Codage par Plage

Samy Jaziri

Sujet de khôlle inspiré du sujet du Concours Centrale Option Informatique 2006 et du cours d'Olivier Spanjaard (LIP6)

Préambule

Tous les algorithmes de ce sujet devront être implémentés en ***C****.*

Pour vous mettre dans les conditions du concours, vous n'avez pas le droit d'accéder aux ressources en ligne.

Vous indiquerez vos réponses sur la fiche réponse qui vous a été fournie en utilisant, en entrée de vos programmes, le numéro u_0 inscrit sur cette fiche. Vous remettrez cette fiche à l'examinateur en fin de séance. Une fiche réponse est mise à disposition à la fin du sujet en tant qu'exemple des réponses attendues pour un $\widetilde{u_0}$ particulier.
En ce qui concerne les questions orales de la khôlle, lorsque la description d'un algorithme est demandée, vous devez présenter son fonctionnement de fa'c'on schématique, courte et précise. Vous ne devez pas expliquer votre code ligne par ligne! Quand la complexité d'un algorithme est demandée en temps ou en mémoire en fonction d'un paramètre n, on demande l'ordre de grandeur en fonction du paramètre, donné en notation de Landau ($\mathcal{O}(n), \mathcal{O}(\log(n)), \dots$). Prenez des notes lorsque vous préparez une question orale pour retrouver plus rapidement les grandes lignes de votre explication lorsque l'examinateur passe vous voir.
Il est recommandé de **tester vos programmes sur des petits exemples** que vous aurez résolus préalablement à la main ou bien à l'aide de la fiche réponse type fournie en annexe.
Il vous est demandé d'aborder les questions dans l'ordre et de noter vos difficultés à répondre à une question avant de passer à la suivante. Vous pourrez alors les aborder avec l'examinateur.
Attention, une attention particulière sera portée aux problèmes de fuite de mémoire lorsque vous utiliserez l'allocation dynamique
Les deux parties sont indépendantes.

1 Introduction

Motivation

Une technique de réduction de la taille d'une donnée consiste à comprimer les plages de valeurs identiques en signalant le nombre de fois qu'une valeur devrait être répétée. Les algorithmes de compression de données basés sur ce principe dont nommées *codage par plage*. Ce sont des algorithmes sans pertes de données : après décompression, la donnée originelle est restituée à l'identique. Dans ce sujet, nous nous intéresserons à deux codages par plages (qui ne sont pas utilisés en pratique) sur des images en noir et blanc.

Préliminaires

Une image sera représentée dans ce sujet par une chaîne de caractère. Les chaînes de caractère en C sont des zones contiguës de la mémoire où chaque octet représente un caractère dans le codage ASCII. La chaîne se termine par un caractère `'\0'` (code `0`).
Dans ce sujet le codage ne se fera pas, comme c'est habituellement le cas, vers une autre chaîne, mais vers une structure de donnée différente. Nous nous intéresserons alors à la mémoire occupée par la chaîne initiale et par la structure qui contient la version encodée de la chaîne.
Rappelons que la fonction `sizeof` en C permet de connaître la taille en mémoire d'un type de donnée en nombre d'octets, *e.g.* `sizeof(int)` renvoie `4`.
Pour raccourcir l'écriture du type des entiers non signés on définira dans ce sujet le type `uint` comme suit :

```c
typedef unsigned int uint;
```

2 Compression des séquences identiques

Structure de Pile

Pour le premier codage, nous allons compresser l'espace occupé par une séquence de caractères identiques. Pour cela nous utiliserons une pile de couples (c, n) où c est un caractère et n est un entier non signé qui représente le nombre de répétitions de c. **Attention** $(c, 0)$ signifie que c apparaît **1 fois**, $(c, 3)$ signifie que c est répété 3 fois, donc que c apparaît **4 fois**.

Question 1

En utilisant une liste simplement chaînée, implémenter le type `pile` représentant une pile de couples d'un caractère et un entier non signé. La structure de donnée devra supporter les opérations suivantes :

- Le nombre de couples de la pile doit être récupérable en $\mathcal{O}(1)$
- La taille de la pile, *i.e* le nombre de caractères stockés dans la pile, avec leur répétition, doit être récupérable en $\mathcal{O}(1)$.
- `pile* creer_pile_vide()` renvoie un pointeur vers une pile vide allouée dynamiquement.
- `void detruire_pile(pile*)` libère la mémoire utilisée par une pile.
- `void empile(pile*, char)` empile un caractère c sur le sommet de la pile :
  - Si le sommet est un couple (c', n) avec $c' = c$, alors incrémenter n.
  - Sinon, empiler le couple $(c, 0)$ sur le sommet de la pile.
- `char depiler(pile*)` dépile un caractère du sommet d'une pile considérée non vide :
  - Si le sommet est un couple (c, n) avec $n > 0$, alors décrémenter n et renvoyer c.
  - Sinon, supprimer le couple $(c, 0)$ du sommet de la pile et renvoyer c.

