Épreuve d'informatique de l'X - 2013 - MP/PC

Partie I. Recherche de point fixe : cas général

Question 1. On parcourt le tableau dans la boucle des lignes 3 à 5. Dès que l'on a trouvé un point fixe, on quitte le programme en retournant la valeur **vrai** (ligne 4). Si on a parcouru tout le tableau sans rencontrer de point fixe, on retourne la valeur **faux** (ligne 6).

```

1 admet_point_fixe := proc(t)
2   local i;
3   for i from 0 to n-1 do
4     if t[i] = i then return(true) fi;
5   od;
6   return(false)
7 end;
```

Question 2. On parcourt le tableau **t** à l'aide de la boucle des lignes 3 à 5 et on comptabilise le nombre de points fixes à l'aide du compteur **compt**. On incrémente le compteur **compt** à chaque fois que l'on trouve un point fixe (ligne 4).

```

1 nb_points_fixes := proc(t)
2   local i, compt := 0;
3   for i from 0 to n-1 do
4     if t[i] = i then compt := compt+1 fi
5   od;
6   return(compt);
7 end;
```

Question 3 : On calcule $f^k(x)$ de façon itérative à l'aide de la boucle des lignes 3 à 5. On initialise la variable **res** à x puis on calcule les valeurs successives de $f^i(x)$ jusqu'à la valeur $f^k(x)$ (ligne 4).

```

1 itere := proc(t,x,k)
2   local i, res := x;
3   for i from 1 to k do
4     res := t[res]
5   od;
6   return(res);
7 end;
```

Question 4 : Pour calculer le nombre de points fixes de f^k , on parcourt le tableau **t** dans la boucle des lignes 3 à 5 et pour chaque x vérifiant $f^k(x) = x$, on incrémente **compt** le compteur de points fixes de f^k . On retourne la valeur du compteur après l'exécution de la boucle (ligne 5).

```

1 nb_points_fixes_iteres := proc(t,k)
2   local x, compt := 0;
3   for x from 0 to n-1 do
4     if itere(t,x,k) = x then compt := compt+1 fi;
5   od;
6   return(compt);
7 end;
```

Question 5 : Si $f : E_n \rightarrow E_n$ admet un attracteur principal z alors nécessairement $\forall x \in \llbracket 0, n-1 \rrbracket$, $f^n(x) = z$. Dans le programme ci-dessous, on commence par calculer $z = f^n(0)$ que l'on représente par la variable **z**. On vérifie que le z candidat est bien un point fixe (ligne 4) sinon, on retourne **faux**. Puis, dans la boucle des lignes 5 à 7, on vérifie que $\forall x \in \llbracket 1, n-1 \rrbracket$, $f^n(x) = z$. Si ce n'est pas le cas, on retourne **faux** (ligne 6) et si c'est le cas, on retourne **vrai** (ligne 8).

```

1 admet_attracteur_principal := proc(t)
2   local x, z;
3   z := itere(t,0,n);
4   t[z] <> z then return(false) end;
5   for x from 1 to n-1 do
6     if itere(t,x,n) <> z then return(false) fi;
7   od;
8   return(true)
9 end;
```

Question 6 : Voici une première version récursive exploitant la remarque de l'énoncé.

```

1 temps_de_convergence := proc(t,x)
2   if t[x] = x
3     then return(0)
4     else return(1+temps_de_convergence(t, t[x]))
5   fi;
6 end;
```

Voici une deuxième version itérative.

```

1 temps_de_convergence := proc(t,x)
2   local compt := 0, y := x, ty := t[y];
3   while y <> ty do
4     y := ty;
5     ty := t[y];
6     compt := compt+1;
7   end;
8   return(compt);
9 end;
```

