Session 2007

Filière : 2ème concours

ENS de Lyon

Epreuve d'informatique

Durée : 3 heures

Ce livret comprend 7 pages numérotées de 1 à 7

L'usage de documents et de calculatrice est interdit

Évaluation de polynômes

Il est recommandé de lire tout le sujet avant de commencer la rédaction.
Pour les questions qui demandent l'écriture d'un pseudo-programme : il s'agit d'exprimer votre algorithme dans un langage de votre choix, avec les structures de données et de contrôle classiques. Tout programme devra être judicieusement commenté. Une description algorithmique ne sera acceptée que si elle est d'un niveau de détail suffisant.

I Généralités

On va étudier différentes manières d'évaluer un polynôme de degré n. Dans cette partie, la qualité de chaque technique d'évaluation sera mesurée uniquement par le nombre total d'opérations $+$ et $\times$.

I.1 Évaluation de la forme développée

Soit la forme développée suivante d'un polynôme P de degré n :

$$P(x) = a_0 + a_1 x + a_2 x^2 + ... + a_n x^n$$

Question I-1 *Écrivez un pseudo-programme qui évalue le polynôme sous cette forme. Les entrées seront n, x, et un tableau ou une liste contenant les coefficients. On tâchera de réutiliser les valeurs de x^k déjà calculées.*

Question I-2 *Donnez le nombre d'opérations en fonction du degré n.*

I.2 Schéma de Horner

Le schéma d'évaluation de Horner est un reparenthésage de la forme développée de la manière suivante :

$$P(x) = a_0 + x \times (a_1 + x \times (a_2 + ... + x \times (a_{n-1} + x \times a_n)) ...)$$

Question I-3 *Écrivez un pseudo-programme qui évalue le polynôme sous cette forme.*

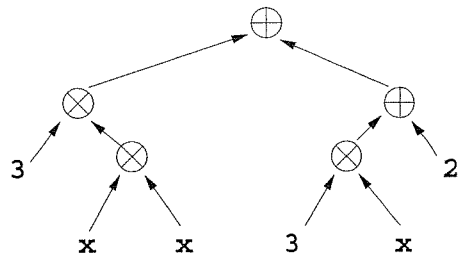
Question I-4 *Donnez le nombre d'opérations en fonction du degré n.*

I.3 Polynômes particuliers

Question I-5 *Pouvez-vous donner un schéma d'évaluation de $P(x) = x^3 + 3x^2 + 3x + 3$ en moins d'opérations que le schéma de Horner ?*

II Arbres polynômieurs

L'avantage est pour l'instant dans le cas général à la forme de Horner, qui minimise le nombre d'opérations. On va à présent s'intéresser à la formation d'arbres évaluant les polynômes. Un tel arbre sera un arbre binaire dont les feuilles sont une variable (ici x ou bien l'un des coefficients a_i), et les nœuds internes sont des opérations $+$ ou $\times$. Voici par exemple un arbre qui évalue le polynôme $3x^2 + 3x + 2$

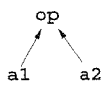


Pour un tel arbre, on s'intéresse toujours au nombre d'opérations, mais aussi à la hauteur de l'arbre, qui déterminera le temps d'exécution si cet arbre est traduit en un circuit électronique. La hauteur est également définie comme un nombre d'opérations : on considère tous les chemins d'une feuille à la racine de l'arbre, et la hauteur est le nombre d'opération maximal parmi tous ces chemins. Par exemple, la hauteur de l'arbre ci-dessus est de 3 opérations.

Question II-1 *Dessinez un arbre pour le schéma de Horner du polynôme $3x^2 + 3x + 2$. Comparez l'arbre donné en exemple et cet arbre de Horner en termes de nombres d'opérations et de hauteur.*

Question II-2 *Définissez soigneusement dans le langage choisi la structure de donnée et le type* `Arbre` *qui permettra de manipuler de tels arbres binaires. Les variables seront identifiées par des chaînes de caractère.*

On pourra utiliser dans la suite les fonctions primitives suivantes sur les arbres (suivant le langage utilisé, toutes ces fonctions ne seront pas utiles) :

- `estFeuille(a)` est une fonction qui renvoie une valeur de vérité, vraie si et seulement si l'arbre `a` est une feuille.
- `variable(a)` est une fonction qui extrait d'une feuille la chaîne de caractère correspondant à la variable. Si `a` n'est pas une feuille cette fonction produit un message d'erreur et vous avez perdu des points.
- `ssaD(a)` est une fonction qui renvoie le sous-arbre droit de l'arbre `a`. Si `a` n'est pas un nœud interne cette fonction produit un message d'erreur et vous avez perdu des points.
- `ssaG(a)` est similaire à `ssaD(a)` et renvoie le sous-arbre gauche de l'arbre `a`.
- `op(a)` renvoie l'opération sous forme d'un caractère.
- `nouvelleFeuille(s)` est une fonction prend une chaîne de caractère et renvoie un arbre composé d'une feuille.
- `nouvelleOp(a1, a2, op)` est une fonction prend deux arbres et une opération et renvoie l'arbre 

