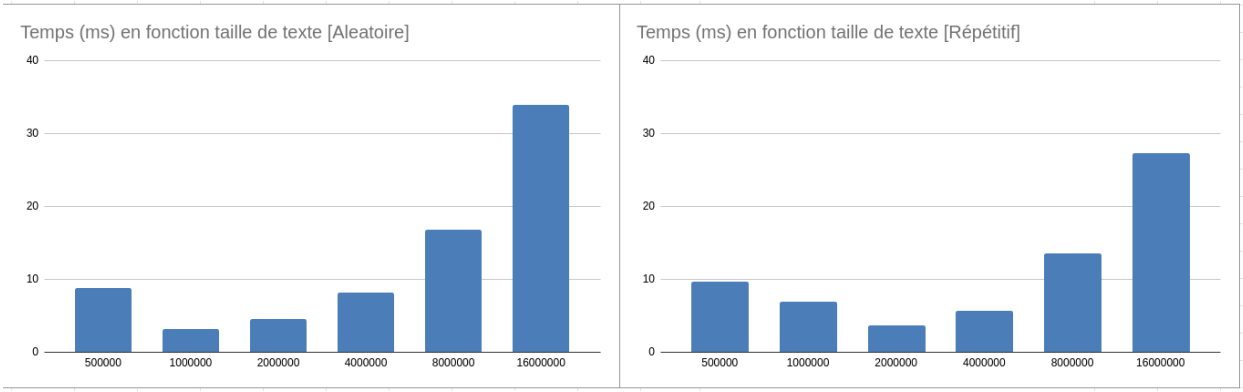
L'algorithme naïf compare le motif avec chaque sous-chaîne possible du texte, caractère par caractère, en décalant le motif d'une position à chaque tentative.

3.2 Résultats expérimentaux

NaiveAlgo	taille de texte	O(n?)	# cpt	# cpt/n	constante	# temps(ns)	temps(ms)
texte répétitif	500000	O(n × m)	499999	0,999998	≈1	9564334	9,564334
texte répétitif	1000000	O(n × m)	999999	0,999999	≈1	6836565	6,836565
texte répétitif	2000000	O(n × m)	1999999	0,9999995	≈1	3573413	3,573413
texte répétitif	4000000	O(n × m)	3999999	0,99999975	≈1	5668588	5,668588
texte répétitif	8000000	O(n × m)	7999999	0,999999875	≈1	13455945	13,455945
texte répétitif	16000000	O(n × m)	15999999	0,9999999375	≈1	27257060	27,25706
texte aléatoire	500000	O(n × m)	499998	0,999996	≈1	8712028	8,712028
texte aléatoire	1000000	O(n × m)	999998	0,999998	≈1	3072678	3,072678
texte aléatoire	2000000	O(n × m)	1999998	0,999999	≈1	4479440	4,47944
texte aléatoire	4000000	O(n × m)	3999998	0,9999995	≈1	8072733	8,072733
texte aléatoire	8000000	O(n × m)	7999998	0,99999975	≈1	16700962	16,700962
texte aléatoire	16000000	O(n × m)	15999998	0,999999875	≈1	33893194	33,893194

Graphique – nombre d'opérations en fonction de la taille du texte





3.3 Commentaires

Les résultats montrent que le temps d'exécution augmente avec la taille du texte, ce qui confirme un comportement essentiellement linéaire de l'algorithme naïf. Les textes aléatoires sont légèrement plus coûteux à traiter que les textes répétitifs, mais la taille du texte reste le facteur le plus déterminant.

