

AGP - examen 2012-2013

Durée : 2 heure . tous documents autorisés

Remarques :

- Le barème est donné à titre indicatif (il pourra être modifié éventuellement).
- Pour évaluer les programmes C, on prendra en compte (dans l'ordre décroissant d'importance) : la validité de l'algorithme, la validité syntaxique du programme, la clarté du programme et les commentaires, la forme générale de la présentation.

1 Arbre binaire (4pts)

On rappelle que dans un arbre binaire, chaque nœud a *au plus* deux fils. On utilise la structure de donnée suivante pour stocker un arbre binaire :

```
struct model_noeud
{
    int val;
    struct model_noeud *filsGauche;
    struct model_noeud *filsDroit;
} ;

typedef struct model_noeud NOEUD;

typedef NOEUD *ARBRE;
```

Écrire une fonction C récursive : `int nbFeuille(ARBRE racine)` qui calcule le nombre de feuille de l'arbre dont la racine est `racine`

2 Graphes (6pts)

2.1 Algorithme de Dijkstra

Considérons le graphe de la figure 1. On rappelle l'algorithme de Dijkstra pour calculer les plus courts chemins entre un sommet S et tous les autres sommets d'un graphe G , avec $G = (V, E, P)$ P étant le poids des arêtes et $\Gamma^-(x)$ représentant les prédécesseurs du sommet x :

1. Chaque nœud x est initialisé à une distance infinie de S : $\forall x \in V, x \neq s \text{ } dist_0(x) = \infty$, et $dist_0(S) = 0$.
2. À chaque étape i :

- on réévalue $dist$ pour les sommets non marqués :

$$dist_i(x) = \min(dist_{i-1}(x), \min_{y \in \Gamma^-(x)}(dist_{i-1}(y) + P(xy)))$$

- On marque les sommets x de distance $dist(x)$ minimum
3. On s'arrête au bout de $|V| - 1$ étapes ou quand tous les sommets sont marqués.

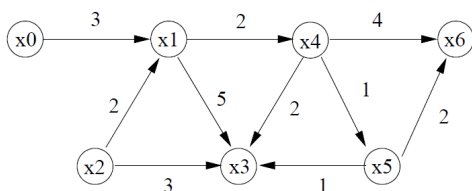


FIGURE 1 – Graphe pour la question 2.1 (Dijkstra)

1. Appliquer l'algorithme de Dijkstra partir du sommet x_0 , recopiez sur votre feuille et complétez le tableau ci-dessous en donnant les valeurs des distances pour chaque sommet à chaque étape.

Sommet	x0	x1	x2	x3	x4	x5	x6
dist à l'étape 1	0	∞	∞	∞	∞	∞	∞
dist à l'étape 2	0						
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots

2.2 Modélisation

Soit la matrice booléenne suivante M :

$$M = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

1. Si l'on considère cette matrice comme la matrice d'adjacence d'un graphe G , dessinez le graphe G
2. Quel est l'information donnée par M^2 , M^3 et M^{17} sachant que les opérations $+$ et \times utilisées dans les multiplications de matrices sont le 'ou' logique et le 'et' logique (il n'est pas nécessaire de calculer ces matrices pour répondre à cette question).
3. Donner la matrice d'adjacence de la fermeture transitive de G , comment peut-on la calculer à partir de M ?

3 Arithmétique Sumérienne (10pts)

La civilisation sumérienne s'est épanouie dans le sud de la mésopotamie vers 3500 à 2000 avant J.-C. Cette civilisation avait élaboré un système de numération à la fois décimal (base 10) et sexagésimal (base 60). Il nous reste de ce système un outil que l'on utilise tous les jours, je vous laisse trouver lequel...







1	10	60	600	3600	36000
					

FIGURE 2 – Symboles utilisés par les sumériens

Le système de numération contenait des signes spécifiques pour 1, 10, 60, 600 et 3600 représentés sur la figure 2 (pour l'exercice, on se limitera à 1, 10 et 60). Plutôt que d'utiliser les symboles sumériens de la figure 2, on utilisera les symboles 1, *X* et *Y*. '1' étant le symbole pour le nombre 1, '*X*' étant le symbole pour le nombre 10 et '*Y*' étant le symbole pour le nombre 60. Le système de numération est additif, c'est à dire que pour représenter un nombre, on répète autant de fois qu'il faut les symboles qu'il faut, c'est à dire :

- le nombre 1 s'écrit : 1
- le nombre 3 s'écrit : 1 1 1
- le nombre 13 s'écrit : *X* 1 1 1
- le nombre 23 s'écrit : *X X* 1 1 1
- le nombre 63 s'écrit : *Y* 1 1 1
- le nombre 163 s'écrit : *Y Y X X X X* 1 1 1

3.1 Structure de donnée (5pts)

On va utiliser la structure de donnée suivante pour stocker un entier en notation sumérienne :

- une liste chaînée de structure **CHIFFRE**. Chaque structure **CHIFFRE** contenant le caractère le représentant symboliquement ('1', '*X*' ou '*Y*'), la valeur du chiffre en décimal et un pointeur vers le chiffre suivant dans le nombre.
- Une structure **NOMBRE** pour un nombre. Cette structure **NOMBRE** est composée d'un champ indiquant le nombre de chiffre du nombre, et un pointeur vers le premier chiffre. Les chiffres de poids fort sont stocké en tête de liste (ou de manière équivalente, les unités en fin de liste).

La figure 3 représente le type **C** que vous utiliserez pour vos fonctions (à gauche) et la représentation (à droite) en mémoire du nombre 163 (nombre à 9 chiffres en sumérien).

