Informatique
Devoir surveillé 3
16 décembre 2021

MP2I 2021/2022

CIV

Durée totale 2h.
Tout document interdit. Tout dispositif électronique interdit.
On vous demande d'écrire lisiblement et de bien présenter. Les programmes en particulier doivent être bien indentés et lisibles. Utilisez de préférence plusieurs couleurs. **Commentez** votre code dès que ce qu'il fait n'est pas évident.

1 Pirelli

On considère la fonction suivante :

```
let expo_naive x n =
    let p = ref 1 in
    for i = 1 to n do
        p := !p * x
    done;
    !p
```

Q01 Montrer que la fonction précédente calcule x^n en considérant sa terminaison acquise.

Q02 Combien de multiplications effectue cette fonction pour calculer x^n ?

On considère maintenant la fonction :

```
let expo_binaire x n =
    let p = ref 1 in
    let a = ref x in
    let k = ref n in
    while !k > 0 do
        if !k mod 2 = 1
        then p := !p * !a;
        a := !a * !a;
        k := !k / 2
    done;
    !p
```

Q03 Donner l'état des variables à chaque itération dans l'appel de `expo_binaire 3 15`.

Q04 Prouver la correction de cette fonction :
1. sa terminaison ;
2. le fait qu'elle calcule x^n.

Soit $n \in \mathbb{N}$, on note
- $\lfloor \log_2 n \rfloor$ l'unique entier p tel que $2^p \leqslant n < 2^{p+1}$.
- $\lceil \log_2 n \rceil$ l'unique entier p tel que $2^{p-1} < n \leqslant 2^p$.
- $\nu(n)$ le nombre de 1 dans l'écriture binaire de n. Plus précisément, si $n = a_0 + a_1 2 + a_2 4 + \dots a_{d-1} 2^{d-1}$ avec $a_0, \dots, a_{d-1} \in \{0, 1\}$ alors il y a exactement $\nu(n)$ des a_i qui valent 1.

Q05 On suppose que $n > 0$. Montrer que la fonction `expo_binaire` effectue $\lceil \log_2 n \rceil + \nu(n)$ multiplications pour calculer x^n.

L'algorithme précédent effectue des multiplications non nécessaires pour gérer les variables : le premier calcul $1 \times x$ et la dernière mise au carré. Si on effectue ce principe à la main, pour calculer x^{15} on place x dans p (pas de multiplications cf. remarque), puis on calcule x^2, on place $x^3 = x \times x^2$ dans p, puis on calcule x^4, on place $x^7 = x^3 \times x^4$ dans p, on calcule x^8 et on place $x^{15} = x^7 \times x^8$ dans p. Cette méthode de décomposition en binaire calcule ainsi x^{15} en six multiplications.
On peut faire mieux, en effet, on peut remarquer que $15 = 3 \times 5$. Pour cela, on peut calculer $x^3 = x \times x \times x = y$ en deux multiplications et $x^{15} = y^5 = (y^2)^2 \times y$ en trois multiplications en calculant $x^6 = x^3 \times x^3$, $x^{15} = x^6 \times x^6 \times x^3$. Au total, on aura ainsi effectué cinq multiplications.
La méthode qu'on vient d'utiliser est appelée la méthode du facteur :

— si n est non premier et > 1, on écrit $n = p \times q$ avec p le plus petit facteur premier de n. On a alors $q > 1$. Pour calculer x^n, on commence par calculer $x^p = y$ puis $y^q = (x^p)^q = x^{pq} = x^n$ par cette méthode.
— si n est premier, on écrit $x^n = x \times x^{n-1}$ et on applique la méthode à $n - 1$.
— si $n = 1$, on a directement $x = x^1$.

Q06 Comparer les deux méthodes pour $n = 55$ et $n = 33$. Que peut-on en conclure ?

Pour noter ces calculs de puissances, on peut considérer la suite finie des exposants utilisés. Ainsi, pour calculer x^{15} par la méthode binaire on obtient la chaine $(1, 2, 3, 4, 7, 8, 15)$ qui est de longueur 7. En effectuant la méthode du facteur, on obtient la chaine $(1, 2, 3, 6, 12, 15)$ de longueur 6.

Q07 Donner, sans les justifier, les chaines correspondant aux deux méthodes pour 16, 27 et 125.

