Travaux Dirigés d'algorithmique n°2

Cours d'informatique de deuxième année

—DEUG Sciences S3-MI—

Preuves d'algorithmes et complexité

Nous introduisons dans ce TD la notion "d'invariant de boucle" qui permet de montrer qu'un algorithme itératif est correct. Nous parlerons également de complexité.

▸ **Exercice 1. (Révisions)** *Qu'affiche le programme suivant :*

```
#include <stdio.h>
#define MAX 2

void toto(int *n, int p)
{ *n = 3; p = 2; *n = *n + p; }

void truc(int *n, char *c)
{ *n = *n+1; *c = 'X'; }

int main(void)
{ char t[MAX]; int i;

  i = 0;
  toto(&i, i); printf("i=%d", i);
  t[0]='a'; t[1]='b'; printf("%c %c \n", t[0], t[1]);
  i = 0; truc(&i, &t[i]); printf("%c %c \n", t[0], t[1]);
}
```

▸ **Exercice 2.** *On considère l'algorithme suivant :*

fonction Mult(n, m *entiers*, $n \geq 0$) *retourne un entier*

{

  $x \leftarrow n$; $y \leftarrow m$; $res \leftarrow 0$;

  tantque $x > 0$ *faire* {

    si x *est impair alors* $res \leftarrow res + y$;

    $x \leftarrow$ *quotient*$(x, 2)$;

    $y \leftarrow 2y$;

```
    }
    retour(res);
}
```

1. *Calculer Mult$(19,25)$;*

2. *Que fait la fonction ? Faire une preuve en établissant un invariant de boucle;*

3. *Évaluer la complexité en temps;*

4. *Traduire cet algorithme en C.*

Soit t un tableau de taille n. On suppose qu'un élément peut apparaître plusieurs fois dans t et soit k le nombre d'éléments distincts de t. Supprimer les doublons dans t signifie placer les k éléments distincts de t en début du tableau t lui-même. La taille du tableau ainsi modifié doit être k. De plus, si le tableau t, de taille n est trié au départ, alors le tableau t de taille k doit l'être également après la transformation. Par exemple, si

t	10	10	10	10	15	15	15	20	30	30	$n = 10$ au départ
t	10	15	20	30	zz	zz	zz	zz	zz	zz	$k = 4$ après transformation.

où zz est une valeur quelconque.

L'ordre dans lequel apparaissent les éléments $(10, 15, 20, 30)$ après la transformation, n'a aucune importance si le tableau de départ n'est pas trié.

Le but des exercices suivants est de supprimer les doublons dans un tableau t quelconque.

▸ Exercice 3. *(doublons dans un tableau trié)*
On suppose ici, que le tableau $t[0,\dots,n-1]$ est trié.

1. *Écrire un algorithme pour résoudre le problème. (Idée : pour tout élément x apparaissant dans t, on ne garde que sa première occurrence);*

2. *Montrer que l'algorithme est correct et en évaluer sa complexité;*

3. *Le traduire en C (on supposera que t contient des entiers).*

▸ Exercice 4. *(doublons dans un tableau non trié)*
On suppose ici, que le tableau $t[0,\dots,n-1]$ n'est pas trié.

1. *Écrire un algorithme pour résoudre le problème en utilisant un tableau intermédiaire;*

2. *sans utiliser de tableau intermédiaire. (Idée : pour tout élément x apparaissant dans t, on ne garde que sa dernière occurrence) ;*

3. *Évaluer la complexité des algorithmes donnés et montrer qu'ils sont corrects ;*

4. *Les traduire en C (on supposera que t contient des entiers).*