TD12 - Tri

Exercice 1 - Algorithmes de tri

1. Écrire l'algorithme de tri par sélection et donner sa complexité dans le pire des cas.
2. Écrire l'algorithme de tri à bulle et donner sa complexité dans le meilleur et dans le pire des cas.
3. Écrire l'algorithme de tri par insertion et donner sa complexité moyenne et dans le pire des cas.

Exercice 2 - Algorithme de tri fusion

Le principe de l'algorithme de tri fusion est de scinder le tableau initial T de n éléments en deux sous-tableaux ayant respectivement $\lceil \frac{n}{2} \rceil$ et $\lfloor \frac{n}{2} \rfloor$ éléments. Le tri de chacun de ces sous-tableaux pris séparément suivi de l'interclassement des deux sous-tableaux triés fournit le tableau trié résultant.

1. Écrire l'algorithme *Tri_Fusion(T : tableau ; i,j : entier)* qui trie le tableau T depuis l'indice i jusqu'à l'indice j. On suppose qu'on dispose de l'algorithme *Interclassement(T : tableau ; i, j, k : entier)*.
2. Écrire l'algorithme *Interclassement(T : tableau ; i, j, k : entier)*. On utilisera un tableau intermédiaire R pour réaliser l'interclassement. Les valeurs de $T[i \cdots k]$ seront copiées en $R[i \cdots k]$ et les valeurs de $T[k+1 \cdots j]$ seront inversées et copiées en $R[k+1 \cdots j]$.
3. Déterminer la complexité de ces deux algorithmes.

Exercice 3 - Algorithme de tri par champs

On considère un ensemble $\{O_1, \ldots, O_n\}$ de n objets placés initialement dans un tableau $T[1 \cdots n]$ tel que $T[i] = O_i$. Chaque objet O_i possède un numéro i et une clé $c_i = (c_i^1, \cdots, c_i^g)$ où $c_i^p \in [0; K-1], K > 0, g > 0$. On définit la relation d'ordre $<_1$ de la manière suivante : $O_i <_1 O_j \iff \exists p \in [1; g],\ c_i^1 = c_j^1, \cdots, c_i^p < c_j^p$. On propose l'algorithme *Tri_par_Champs(T : tableau ; n, g, K : entier)* qui ordonne un tableau de valeurs T suivant la relation $<_1$. Cet algorithme utilise K files $F_0, \cdots, F_{K-1}$ initialement vides. $T[i].num$ est le numéro de l'objet placé dans $T[i]$.

```
Procedure Tri_par_Champs(T : tableau ; n, g, K : entier)
  r, i, k : entier
  Pour r := g..1 par pas de -1 faire
    Pour i := 1..n faire
      enfiler(T[i], F_c^r_{T[i].num})
    FinPour
    i := 1
    Pour j := 0..K-1 faire
      TantQue Est_Vide(F_j) = faux faire
        T[i] := Defiler(F_j)
        i := i +1
      FinTantQue
    FinPour
  FinPour
```

1. Exécuter l'algorithme *Tri_par_Champs* sur l'exemple suivant : $n = 10, g = 3, K = 10, T = [O_1, \cdots, O_{10}], c_1 = (2,1,4), c_2 = (2,9,7), c_3 = (0,3,3), c_4 = (2,9,8), c_5 = (2,1,4), c_6 = (2,0,4), c_7 = (9,9,9), c_8 = (2,2,2), c_9 = (1,3,5), c_{10} = (9,9,9)$.
2. Démontrer que cet algorithme permet de trier les n objets dans T suivant la relation $<_1$.
3. Déterminer la complexité de cet algorithme.