---

4. Algorithme de Knuth-Morris-Pratt (KMP)

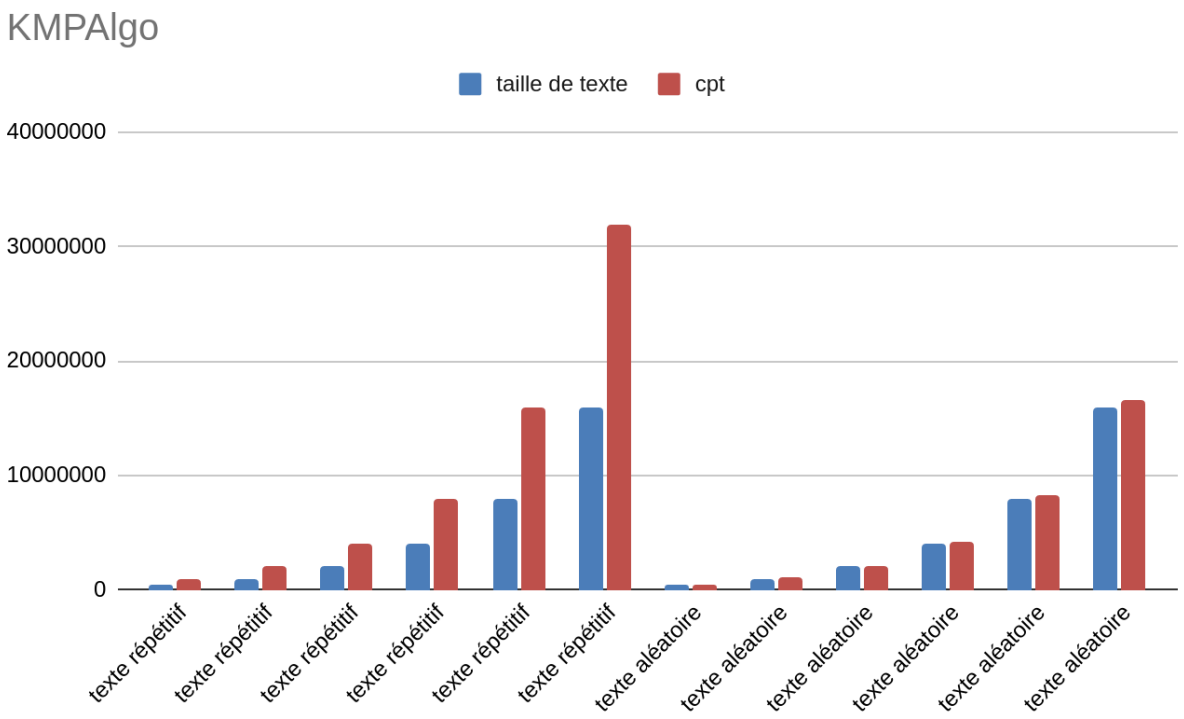
4.1 Principe

L'algorithme KMP améliore la recherche en évitant les comparaisons inutiles grâce à l'utilisation d'un tableau de préfixes.