Définition 1. *Une suite finie* $(n_0, \dots, n_r)$ *est appelée une chaine de calcul pour* x^n *si :*
— $n_0 = 1$
— $n_r = n$
— $\forall k \in [\![1, r]\!], \exists i, j \in [\![0, k-1]\!], n_k = n_i + n_j$

La dernière condition exprime le fait qu'on a pu calculer $x^{n_k} = x^{n_i} \times x^{n_j}$.

Q08
1. Montrer par récurrence sur r que toute chaine $(n_0, \dots, n_r)$ de calcul pour x^n vérifie
$$\forall k \in [\![0, r]\!], n_k \leqslant 2^k$$
2. En déduire que tout calcul de x^n effectue au moins $\lceil \log_2 n \rceil$ multiplications.
3. Donner une infinité de valeurs de n pour lesquelles ce minorant est atteint.

2 Breizhsort

Si `t` est un tableau et $0 \leqslant i \leqslant j < |t|$ des indices, on notera `t[i:j]` le sous-tableau de `t` composé des valeurs entre les indices `i` et `j` inclus.
On va définir dans la suite du sujet une fonction

```
retourner : 'a array -> int -> int -> unit
```

telle que `retourner t i j` retourne le sous-tableau `t[i:j]`.
Ainsi, si `t = [| 1; 2; 3; 4; 5; 6 |]` alors `retourne t 2 4` change `t` en `[| 1; 2; 5; 4; 3; 6 |]` car `t[2:4] = [| 3; 4; 5 |]`.

Q09 Quel est le contenu de `t` à l'issue du programme suivant ?

```
let t = [| 5; 8; 2; 0; 3; 6; 5; 2; 1 |];;
retourner t 1 7;;
```

Note : vous pouvez ignorer les `;;` ils ne sont là que pour s'assurer qu'on puisse avoir cette interaction dans un toplevel `OCaml`.

On considère les fonctions suivantes :

```
let echanger t i j =
  let temp = t.(i) in
    t.(i) <- t.(j);
    t.(j) <- temp

let rec retourner t i j =
    if i < j
    then begin
      echanger t i j;
      retourner t (i+1) (j-1)
   end
```

Q10 Montrer que `retourner` termine.

Q11 Montrer que `retourner` correspond bien à la spécification de l'énoncé. On pourra montrer par récurrence

sur $\texttt{j} - \texttt{i} + 1$ [a] que

$$\forall \texttt{k} \in [\![\texttt{i}, \texttt{j}]\!], \texttt{t.(k)} = \texttt{t'.(i+j-k)}$$

où `t'` est la valeur initiale de `t`.

a. le $+1$ n'est là que pour déplacer l'initialisation au cas de base

On considère maintenant les fonctions suivantes :

```
let indice_max t k =
    let m = ref 0 in
    for i = 1 to k do
        if t.(!m) < t.(i)
        then m := i
    done;
    !m

let placer t k =
    let i = indice_max t k in
    if i <> k
    then begin
        retourner t 0 i;
        retourner t 0 k
    end
```

Q12 Montrer qu'en sortie de l'appel à `placer t k` le tableau `t` vérifie :

$$\forall \texttt{i} \in [\![0, \texttt{k}]\!], \texttt{t.(i)} \leqslant \texttt{t.(k)}$$

On en déduit alors un algorithme de tri, dit *tri du crêpier* car il consiste à faire des retournements successifs comme une pile de crêpes.

```
let rumman t =
   for k = Array.length t - 1 downto 0 do
      placer t k
   done
```

Q13 Donner l'état du tableau à chaque itération dans l'appel à `rumman t` où `t = [| 0; 5; 7; 1; 2 |]`.

Q14 En utilisant un invariant de boucle, montrer que la fonction `rumman` trie le tableau passé en argument dans l'ordre croissant.

Q15 Donner, en les justifiant, les complexités temporelles dans le pire des cas des fonctions `retourne`, `placer`, `indice_max` et `rumman`.

`lawful_good`	`neutralGood`	`chaotic_Good`
`fLawfulneutral`	`trueneutral`	`cHaOtIcNeUtRaL`
`LAWFULEVIL`	`n3UTR4LeV1L`	`fCh_40T1C_Ev1l`