La suite de Syracuse [it06] - Exercice

Karine Zampieri, Stéphane Rivière

Unisciel algoprog Version 17 mai 2018

Table des matières

alg - La suite de Syracuse (Solution)

Mots-Clés Schéma itératif, Suite de Syracuse ■
Requis Structures de base, Structures conditionnelles, Structures répétitives ■
Difficulté • • ∘ (25 min) ■

Objectif
Cet exercice propose quelques problèmes autour de la suite de Syracuse.

http://images.math.cnrs.fr/Le-probleme-3n-1-elementaire-mais.html

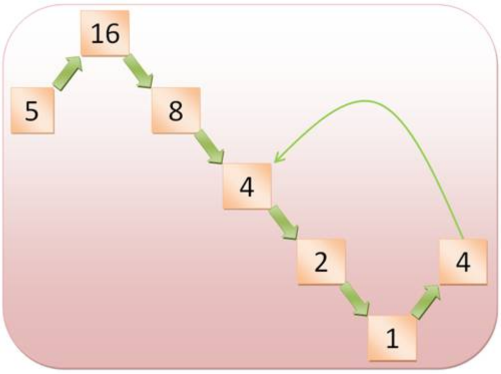
1 La suite de Syracuse / pgsyracuse

1.1 Présentation du problème

Définition
Étant donné un entier initial $u_0 > 0$, les termes de la **suite de Syracuse** (appelée aussi suite de Collatz ou suite des grelons) sont donnés par la récurrence :

$$u_{n+1} = \begin{cases} u_n \text{ div } 2 & \text{si } u_n \text{ pair} \\ 3u_n + 1 & \text{si } u_n \text{ impair} \end{cases}$$

Exemple
Avec $u_0 = 5$ elle donne $5, 16, 8, 4, 2, 1, 4, 2...$ et pour $u_0 = 13$ elle génère $13, 40, 20, 10, 5$ qui génère finalement le cycle $1, 4, 2, 1, 4, 2, 1, 4, 2...$



Conjecture de Collatz
Elle stipule que la suite de Syracuse donne un terme égal à 1 en un temps fini pour tout entier naturel u_0. On tient cette conjecture pour vraie jusqu'à preuve du contraire.

Objectif
Étant donné un germe u_0, calculer : (a) les termes de la suite, (b) le nombre de termes ainsi que (c) le plus grand des termes.

...(suite page suivante)...

1.2 Termes de la suite

Écrivez un algorithme de sorte qu'il saisit le terme initial u_0 dans un entier `u0` **tant qu'il n'est pas** (ou **jusqu'à** ce qu'il soit) > 0. Affichez l'invite :

```
Germe initial?
```

Calculez et affichez les termes de Syracuse issus du germe `u0`.
N'oubliez pas d'afficher le dernier (ou premier) terme.

Aide détaillée
Il faut :

1. Déclarer une variable `un` (pour u_n) et l'initialiser avec le germe `u0` (de u_0).
2. Tant que 1 n'est pas atteint (conjecture de Collatz) pour `un` :
   - Afficher le terme courant `un`.
   - Avancer au terme suivant d'après la relation :

$$u_{n+1} = \begin{cases} u_n \text{ div } 2 & \text{si } u_n \text{ pair} \\ 3u_n + 1 & \text{si } u_n \text{ impair} \end{cases}$$

On sort de la boucle pour `un` valant 1 : il faut donc afficher ce dernier terme (qui fait également parti de la suite). Notez que si, dans la boucle, on inverse l'affichage et le calcul du terme suivant, il faut alors afficher le premier terme puisque la boucle détermine le terme u_{n+1}.

Testez. Exemple d'exécution :

```
Germe initial? 25
25 76 38 19 58 29 88 44 22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1
```

1.3 Nombre de termes

Complétez votre algorithme de sorte qu'il calcule le **nombre de termes** de la suite issu du germe `u0`. (« Complétez » signifie qu'il faut rajouter des instructions et n'en supprimer aucune.)

Aide simple
Déclarez et calculez une variable `nt` qui mémorisera le nombre de termes et lors de chaque affichage de `un`, incrémenter `nt` de 1.

Testez. Exemples d'exécution :

```
Germe initial? 25
==> 24 termes
```

```
Germe initial? 1132
==> 63 termes
```

1.4 Plus grand terme

Enfin **complétez** votre algorithme de sorte qu'il calcule et renvoie la valeur du **plus grand terme** de la suite issu du germe `u0`.

Aide détaillée

Déclarez et calculez une variable `tmax` qui mémorisera le plus grand terme. On pourra initialiser `tmax` soit avec 0 (la suite est définie sur les entiers naturels), soit avec le premier terme `u0` (initialisation à un terme utile). Dans la boucle, il faut tester le terme `un` par rapport à `tmax` : s'il est plus grand, c'est que l'on a trouvé mieux et donc actualiser `tmax`.

Testez. Exemples d'exécution :

```
Germe initial? 25
==> Plus grand 88
```

```
Germe initial? 1132
==> Plus grand 9232
```

Validez votre algorithme avec la solution.

Solution alg @[pgsyracuse.alg]

```
Algorithme pgsyracuse
Variable u0 : Entier
Variable un : Entier
Variable nt : Entier
Variable tmax : Entier
Début
  |  Répéter
  |     |  Afficher ( "Germe initial? " )
  |     |  Saisir ( u0 )
  |  Jusqu'à ( u0 > 0 )
  |  un <- u0
  |  nt <- 1
  |  tmax <- u0
  |  TantQue ( un <> 1 )
  |     |  AfficherSS ( un , " ")
  |     |  Si ( Modulo ( un , 2 ) = 0 ) Alors
  |     |  |  un <- DivEnt ( un, 2 )
  |     |  Sinon
  |     |  |  un <- 3 * un + 1
  |     |  FinSi
  |     |  nt <- nt + 1
  |     |  Si ( tmax < un ) Alors
  |     |     |  tmax <- un
  |     |  FinSi
  |  FinTantQue
  |  Afficher ( un )
  |  nt <- nt + 1
  |  Afficher ( "==> " , nt , " termes" )
```

```
  |  Afficher ( "==> Plus grand " , tmax )
Fin
```

2 Références générales

Comprend [http://images.math.cnrs.fr/Le-probleme-3n-1-elementaire-mais.html] ■