Question II-3 *Donnez un pseudo-programme construisant récursivement un arbre P_k calculant x^k, de hauteur logarithmique en k. On n'hésitera pas à expliquer l'algorithme par des petits dessins.*

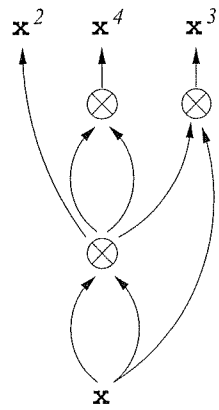
Question II-4 *Soit l'écriture binaire de* k *:* $k = \sum_{i=0}^{p} 2^i k_i$*, avec* $k_i \in \{0, 1\}$*. Pouvez-vous relier la hauteur de l'arbre produit par le programme précédent à l'écriture binaire de* k *? Justifiez soigneusement.*

Question II-5 *On se donne un tableau d'arbres* $t_1 ... t_n$*. Décrivez une construction récursive d'un arbre de hauteur logarithmique calculant la somme de ces de* n *arbres.*

Question II-6 *En déduire la construction d'un arbre évaluant un polynôme arbitraire de degré* n *de hauteur logarithmique en* n*.*

III Arbres économes

On se permet désormais de partager des sous-arbres, comme dans l'exemple suivant.



Dans ce schéma, x, x^2, x^3 et x^4 sont disponibles dans différents nœuds de l'arbre. La feuille x comme le sous-arbre calculant x^2 sont partagée entre plusieurs opérateurs. Il s'agit, lorsqu'on traduit un arbre en circuit électronique, d'économiser des opérateurs. Un tel schéma est appelé un *graphe acyclique dirigé* ou GAD. Il peut possèder plusieurs racines et plusieurs feuilles, et sa hauteur est définie comme le plus long chemin d'une feuille à une racine. On remarquera qu'un arbre est un GAD.

Question III-1 *Dessinez un GAD, le plus petit possible, qui évalue toutes les puissances de* x *entre* x *et* x^8*.*

Question III-2 *Décrivez la construction d'un GAD évaluant toutes les puissances de* x *entre* x *et* x^n *pour* $n = 2^p$*, de hauteur* p *et en* $n - 1$ *opérations. On prouvera formellement que sa hauteur est* p *et que le nombre d'opérations est* $n - 1$*.*

Question III-3 *Déduisez-en la construction d'un GAD de hauteur minimale pour l'évaluation d'un polynôme quelconque de degré* 2^p*. Quel est sa hauteur ? Quel est le nombre d'opérations dans ce GAD ?*

IV Évaluation sur un processeur simple

On va à présent étudier l'évaluation d'un polynôme sur un processeur. Il faudra pour cela se ramener au langage du processeur, qui est composé d'*instructions machines*.

On se limitera aux instructions machines de l'une des formes suivantes :

- $\boxed{r \leftarrow t}$ où r et t sont des variables arbitraires,
- $\boxed{r \leftarrow t_1 \text{ op } t_2}$ où
  - op est soit $+$, soit $\times$
  - r, t_1 et t_2 sont des variables arbitraires

Pour le processeur traité dans cette section, un programme est une suite d'instructions. Par exemple, voici un programme machine évaluant un polynôme de degré 2 par le schéma de Horner.

```
Entrée : x, a0, a1, a2

        p ← a2;
        p ←  x × p;
        p ← a1 + p;
        p ←  x × p;
        p ← a0 + p;

Résultat : p
```

Question IV-1 *Écrivez une fonction* `codeMachine(a)` *qui prend en entrée un arbre* `a`*, qui écrit à l'écran un programme machine évaluant cet arbre, et qui retourne le nom de la variable de ce code contenant le résultat. On utilisera le type* `Arbre` *défini en II. On pourra utiliser, sans la définir, une fonction* `nouvelleVariable()` *qui retourne un nouveau nom de variable jamais encore utilisé. On vérifiera bien que les instructions écrites à l'écran le sont dans le bon ordre, c'est-à-dire qu'une variable n'est jamais utilisée avant d'être définie.*

V Dépendances de données

Dans toute la suite on ne va plus travailler que sur du code machine à *assignation unique* : dans un programme, une variable r n'apparaît qu'une seule fois à gauche de la flèche dans une instruction $r \leftarrow v$ op w ou $r \leftarrow v$.

On commence par s'intéresser aux conditions sous lesquelles on peut exécuter plusieurs instructions en même temps.

