Tri fusion et Tri rapide : récursivité

Tri Fusion :

Principe : Le tri fusion utilise la stratégie dite « diviser pour régner ».

- Etape1 : On divise le tableau en deux sous-tableaux.
- Etape2 : On trie (par récursivité) chaque sous-tableau.
- Etape3 : On fusionne les deux sous-tableaux, pour obtenir un tableau complet trié.

Désignons par a et b (avec a≤b) les indices, initialement à 0 et n-1, indiquant le premier élément de T (ou d'un sous-tableau de T) et le dernier élément.

L'implémentation de ces 3 étapes donne l'algorithme suivant :

```
Algorithme : tri_fusion(T,a,b)
    si a<b alors
        m=[(a+b)/2]
        afficher("appel tri_fusion:","a=",a,"b=",m,T)
        TriFusion(T,a,m)
        afficher("appel tri_fusion:","a=",m+1,"b=",b,T)
        TriFusion(T,m+1,b)
        Fusion(T,a,b,m)                                          ;
        afficher("fusion:","a=",a,"m=",m,"b=",b,T)
    fin
```

a- Exécutez l'algorithme ci-dessus avec T=[3,1,0,9,1,2,6,8,3,4,7],

b – Écrire en C++ la fonction fusion [void fusion (int T[],int a,int b,int m)] qui fusionne les éléments partant de l'indice a jusqu'à l'indice m (supposés triés entre eux) avec les éléments partant de l'indice m+1 à l'indice b (supposés triés entre eux).

c- Écrire en C++ la fonction récursive permettant de réaliser le tri_fusion d'un tableau.

```
void  tri_fusion(int T[],int a,int b) ;
```

Tri rapide:
Principe : Soit T un tableau à trier. On se donne un élément quelconque du tableau (en général le premier `a gauche) nommé pivot. On effectue une partition du tableau consistant à réorganiser les éléments de T de la façon suivante : T[0...m − 1], T[m], T[m + 1...n-1] T[m] contient le pivot choisi. T[0...m − 1] contient les éléments de T inférieurs à T[m] et T[m + 1...n-1] ceux supérieurs. Notez que les valeurs dans T[0...m−1] et T[m+1...n-1] ne sont pas nécessairement triées. Pour les trier on réitère récursivement ce qui a été fait sur T sur chacun de ces sous-tableaux. L'algorithme est alors le suivant :

```
Algorithme : TriRapide(T,a,b)
    si a < b alors
        afficher("avant appel partitionnement",a,b,T)
        m=Partitionnement(T,a,b)
        afficher("apres appel partitionnement",a,b,m,T)
        afficher("appel tri_rapide à gauche",a,m-1)
        TriRapide(T,a,m-1)
        afficher("appel tri_rapide à droite ",m+1,b)
        TriRapide(T,m+1,b)
    fin
```

Un algorithme de partitionnement possible est le suivant : On considère comme pivot le premier élément du tableau en cours de traitement, T[a]. On utilise deux compteurs l et k initialisés aux deux extrémités (gauche pour l, et droite pour k) du tableau. Tant que T[l] ≤ T[a], l est incrémenté. Inversement tant que T[k] > T[a], k est décrémenté. On échange alors T[l] et T[k] puis on continue de façon analogue jusqu'à ce que les indices l et k se croisent.

a- Exécutez l'algorithme pour T=[3,5,8,9,1,5,2,2,10]

b- Écrire en C++ la fonction partitionnement(int T[],int a,int b):
c- Écrire en C++ la fonction tri_rapide(int T[], int a, int b)