1ère Année ENSEIRB

Algorithmique et structure de données –

Corrigé du TD 1 : Algorithmes simples

1 Rendre la monnaie

On se propose d'écrire un algorithme permettant d'obtenir la suite des billets totalisant une somme donnée (dont on suppose qu'elle est un multiple de 10*). Les espèces disponibles sont des billets de 50, 20 et 10 euros Le principe est de donner le billet de valeur la plus grande possible inférieure ou égale à la somme à rendre et de poursuivre la même stratégie avec la somme restante jusqu'à ce que la somme restnate soit nulle.*

1. *Écrire cet algorithme en utilisant les structures de contrôle suivantes :* `while` *,* `if` *et* `else if` *.*

   On suppose que la somme donnée est un multiple de 10, cette somme sera notée `s` dans l'algorithme, les nombres de billets à rendre seront calculés à l'aide de 3 variables `n1, n2, n3` que l'on incrémentera à chaque tour de boucle en même temps que la somme sera diminuée de la valeur du billet trouvée.

```
while (s != 0) {
     if (s >= 50) {
        s = s - 50;
        n1++;
     }
     else if (s >= 20 ){
        s = s -20;
        n2++;
     }
     else {
        s = s - 10;
        n3++;
     }
}
Afficher (n1, "billets de 50 euros);
Afficher (n2, "billets de 20 euros);
Afficher (n3, "billets de 10 euros);
```

2. *Déterminer le nombre de soustractions effectuées par l' algorithme, en utilisant des divisions entières par les nombres 50, 20 et 10.*

Soit q_1 le quotient entier de s par 50, $q2$ celui de $s - 50q_1$ par 20, et q_3 la valeur de $\frac{s-50q_1-20q_2}{10}$ le nombre de soustractions effectuées est $q_1 + q_2 + q_3$ qui inférieur ou égal à

$$\frac{s}{50} + 3$$

3. *Généraliser votre algorithme au cas où les valeurs de billets ou pièces disponibles sont en nombre quelconque et figurent dans un tableau à valeurs décroissantes* `val[0], val[1], ..., val[k-1]`. *Ainsi dans l'exemple considéré plus haut on aurait :* `k = 3` *et* `val[0] = 50, val[1] = 20, val[2] = 10`.

   Il faut à chaque tour de boucle retrouver la valeur de billet la plus grande inférieure à `s`.

```
k = 0;
while (s > 0) {
   while (val[k] > s) k++;
   tab[k]++;
   s = s- val[k];
}
Afficher(tab);
```

4. *Montrer qu' il existe des valeurs de billets et une somme à rendre pour lesquels cet algorithme ne donne pas le nombre minimum de billets ou pièces à rendre.*

   Il y a en effet des cas où cet algorithme ne donne pas le plus petit nombre de billets à rendre, par exemple si on dispose de billets de 50, 20, 12, 1, et que la somme à rendre est de 24, l'algorithme donnera 5 billets à rendre (1 de 20 et 4 de 1) alors que l'on peut faire avec 2 billets de 12.

   Cette situation ne se produit pas si chaque valeur de billet est supérieure ou égale au double de la valeur du billet immédiatement inférieure.

2 Calcul de la puissance nième

1. *Écrire un algorithme permettant de calculer rapidement la puissance nième d'un nombre entier en s'inspirant de l'algorithme vu en cours pour la représentation d'un nombre en base 2.*

   On suppose que l'on veut élever `a` à la puissance n, n s'écrit en base 2 par;

$$n = \sum_{k=1}^{p} x_k 2^k$$

   où les x_k ont pour valeur 0 ou 1.

   On a alors

$$a^n = \prod_{k=1}^{p} (a^{2^k})^{x_k}$$

   Ainsi on calcule les puissances a^{2^k} de a en effectuant les carrés successifs des valeurs trouvées et on multiplie le résultat partiel par la valeur obtenue si et seulement si $x_k = 1$. Soit:

```
res =  1;
while (n > 0){
    if (n%2 != 0) res = res*a;
    a = a*a;
    n = n/2;
}
Affiche(res);
```

2. *Donner l'ordre de grandeur du nombre d'opérations à effectuer en fonction de n.*

   Le nombre de multiplications effectueés est au plus $2p$ où p est le plus petit entier tel que $2^p \geq n$, (c'est ce qu'on appelle le logarithme en base 2 de n).

3 Anagrammes

On vous propose maintenant d'écrire un algorithme permettant de vérifier si deux mots (représentés sous la forme de tableaux de caractères) sont l'anagramme l'un de l'autre. C'est à dire contiennent exactement les mêmes caractères mais dans un ordre différent. Par exemple Marion *est anagramme de* Romain, arbre *est anagramme de* barre.

```
for (i = 0; i < n; ++i){
      a = f[i]; b = g[i];
      nbf[a]++; nbg[b]++;
}
resultat = true;
for (j = 0; j < 26; ++j)
    if (nbf[j] != nbg[j])
       resultat = false;
if (resultat)
    Afficher("Sont  anagrammes");
else
    Afficher("Ne sont pas anagrammes");
```