Devoir de Programmation : Tries

Marco PONTI
Nicolas SERRES

December 2016

1 Introduction

Dans ce Devoir de Programmation, l'objectif est d'implémenter et de comparer les deux structures d'arbres de Patricia-Tries et de Hybride-Tries, afin de déterminer lequel est meilleur.

2 Choix de la Structure

Le Patricia-Tries

Lors de l'implémentation de cet algorithme un problème c'est posé, le choix de la structure.

Premièrement, la structure était un objet composé d'un préfixe et d'un vecteur de caractère ainsi que d'un second vecteur contenant des pointeurs vers les Patricia-tries fils, le problème étant la taille du vecteur en mémoire ainsi que le nombre de pointeur vers null. De plus, en java, un caractère ne peut etre mis à null, une simple modification aurait été de changer le vecteur de caractère en vecteur de chaine, mais la mémoire restait un problème.

La seconde était d'utiliser des Array-list, mais cela allait à l'encontre des arbres et des exemples montrés par les nos professeurs, de plus la complexité en pire cas aurait été de parcourir chaque caractères * la profondeur.

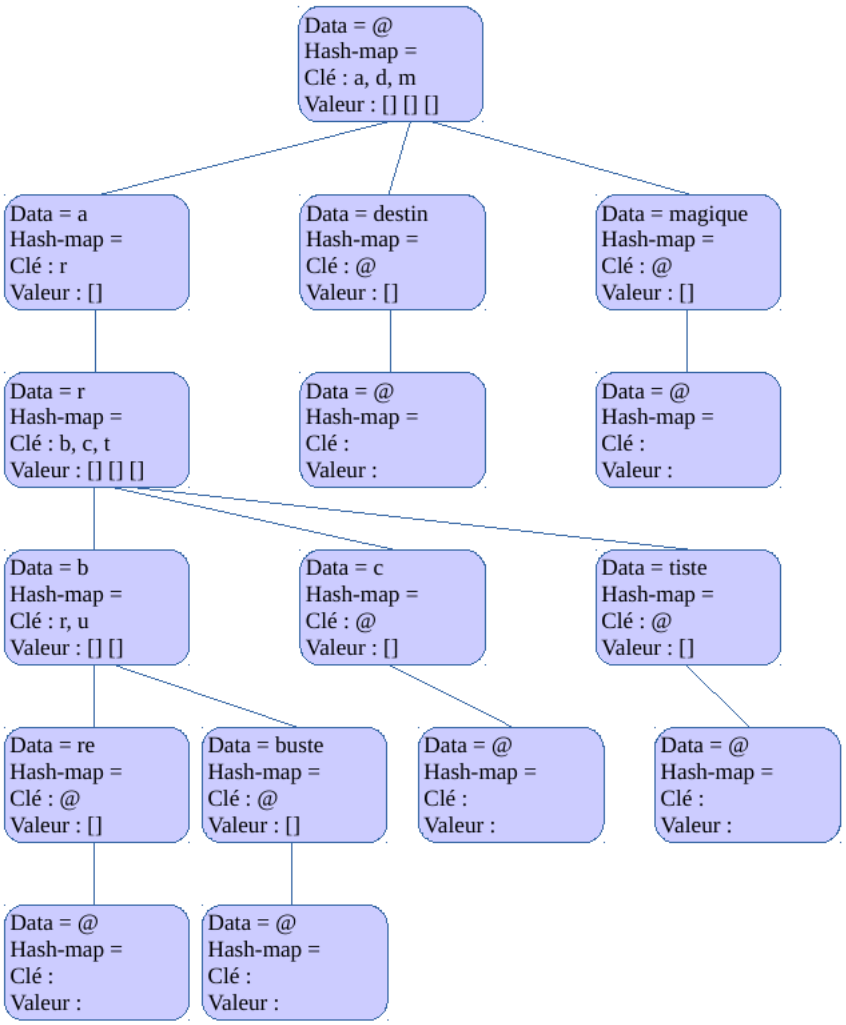
La troisième structure implémentée a été une simple hash-map avec comme clé le préfixe et en valeur un Patricia-Tries, malheureusement cette structure a été abandonné, car elle forait la plupart des algorithmes à devoir parcourir toute les clés de la hash-map comme le dernier algorithme ce qui faisait perdre tout l'intéret d'utiliser des hash-map.