Question 7 : On commence par créer et initialiser avec des -1 un tableau tab qui va contenir au final les temps de convergence des éléments de E_n (lignes 3 à 6). On calcule ensuite le temps de convergence de 0 (ligne 7) et on affecte les temps de convergence de 0 et de ses images par f jusqu'au point fixe (boucle des lignes 10 à 13). Dans la boucle des lignes 14 à 29, on étudie les points pour lesquels on ne dispose pas du temps de convergence. Dans la boucle conditionnelle des lignes 19 à 22, on cherche la première des images itérées de i pour laquelle le temps de convergence a déjà été calculé. On établit le temps de convergence de i à la ligne 23. On change éventuellement le temps de convergence maximal à la ligne 24. Dans la boucle des lignes 25 à 27, on affecte les temps de convergence des images de i qui n'avaient pas été calculés auparavant. À la ligne 30, on retourne le résultat.

```

1 temps_convergence_max := proc(t)
2   local tmax := 0, tab, n, i, j, k, tc, ti, x;
3   tab := allouer(n);      # tableau des temps de convergence
4   for i from 0 to n-1 do
5     tab[i] := -1
6   end;
7   tc := temps_de_convergence(t,0);
8   tmax := tc;
9   x := 0;
10  for i from 0 to tc do    # on initialise les temps de convergence
11    tab[x] := tc-i;      # de 0 et de ses images.
12    x := t[x]
13  od;
14  for i from 1 to n-1 do  # on parcourt le tableau tab.
15    if tab[i] = -1      # cas où le temps de convergence
16      then              # de i n'est pas établi.
17        k := 0;
18        x := i;
19        while tab[x] = -1 do # on recherche la première des images
20          k := k+1;        # de i dont on dispose du temps
21          x := t[x];      # de convergence.
22        od;
23        ti := k+tab[x];   # temps de convergence de i.
24        tmax := max(tmax, ti);
25        for j from 1 to k-1 do # on initialise les temps
26          tab[i] := ti-j;   # de convergence de i
27        od;                # et de ses images.
28      fi;
29    od;
30    return(tmax);
31 end;
```

Partie II. Recherche efficace de points fixes

Premier cas.

Question 8 : Pour vérifier qu'une fonction $f : E_n \rightarrow E_n$ représentée par un tableau t est croissante, il suffit de vérifier que $\forall i \in \llbracket 0, n-2 \rrbracket$, $t[i] \leq t[i+1]$. Si l'une de ses conditions n'est pas vérifiée, on quitte le programme en retournant la valeur **faux** (ligne 4). Sinon, tout le tableau est parcouru et on retourne la valeur **vrai** à la ligne 6.

```
1 est_croissante := proc(t)
2   local i;
3   for i from 0 to n-2 do
4     if t[i] > t[i+1] then return(false) fi;
5   od;
6   return(true);
7 end;
```

Question 9 : On effectue une recherche dichotomique. Au départ, on a nécessairement $f(0) \geq 0$ et $f(n-1) \leq n-1$. On remarque aussi que, comme la fonction est croissante, si $f(a) \geq a$ et $f(b) \leq b$ avec $a \leq b$ alors f admet nécessairement un point fixe entre a et b . Dans la boucle conditionnelle des lignes 4 à 13, on commence par calculer m le point médian de l'intervalle $\llbracket a, b \rrbracket$. Si $f(m) = m$, on a trouvé un point fixe, on retourne sa valeur (ligne 7). Si $f(m) > m$ alors on a un point fixe dans l'intervalle $\llbracket m+1, b \rrbracket$ qui devient notre nouvel intervalle de recherche. Si $f(m) < m$, un point fixe se trouve dans l'intervalle $\llbracket a, m-1 \rrbracket$ qui devient dans ce cas notre nouvel intervalle de recherche (ligne 10). Si on quitte la boucle conditionnelle sans avoir trouvé de point fixe, c'est que certainement la fonction f représentée n'était pas croissante (ligne 14).

```
1 point_fixe := proc(t)
2   local i, a:=0, b, m;
3   b := n-1;
4   while a <= b do
5     m := floor((a+b)/2); # calcul point médian
6     if t[m] = m
7       then return(m); # le point médian est point fixe
8     else if t[m] > m
9       then a := m+1 # le point fixe est entre m+1 et b
10      else b := m-1 # le point fixe est entre a et m-1
11      fi;
12    fi;
13  od;
14  print("il y a un problème");
15 end;
```

Question 10 : On considère l'entier $d = b - a$. Initialement, avant de rentrer dans la boucle conditionnelle, d vaut $n - 1$. Par définition de m , en début de boucle, on a $m - 1 < m = \lfloor \frac{b+a}{2} \rfloor \leq \frac{b+a}{2} < m + 1$ et $a \leq \frac{a+b}{2} \leq b$.

Dans le cas où m n'est pas point fixe,

- si b prend la valeur $m - 1$ alors le nouvel intervalle de recherche a pour longueur $m - 1 - a \leq \frac{b+a}{2} - a = \frac{b-a}{2}$.
- si a prend la valeur $m + 1$ alors le nouvel intervalle de recherche a pour longueur $b - (m + 1) \leq b - \frac{b+a}{2} = \frac{b-a}{2}$.