On définit la relation de *dépendance directe* comme suit : une instruction $r_2 \leftarrow v_2$ op$_2$ w_2 est directement dépendante d'une instruction $r_1 \leftarrow$ ... si et seulement si $r_1 \in \{v_2, w_2\}$. De même, une instruction $r_2 \leftarrow v_2$ est directement dépendante d'une instruction $r_1 \leftarrow$ ... si et seulement si $r_1 = v_2$.

On définit la relation de *dépendance* comme la fermeture transitive de la relation de dépendance directe (une instruction I_2 est dépendante d'une instruction I_1 si elle est directement dépendante de I_1, ou si elle est directement dépendante d'une instruction I_3 elle-même dépendante de I_1).

Enfin, deux instructions I_1 et I_2 sont indépendantes si I_1 n'est pas dépendante de I_2 et I_2 n'est pas dépendante de I_1.

Question V-1 *Dessinez, par des flèches entre les variables, toutes les relations de dépendance directe sur le code suivant :*

```
x2   ← x × x;
a2x2 ← a2 × x2;
a1x  ← a1 × x;
p01  ← a0 + a1x;
p02  ← p01 + a2x2;
```

Question V-2 *Certaines instructions, dans le code précédent, peuvent être échangées sans que la valeur calculée par ce code dans* `p02` *ne soit changée. Donnez un exemple d'une telle réécriture du code.*

Question V-3 *Montrez que si deux instructions sont indépendantes, elles peuvent s'exécuter dans n'importe quel ordre. En déduire que deux instructions indépendantes peuvent s'exécuter en même temps.*

Question V-4 *Donnez un arbre ou un GAD correspondant au code ci-dessus. Vous annoterez les nœuds avec les variable du code. Quelle remarque faites-vous ?*

VI Évaluation sur un processeur superscalaire

On considère à présent un processeur qui possède 4 unités flottantes identiques, toutes capables de calculer $+$ ou $\times$, et capables de calculer en même temps. Le but est d'écrire du code pour ce processeur utilisant au mieux ces 4 unités. Le langage machine de ce processeur est défini comme suit :

- Un programme est composé de lignes qui sont exécutées en séquence.
- Chaque ligne se compose d'au plus 4 instructions, les instructions étant celles du IV.
- Toutes les instructions d'une ligne sont exécutées en même temps.

On a de plus toujours la propriété d'assignation unique : dans un programme, chaque variable n'apparait qu'une seule fois à gauche de la flèche $\leftarrow$.

Voici un exemple de code évaluant un polynôme de degré 3 en par un schéma adapté à ce processeur :

```
Entrée : x, a0, a1, a2, a3

x2    ← x  × x;     a1x ← a1 × x;    a3x ← a3 × x;
p01   ← a0 + a1x;   p23 ← a2 + a3x;
x2p23 ← x2 × p23;
p03   ← p01 + x2p23;

Résultat : p
```

Question VI-1 *Dessinez par des flèches les relations de dépendance directe sur le code précédente.*

Question VI-2 *Cet exemple n'utilise jamais les 4 unités flottantes, seulement 3 au plus. Prouvez que le nombre de lignes ne peut être réduit en déplaçant des instructions dans des emplacements vides.*

Le temps d'exécution est identifié au nombre de lignes. On définit aussi l'*utilisation* du processeur comme le nombre total d'instructions divisé par 4 fois le nombre de lignes. Par exemple, l'utilisation du programme ci-dessus est de $7/16$.

Question VI-3 *Quelle serait l'utilisation du code évaluant ce polynôme de degré 3 selon le schéma de Horner sur ce processeur ? Quel serait son temps d'exécution ?*

Question VI-4 *Donnez du code minimisant le nombre de lignes pour un polynôme quelconque de degré 5, puis pour un polynôme quelconque de degré 7. On essaiera de généraliser l'exemple, et d'utiliser des noms de variables qui généralisent ceux de l'exemple. Quels sont leurs nombres d'opérations ? Comparez avec Horner.*

Question VI-5 *Donnez du code machine implémentant au mieux sur ce processeur le GAD déterminé question III-3 pour le degré 7. Comparez avec le code de la question précédente en terme de nombres de lignes et de nombre d'opérations.*

On s'intéresse à présent au code pour un polynôme quelconque de degré $n = 2^p - 1$, pour p entier arbitraire. On cherche à minimiser le nombre de lignes de 4 instructions.

Question VI-6 *Décrivez un algorithme qui imprime à l'écran le code pour un polynôme de degré $n = 2^p - 1$, pour p entier arbitraire. On pourra définir d'abord des ensembles d'instructions indépendantes, puis les partitionner en lignes de taille 4. On pourra aussi raisonner sur un GAD.*

Question VI-7 *Si n est très grand, vers quoi tendent le nombre d'opérations, le nombre de lignes et l'utilisation ?*