La dernière implémentation, donc celle utilisée dans notre code, est un préfixe avec une hash-map composé de clés valant la première lettre du préfixe du fils et en valeur le fils.
L'arbre initial possède pour racine le mot vide, la structure ne présente aucun pointeur vers null, car tous les Patricias-Tries sont créés s'il y a l'existence d'une lettres et la hash-map est créé par défauts meme s'il se trouve etre vide.

Explication de la fonction de fusion :
La fonction parcours les deux arbres chacun au même niveau et test si les préfixes sont similaires.
Si oui elle insère chaque clé n'étant pas présente dans le premier arbre dans celui-ci, si elle existe, la fonction de fusion est alors lancée sur les deux fils.
Si les préfixes sont différents, la fonction sépare les préfixes en deux parties la racine commune et le reste des deux préfixes, puis une récursion est lancée sur

le nouveau noeud de la racine commune.

Exemple graphique d'un Patricia-Tries avec les mots :
arbre, arbuste, arc, artiste, destin et magique



Le Hybride-Tries

Chaque noeud hybride est constitué d'une lettre, d'un bollean pour savoir si on est a la fin d'un mot, d'un fil et de 2 freres.
Lorsque qu'un noeud a une lettre égale a null c'est que l'arbre est vide.

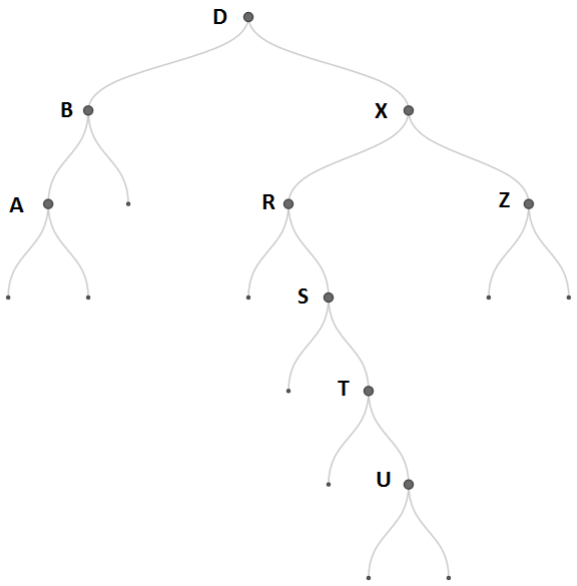
L'implémentation est ensuite en accord avec la structure des tries hybrides vu en cours.

Toutes les fonctions qui nécessitent de parcourir l'arbre utilise la récursivité afin d'avoir le code le plus simple possible.

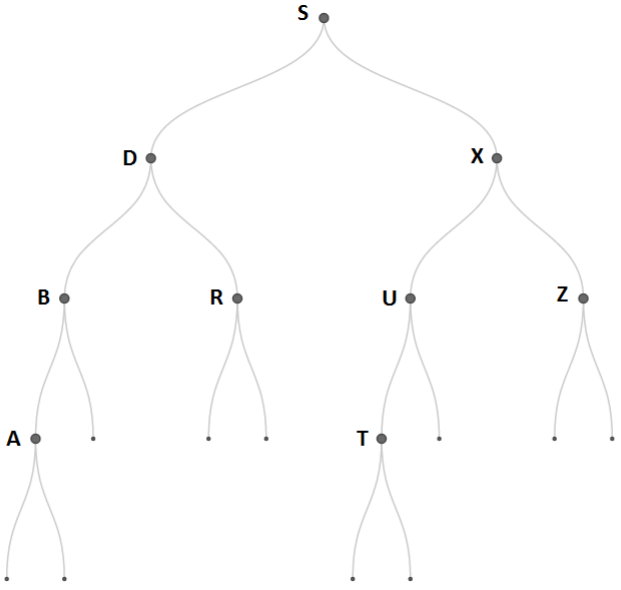
Pour les fonctions avec un déplacement spécial, il est expliqué directement dans le code par des commentaires.

