Projet 2

Mémorisation bigtable vs hashtable pour le problème du traversier

CSI 2510 - Structure de données et algorithmes

Automne 2020

Université d'Ottawa

Professeur: Robert Laganière

Mark-Olivier Poulin 300058025

Date de soumission: December 4th 2020

Partie 3:

a) La conception de l'algorithme avec une matrix de grandeur N+1 * L+1 (BigTable) où N représente le nombre de voitures et L la longueur du traversier, passe tous les tests du juge en ligne avec un temps d'exécution de 0.340 secondes. Voici la capture d'écran du juge en ligne:

#		Problem	Verdict	Language	Run Time	Submission Date
25798510	10261	Ferry Loading	Accepted	JAVA	0.340	2020-12-02 19:46:53
25798492	10261	Ferry Loading	Wrong answer	JAVA	0.370	2020-12-02 19:42:59
25798466	10261	Ferry Loading	Wrong answer	JAVA	0.380	2020-12-02 19:37:50
25798460	10261	Ferry Loading	Wrong answer	JAVA	0.340	2020-12-02 19:36:50
25795332	10261	Ferry Loading	Wrong answer	JAVA	0.420	2020-12-02 06:09:36
25795318	10261	Ferry Loading	Wrong answer	JAVA	0.540	2020-12-02 06:02:40
25795317	10261	Ferry Loading	Wrong answer	JAVA	0.430	2020-12-02 06:02:08
25795308	10261	Ferry Loading	Wrong answer	JAVA	0.360	2020-12-02 05:55:50
25795297	10261	Ferry Loading	Wrong answer	JAVA	0.540	2020-12-02 05:48:40
25795294	10261	Ferry Loading	Wrong answer	JAVA	0.530	2020-12-02 05:46:42
25795288	10261	Ferry Loading	Wrong answer	JAVA	0.350	2020-12-02 05:45:25
25795284	10261	Ferry Loading	Wrong answer	JAVA	0.540	2020-12-02 05:44:10
25795259	10261	Ferry Loading	Wrong answer	JAVA	0.430	2020-12-02 05:39:39
25795256	10261	Ferry Loading	Runtime error	JAVA	0.000	2020-12-02 05:39:13
25795254	11304	Difussing Nuclear Bombs on Planet X	Runtime error	JAVA	0.000	2020-12-02 05:38:32
25790646	10261	Ferry Loading	Wrong answer	JAVA	0.350	2020-12-01 05:29:44
25790566	10261	Ferry Loading	Wrong answer	JAVA	0.560	2020-12-01 05:14:58
25790553	10261	Ferry Loading	Wrong answer	JAVA	0.340	2020-12-01 05:10:58
25790552	10261	Ferry Loading	Wrong answer	JAVA	0.340	2020-12-01 05:10:07
25790548	10261	Ferry Loading	Wrong answer	JAVA	0.320	2020-12-01 05:07:33
25784807	10261	Ferry Loading	Wrong answer	JAVA	0.120	2020-11-30 03:54:46

Bigtable

<< Start < Prev 1 Next > End >>

b) La conception de l'algorithme avec une hashmap de grandeur n où n représente le nombre de paires (k (nombre de voitures ajouté au traversier) + s (espace restant), boolean), passe tous les tests du juge en ligne avec un temps d'exécution moyen de 0.180 secondes ((0.170 + 0.190) / 2). Le facteur de réduction de la mémoire avec le fichier « input1.txt » est de 2105,26 ((8 * 5000) / 19). Voici la capture d'écran du juge en ligne:

