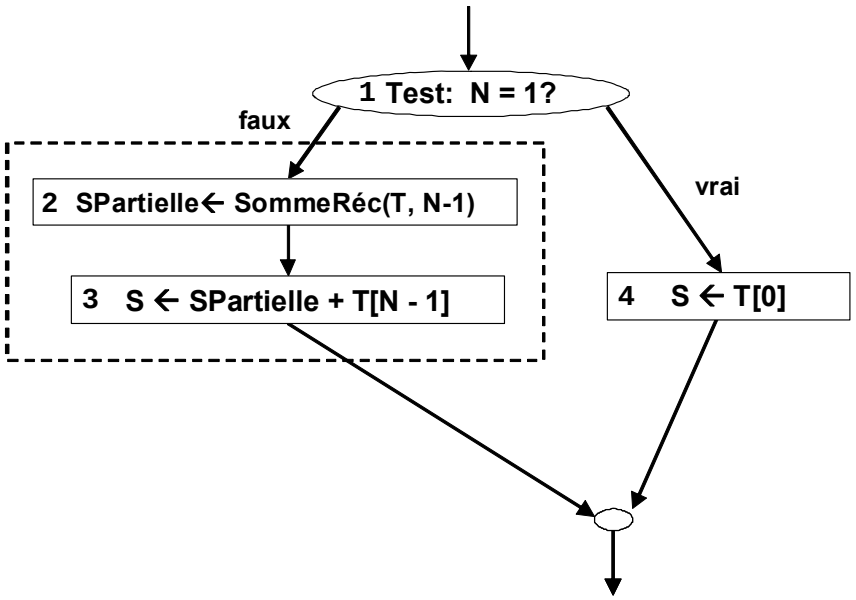
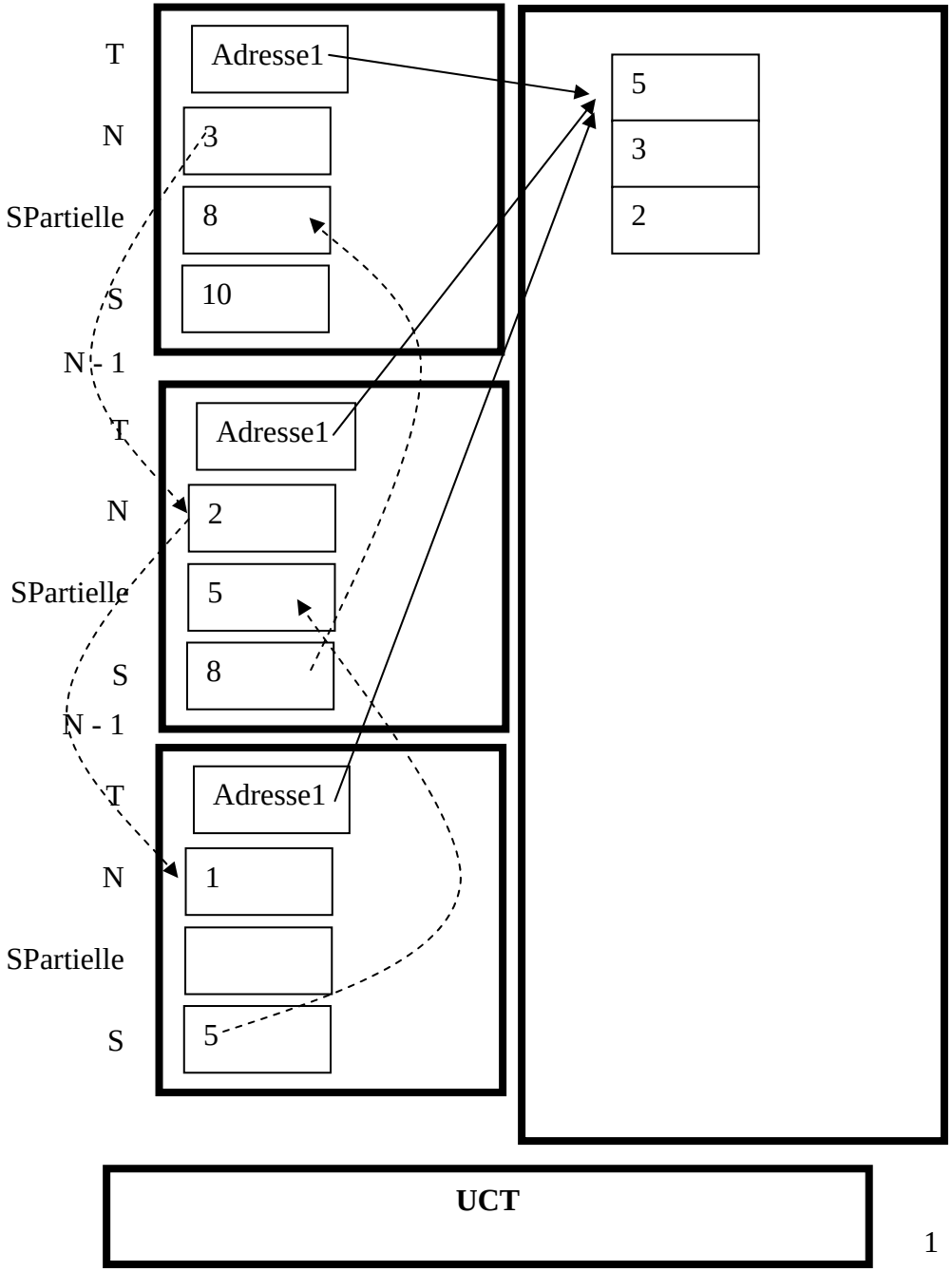
Mémoire de programme **Exercice 8-1 – Algorithme pour somme récursive** **Mémoire de travail** **Mémoire globale**

