Boucles et récurrence

Vallon

2 septembre 2017

Objectifs :

- Une boucle peut être décrite par une suite récurrente
- La technique de démonstration par récurrence peut nous aider à prouver que la boucle calcule bien ce qu'elle prétend calculer
- Mieux Ce que l'on veut calculer, vue comme une propriété héréditaire ou invariant de boucle nous aide à trouver les instructions à l'intérieur de la boucle
- Regardons cela sur plusieurs exemples

- **Problème 1 :** Trouver le plus petit élément d'une liste d'entiers $[L[0], L[1], ..., L[f]]$
- Nous allons faire une boucle pour parcourir la liste du premier élément $L[0]$ jusqu'au dernier $L[f]$.
  Supposons que nous ayons parcouru la liste de $L[0]$ jusqu'à $L[k]$ et que nous avons mémorisé le minimum de la sous-liste $[L[0], L[1], ..., L[k]]$ dans une variable appelée min, que devons nous faire pour que après examen de $L[k+1]$, min contient le minimum de $[L[0], L[1], ..., L[k+1]]$?
- Il suffit de comparer $L[k+1]$ à min pour conserver dans min le plus petit de $L[k+1]$ et min

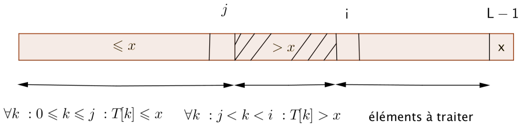
Algorithme 1 :

Données : Une liste d'entiers naturels $[L[0], L[1], ..., L[f]]$
Résultat : La plus petite valeur de cette liste
début
    min $\leftarrow L[0]$
    pour x *dans liste* **faire**
        si $x < min$ **alors**
            min $\leftarrow x$
        fin
    fin
fin

Preuve de l'algorithme du minimum d'une liste d'entiers

- Soit $I(n)$: min contient la plus petite valeur de $[L[0], L[1], ..., L[n]]$ montrons par récurrence que $I(n)$ est vraie pour $n = 0$ jusqu'à $n = f$
- $I(0)$ est vraie car min est initialisée avec $L[0]$
- Supposons $I(n)$ vraie montrons que $I(n+1)$ l'est aussi pour $n \geqslant 0$: Si min contient le plus petit élément de $[L[0], L[1], ..., L[n]]$, au prochain tour de boucle on examine $L[n+1]$ que l'on compare à min donc min contiendra le plus petit élément de $[L[0], L[1], ..., L[n], L[n+1]]$
- La propriété I(n) est donc héréditaire elle est donc vraie de $n = 0$ jusqu'à $n = f$
- $I(f)$: min contient la plus petite valeur de $[L[0], L[1], ..., L[f]]$
- On dit que $I(n)$ est un invariant de boucle

- **Problème 2 :** Etant donné une liste de nombres de longueur L, $[T[0], T[1], \ldots, T[L-1]]$ il s'agit de partitionner le tableau en deux en fonction du dernier élément de la liste $T[L-1]$
- Une première partie au début de la liste(éventuellement vide) contenant les éléments de la liste inférieurs ou égaux à $T[L-1]$
- Le reste de la liste contient les nombres strictement supérieurs à $T[L-1]$
- Ensuite on insère $T[L-1]$ entre les deux parties

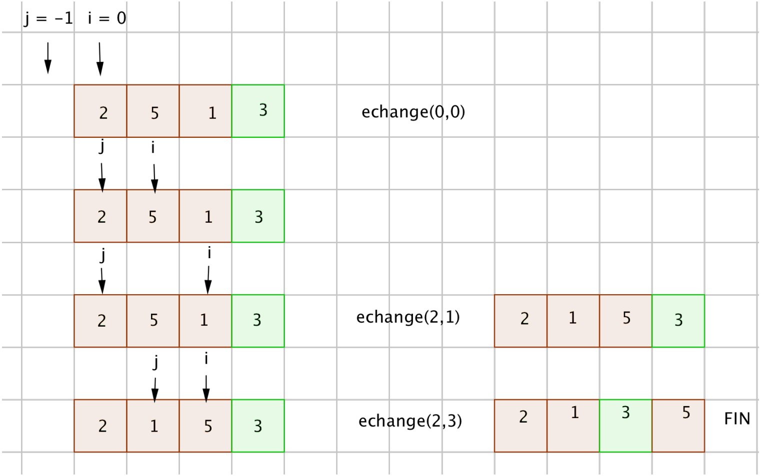


- L'invariant de boucle étant déterminé par la situation finale, c'est à dire la partition de la liste en deux blocs, les nombres inférieurs ou égaux à $T[L-1]$, puis les nombres strictement supérieurs à $T[L-1]$, il s'agit de maintenir invariante cette partition à chaque tour de boucle
- Supposons que nous ayons parcouru la liste de $T[0]$ jusqu'à $T[i-1]$ et que "le travail a été fait" c'est à dire de 0 à $i-1$ les nombres sont rangés en deux blocs , le premier de 0 à j, la variable j mémorise la fin du premier bloc), puis le second de $j+1$ à $i-1$ contient les nombres strictement supérieur à $T[L-1]$ que devons nous faire après examen de la couleur en i pour que l'ordre soit conservé ?

