Travaux Dirigés d'algorithmique n°1

Cours d'informatique de deuxième année

—DEUG Sciences S3-MI—

Quelques exercices d'introduction

Quelques exercices de remise en forme et introduction de la notion de complexité en temps d'un algorithme, c'est-à-dire le temps nécessaire à son exécution.

▸ Exercice 1. (Révisions)

Que font les deux programmes suivants ?

1. *1^{er} programme :*

```c
#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,j;

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

2. *2^{me} programme :*

```c
#include <stdio.h>

void bidule(int y, int *x)
{
  int p=1,i=1;

  if (y<=0) return 0;

  while (y>p)
  {
    i++;
    p=p*i;
  }
  if (y!=p) return 0;

  *x=i;
  return 1;
}
int main(void)
{
  int a,b,r;

  printf("Donner un entier:");
  scanf("%d",&a);
  b=-1;
  r=bidule(a,&b);
  printf("%d %d %d \n",b,a,r);
}
```


▸ Exercice 2. (Multiplication de deux entiers)

1. *On suppose que la touche `'*'` de votre ordinateur est en panne. Comment faire pour calculer le produit de deux entiers n et m ? Quelle est la complexité de votre méthode ?*
2. *Multiplication égyptienne*
   Les anciens Egyptiens savaient additionner deux entiers, mutiplier et diviser un entier par 2. Pour multiplier deux entiers, ils utilisaient l'algorithme suivant :

```
fonction Mult(n,m entiers, n ≥ 0) retourne un entier
{
   x ← n ; y ← m ; r ← 0 ;
   tantque x > 0 faire {
      si x est impair alors r ← r + y ;
      x ← quotient(x, 2) ;
      y ← 2y ;
   }
   retour(r) ;
}
```

   (a) *Décomposer le calcul pour $n = 19$ et $m = 25$.*

   (b) *évaluer la complexité en temps de cet algorithme.*

   (c) *Le traduire en C. Que faut-il modifier pour avoir le produit de deux entiers quelconques ?*

▸ Exercice 3. *(Rotation dans un tableau)*

Le but de cet exercice est d'effectuer, dans un tableau de n caractères, la rotation à gauche de k positions ($0 < k < n$).

Par exemple, pour $n = 8$ et $k = 3$, le tableau [A|B|C|D|E|F|G|H] *devient, après rotation,* [D|E|F|G|H|A|B|C]

Pour celà on propose trois solutions :

1. *Utiliser un tableau auxiliaire.*
2. *Ecrire tout d'abord une fonction qui effectue une rotation à gauche d'une seule position puis l'utiliser pour résoudre le problème.*
3. *Ecrire une fonction qui inverse les éléments de la portion de tableau délimitée par deux indices i et j (par exemple si $i = 2$ et $j = 5$, la fonction transforme le tableau précédent en* [A|B|F|E|D|C|G|H]*), pui utiliser cette fonction pour résoudre le problème.*