Codage de Huffman

abracadabra

```
01100001 01100010 01110010 01100001
01100011 01100001 01100100 01100001
01100010 01110010 01100001
```

ASCII

caractère	décimal	binaire
a	97	01100001
b	98	01100010
c	99	01100011
d	100	01100100
r	114	01110010

= 88 bits

Codage de Huffman

abracadabra

```
000 001 100 000 010 000 011 000 001
100 000
```

caractère	binaire
a	000
b	001
c	010
d	011
r	100

= 33 bits

Codage de Huffman

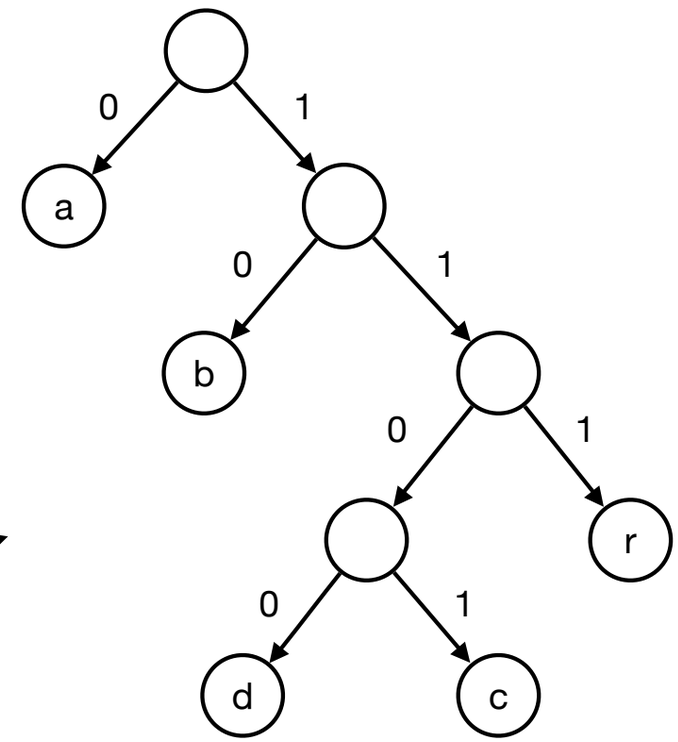
abracadabra

```
0 10 111 0 1101 0 1100 0 10 111 0
```

= 23 bits

caractère	binaire
a	0
b	10
c	1101
d	1100
r	111

arbre de Huffman
adapté à la chaîne
abracadabra



Codage de Huffman

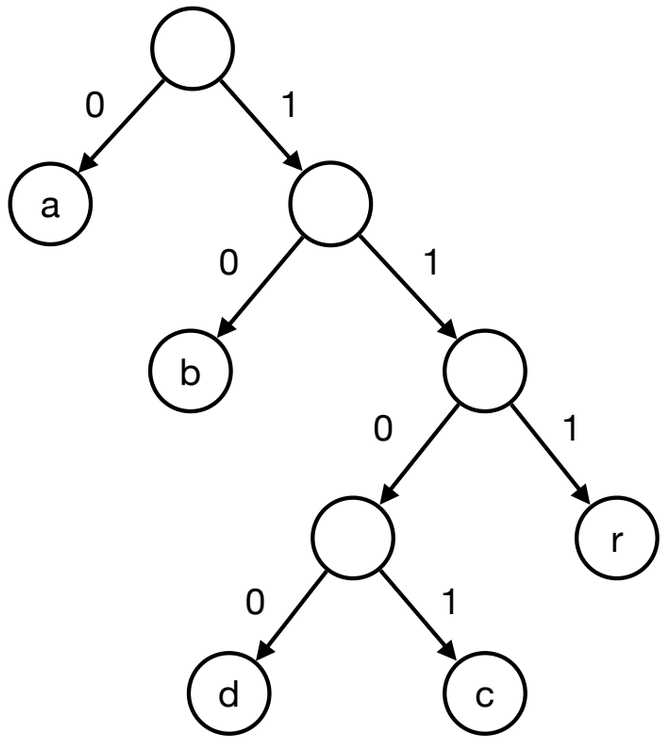
Encodage

Lire le fichier d'entrée et **construire l'arbre de Huffman** correspondant
Écrire l'arbre de Huffman sur le fichier de sortie
Relire le fichier d'entrée, l'**encoder** et écrire le texte codé sur le fichier de sortie

Décodage

Lire l'arbre de Huffman à partir du fichier codé
Lire et **décoder** le texte codé et écrire le texte décodé sur le fichier de sortie

- Chaque texte peut donner lieu à un arbre différent
- Il peut y avoir plusieurs arbres correspondant au même fichier d'entrée



Informations pratiques

- Énoncé sur elearning
  - également des exemples de fichiers de texte
- Projet en binôme à constituer sur elearning **avant le 4 mai**
  - pas le même binôme deux fois pendant les trois années
- Soutenance bêta fin mai (?)
  - au moins la partie décrite dans la section 3 de l'énoncé (Construction de l'arbre de Huffman)
- Rendu final sur elearning
  - par les deux membres du binôme avant la date limite indiquée

Conseils pratiques

- Trouvez votre binôme rapidement
- Lire et relire l'énoncé
- Aucun groupe de 1 personne ou 3 personnes
- Éviter de former des groupes comprenant 2 apprentis en difficulté en C
  - il faut comprendre plein de choses qui ne sont pas liées au C
- Éviter de former des groupes trop déséquilibrés
  - on posera des questions individuelles pendant les soutenances

Conseils

- On étudie des fonctions utiles au td et tp les semaines qui viennent
  - coder / décoder un arbre par une suite d'entiers
  - visualisation d'un arbre
- Utiliser initialement des fichiers codés en ASCII, sans accents
- Écrire initialement les 0 et les 1 comme des caractères (8 bits)
  - évidemment cela ne compresse pas le fichier
  - estimer la compression en divisant la taille du fichier de sortie par 8
- Quand tout marche bien
  - essayer de coder d'autres types de fichiers
  - implanter la lecture et l'écriture en binaire
    (avec un nombre de bits variable)

Rendu

- Vous rendez une archive `login1_login2.zip` où `login1` et `login2` sont les logins des deux membres du binôme en ordre lexicographique avec tous vos fichiers, un Makefile inclus.
- Compte rendu expliquant
  - comment compiler et exécuter vos programmes
  - les bugs connus
  - lesquels parmi les fichiers d'exemples fournis sur elearning peuvent être correctement traités
  - les difficultés rencontrées et solutions apportées
  - les améliorations implantées

Rendu

- Un rendu final **minimal** consiste en un encodeur et un décodeur (potentiellement le même exécutable) qui permettent de
  - coder le fichier `2city11.txt` et
  - ensuite le décoder (de la manière décrite dans l'énoncé)
- Il faut que le décodeur reproduise **exactement le même fichier**. Cela se vérifie avec la commande suivante :

```
diff 2city11.txt votrefichier.txt
```

Rendu

- Tout retard sera sanctionné.
- Un rendu avec une archive `.rar` ou `.7z` ou `.bz2` sera sanctionné.
- Un projet non rendu sera noté 0.
- Un programme qui termine avec une erreur de segmentation sur le fichier `banana01.txt` sera noté 0.
  - testez vos programmes sous plusieurs systèmes d’exploitation
  - utiliser valgrind