G. Hunault
Licence L3 Informatique

Angers, juin 2020

Décomposition, Conception et Réalisation d'Applications

1. Un peu d'APL

On suppose que V est un vecteur non vide d'entiers, par exemple V ← 1 4 5 5 3.

Que renvoie numériquement l'expression APL $(+/\mathbf{V}=\mathbf{1}) + (+/\mathbf{V}=\mathbf{5})$ pour ce vecteur ?

Détaillez votre réponse.

Et dans le cas général, que calcule cette expression APL ?

2. Pratique de GREP et AWK

2.1 Commande GREP

On dispose d'un fichier nommé `listeFic.lst` qui contient une liste de fichiers générée sous Unix par la commande ls avec l'option -l.

Quelle commande grep permet de compter dans cette liste le nombre de fichiers dont le nom finit par .txt ? Attention, il ne faut pas comptabiliser un fichier nommé abctxt car il doit y avoir le caractère "point" de .txt.

2.2 Script AWK

On applique maintenant le programme AWK suivant sur ce même fichier. Qu'obtient-on comme affichage ?

```
BEGIN                    { nbf = 0 }
($9 ~ /^xmp.*\.tex$/)    { nbf++   }
END                      { print nbf " fichiers correspondent." }
```

3. Un peu de PYTHON

Pour ce qui suit, le terme intervalle désigne toujours un intervalle fermé fini, éventuellement vide. La notation **[a,b]** est associée à l'intervalle des nombres supérieurs ou égaux à **a** et inférieurs ou égaux à **b**.

3.1 - *Calculs de l'intersection de deux intervalles*

Quel est l'intersection des deux intervalles [1,5] et [2.375,3] ?

Quel est l'intersection des deux intervalles [1,5.17777] et [2,6] ?

Est-ce que l'intersection de deux intervalles est toujours un intervalle ?

3.2 - *Méthode simple pour programmer l'intersection de deux intervalles*

On décide d'implémenter les intervalles non vides par un tableau à deux éléments qui correspondent aux bornes de l'intervalle ; le premier élément du tableau est par construction toujours plus petit ou égal au deuxième élément du tableau. Un intervalle vide est implémenté par un tableau vide.

Expliquer quelle méthode vous utiliseriez pour construire l'intervalle [e,f] intersection des deux intervalles [a,b] et [c,d].

3.3 - *Script python du calcul de l'intersection de deux intervalles de nombres*

Donner le script d'une fonction inter() en PYTHON qui, à l'appel de de inter(i1,i2) renvoie l'intervalle correspondant à l'intersection des intervalles i1 et i2 selon la méthode que vous avez explicitée en 3.2.

3.4 - *Une autre implémentation*

On décide d'implémenter maintenant un intervalle par un objet avec deux attributs deb et fin correspondant aux bornes de l'intervalle. Donner le code PYTHON qui définit la classe intervalle avec les méthodes __init__, getDeb, getFin, setDeb et setFin.

3.5 - *Autre script python du calcul*

Donner le script d'une méthode inter() en PYTHON qui, à l'appel de de i1.inter(i2) renvoie l'intervalle correspondant à l'intersection des intervalles i1 et i2 selon la méthode que vous avez explicitée en 3.2.

3.6 - *Comparaison des deux implémentations*

Quelle est, selon vous, la meilleure implémentation et pourquoi ?

4. Un peu de R

Donner le script d'une fonction inter() en R qui, à l'appel de inter(i1,i2) renvoie l'intervalle correspondant à l'intersection des intervalles i1 et i2 selon la méthode que vous avez explicitée en 3.2. On implémentera les intervalles par des vecteurs de longueur 2 ou 0.

5. Intersection d'intervalles dans une page Web

On décide maintenant de fournir une page Web pour calculer des intersections d'intervalles.

Quelle interface (input, textarea...) utiliseriez-vous représenter les intervalles ?

A-t-on besoin de PHP ou peut-on se contenter de Javascript ?

Comment afficher l'intervalle vide ?

Combien de tests (et lesquels) faut-il écrire pour garantir que vous couvrez tous les cas d'intersections ?

6. Discussion culturelle

Essayez de répondre à la question suivante :

les méthodes AGILE sont-elles utilisables pour les petites applications ?

Votre réponse devra essayer de mettre en évidence votre culture naissante, votre recul et votre esprit de synthèse en matière de modélisation, de traitement de l'information et de la programmation.

Cette réponse devra faire 10 lignes au minimum, sans limite de maximum. On utilisera au moins 3 mots de 4 syllabes ou plus pour « transmettre un contenu rédactionnel fort ».

ESQUISSE DE SOLUTION

1. Un peu d'APL

Si V est le vecteur 1 4 5 5 3, alors V=1 qui compare chacun des éléments de V à 1 est le vecteur 1 0 0 0 0 puisque qu'en APL 0 correspond à FAUX et 1 à VRAI.

+/V=1 est la somme de ce vecteur, soit 1.

De la même façon, V=1 est le vecteur 0 0 1 1 0 dont la somme est 2.

Au final **(+/V=1) + (+/V=5)** vaut 1 + 2 soit 3.