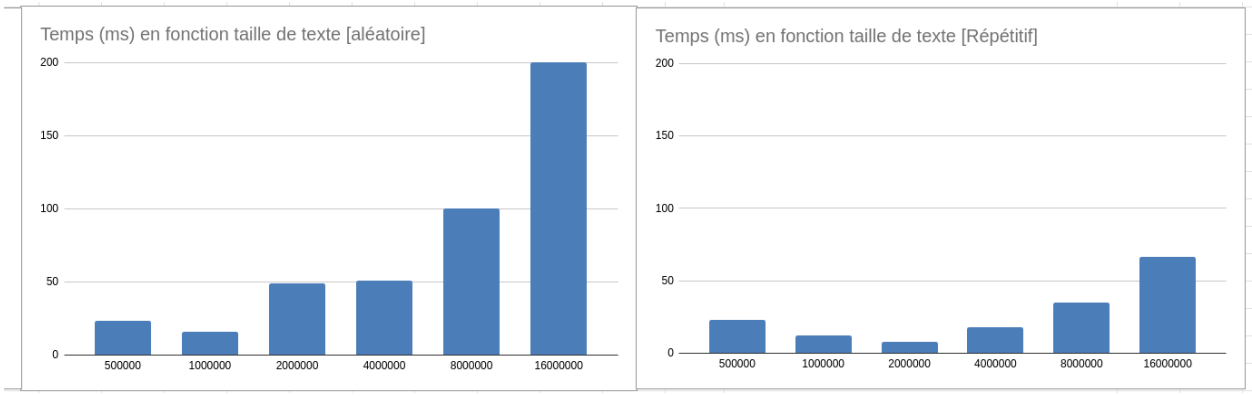
4.2 Résultats expérimentaux

kmpAlgo	taille de texte	O(n?)	# cpt	# cpt/n	constante	temps(ns)	temps(ms)
texte répétitif	500000	O(n)	999992	1,999984	2	23154295	23,154295
texte répétitif	1000000	O(n)	1999992	1,999992	2	15492623	15,492623
texte répétitif	2000000	O(n)	3999992	1,999996	2	48696357	48,696357
texte répétitif	4000000	O(n)	7999992	1,999998	2	50530678	50,530678
texte répétitif	8000000	O(n)	15999992	1,999999	2	99925554	99,925554
texte répétitif	16000000	O(n)	31999992	1,9999995	2	199965760	199,96576
texte aléatoire	500000	O(n)	519307	1,038614	1	22595690	22,59569
texte aléatoire	1000000	O(n)	1038774	1,038774	1	12145799	12,145799
texte aléatoire	2000000	O(n)	2076561	1,0382805	1	7633950	7,63395
texte aléatoire	4000000	O(n)	4153349	1,03833725	1	17627345	17,627345
texte aléatoire	8000000	O(n)	8306602	1,03832525	1	34969266	34,969266
texte aléatoire	16000000	O(n)	16614556	1,03840975	1	66307577	66,307577

Graphique – nombre d'opérations en fonction de la taille du texte



Graphique – temps d'exécution en fonction de la taille du texte



4.3 Commentaires

Les résultats montrent que l'algorithme KMP reste efficace même lorsque la taille du texte augmente, avec une croissance maîtrisée du temps d'exécution. Les textes répétitifs sont traités plus rapidement que les textes aléatoires, ce qui illustre l'avantage de KMP à exploiter les répétitions et à éviter les comparaisons inutiles.