La fonction de rééquilibrage est a détaillée :
Le rééquilibrage d'un trie hybride peut se faire au niveau des freres en les considérants comme des noeuds qu'on pourrait disposer de facon a obtenir un arbre complet.
Pour voir si les freres sont déséquilibrés (cf. ne forme pas un arbre complet) on regarde si la profondeur totale des freres est supérieure au log en base 2 du nombre de frere.
- ligne 604 : if (profondeurMaxFrere¿Math.abs(log2(nbFrere)))
Dans ce cas la, on appel listesFrereSorted qui récupere tous les freres, supprime leur lien frereGauche et frereDroit, puis les classe par ordre de leur lettre.
Puis on renvoie reformeFrere qui s'occupe de replacer les noeuds dans un arbre complet.
reformeFrere prenant chaque fois le milieu de la liste avec pour frere gauche le milieu de la liste gauche et pour frere droit le milieu de la liste droite.
On effectue ce test de rééquilibrage pour chaque liste de frere c'est a dire a chauque déplacement vers le bas dans l'arbre :
- arbre.fils= reequilibrageFrere(arbre.fils);

Hybrid-Tries avant rééquilibrage



Hybrid-Tries après rééquilibrage



3 Réponse aux Questions

Question 1.1 : Le caractère pour le fin d'un mot est '@'

Question 3.9 : On peut reequilibrer un trie hybride au niveau des freres. En effet si l'arbre est parfaitement équilibre tous les frere d'un meme noeud formerons un arbre binaire complet. C'est en ce basant sur cette logique qu'on trouve que le seuil est dépassé lorsque la profondeur max des freres est supérieure a la valeure absolue du log en base 2 du nombre total de frere.

Question 4.10 : Calcule des complexitées PatriciaTrie :

findPrefix = O (longeur du mot)
(Parcours d'une chaine de caractère et comparaisons pour chaque caractère)

displayPtree = O (nb max de caractère * profondeur)
(Parcours d'une Hashmap au pire des cas contenant tout les caratères possible)

cloneAll = O (nb max de caractère)
(Parcours d'une Hashmap au pire des cas contenant tout les caratères possible)

search = O (4 * (longeur du mot))
(Une comparaison puis lancement de la sous fonction 4 comparaisons pour chaque caractère du mot aux pire cas)

delete = O (4 * (longeur du mot))
(Une comparaison puis lancement de la sous fonction 4 comparaisons pour chaque caractère du mot aux pire cas)

insert = O ((5 * cloneAll) * longeur du mot)
(5 comparaison puis lancement de la fonction cloneAll, au pire cas pour chaque caractères du mot)

countWord = O (nb max de caractère * profondeur)
(Une comparaisons et parcours d'une Hashmap au pire des cas contenant tout les caratères possible pour toute la profondeur de l'arbre)

countDeep = O (nb max de caractère * profondeur)
(Une comparaisons et parcours d'une Hashmap au pire des cas contenant tout les caratères possible pour toute la profondeur de l'arbre)

arrayWord = O (nb max de caractère * profondeur)
(Une comparaisons et parcours d'une Hashmap au pire des cas contenant tout

les caratères possible pour toute la profondeur de l'arbre)

allWord = O (nb max de caractère * profondeur)
(Une comparaisons et parcours d'une Hashmap au pire des cas contenant tout les caratères possible pour toute la profondeur de l'arbre)

copy = O (nb max de caractère * profondeur)
(Parcours d'une Hashmap au pire des cas contenant tout les caratères possible pour toute la profondeur de l'arbre)

split = O (cloneAll)
(Deux comparaisons et lancement de la fonction cloneAll)

fusion = O (2 * (nb max de caractère * profondeur-min-d'un-des-deux-arbres))
(Deux comparaisons puis, parcours d'une Hashmap au pire des cas contenant tout les caratères possible pour toute la profondeur de l'arbre le plus court)

