Adequin Renaud
Toussaint Etienne

Compression de Huffman

1. Compression

```
./huff fichier1 fichier2 dossier1
```

Le programme implémente deux façon de créer l'arbre des fréquences avec un tas min ou alors avec un tas de Fibonacci.
Le tas de Fibonacci est la méthode utilisé dans le programme même si les deux tas sont présent dans le fichier **`struct.ml`**
En effet le tas de Fibonacci est une méthode de meilleurs compléxité en temps pour la construction de l'arbre, mais son implémentation étant beaucoup plus impérative que celle du tas min nous avons fait le choix de proposer les deux.

1) Tas min

Structure du TAS

L'implémentation du tas min possède une structure

`'a tree` qui sera un **`int * int`** de `octet * fréquence`

```
type 'a tree =
| Feuille of 'a
| Node of int * 'a tree * 'a tree;;
```

Le Tas est représenté par un array de **`'a tree`** et un entier qui représente le nombre de nœud restant dans le tableau.

Extraction minimum

Le minimum du tas est sa racine (élément d'indice 0), l'extraction du minimum se fait donc en récupérant le premier élément du tableau et de mettre le dernier élément de ce tableau à la place du premier.

Pour finir on appel la fonction entasser minimum pour mettre à jour notre Tas

Entasser minimum

Une fois le minimum extrait cette fonction vérifie récursivement que chaque fils a une valeur plus grande que son père sinon il les échanges et rappel entasser min

2) Tas de Fibonacci

Structure du TAS

Notre implémentation du Tas de Fibonacci possède 3 structures de données :

- Le type **`'a tree`** du tas min
- Un type nœud qui contient donc :
  - '**`a tree`** `key_val`, des **`int`** qui sont `pere, fils, frere_droit, frere_gauche` qui sont les indice dans le tableau des nœud
  - et un **`int`** `degree` qui est le nombre de fils direct du nœud
- Et le **`type heap`** qui est la structure principal du tas qui contient un **`array`** de nœud, un **`Hashtbl`** de racine, accompagné d'un **`array`** contenant les degrés, et enfin un **`int`** que contient l'indice du minimum dans le tableau des nœuds.

Trouver minimum

Avec le tas de Fibonacci la fonction trouver le minimum a un coût constant car le minimum est directement accessible dans la structure du tas

Extraire minimum

Avec l'indice du minimum dans le tableau, l 'extraction de ce minimum se fait en 2 étapes :

- Ensuite les fils du nœud minimum sont ajouter a la racine en supprimant leur relation fraternelle
- Enfin le nœud minimum est retirer de la liste des nœud racines.

Consolider

Une fois le minimum extrait et ses fils remis dans la racine la phase de consolidation consiste a parcourir le tableau des racine si deux nœud on le même degré (Le degré d'un nœud est le nombre de fils « direct » qu'il possède) on link ces deux nœud, c'est a dire que le nœud avec la plus grande valeur deviendra le fils du second nœud et sera retiré de la liste des racines. Ce qui permet de garantir que le

minimum est toujours présent dans les liste des racine. Cette phase permet de réduire au minimum le nombre d'élément de root et donc accélérer la recherche du minimum.

Une fois la phase de consolidation fini le l'indice du minimum est donné a la structure du Tas.

2. Décompression

```
./huff –d fichier1.hf fichier2.hf dossier1.hf
```

Lors de la décompression une fois l'arbre des fréquences récupéré la décompression du fichier se fait bit à bit en parcourant l'arbre et en écrivant directement dans le fichier de sortie aucune autre structure de donnée intermédiaire n'est créée.

En mode décompression il n'est pas neccesaire de passer par une structure de donnée de type impérative tel que array. En effet après avoir récuperer l'arbre nous lisons le fichier bit a bit et lorsque l'on rencontre un feuille nous l'écrivons dans le fichier de sorti.

3. Extensions

1) Liste de fichier

On peux donner une liste de fichiers/dossiers en entrée

```
./huff fichier1 fichier2 dossier1
```

En mode compression ou décompression les fichiers seront traité les uns après les autres. Si un fichier possédant le même nom existe déjà il passera directement au fichier suivant.

Interopérabilité

Chaque fichier de la liste peut être décompressé avec le programme test donné avec l'énoncé.

2) Dossiers

Le programme est capable de compresser un dossier passé en argument.

L’arborescence des fichiers est stocké dans le premier emplacement de Bits ignorés « début de fichier. »

En mode compression :

Le programme utilise la compression de Huffman pour compresser cette arborescence et teste quelle version de arborescence est la moins coûteuse en espace car l'arborescence compressée possède son propre arbre.

Afin de faire la distinction entre les différents fichier on repère dans le tableau des fréquences un caractère non présent qui va être utiliser pour séparer les différent fichiers.

Cependant il peut arriver que les 256 caractères possibles soient utilisé dans l'arbre « ex : **`*.bmp`** », dans ce cas le caractère de fin de fichier **`EOF`** est utilisé et l'interopérabilité est perdu.

En mode décompression :

Le programme vérifie si le fichier possède une arborescence, si oui le programme récupère cette arborescence « la décompresse si besoin ».

Le programme est capable de restituer tout les fichier de l'archive contrairement au programme donnée avec l'énoncé qui lui se contentera de tout décompresser dans un unique fichier

Interopérabilité

Si le caractère de fin de fichier pour séparé les fichiers, lors de la décompression le programme test donné avec l'énoncé ne sera pas distinguer la « vrai » fin d'un fichier. Il ne décompressera que le premier fichier et quittera avec une erreur : « Données supplémentaires à la fin du fichier. »

3) HashCode

En mode compression :

Le programme écrit un hashcode des fichiers qu'il place a la fin du fichier « **`.hf`** » dans les Bits ignorés.

En mode décompression :

Le programme vérifie si le fichier possède un hascode, si oui en fin de décompression il vérifie que les empreinte sont bien identique et affiche une message en conséquence.

Interopérabilité

Si l'on décompresse un fichier qui ne contient pas de hashcode alors le programme saute alors cette étape pour rester interopérable.