Dans tous les cas de figure, la longueur de l'intervalle de recherche est à chaque étape au moins divisé par 2. Si on note d_i la longueur de l'intervalle à la fin du i -ème passage dans la boucle, on a $d_i \leq \frac{(b-a)}{2^i}$. Si $d_i = 0$, l'intervalle d'étude est réduit à un point au $(i+1)$ -ème passage dans la boucle, le point est nécessairement point fixe. Comme d_i est un entier, il suffit d'avoir $d_i < 1$ pour avoir $d_i = 0$ donc $d_i = 0$ si $\frac{(b-a)}{2^i} < 1 \iff i > \ln(b-a)/\ln(2)$. On trouve donc un point fixe en au plus $\lceil \frac{\ln(b-a)}{\ln(2)} \rceil + 1$ étapes, le temps de calcul est bien logarithmique.

Deuxième cas.

Question 11 : Soit m le plus petit élément de $(E, \preceq)$, $f : E \rightarrow E$ une fonction croissante au sens de $\preceq$ et x un point fixe de f .

$$\begin{aligned} m \preceq x &\Rightarrow f(m) \preceq f(x) = x \\ &\Rightarrow f^2(m) = f(f(m)) \preceq f(x) = x \quad (\text{par croissance de } f) \\ &\Rightarrow \dots \Rightarrow \forall p \in \mathbb{N}, f^p(m) \preceq x \quad (\text{par récurrence sur } p). \end{aligned}$$

Donc si k est tel que $f^k(m)$ est un point fixe, on a $f^k(m) \preceq x$ pour tout point fixe x de f .

conclusion : $f^k(m)$ est le plus petit des points fixes de f .

Question 12 : Soit $x_1, \dots, x_m$ les points fixes de f et $x = \text{pgcd}(x_1, \dots, x_m)$. On a $\forall k \in \llbracket 1, m \rrbracket$, $1 \preceq x \preceq x_k$. Par croissance de f , on a $f(1) \preceq f(x) \preceq f(x_k) = x_k$ et donc, par propriété du pgcd, $f(x) \preceq \text{pgcd}(x_1, \dots, x_m) = x$. D'où $\forall k \in \llbracket 1, m \rrbracket$, $f(1) \preceq f(x) \preceq x \preceq x_k$.

Par récurrence sur p , on montre que $\forall p \in \mathbb{N}$, $\forall k \in \llbracket 1, m \rrbracket$, $f^p(1) \preceq f(x) \preceq x \preceq x_k$. On considère p tel que $f^p(1)$ soit le plus petit point fixe de f (p existe d'après la question précédente). $f^p(1)$ étant un point fixe, on peut choisir $k_0 \in \llbracket 1, m \rrbracket$ tel que $f^p(1) = x_{k_0}$. D'après ce qui précède, on a :

$$x_{k_0} \preceq x \preceq x_{k_0} \Rightarrow f^p(1) = x_{k_0} = x = \text{pgcd}(x_1, \dots, x_m).$$

conclusion : $\text{pgcd}(x_1, \dots, x_m) = f^p(1) =$ le plus petit des points fixes de f .

Question 13 : Comme 1 est le plus petit élément de E au sens de la divisibilité, d'après la question 11, la suite $(f^k(1))$ est stationnaire à partir d'un certain rang et égal au plus petit point fixe au sens de la divisibilité, c'est-à-dire au pgcd des points fixes de f , d'après la question précédente.

```

1  pgcd_point_fixes := proc(t)
2      local x := 1;
3      while t[x] <> x do
4          x := t[x]
5      od;
6      return(x)
7  end;
```

Question 14 : L'entier 1 est le plus petit élément de $(E, \preceq)$, on a $1 \preceq f(1)$. Par croissance de l'application f et par récurrence sur p , $\forall p \in \mathbb{N}$, $f^p(1) \preceq f^{p+1}(1)$, c'est-à-dire $f^p(1)$ divise $f^{p+1}(1)$. Si $f^p(1) \neq f^{p+1}(1)$ alors $f^{p+1}(1) = 0$ ou $f^{p+1}(1) \geq 2 \cdot f^p(1)$. Soit k tel que $f^k(1)$ soit point fixe. $1, f(1), \dots, f^{k-1}(1)$

sont k valeurs distinctes non nul et donc $f^{k-1}(1) \geq 2^{k-1}$ comme $f^{k-1}(1) \in \llbracket 0, n-1 \rrbracket$, on a $2^{k-1} \leq n \Rightarrow k-1 \leq \log_2(n) \Rightarrow k \leq \log_2(n) + 1$. Le temps de calcul est bien logarithmique.