---

5. Algorithme de Rabin-Karp

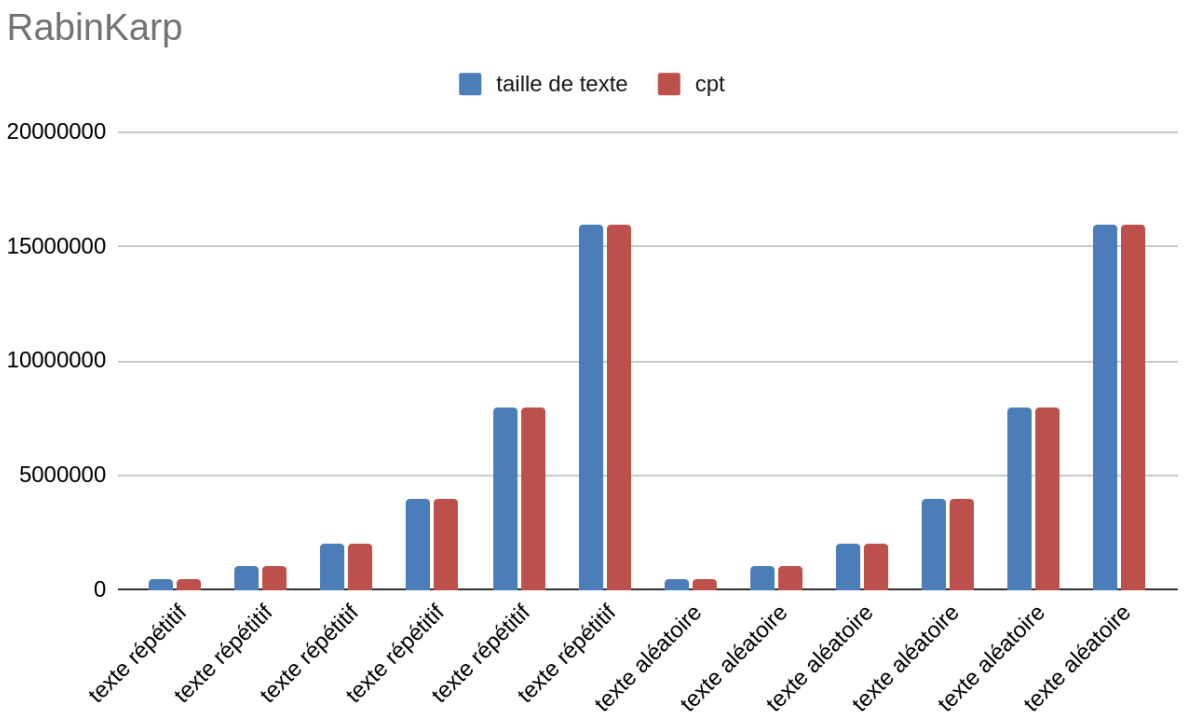
5.1 Principe

L'algorithme de Rabin-Karp utilise une fonction de hachage déroulante pour comparer rapidement le motif avec des portions du texte.

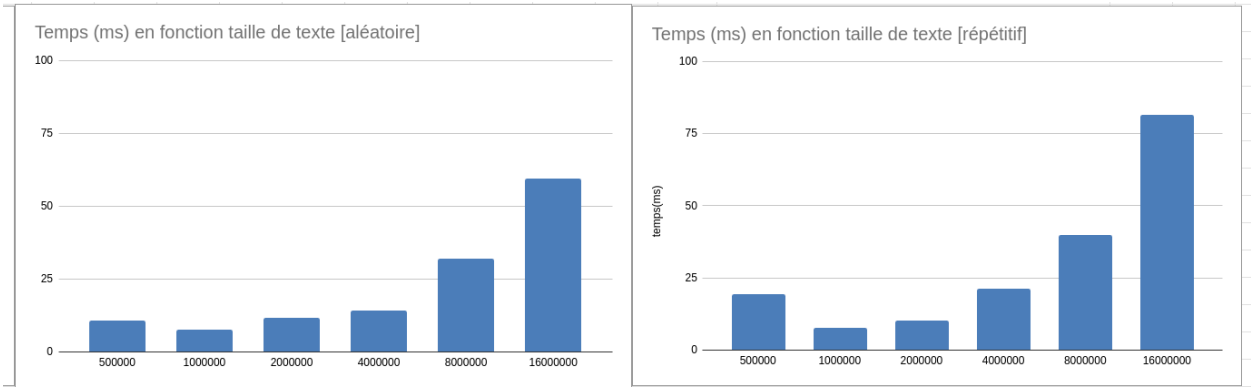
5.2 Résultats expérimentaux

RabinKarpAlgo	taille de texte	O(n?)	#	cpt	#	cpt/n	constante	#	temps(ns)	temps(ms)
texte répétitif	500000	O(n × m)		500001		1,000002	≈1		19328283	19,328283
texte répétitif	1000000	O(n × m)		1000001		1,000001	≈1		7550108	7,550108
texte répétitif	2000000	O(n × m)		2000001		1,0000005	≈1		10211293	10,211293
texte répétitif	4000000	O(n × m)		4000001		1,00000025	≈1		21218475	21,218475
texte répétitif	8000000	O(n × m)		8000001		1,000000125	≈1		40020527	40,020527
texte répétitif	16000000	O(n × m)		16000001		1,000000063	≈1		81435418	81,435418
texte aléatoire	500000	O(n × m)		500100		1,0002	≈1		10695791	10,695791
texte aléatoire	1000000	O(n × m)		1000190		1,00019	≈1		7638049	7,638049
texte aléatoire	2000000	O(n × m)		2000367		1,0001835	≈1		11502251	11,502251
texte aléatoire	4000000	O(n × m)		4000706		1,0001765	≈1		14109136	14,109136
texte aléatoire	8000000	O(n × m)		8001453		1,000181625	≈1		31819132	31,819132
texte aléatoire	16000000	O(n × m)		16002782		1,000173875	≈1		59335455	59,335455