Dans le cas général, l'expression renvoie le nombre de fois où on rencontre 1 ou 5 dans le vecteur V.

2. Pratique de GREP et AWK

La commande grep demandée est

```
grep -c "\.txt$" listeFic.txt
```

En effet, l'option **-c** comptabilise le nombre d'occurences trouvées, dans l'expression régulière de recherche le symbole dollar signifie "qui se termine par" et la séquence **\.** indique le caractère point.

Le script AWK affiche « *xxx fichiers correspondent.* » où *xxx* représente le nombre de fichiers dont le nom commence par **xmp** et se termine par **.tex**. En effet, avec la commande **ls -l** le neuvième mot affiché est le nom du fichier. Voici un exemple de sortie de cette commande avec les mots numérotés :

```
$1          $2  $3   $4       $5 $6    $7  $8   $9
-rw-rw-r-- 1   gh   gh       80 nov.  29  2015 packages3b.txt
-rw-rw-r-- 1   gh   gh      137 nov.  29  2015 packages3.txt
-rw-r--r-- 1   gh   gh    16345 oct.   8  2011 sen2011_1.txt
```

3. Un peu de PYTHON

Avec un peu de réflexion, la formule de l'intersection [e,f] de [a,b] et de [c,d] est définie par [e,f] = [max(a,c), min(b,d)]. Si e=max(a,c) est plus petit ou égal à f=min(b,d) alors il s'agit d'un "vrai" intervalle. Sinon, c'est l'intervalle vide.

L'intersection des deux intervalles [1,5] et [2.375,3] est donc [max(1,2.375), min(5,3)] = [2.375,3].

Celle de [1,5.17777] et [2,6] est [max(1,2), min(5.17777,6)] = [2,5.17777].

Si on admet que [x,x] est un intervalle et que l'intervalle vide, noté [ ], qui correspond à [x,y] avec y<x est aussi un intervalle, alors oui, l'intersection de deux intervalles est un intervalle.

Pour calculer l'intersection de deux intervalles [a,b] et de [c,d], une méthode simple consiste à calculer e=max(a,c) et f=min(b,d). Si e est plus grand que f, on renvoie un tableau vide, sinon on renvoie le tableau [e,f].

Sans vérification que i1 et i2 sont des tableaux, la fonction PYTHON associée est :

```
def inter(i1,i2) :

  e = max(i1[0],i2[0])
  f = min(i1[1],i2[1])
  if (e>f) :
     return([])
  else :
     return([e,f])
  # fin de si

# fin de fonction inter
```

Le code pour la classe intervalle peut être le suivant. On notera que nous avons ajouté une méthode affiche() :

```
class intervalle :

  def __init__(self) :
      self.Deb = 0
      self.Fin = 1

  def getDeb(self) :
      return(self.Deb)

  def getFin(self) :
      return(self.Fin)

  def setDeb(self,valeur) :
      self.Deb = valeur

  def setFin(self,valeur) :
      self.Fin = valeur

  def affiche(self) :
      print("[",self.Deb,",",self.Fin,"]")

# fin de classe intervalle
```

Le code pour la méthode inter n'est pas plus compliqué. On le trouve ci-dessous avec des exemples d'utilisation.

```
# définition de la méthode inter (dans la classe intervalle) :

        def inter(self,autreIntervalle) :
            e = max(self.Deb,autreIntervalle.getDeb())
            f = min(self.Fin,autreIntervalle.getFin())
            r = intervalle()
            if (e>f) :
               return(r)
            else :
               r.setDeb(e)
               r.setFin(f)
               return(r)
            # fin si

# exemple d'utilisation de la fonction inter :

    print( inter([1,5],[2,6]) )

# exemple d'utilisation de la classe intervalle et de la méthode inter :

   i1 = intervalle() ;  i1.setDeb(1) ;  i1.setFin(5)
   i2 = intervalle() ;  i2.setDeb(2) ;  i2.setFin(6)

   i3 = i1.inter(i2) ; i3.affiche()
```

4. Un peu de R

Le code R, implémentant la même méthode que PYTHON, est donc très similaire.

```
inter <- function(i1,i2) {

  e <- max(i1[1],i2[1])
  f <- min(i1[2],i2[2])
  if (e>f) {
    return( c() )
  } else {
    return( c(e,f) )
  } # fin si

} # fin de fonction inter

## exemple d'utilisation :
# print(inter( c(1,5), c(2,6) ))
```

5. Intersection d'intervalles dans une page Web

Un intervalle est défini par deux valeurs numériques, donc pour chaque intervalle, deux élements `<input type="text" />` suffisent. Le calcul de l'intersection ne requiert que le min, le max et des tests en si, donc Javascript est largement suffisant ici, pas besoin de PHP.

Afficher l'intervalle vide n'est pas simple. Ecrire "vide" disons en rouge dans chaque `input` est une solution possible.

Il faut sans doute une dizaine de tests fonctionnels et unitaires. Par exemple pour l'intervalle $[a, b]$ il faut tester que a et b sont des nombres, que b est supérieur ou égal à a. Ensuite il faut des tests pour vérifier qu'on couvre tous les cas possibles : intervalles disjoints, intervalles emboités, intervalles quelconques...