

Corrigé du TP Informatique 13

Exercice 1

```
1. def est_libre(T,d) :  
    for i in range(len(T)) :  
        if T[i]<=d :  
            return i  
    return -1
```

```
2. def allocation(I) :  
    S=[]  
    T=[]  
    for k in range(len(I)) :  
        i=est_libre(T,I[k][0])  
        if i>=0 :  
            S[i].append(I[k])  
            T[i]=I[k][1]  
        else :  
            S.append([I[k]])  
            T.append(I[k][1])  
    return S
```

Exercice 2

```
1. def fraction1(a,b) :  
    x,res,k=a/b, [], 1  
    while x!=0 :  
        while x<1/k :  
            k+=1  
        x=x-1/k  
        res.append("+1/"+str(k))  
    return res
```

La fonction renvoie sans problème les deux premiers résultats demandés, mais pas le troisième : en effet, $2/5$ n'est pas un nombre dyadique, sa représentation en mémoire conduit à une légère erreur, qui se répercute lors des itérations, et empêche le programme de terminer.

2. On esquivé le problème en représentant x sous la forme d'un couple (p, q) où $x = \frac{p}{q}$. Les calculs sont font en nombres entiers en observant que $x - \frac{1}{k} = \frac{pk-q}{qk}$

```
def fraction2(a,b) :  
    p,q,res,k=a,b, [], 1  
    while p!=0 :  
        while p*k-q<0 :
```

```
        k+=1
        p,q=p*k-q,q*k
        res.append("+1/"+str(k))
    return res
```

En exécutant le programme avec les exemples précédents, les résultats s'obtiennent rapidement. Mais on rencontre une nouvelle difficulté : le dénominateur des fractions calculées grandit très rapidement, ce qui pose d'importants problèmes de complexité. Tester par exemple la fonction avec $x = \frac{5}{31}$.

Il existe des améliorations (non gloutonnes) de cet algorithme, et il est possible aussi de l'adapter pour déterminer le développement d'un rationnel ≥ 1 en somme de fractions égyptiennes.