Université de Paris

L2 informatique - Année 2021–2022

TD d'Éléments d'Algorithmique n° 6

*** Les exercices marqués d'une étoile sont à faire à la maison.**

Exercice 1. *Le Terminal.*

Nous considérons les algorithmes suivants :

Algorithm 1 Algorithme PUISS

Entrée : n et k deux entiers naturels.

```
1: fonction PUISS(k, n)
2:     si n = 0 alors
3:         retourne 1
4:     sinon
5:         retourne (k · PUISS(k, n − 1))
```

Algorithm 2 Algorithme PUISSAUX

Entrée : n, k, et a trois entiers naturels.

```
1: fonction PUISSAUX(k, n, a)
2:     si n = 0 alors
3:         retourne a
4:     sinon
5:         retourne PUISSAUX(k, (n − 1), (a ∗ k))
```

1. Comment réutiliser PUISSAUX pour créer un algorithme équivalent à PUISS (on appelera cet algoritme PUISSTER) ?
2. Calculer à la main, PUISS(5,3), et PUISSTER(5,3), en suivant rigoureusement les instructions. Que constatez-vous ?
3. Prouver que, étant donné un tableau T de longueur t et $n \in \{0, \ldots, t-1\}$, l'algorithme SOMME(T, n) ci-dessous calcule la somme des éléments du sous-tableau T$[n, \ldots, t-1]$.
4. En vous inspirant de la question 1, et de PUISSAUX, adaptez l'algorithme SOMME afin de le rendre moins gourmand en mémoire.

Algorithm 3 Algorithme SOMME

Entrée : T Un tableau de taille t, n un entier.

```
1: fonction SOMME(int [] T, int n)
2:     t ← longueur de T
3:     si n ≥ t alors
4:         retourne 0
5:     sinon
6:         retourne (T[n] + SOMME(T, n + 1))
```

← cas de base

t − n = 0 l'algo retourne 0. Correct car dans ce cas n ≥ t.

Dans les exercices suivants, on utilise les classes `Liste` et `Cellule` pour les listes chaînées.

```
class Liste {
  Cellule head;
}
```

```
class Cellule {
  int key;
  Cellule next;
}
```

Exercice 2. *Bâteau.*

1. Quelle liste est stockée dans `L` après la suite d'instructions suivantes ? (Faites un dessin.)

```
1 a := new Cellule(1, nil)
2 b := new Cellule(2, a)
3 c := new Cellule(3, nil)
4 a.next := c
5 b.key := 4
6 L := new List(b)
```

2. On suppose que la liste `M` contient la suite de valeurs $(1, 2, 3, 4, 5)$. Écrivez la suite d'instructions qui mute `M` pour qu'elle contienne $(1, 2, 4, 5)$ sans créer aucune nouvelle cellule et sans jamais modifier la valeur contenue dans une cellule.
3. On suppose maintenant que la liste `N` contient la suite de valeurs $(1, 2, 4, 5)$. Écrivez la suite d'instructions qui mute `N` pour qu'elle contienne $(1, 2, 3, 4, 5)$, en créant une seule nouvelle cellule.

Exercice 3. *Manipulation de listes.*

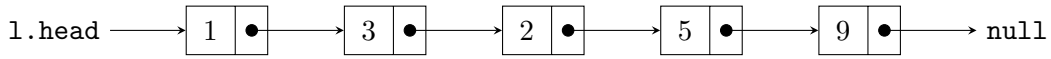
1. Écrivez un algorithme qui prend en entrée une liste `L` non vide et retourne son élément maximal.
2. Écrivez un algorithme qui prend en entrée une liste `L` et retourne 1 si `L` est triée, et 0 sinon (la liste vide est triée).
3. Écrivez un algorithme qui prend en entrée une liste `L` et retourne une liste contenant les mêmes éléments mais dans l'ordre inverse.

Exercice 4. *Fonction mystère – contrôle continu 2020.*

```
boolean auxMystere(int a, int b, boolean c){
  return  ((c and  a < b) or  ( (not c)  and  b < a));
}

boolean mystere(Cellule c, boolean val){
  if (c==null or c.next==null) return true;
  if (auxMystere(c.key,c.next.key, val))
     return mystere(c.next,  not val);
  return false;
}
```

On considère une liste chaînée `l` de type `Liste` contenant un pointeur vers la cellule de tête et dont les cellules, de type `Cellule`, contiennent dans l'ordre des clefs de valeurs 1, 3, 2, 5, 9.



1. Quelle est la valeur de la clé `l.head.next.next.key` ?
2. Lister tous les appels récursifs des fonctions `auxMystere` et `mystere` effectués lors de l'appel `mystere(l.head,true)`.
3. Quel est le résultat renvoyé par l'appel `mystere(l.head,true)` ?
4. Pour quelles listes `liste` l'appel `mystere(liste.head,true)` renvoie-t-il `true` ?
5. Pour quelles listes `liste` l'appel `mystere(liste.head,false)` renvoie-t-il `true` ?
6. L'algorithme est-il récursif terminal ?

Exercice 5. *Tri Fusion.*

1. Ecrire un algorithme `fusion` qui prend en entrée deux tableaux triés T et T' et qui renvoie la fusion triée de ces tableaux (c'est à dire le tableau trié contenant tous les éléments de T, et T').
2. ***** Implémenter un algorithme récursif de tri qui utilise la fonction `fusion`. Prouvez la correction de votre algorithme par récurrence et écrivez la preuve dans un commentaire.

Exo 1 :

1) L'algorithme PUISS(k,n) calcule k^n.