1. On suppose que nous avons une fonction echange(liste,i,j) qui échange les nombres d'indice i et j dans la liste
2. On pose $x = T[L-1]$
3. Si $T[i] > x$ on laisse le nombre à sa place dans la liste et on incrémente i (cela signifie que $i \leftarrow i + 1$)
4. Si $T[i] \leqslant x$ on incrémente j et on fait echange(T,i,j)

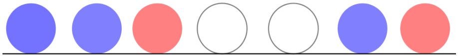
Algorithme 2 :

Données : Une liste de nombres $T[0], \dots . T[L-1]$
Résultat : La liste est partitionnée en fonction du dernier élément $T[L-1]$
début
  $j \leftarrow -1$
  pour *i de 0 jusqu'à L-2* **faire**
    si $T[i] \leqslant T[L-1]$ **alors**
      $j \leftarrow j+1$
      echanger(T,i,j)
    fin
  fin
  echanger(T, j+1,L-1)
fin

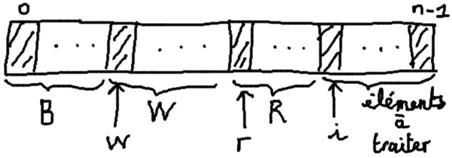

j = −1
i = 0
2 5 1 3
echange(0,0)
j
i
2 5 1 3
j
i
2 5 1 3
echange(2,1)
2 1 5 3
j
i
2 1 5 3
echange(2,3)
2 1 3 5
FIN

Preuve de l'algorithme de partition

1. Initialisation : Avant la première itération $j = -1$ et $i = 0$ et les blocs sont vides donc les conditions de l'invariant de boucle sont vraies
2. Hérédité : Exercice
3. Terminaison : A la fin $i = L - 2$ le tableau est partitionné



- **Problème 3 :** Ranger les balles sur une étagère dans l'ordre suivant (en regardant de la gauche vers la droite), les bleues, puis les blanches et enfin les rouges.
  Contraintes :
  - On ne peut examiner la couleur d'une balle qu'une fois
  - Pour ranger les balles on ne peut qu'échanger deux balles de place
  Structure des données : L'ensemble des balles est une liste de couleurs [C[0],C[1],....C[nbBalles-1]] où C[0] est la balle la plus à gauche et C[nbBalles-1] la balle la plus à droite



- L'invariant de boucle étant déterminée par la situation finale, c'est à dire la partition de la liste en trois blocs, les balles bleues, puis les blanches, puis les rouges, il s'agit de maintenir invariante cette partition à chaque tour de boucle
- Supposons que nous ayons parcouru la liste de $C[0]$ jusqu'à $C[i-1]$ et que "le travail a été fait" c'est à dire de 0 à $i-1$ les balles sont rangées en trois compartiments les bleues de 0 à $w-1$ (la variable w mémorise la première balle blanche), les blanches de w à $r-1$ (la variable r mémorise la première balle rouge), puis les rouges de r à $i-1$, que devons nous faire après examen de la couleur en i pour que l'ordre soit conservé ?

- On a une fonction echange(liste,j,k) qui échange les balles de position j et k , avec j et k entre 0 et f (pas forcément différents)
- Si la balle en i est de couleur blanche, on échange cette balle blanche avec la première balle rouge en faisant echange(liste,i,r), et on met à jour r en faisant $r \leftarrow r+1$
- Si la balle en i est de couleur bleue,
  - (première possibilité) on échange cette balle bleue avec la première balle blanche, en faisant echange(liste,i,w), on met à jour . Par contre on se retrouve dans la situation précédente avec une balle blanche en i donc on fait echange(liste,i,r), et on met à jour r et w en faisant $w \leftarrow w+1$ et $r \leftarrow r+1$
  - (deuxième possibilité) on échange cette balle bleue avec la première balle rouge, en faisant echange(liste,i,r), et on a une balle bleue entre les rouges et les blanches donc on fait echange(liste,r,w), et on met à jour r et w en faisant $r \leftarrow r+1$ et $w \leftarrow w+1$
  - On choisit la deuxième possibilité car dès le premier échange les balles rouges sont en place
- Si la balle est rouge on ne fait rien

- L'invariant de boucle est :
- $0 \leqslant w \leqslant r \leqslant i$
- Pour tout j tel que $0 \leqslant j < w$ on a $C[j] = "B"$
- Pour tout j tel que $w \leqslant j < r$ on a $C[j] = "W"$
- Pour tout j tel que $r \leqslant j < i$ on a $C[j] = "R"$

Algorithme 3 :

Données : Une liste de lettres $C[0], \dots. C[f]$ où $C[i]$ vaut "B" ou "W" ou "R"

Résultat : La liste est triée en trois blocs, éventuellement vide d'abord le bloc "B" puis "W" puis "R"

début

fin

Preuve de l'algorithme du drapeau

- **Problème 4 :** Etant donné deux entiers naturels a et b déterminer le plus grand commun diviseur d de a et b et l'expression de d sous la forme d'une combinaison linéaire de a et b :

$$d = au + bv$$

  où u et v sont des entiers relatifs
- Partir de l'algorithme d'Euclide du pgcd puis ajouter deux variables u et v
- A l'état initial $d \leftarrow a$, $u \leftarrow 1$ et $v \leftarrow 0$
- En exercice utiliser ce qui a été vu précédemment pour trouver les instructions de la boucle en utilisant comme invariant de boucle

$$d = au + bv$$