Analyse de la procédure tri_shaker:

S	DEF PROC tri_shaker(VAR T:tab, n:entier) L.D.E	O.U
1	**Résultat=[m=1] Répéter** [Echange← faux] **Pour** i de m à n-1 **faire** [] **Si** (T[i] > T[i+1])**Alors** Permute(T[i], T[i+1]) Echange ← vrai **FinSi** **FinPour** **n← n-1** **Pour** j de n à m+1 (pas=-1) **faire** [] **Si** (T[j] < T[j-1])**Alors** Permute(T[j], T[j-1]) Echange ← vrai **FinSi** **FinPour** **m← m+1**	Echange m i j
2	**Jusqu'à (**Echange = Faux) ou (n<=m) **FIN Tri_shaker**	

Tableau de déclaration des objets: T.D.O. Locaux :

Objet	Type/Nature
Echange	Booléen
m, i,j	Entier

En Pascal:

```
procedure tri_shaker(var t:tab; n:integer);
  var i,j,m:integer; echange :boolean ;
begin
   m:=1;
 repeat
      echange :=false ;
    for i :=m to n-1 do
        begin
         if ( T[i] > T[i+1]) then
                                 begin
                     Permute(T[i], T[i+1]) ;
                           Echange :=true;
                                 End ;
       End ;       n :=n-1 ;
   for j :=n to m+1 do
       begin
         if ( T[j] < T[j-1]) then
                                 begin
                     Permute(T[j], T[j-1]) ;
                           Echange :=true;
                                 End ;
       End ;       m :=m+1 ;
  Until  (echange = false) or (n<=m) ;
End ;
```

Tri à Bulles bidirectionnel(cocktail shaker)

Le tri bidirectionnel ou cocktail shaker est une variante de l'algorithme du tri à bulles.

Il consiste à parcourir le tableau de gauche à droite , puis de droite à gauche, le changement de direction ayant lieu chaque fois que l'une des extrémités est atteinte.(les plus petits éléments du tableau descendent au même rythme que remonte les plus grands éléments.

Tri par insertion (utilisant la dichotomie):

Optimisation de la recherche du point d'insertion

La recherche du point d'insertion k peut se faire séquentiellement ; mais on peut employer une recherche dichotomique, qui est plus efficace.

➔La procédure tri insertion appelle la procédure inserer_trie(T,i) pour insérer l' élément T[i] dans sa position dans la partie triée entre 1 et i-1.

➔Dans la procédure insere_trie :

- On met l'élément à insérer dans x puis on recherche la position d'insertion (k)de x dans le tableau T entre la position 1 et i par la fonction posit_ins
- Puis on décale les valeurs de la position k à i-1 d'un pas à droite (`for j := i downto k+1 do T[j] := T[j-1];`)
- En fin on insère x dans sa position (T[k] :=x)

➔Dans la procédure posit_ins on va trouver la position d'insertion de x dans T entre la position 1 et i:

- On commence par traiter le cas des extrémités (1 et i)
- Puis on initialise les deux bornes inf et sup respectivement par 2 et i-1 (par ce qu'on a déjà traité le cas des extrémités)
- Et on recherche la position de m tq T[m-1]<=x<T[m] par une variante de dichotomie sans utiliser une variable booléenne.

```
FUNCTION posit_ins(var T : Tab;i:integer;
                   x : type_element) : integer;
   VAR inf, sup, m : integer;
   BEGIN

   if x < T[1] then posit_ins := 1
   else if T[i-1] <= x then posit_ins := i
   else
begin
  inf := 2; sup := i-1;

       while inf < sup do
          begin
          m := (inf + sup) div 2;
        if T[m] <= x
      then inf := m+1
      else sup := m;
          end;
  posit_ins := sup;
end;
   END;
```

```
0)Def proc insere(var t:tab, i:entier, x:entier)
1) i=1
   tant que (t[i]<=x) et (i<=n) faire
    i ←i+1
    fin tanque
si t[i]>=x alors    pour j de n+1 à i+1 faire
                     t[j] ←T[j-1]
                    finPour
finsi
    T[i]←x
2)Fin insère
```

```
PROCEDURE inserer_trie(var T : Tab; i:integer);
   VAR j, k : integer;
   x : type_element;
   BEGIN
   { élément à insérer }
   x := T[i];
   { recherche position d'insertion de x }
   k := posit_ins (T, i, x);
   { décalage : en descendant }
   for j := i downto k+1 do T[j] := T[j-1];
   { insertion }
   T[k] := x;
   END;


PROCEDURE tri_insertion(var T : Tab;n:integer);
   VAR i : integer;
   BEGIN
   for i := 2 to n do
   inserer_trie (T, i);
   END;
```