Rapport TPZ — Optimisation de la recherche dans un arbre binaire

Maram_Daas et Hibatallah_Khadir groupe : 13

Remarque: Les nœuds des BST incluent un pointeur vers le parent

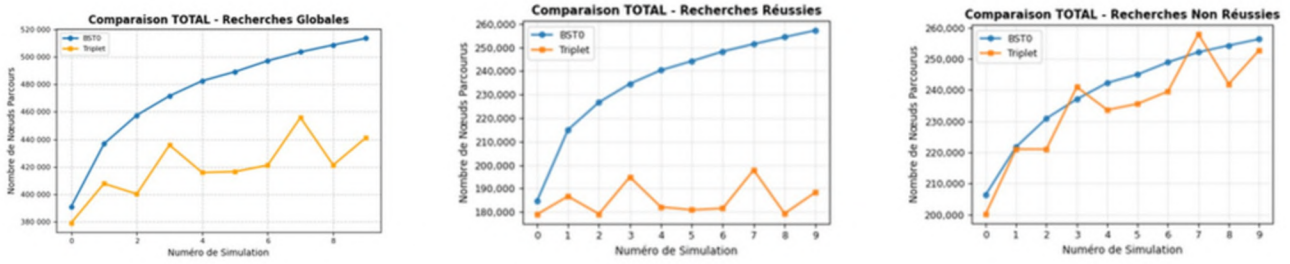
Modification dans l'implémentation

- Les modules de recherche ont été modifiés pour retourner le **nombre de nœuds parcourus**.
- Le module genererFichier accepte prends en parametres le nombre des mots et le nom du fichier
- On a ajouté RangeQuery_param et RangeQuery_optimal_param qui prennent les mots bornes comme paramètres, pour permettre l'automatisation dans les simulations.
- On a séparé les tests des modules et la simulation, et dans la simulation du range query, on a mis un mode complet (10 fois, long ~15min) et un mode rapide (1 fois) pour tester sans attendre.

Résultats de Performance – Recherche d'un seul mot

	BST0				Triplet			
Type	**EQXYZ**	**GTXYZ**	**LTXYZ**	**TOTAL**	**EQXYZ**	**GTXYZ**	**LTXYZ**	**TOTAL**
Successful	62405	17564	104713	184682	53115	21597	104304	179016
Unsuccessful	71148	19247	115939	206334	61542	23186	115428	200156
All	133553	36811	220652	391016	114657	44783	219732	379172

Analyse Graphique : Resultats pour X='q' , Y='r' , Z='s'



(a) Tous (b) Recherches réussies (c) Recherches non réussies

Figure 1 – Comparaison BST0 vs Triplet : Nombre total de nœuds parcourus

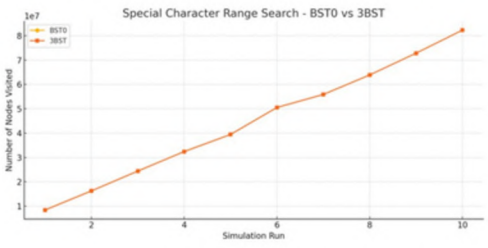
1 Observation

La performance globale du Triplet est meilleure que celle du BST0. Cela est vrai pour les mots inférieurs ou égaux à X, Y, Z (BST1 et BST3). En revanche, pour les cas supérieurs à X, Y, Z (BST2), la recherche avec BST0 s'avère plus optimale.

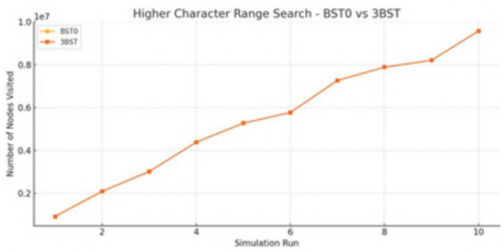
Résultats de Performance – Rangequery
(resultas du test complet dans le fichier results_rangequery)

	BST0				3BST			
Type	**Special**	**higher**	**lower**	**total**	**Special**	**higher**	**lower**	**total**
Successful	446,378,245	54,361,115	1,328,959,613	1,829,698,973	446,307,663	54,365,984	1,328,899,170	1,829,572,817
Unsuccessful	0	73	0	73	0	78	0	78
All	446,378,245	54,361,188	1,328,959,613	1,829,699,046	446,307,663	54,366,062	1,328,899,170	1,829,572,895

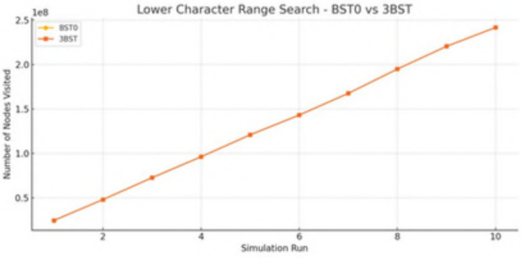
(a) Special = caractères choisis



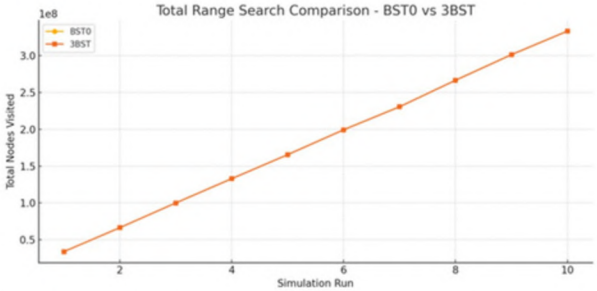
(b) Higher = caractères supérieurs



(c) Lower = caractères inférieurs



(d) total : somme des trois categories



Observation 2:

BST0 et le triplet offrent des performances très proches, avec un écart total de +126 151 nœuds en faveur du triplet principalement sur les caractères special (+70 582) et lower (+60 443). On note toutefois un léger avantage pour BST0 en higher (–4 874). Sur 1,83 milliard de nœuds, cet écart reste marginal, signalant une optimisation limitée dans les range queries aléatoires (≈ 0,01 %).

Figure2– Comparaison BST0 vs Triplet : Nombre total de nœuds visitees