Graphique – nombre d'opérations en fonction de la taille du texte



Graphique – temps d'exécution en fonction de la taille du texte



5.3 Commentaires

Les résultats indiquent que l'algorithme de Rabin-Karp présente une augmentation du temps d'exécution avec la taille du texte, tout en restant globalement efficace. Les performances sont comparables pour les textes répétitifs et aléatoires, ce qui montre que l'algorithme est peu sensible à la nature du texte, la taille étant le facteur principal.

---

6. Algorithme de Boyer-Moore

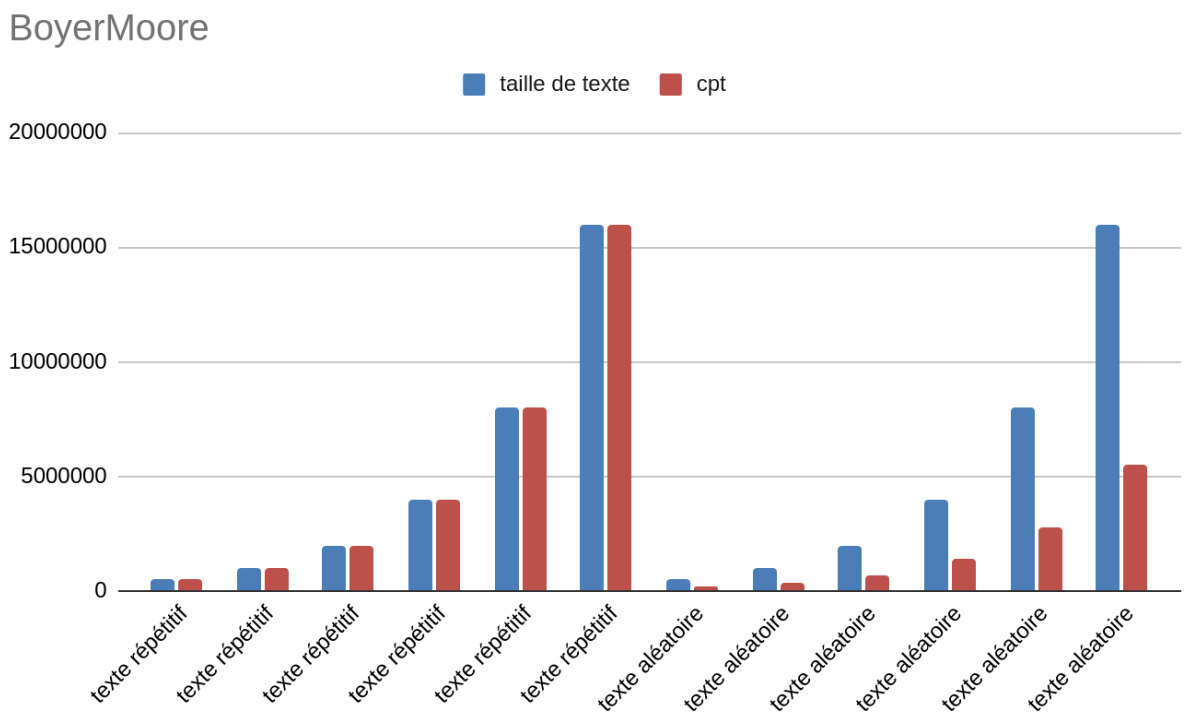
6.1 Principe

L'algorithme de Boyer-Moore compare le motif en partant de la fin et utilise des règles de décalage avancées pour accélérer la recherche.