Codage

Nous considérons le codage suivant : chaque séquence de caractère c de la chaîne est représenté par un maillon (c, n) où n est le nombre de répétitions de c. Par exemple la chaîne :

"00000011100000000111110000"

sera représentée par la pile $('0', 5) \cdot ('1', 2) \cdot ('0', 6) \cdot ('1', 4) \cdot ('0', 3)$.

Question 2

Implémenter une fonction `pile* codage_chaine(char*);` qui prend en argument une chaîne de caractère et renvoie un pointeur vers une pile correspondant un codage de la chaîne.

Question 3

Implémenter une fonction `char* decodage_chaine(pile*);` qui prend en argument un pointeur vers une pile correspondant au codage d'une chaîne de caractère et renvoie la chaîne originelle.

Préparer une réponse à donner à l'oral

- Quelle est la complexité temporelle au cas pire de la fonction de codage en fonction de la longueur de la chaîne ?
- Quelle est la complexité temporelle au cas pire de la fonction de décodage en fonction

de la longueur de la chaîne encodée ?

- Est-ce que la mémoire occupée par la pile représentant le code d'une chaîne est toujours plus faible que la mémoire occupée par la chaîne elle-même ?
- Donner une liste informelle de conditions que la chaîne en entrée doit vérifier pour que le codage occupe moins de mémoire que la chaîne.

Listes aléatoires

Considérons $(u_k)_{k\geq 0}$ la suite d'entiers définie par :

$$u_k = \begin{cases} u_0 \text{ (sur votre fiche réponse)} & \text{si } k = 0 \\ 15091 \times u_{k-1} \mod 64007 & \text{si } k > 0 \end{cases}$$

On défini la suite $(c_k)_{k\geq 0}$ de caractère par :

$$c_k = \begin{cases} \text{'}N\text{'} & \text{si } u_k \equiv 0[144] \\ \text{'}B\text{'} & \text{sinon} \end{cases}$$

Pour tout $n \in \mathbb{N}$, on définit la chaîne de caractère :

$$S_n = c_0 \dots c_n$$

Question 4

Pour chaque chaîne donner le nombre de couples de son codage
a) S_{1000} **b)** S_{12000} **c)** S_{100000}

3 Codage par nombre optimal de carrés blancs

On propose ici un autre codage. Une image en noir et blanc est représentée par un tableau à deux dimensions de caractères. Chaque caractère représente un pixel, 'N' pour un pixel noir, 'B' pour un pixel blanc. Dans une image de largeur l et de hauteur h, le pixel de coordonnée $(0, 0)$ est le pixel dans le coin supérieur gauche de l'image et le coin inférieur droit a pour coordonnées (l, h). On dit que la dimension de l'image est $l \times h$ dans le cas où l et h sont respectivement la largeur et la hauteur de l'image.
Pour notre codage, nous allons chercher à calculer un ensemble de carrés blancs qui couvrent tous les pixels blancs de l'image. Un algorithme naïf consisterait à essayer toutes les combinaisons de carrés blancs pour en trouver une couvrante et utilisant le nombre minimal de carrés blancs. La complexité d'un tel algorithme sur une image $l \times h$ serait en $\mathcal{O}(l\,h\,\max(l, h)^{l\,h})$.
On cherche un algorithme plus efficace en utilisant la programmation dynamique et une heuristique gloutonne.

Type Image

Le type `image` est défini comme suit :

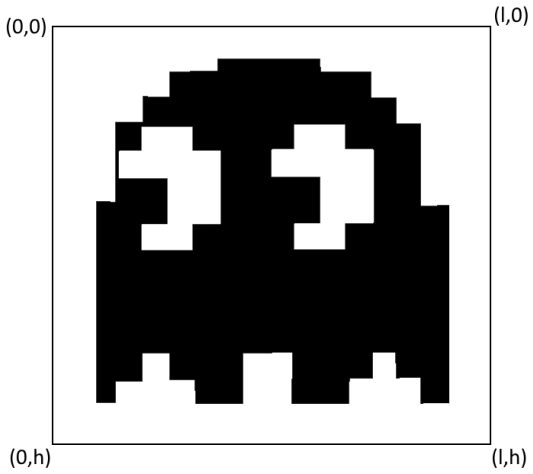

Figure 1: Image

```c
typedef struct image {
    char** pixels;
    uint largeur;
    uint hauteur;
} image;
```

Question 5

Implémenter une fonction `image* creer_image_noire(uint, uint)` qui prend en argument une largeur l et une hauteur h et renvoie un pointeur vers une image de largeur l et hauteur h dont tous les pixels sont noirs.

