Problème de tri

Trier un tableau: ordonner ses éléments par ordre croissant ou ordre décroissant On suppose que le tableau contient des valeurs pouvant avoir n'importe quel type pour lequel est définie une relation d'ordre

Tri par sélection

Idée : sélectionner le minimum et le déplacer au début du tableau T (de taille N).

```
void tri_selection (int tab[],int n)
{
    int i, temp, min;
    for (i=0; i<n ;i++)
    {
        min = recherche_min (tab,i,n);
        if(tab[min]<tab[i])
        {
            temp = tab[min];
            tab[min] = tab[i];
            tab[i] = temp;
        }
    }

}
int recherche_min(int tab[],int rang, int n)
{
    int i,min;
    for (min=rang+1,i=rang+2;i<n;i++)
    if (tab[i]< tab[min])
    min=i;
    return min;

}
```

tri rapide

Le principe de cet algorithme est de diviser un ensemble d'éléments en deux parties. Pour faire cette séparation, une valeur pivot est choisie. Les valeurs sont séparées par le pivot suivant qu'elles lui soit supérieures ou inférieures. Ensuite, on fait la même chose pour les deux sous ensembles. Cet algorithme est donc récursif. Le choix du pivot est un problème majeur, le mieux serait de pouvoir séparer les ensembles en deux parties égales. Toutefois une recherche s'avererait trop coûteuse. C'est pour cela qu'on choisit le premier élément ou le dernier élément.

```
int partition(int tableau[], const int debut, const int fin)
{
   int compteur = debut;
   int pivot = tableau[debut];
```

```
    int i;

    for(i=debut+1; i<=fin; i++)
    {
        if(tableau[i]<pivot) // si élément inférieur au pivot
        {
            compteur++; // incrémente compteur cad la place finale du pivot
            echanger(tableau, compteur, i); // élément positionné
        }
    echanger(tableau, compteur, debut); // le pivot est placé
    return compteur; // et sa position est retournée
}

void triRapideAux(int tableau[], const int debut, const int fin)
{
    if(debut<fin) // cas d'arrêt pour la récursivité
    {
        int pivot =partition(tableau, debut, fin); // division du tableau
        triRapideAux(tableau, debut, pivot-1); // trie partie1
        triRapideAux(tableau, pivot+1, fin); // trie partie2
    }
}

void triRapide(int tableau[], const int longueur)
{
    triRapideAux(tableau, 0, longueur-1);
}
```

Tri par échange (Tri à Bulles)

Idée : L'idée est de faire remonter les grands éléments à la fin du tableau. Comparer toute paire d'éléments contigus et les permuter s'ils ne sont pas dans le bon ordre. Répéter le processus jusqu'à ce qu'il n'y ait plus de permutations.

```
do{
      permute = 0 ;
      for ( i = 0 ; i < N-1 ; i = i+1)
if ( T[i] > T[i+1] )
      {
            X = T[i] ; T[i] = T[i+1] ; T[i+1] = X ;
            permute = 1;
      }
} while ( permute) ;
```