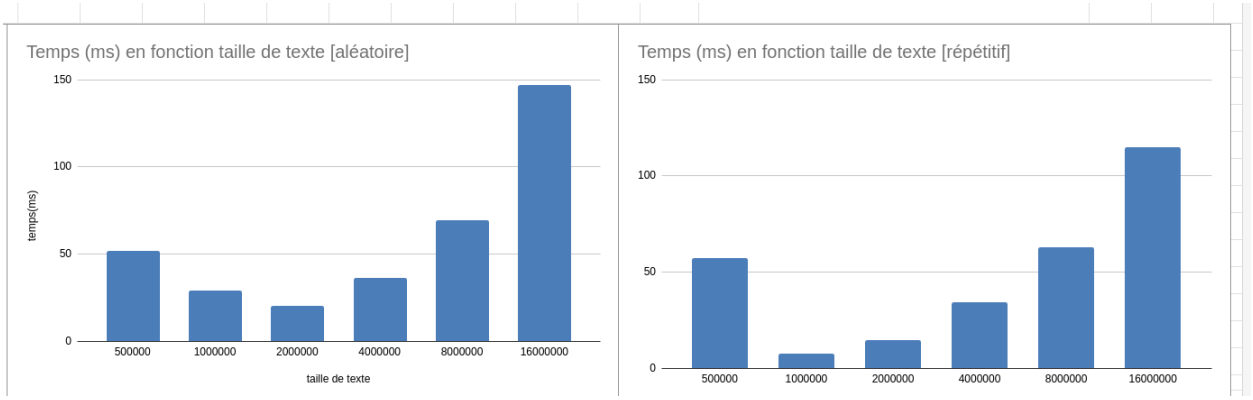
6.2 Résultats expérimentaux

BoyerMooreAlgo	taille de texte	O(n?)	# cpt	# cpt/n	constante	# temps(ns)	temps(ms)
texte répétitif	500000	O(n)	499999	0,999998	1	51697039	51,697039
texte répétitif	1000000	O(n)	999999	0,999999	1	29213820	29,21382
texte répétitif	2000000	O(n)	1999999	0,9999995	1	20231074	20,231074
texte répétitif	4000000	O(n)	3999999	0,99999975	1	36323557	36,323557
texte répétitif	8000000	O(n)	7999999	0,999999875	1	69459698	69,459698
texte répétitif	16000000	O(n)	15999999	0,9999999375	1	147090041	147,090041
texte aléatoire	500000	O(n)	173333	0,346666	0,35	57246514	57,246514
texte aléatoire	1000000	O(n)	346520	0,34652	0,35	7465514	7,465514
texte aléatoire	2000000	O(n)	693097	0,3465485	0,35	14460619	14,460619
texte aléatoire	4000000	O(n)	1386431	0,34660775	0,35	34366542	34,366542
texte aléatoire	8000000	O(n)	2773673	0,346709125	0,35	62607764	62,607764
texte aléatoire	16000000	O(n)	5547213	0,3467008125	0,35	114744654	114,744654

Graphique – nombre d'opérations en fonction de la taille du texte



Graphique – temps d'exécution en fonction de la taille du texte



6.3 Commentaires

Les graphiques de Boyer-Moore-Horspool montrent que le temps d'exécution augmente avec la taille du texte, mais de manière plus modérée que pour l'algorithme naïf. L'algorithme est particulièrement performant sur les textes aléatoires, où les décalages importants réduisent le nombre de comparaisons. En revanche, sur les textes répétitifs, les gains sont plus limités, car les décalages sont plus courts, ce qui entraîne davantage de comparaisons.

Conclusion

Ce travail a permis de mettre en évidence les différences d'efficacité entre plusieurs algorithmes de recherche de sous-chaîne. Les résultats montrent que les algorithmes optimisés (KMP, Rabin-Karp, Boyer-Moore) sont globalement plus performants que l'algorithme naïf, en particulier pour les grands textes et selon la nature des données.

Une analyse plus fine montre que le choix de l'algorithme dépend fortement du type de texte et du motif recherché.