getDeep = O (nb max de caractère * profondeur)
(Une comparaisons et parcours d'une Hashmap au pire des cas contenant tout les caratères possible pour toute la profondeur de l'arbre le plus court)

mediumDeep = O (getDeep * profondeur)
(Appel getDeep puis Parcours la liste obtenu)

getPrefix = O (3 * longeur du mot)
(Trois comparaisons pour chaque caractère du mot)

convert = O ((nb caractère du préfix + nb max de caractère) * profondeur)
(Parcours du préfixe et parcours de la Hashmap des fils au pire des cas contenant tout les caratères possible pour toute la profondeur de l'arbre)

Calcule des complexitées des Tries Hybrides : (en nombre de comparaison)

ajouterMot = O (5 * profondeur de l'arbre + longeur du mot)
(5 comparaisons par noeud puis par lettre si nouveau mot sans lettre existante)

recherche = O (4 * (longeur du mot))
(5 comparaisons par lettre)

comptageMots = O (2 * nb de noeud)

listeMots = O (1 + (4 * nb de noeud))

comptageNil = O (nb de noeud)

hauteur = O (nb de noeud)

profondeurMoyenne = O (1 + (4 * nb de noeud))

prefixe = O ((2 + (4 * longeur du prefixe)) + (2 + (2 * nb de noeud)))
prefixe = O (4 + (4 * longeur du prefixe) + (2 * nb de noeud))
(complexité de deplacePrefixe puis de comptageMots)

suppression = O (1 + (4 * (longeur du mot)) + (hauteur de l'arbre) + (4 * longeur du mot * hauteur de l'arbre))
(complexité de recherche puis de unSeulMot et enfin de suppression2)

Conversion Hybrides vers Patricia = O (1 + (4 * nb de noeud) + (nb de mot * (5 + cloneAll * (longeur du mot))))
(complexité de listeMots puis de insert)

reequilibrage = O ((4 * Largeur de l'arbre) * nb de noeud)
(nbDeFrere puis profondeurMaxFrere. Ensuite listesFrereSorted et reformeFrere pour chaque branche)

Question 5.11 et 5.12 : Benchmark)

PatriciaTrie Benchmark (en nanoseconde)

file	build	insert	search	delete	fusion	nbword	deep
lll.txt	60158387	97607	21310	18158	73240	3637	9
comedy_errors.txt	30828771	9626	8974	5090	69606	2438	9
richardii.txt	40348389	10544	7622	4548	17864	3513	9
twelfth_night.txt	37910475	26658	25174	22286	58942	3030	10
merchant.txt	31662609	8439	8851	6290	14425	3160	9
merry_wives.txt	36051138	41725	8727	6371	13764	3188	9
henryv.txt	37330309	8890	9749	7000	15736	4393	10
1henryiv.txt	36286181	10872	8540	6038	13234	3723	9
2henryiv.txt	33863083	9354	9559	6408	14089	3963	10
1henryvi.txt	34660882	24048	15093	6288	15346	3725	9
3henryvi.txt	42945457	6891	8627	5857	13398	3448	9
measure.txt	25637238	7733	8691	6518	15188	3229	9
othello.txt	33100723	8314	8850	6166	13775	3665	9
taming_shrew.txt	37049511	5877	8590	6540	34292	3149	9
tempest.txt	17955945	19258	7521	4659	13692	3084	9
macbeth.txt	9685622	11288	12223	11339	13843	3204	9
asyoulikeit.txt	12434415	3797	8788	5691	13590	3170	10
allswell.txt	27530014	8522	9704	6441	16105	3388	9
hamlet.txt	18945840	3867	25312	6402	16437	4554	9
julius_caesar.txt	14094816	3102	44731	5437	13891	2793	9
much_ado.txt	13250183	4658	6783	5894	19631	2907	10
coriolanus.txt	21965026	5222	5145	28218	20585	3883	10
romeo_juliet.txt	16064874	3769	6028	7383	34772	3545	9
timon.txt	11839156	3851	4535	5588	13831	3187	10
winters_tale.txt	16359768	4786	5530	5756	13740	3714	9
pericles.txt	11366084	3340	4090	5486	11625	3144	9
john.txt	13917211	4284	5177	5831	13650	3437	10
2henryvi.txt	15594439	3727	6107	6523	42992	3935	10
richardiii.txt	21512564	4631	5503	6019	79156	3896	9
henryviii.txt	15913056	4445	5278	6162	11195	3515	10
troilus_cressida.txt	16134406	4258	5285	5853	11243	4119	9
midsummer.txt	9666933	4811	5625	5959	11346	2914	9
cymbeline.txt	17531433	4043	5180	5533	11150	4057	10
lear.txt	16782019	5071	5485	6186	11481	4007	9
cleopatra.txt	16959747	4686	5751	6467	34248	3782	9
titus.txt	14760790	3407	4918	6023	25061	3306	9
two_gentlemen.txt	12124928	2593	4287	6635	27877	2638	9
Moyen	23789795	10756	9658	7542	23352	3471	9
Total	880222422	398006	357355	279055	864050	128440	344