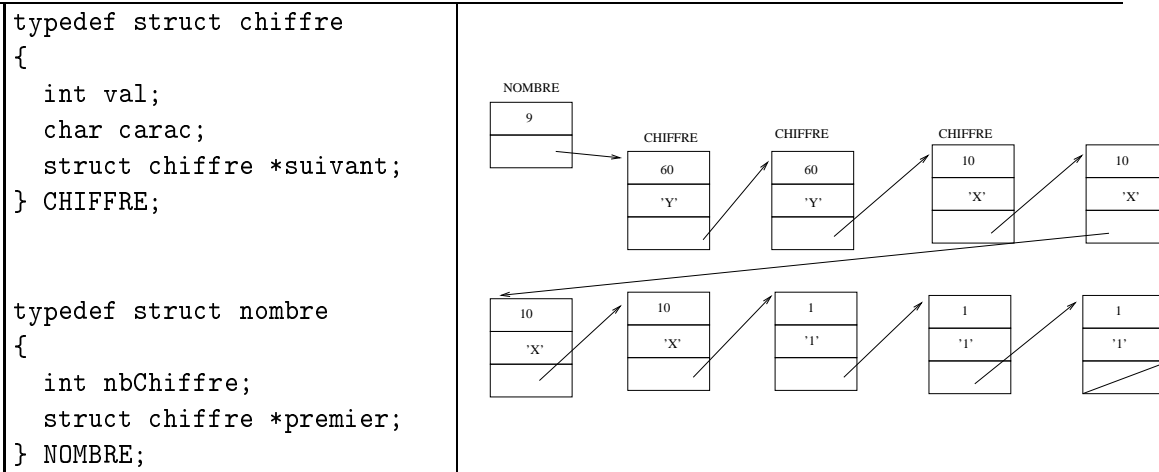


FIGURE 3 – Structure de donnée utilisée pour représenter les nombres sumériens en C et un exemple de représentation en mémoire du nombre à neuf chiffres : 163 soit Y Y X X X X 1 1 1

- Écrivez en C une fonction `afficherNombre` de prototype :
`void afficherNombre(NOMBRE);`
 qui affiche à l'écran un `NOMBRE` (par exemple le nombre 163, aussi représenté dans la liste chaînée de la figure 3 s'affichera comme ceci : Y Y X X X X 1 1 1)
- Expliquez informellement un algorithme de conversion d'un entier en notation sumérienne
- Écrivez un programme C qui implémente cet algorithme dans une fonction `initNombre` de prototype suivant :
`NOMBRE *initNombre(int val);`
 Par exemple, l'appel `initNombre(163)` produira la structure de la figure 3. Pour produire cette fonction on sera amené à produire des sous-fonctions (initialisation d'un chiffre par exemple).
- Expliquez les avantages et les inconvénients du typage proposé, est ce que le stockage des symboles '1', 'X' et 'Y' est nécessaire? proposez une autre structure de donnée et donnez en les avantages par rapport à celle proposée.

3.2 Arithmétique sumérienne (5pts)

- On garde le système de typage proposé dans la figure 3. Proposer un algorithme d'addition pour sumérien, c'est à dire un algorithme qui additionne deux nombres en notation sumérienne sans passer par une traduction en décimal.
- Ecrivez un programme en C qui réalise cette addition
- Proposez des idées pour un algorithme de multiplication.

3.3 Le plus vieux bug du monde (0pts)

(section sans question... qui ne rapporte pas de point non plus)

On sait que les sumériens utilisaient des algorithmes pour compter, on sait aussi que certains de ces algorithmes étaient faux. La tablette représentée en figure 4 représente le partage d'un grenier de grain contenant 1 152 000 "silà" (unité pour mesurer la quantité de grain) entre 7 hommes. le résultat indique 164 160 silà par homme, alors que le résultat exact est 164 571 par homme avec un reste de 3 silà. Paradoxalement, ce *bug* vieux de 4 500 ans permet aux chercheurs de retrouver les algorithmes utilisés par les sumériens pour la division [Cha].

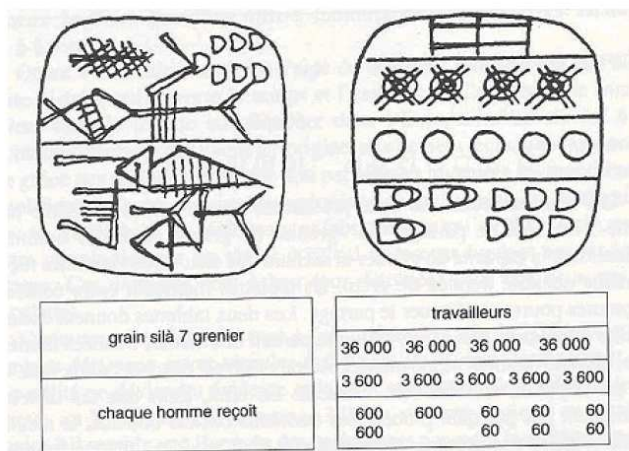


FIGURE 4 – La tablette 651, des tablettes sumériennes de Shuruppak publiées en 1937. Les détails algorithmique viennent du livre “Histoire d’algorithmes : du caillou à puce” [Cha]

Références

[Cha] Jean-Luc Chabert. *Histoire d’algorithmes : du caillou à la puce*. Regards sur la science. Belin 1994 (21-Dijon-Quétigny, Paris. Autre tirage : 1995.