TD1.2 – Écriture et analyse d'algorithmes de tri

Objectifs

À la fin de cette séance, vous devriez être capable de :
- illustrer le fonctionnement d'algorithmes par des schémas ;
- écrire en langage algorithmique des algorithmes simples décrits en langue naturelle ;
- analyser le coût d'algorithmes dans des cas simples ;

Exercice 1 : Tri de Shell

On rappelle ci-dessous l'algorithme de tri par insertion (on suppose ici le tableau **indexé de 1 à** n). Ce tri insère successivement chaque élément du tableau dans la séquence formée par les éléments d'indices inférieurs. Pour cela, on échange l'élément à placer avec celui qui le précède jusqu'à ce qu'il soit à sa place.

```
TRI_INSERTION(tableau T)
for i := 2 to n do
    x := T[i]
    j := i − 1
    while j > 0 et puis T[j] > x do
        T[j + 1] := T[j]
        j := j − 1
    T[j + 1] := x
```

On s'intéresse à une « variante » de ce tri proposée par Donald Shell en 1959 : celle-ci consiste à choisir un entier k et à appliquer dans une première passe l'algorithme du tri par insertion sur chacun des k sous-tableaux T_i constitués des termes d'indices de la forme $i + j \times k$ (pour j entier quelconque). Autrement dit, le sous-tableau T_i commence à l'indice i et contient des éléments séparés d'un pas de k positions.

Par exemple, pour un tableau de 12 éléments avec $k = 5$, le tableau T_1 contient les termes d'indices $1, 6, 11$; T_2 ceux d'indices $2, 7, 12$; T_3 ceux d'indices $3, 8$; et ainsi de suite.

Une fois le tri par insertion réalisé sur tous les tableaux T_i, on décrémente le pas k et on recommence le processus, et ce jusqu'à $k = 1$.

1. Appliquer l'algorithme de tri de Shell sur le tableau $T = [G; C; A; B; E; F; D]$ avec un pas initial $k = 3$. Donner le contenu du tableau T après chaque étape de tri sur chaque sous-tableau T_i et pour toutes les valeurs de k de 3 à 1.
2. Représenter sous forme de schémas le déroulement du tri de Shell.
3. Écrire en détails l'algorithme $triShell(T, n, k)$ où T est le tableau à trier, n est le nombre d'élément de ce tableau, et k la valeur initiale du pas.
4. Pourquoi le tri de Shell réussit-il forcément à trier le tableau donné en entrée ?
   Que se passe-t-il si on ne prend plus comme valeurs de pas $k, k-1, k-2, \ldots, 2, 1$ mais d'autres suites comme par exemple $7, 5, 3, 1$ ou encore $8, 4, 2, 1$?
5. Déterminer la complexité du tri de Shell en nombre de comparaisons entre éléments du tableau :
   — lorsque les pas sont $k, k-1, k-2, \ldots, 2, 1$
   — lorsque les pas sont $n/2, n/4, \ldots, 8, 4, 2, 1$
6. **(à faire seulement si vous êtes en avance)**
   Comparer la complexité calculée avec celle du tri par insertion.
   Expérimentalement, on constate pourtant que le tri de Shell est plus efficace (et on peut le prouver, dans certains cas).
   Essayer le tri de Shell sur un tableau plus grand (15 à 20 éléments) et avec la suite de pas $9, 5, 3, 1$.
   Que constate-t-on lors des deux dernières passes ?

Exercice 2 : Un programme ...

Data : Un tableau T indicé de 1 à n dont les éléments sont compris entre 1 et n
Result : à déterminer
S : un tableau d'entiers indicé de 1 à n
for $i = 1 \dots n$ **do**
  $S[i] := 0$
for $i = 1 \dots n$ **do**
  $S[T[i]] := S[T[i]] + 1$
$i := 1$
$j := 1$
while $i \leq n$ *et* $j \leq n$ **do**
  $k = 0$
  while $k < S[j]$ **do**
    $T[i] := j$
    $i := i + 1$
    $k := k + 1$
  $j := j + 1$

1. En vous aidant de schémas, déterminer ce que fait ce programme.
2. Quel est son coût en fonction de n ?

Membres du groupe

—
—
—
—

Exercice 1 : Écriture du tri de Shell

2. Représenter sous forme de schémas le déroulement du tri de Shell.
3. Écrire en détails l'algorithme $triShell(T, n, k)$ où T est le tableau à trier, n est le nombre d'élément de ce tableau, et k la valeur initiale du pas.