Tries Hybrides Benchmark (en nanoseconde)

file	build	insert	search	delete	nbword	deep
lll.txt	66631126	5400	32479	9944	3634	33
comedy_errors.txt	35247477	4362	8450	9906	2435	24
richardii.txt	58213606	12322	24619	19888	3510	26
twelfth_night.txt	39637143	6437	11022	23118	3027	31
merchant.txt	40999826	4720	36604	3085	3157	28
merry_wives.txt	40514232	4609	2418	4143	3185	27
henryv.txt	51744248	5229	2616	6405	4390	30
1henryiv.txt	72914001	3274	2324	3931	3720	27
2henryiv.txt	39277108	24658	3824	5499	3960	32
1henryvi.txt	45192574	23767	2051	4798	3722	25
3henryvi.txt	46620362	9555	4040	10040	3445	29
measure.txt	38081410	4580	2601	3625	3226	24
othello.txt	49834188	7369	3361	6280	3662	26
taming_shrew.txt	60186504	6991	2155	6883	3146	25
tempest.txt	50169038	8435	8249	6609	3081	27
macbeth.txt	56239471	5156	3291	4477	3201	26
asyoulikeit.txt	47967582	4779	2463	5888	3167	28
allswell.txt	50410989	7507	2326	5551	3385	27
hamlet.txt	66066594	6046	2862	5910	4551	25
julius_caesar.txt	24679242	21777	2212	3582	2790	26
much_ado.txt	18830476	17983	2026	1869	2904	26
coriolanus.txt	21227378	4439	3173	6927	3880	25
romeo_juliet.txt	22072252	3407	2199	4169	3542	26
timon.txt	14088450	4331	3013	4623	3184	26
winters_tale.txt	20839806	3880	2483	5687	3711	28
pericles.txt	17679759	6127	3612	4068	3141	24
john.txt	38864785	7667	3433	4169	3434	27
2henryvi.txt	78783272	6920	8614	6489	3932	32
richardiii.txt	49770234	4147	2623	4113	3893	25
henryviii.txt	27955676	4286	3619	5851	3512	26
troilus_cressida.txt	32187681	4799	2707	5803	4116	27
midsummer.txt	19945487	6235	3323	7147	2911	25
cymbeline.txt	24953860	10306	6786	12717	4054	26
lear.txt	21814093	3153	2240	3296	4004	26
cleopatra.txt	22975703	2505	1963	4496	3779	28
titus.txt	19535893	1598	2284	2449	3303	26
two_gentlemen.txt	12821391	1540	2213	2580	2635	25
Moyen	39053322	7305	5845	6379	3468	26
Total	1444972917	270309	216290	236028	128329	994

4 Conclusion :

Sur quelque instance le Patricia-Tries est meilleur que le Hybride-Tries, néanmoins, en moyen l'Hybride est moins performent en construction mais ensuite il est plus rapide que le Patricia-Tries pour les fonctions d'insertion, d'ajout, de suppression et de recherche