Question 6

Implémenter une fonction `void detruire_image(image*)` qui libère la mémoire occupée par une image.

Plus grand carré blanc

Pour les techniques utilisées dans le codage, nous nous inspirerons d'une solution par programmation dynamique du problème de recherche du plus grand carré blanc de l'image.
On s'intéresse au sous-problème suivant : *pour chaque pixel* (x, y) *quel est le plus grand carré blanc dont le coin inférieur droit est* (x, y) $(\star)$. Si le pixel est noir le plus grand carré blanc aura une taille 0.

> *Préparer une réponse à donner à l'oral*
>
> Proposer une solution au sous-problème $(\star)$ associé au pixel (x, y) en fonction des résultats des sous-problèmes $(\star)$ associés aux pixels $(x-1, y)$, $(x, y-1)$ et $(x-1, y-1)$ lorsqu'ils existent.

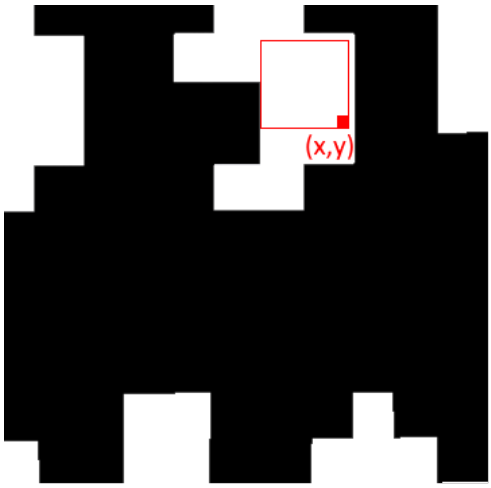

Figure 2: Sous-problème (x, y)

Question 7

Implémenter un algorithme utilisant la programmation dynamique pour trouver le plus grand carré d'une image. Si l'image a pour dimension $l \times h$, vérifier que votre algorithme a une complexité en $\mathcal{O}(lh)$.

Préparer une réponse à donner à l'oral

Quelle est la complexité spatiale de votre algorithme ?

Image noir et blanc aléatoires

On définit l'image I_n de dimension $n \times n$ telle que le pixel (x, y) de l'image soit égal à $c_{x \cdot n+y}$ où c_i est le $i^{\text{ème}}$ caractère de S_{n^2}.

Question 8

Quel est le plus grand carré blanc des images suivantes ?
a) I_{212} **b)** I_{512} **c)** I_{1000}

Codage

Le codage d'un carré blanc consiste en trois entiers x, y et c tels que le carré soit de coté c et le coin inférieur droit du carré est le pixel (x, y). Le codage d'une image de dimension $l \times h$ sera une séquence de triplés d'entiers :

$$t_0 \cdot t_1 \cdot \ldots t_n$$

tels que $t_1, \ldots, t_n$ soit le codage de n carrés blancs couvrant l'ensemble des pixels blancs de l'image et $t_0 = n \cdot h$.

Préparer une réponse à donner à l'oral

En vous inspirant de la solution par programmation dynamique du problème du plus grand carré blanc, proposer, étant donné un pixel (x, y), une heuristique gloutonne pour éliminer, après le calcul du plus sous-problème $(\star)$ associé au pixel (x, y), un maximum de carrés blanc du codage sans perdre la couverture.

Question 9

Implémenter un algorithme de codage d'une image utilisant la programmation dynamique et l'heuristique gloutonne proposée au-dessus.

Préparer une réponse à donner à l'oral

Quelle est la complexité en temps et en espace de votre algorithme en fonction de la largeur et de la hauteur de l'image encodée.

Question 10

Implémenter un algorithme de décodage du code d'une image.

Le taux de compression du codage est lié au rapport entre la mémoire occupée par le codage de l'image et la mémoire occupée par l'image initialement. Si a est la mémoire occupée par la chaîne de caractère S_n et b est la mémoire occupée par la séquence d'entiers non signés constituant le code de I_n, le taux de compression est défini par $1 - \frac{a}{b}$.

Question 11

Calculer le taux de compression du codage des images suivantes :
a) I_{212} **b)** I_{512} **c)** I_{1000}