#		Problem	Verdict	Language	Run Time	Submission Date
25810961	10261	Ferry Loading	Accepted	JAVA	0.170	2020-12-05 01:20:58
25810809	10261	Ferry Loading Hashmap (Java implementation)	Accepted	JAVA	0.190	2020-12-05 00:58:38
25806135	10261	Ferry Loading Custom Hashmap (my implementation)	Time limit exceeded	JAVA	3.000	2020-12-04 06:59:12
25805162	10261	Ferry Loading Hashmap (Java implementation)	Time limit exceeded	JAVA	3.000	2020-12-04 03:33:18
25798510	10261	Ferry Loading	Accepted	JAVA	0.340	2020-12-02 19:46:53
25798492	10261	Ferry Loading	Wrong answer	JAVA	0.370	2020-12-02 19:42:59
25798466	10261	Ferry Loading Bigtable	Wrong answer	JAVA	0.380	2020-12-02 19:37:50
25798460	10261	Ferry Loading	Wrong answer	JAVA	0.340	2020-12-02 19:36:50
25795332	10261	Ferry Loading	Wrong answer	JAVA	0.420	2020-12-02 06:09:36
25795318	10261	Ferry Loading	Wrong answer	JAVA	0.540	2020-12-02 06:02:40
25795317	10261	Ferry Loading	Wrong answer	JAVA	0.430	2020-12-02 06:02:08
25795308	10261	Ferry Loading	Wrong answer	JAVA	0.360	2020-12-02 05:55:50
25795297	10261	Ferry Loading	Wrong answer	JAVA	0.540	2020-12-02 05:48:40
25795294	10261	Ferry Loading	Wrong answer	JAVA	0.530	2020-12-02 05:46:42
25795288	10261	Ferry Loading	Wrong answer	JAVA	0.350	2020-12-02 05:45:25
25795284	10261	Ferry Loading	Wrong answer	JAVA	0.540	2020-12-02 05:44:10
25795259	10261	Ferry Loading	Wrong answer	JAVA	0.430	2020-12-02 05:39:39
25795256	10261	Ferry Loading	Runtime error	JAVA	0.000	2020-12-02 05:39:13
25795254	11304	Difussing Nuclear Bombs on Planet X	Runtime error	JAVA	0.000	2020-12-02 05:38:32
25790646	10261	Ferry Loading	Wrong answer	JAVA	0.350	2020-12-01 05:29:44
25790566	10261	Ferry Loading	Wrong answer	JAVA	0.560	2020-12-01 05:14:58
25790553	10261	Ferry Loading	Wrong answer	JAVA	0.340	2020-12-01 05:10:58
25790552	10261	Ferry Loading	Wrong answer	JAVA	0.340	2020-12-01 05:10:07
25790548	10261	Ferry Loading	Wrong answer	JAVA	0.320	2020-12-01 05:07:33
25784807	10261	Ferry Loading	Wrong answer	JAVA	0.120	2020-11-30 03:54:46

<< Start < Prev 1 Next > End >>

c) La table de hachage est composée de « Integer » comme clé et de « boolean » comme valeur. La fonction de hachage est celle de l'implémentation par défaut de la classe HashMap de Java (voir: https://www.mindprod.com/jgloss/hashcode.html). Les clés sont créées de la manière suivante: k + s où k est le nombre de voitures ajouté au traversier et s, l'espace restant. En créant des clés k + s, on obtient des clés uniques et on n'a pas à emmagasiner les valeur de s associé à k dans une liste chainé. La dimension de la table de hachage est donc de n où n est le nombre de paires(k + s, boolean).

La première tentative (id: 25805162) utilisait l'implémentation par défaut HashMap de Java. La manière donc les paires était emmagasiné était que les clé représentait les k et les valeurs était les différents s. Le problème avec cette technique est que le temps de recherche d'une paire était trop lent (O(n) dans le pire cas) donc ne passe pas les tests du juge en ligne.

La deuxième tentative (id: 25806135) utilisait une classe personalisé de HashMap (voir la fin de Main.java pour l'implémentation). Comme la tentative précédente, les clés représentait les différents k et les valeurs, les différents s. Cette implémentation consistait d'une liste contenant des listes chaînées simples. Comme la tentative précédente, la recherche d'une paire était trop lente (O(n) dans le pire cas) donc ne passe pas les tests du juge en ligne.

La troisième et dernière tentative (ids: 25810809 & 25810961) utilisait encore une fois l'implémentation par défaut HashMap de Java. Cette fois-ci, il fallait réduire le temps de recherche d'une paire. Pour résoudre ce problème, il suffit de combiner k et s ensemble afin de créer une clé et d'assigner « true » comme valeur à cette clé. En faisant une recherche de la paire, on a qu'à combiné k et s. Le temps d'exécution de cette recherche est donc de O(1) donc les tests du juge en ligne passe. On aurait aussi pu faire s - k ou k * s, mais dans ces cas-ci on aurait des clés avec des valeurs négatives ou bien des clés avec des valeurs trop grandes ce qui n'est pas nécessaire.

Note: La classe Main.java de HashTable contient une implémentation commentée de la solution qui permet de débugger avec VS Code et l'extension de Java.