——— // ——— PUISSAUX(k,n,1) calcule n^k

fonction PUISSTER(k,n)

retourne PUISSAUX(k,n,1)

2) PUISS(5,3)

5 · PUISS(5,0) → PUISS(5,0) (1)

5 · PUISS(5,1) → PUISS(5,1) (5)

5 · PUISS(5,2) → PUISS(5,2) (25)

PUISS(5,3) (125) (= 5^3)

PUISSTER(5,3) PUISSAUX(5,0,125) (125)

PUISSAUX(5,1,25)

PUISSAUX(5,2,5)

PUISSAUX(5,3,1)

PUISSTER(5,3)

3) Par recurrence t-n.

Cas de base

- si t-n = 0 alors $n \geq t$ et l'algorithme retourne correctement 0.
- si t-n = k. Alors dans ce cas on suppose que SOMME(T,n) = $T[n]+T[n+1]+\dots+T[t-1]$

$T[n]+T[n+1]+\dots+T[t-1]$

On va prouver la correction pour t-n = k+1.

Dans ce cas, l'algorithme retourne $T[n]$ + SOMME(T, n+1)

$t-(n+1) = t-n-1 = k$

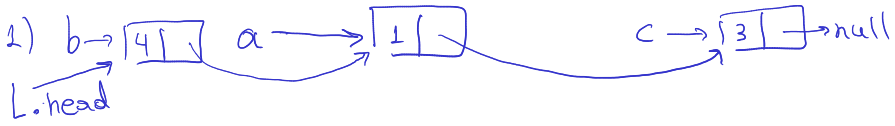
alors par recurrence : $T[n]+SOMME(T,n+1) = T[n]+\underbrace{T[n+1]+T[n+2]+\dots+T[t-1]}_{\text{recurrence.}}$

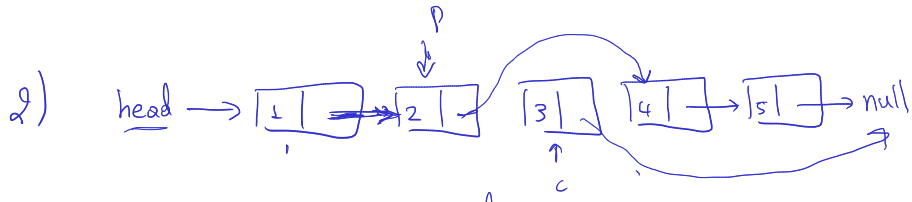
4)

```
fonction SOMMEAUX(T, n, a)
    t ← longueur de T
    si n≥t retourne a
    sinon retourne SOMMEAUX(T, n+1, T[n]+a)
```

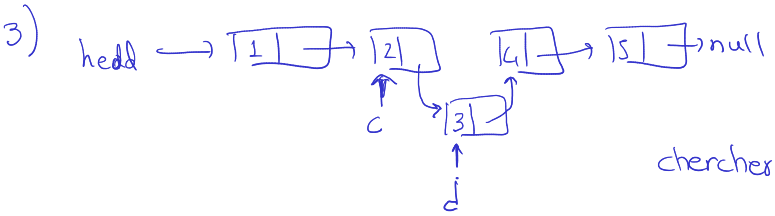
```
fonction SOMMETER(T, n)
    retourne SOMMEAUX(T, n, 0)
```

Exo 2)

1) 

2) 


```
Cellule c = head
Cellule p = head
si (head.key == 3)  head = head.next;
c = head.next;
tant que (c != null)
    si (c.key == 3)
        p.next = c.next;  c.next = null
    sinon  c = c.next;  p = p.next
```

3) 

chercher l'élément 2 et insérer l'élément 3 après.

```
Cellule c = head;
tant que (c != null)
if (c.key == 2)
    Cellule d = new Cellule(3, c.next);
    c.next = d;
```

Exo 3

1) fonction max(L)

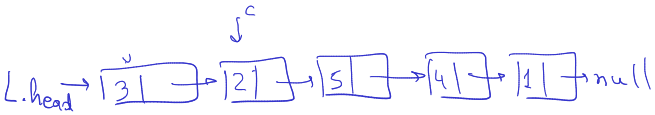
```
Cellule a = L.head;
int max = a.key; // L n'est pas vide
tant que (a.next != null)
    if (max < a.next.key)  max = a.next.key;
    a = a.next;
retourne max;
```

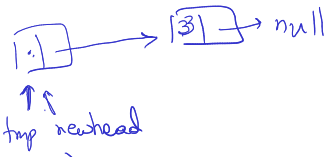
2) fonction triee (Liste L)

```
si (L.head == null) retourne 1;
Cellule a = L.head;
tant que (a.next != null)
    if (a.key > a.next.key) retourne 0;
    a = a.next;
retourne 1;
```

3) fonction inverser (Liste L)

```
Cellule newhead = new Cellule(0, null);
Cellule c = L.head;
tant que (c != null)
    newhead.key = c.key;
    Cellule tmp = new Cellule(0, newhead);
    newhead = tmp;
    c = c.next;
Liste L2 = new Liste(newhead);
```





Exo 5)

fonction fusion(T, T')

```
i=0; j=0; k=0;
R = new Tableau[T.length + T'.length];
tant que( i < T.length et j < T'.length )
    if (T[i] < T'[j])
        R[k] = T[i];
        i++;
        k++;
    else
        R[k] = T'[j];
        j++;
        k++;
tant que( j < T'.length )
    R[k] = T'[j];
    k++;
    j++;
tant que( i < T.length )
    R[k] = T[i];
    i++;
    k++;
```