Projet LSINF1121 - Algorithmique et structures de données

-

Bilan Mission 5

Groupe 26

Laurian Detiffe (6380-12-00)

Sundeep Dhillon (6401-11-00)

Alexis Macq (5910-12-00)

Xavier Pérignon (8025-11-00)

Thibaut Piquard (4634-13-00)

Thomas Wyckmans (3601-12-00)

Année académique 2015-2016

Questions et réponses

1. Dans la technique de compression par un codage de Huffman, il s'avère utile d'inclure dans le fichier comprimé une entête contenant l'information nécessaire au décodage de ce fichier. De quelle information s'agit-il ? Proposez au moins trois alternatives en précisant la nature de l'information et son mode de représentation dans le fichier comprimé. Discutez des avantages et inconvénients de ces alternatives.
   On a besoin de l'arbre pour décompresser. Du coup, on encode notre arbre (ou table de fréquence) dans un fichier en sachant quel caractère correspond à quel symbole et cela, ça dépend de la structure dans notre arbre. On doit juste connaître la structure et quel symbole est dans quel leaf.
2. Peut-on gagner encore en taux de compression si l'on réapplique l'algorithme de compression de Huffman sur un fichier déjà comprimé une première fois ? Que se passe-t-il dans ce cas ? Cela ouvre-t-il la porte vers un algorithme de compression récursif et optimal ?
   Non, on ne peut pas gagner en taux de compression en réappliquant l'algorithme de compression de Huffman sur un fichier déjà comprimé une première fois. Dans ce cas, cela n'ouvre pas la porte vers un algorithme de compression récursif et terminal ! Cela est dû au fait que l'on va retomber sur exactement la même table de Huffman en exécutant l'algorithme. (A CHANGER)
   Si on prend un texte avec "abababababab", on peut représenter un a par 0 et b par 1. Ensuite, quand on compresse avec Huffman, on prend les 8 premiers bits "01010101" et on peut les compresser. On peut compresser tant qu'il y a plusieurs occurrences d'un même caractère et théoriquement, à force, on arrivera à un point où on ne pourra plus compresser.

3. Quel est, approximativement, le taux de compression obtenu si l'on applique l'algorithme de compression de Huffman sur un un fichier comportant une seule chaîne composée du caractère `a` répété un million ($\approx 2^{20}$) de fois, suivi du caractère `b` présent une seule fois ?
   Vu qu'il n'y a que deux caractères, on peut l'écrire sur 2 bits : a serait 0 et b serait 1.
   Le taux de compression obtenu varie-t-il avec la longueur du fichier (par exemple, si le caractère a est répété deux millions de fois) ?
   La taux de compression serait de 1/8.
   A votre avis, quel est le nombre minimal de bits nécessaires pour représenter sous forme comprimée ce fichier ?
   Du coup, on sait l'écrire sur un million et un.
   Peut-on adapter la technique de compression par un codage de Huffman en mesurant la fréquence d'autre chose que les caractères présents ?

   Peut-on utiliser une autre technique de compression qui serait plus efficace dans

ce cas particulier ?

4. Imaginez une implémentation d'une file de priorité par un tas (heap, en anglais) à l'aide d'une structure chaînée pour représenter l'arbre binaire essentiellement complet correspondant au tas. Combien de liens sont nécessaires dans chaque noeud ? Ecrivez le code des méthodes *insert*, *delMax*. Quelle en est la complexité ? Est-il utile de donner la taille maxN dans le constructeur ?

5. Proposez une structure de données qui supporterait les opérations suivantes en temps logarithmique :
   (a) *insertion* :
   (b) *supprimer le maximum* :
   (c) *supprimer le minimum* :
   (d) les opérations suivantes en temps constant : *trouver le maximum* :
   (e) *trouver le minimum* :
6. Imaginez une structure de données qui supporte :
   (a) L'insertion en temps logarithmique : d
   (b) L'opération *trouver la médiane* en temps constant : Queue
   (c) *supprimer la médiane* en temps logarithmique :