Trace call SommeRéc(T, 3)

DONNÉES: T (référence à un tableau d'entiers)
N (nombre d'éléments à additionner dans le tableau)
RÉSULTAT: S (somme des N éléments du tableau)
INTERMÉDIAIRES:
SPartielle (somme partielle des N-1 éléments)
EN-TÊTE:
S ← SommeRéc(T, N)





Exercice 8-1 – Trace, Table 1 – SommeRéc(T, 3)

Instructions	Réf. par T	N	S	Somme
Valeurs initiales	{ 5, 3, 2 }	3	?	?
1. test: N = 1? faux				
2. S ← SommeRéc(T,N-1) voir Table 2			8	
3. Somme ← S + T[N-1]				10

S ← SommeRéc(T,N-1)

8 {5,3,2} 2

Somme ← SommeRéc(T,N)

Exercice 8-1 – Trace, Table 2 – SommeRéc(T, 2)

Instructions	Réf. par T	N	S	Somme
Valeurs initiales	{ 5, 3, 2 }	2	?	?
1. test: N = 1? faux				
2. S ← SommeRéc(T,N-1) voir Table 3			5	
3. Somme ← S + T[N-1]				8

S ← SommeRéc(T,N-1)

5 {5,3,2} 1

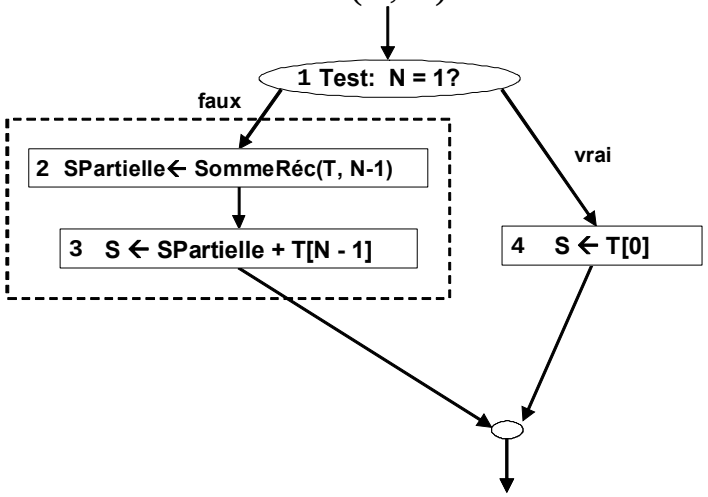
Somme ← SommeRéc(T,N)

Exercice 8-1 – Trace, Table 3 – SommeRéc(T, 1)

Instructions	Réf. par T	N	S	Somme
Valeurs initiales	{ 5, 3, 2 }	1	?	?
1. test: N = 1? vrai				
4. Somme ← T[0]				5

Mémoire de programme **Exercice 8-2 – Traduction de SommeRec en Java** **Mémoire de travail** **Mémoire globale**

DONNÉES: T (référence à un tableau d'entiers)
N (nombres d'éléments à additionner dans le tableau)
RÉSULTAT: S (somme des N elements)
INTERMÉDIAIRES:
SPartielle (somme de N-1 éléments dans le tableau)
EN-TÊTE:
S ← SommeRéc(X, N)



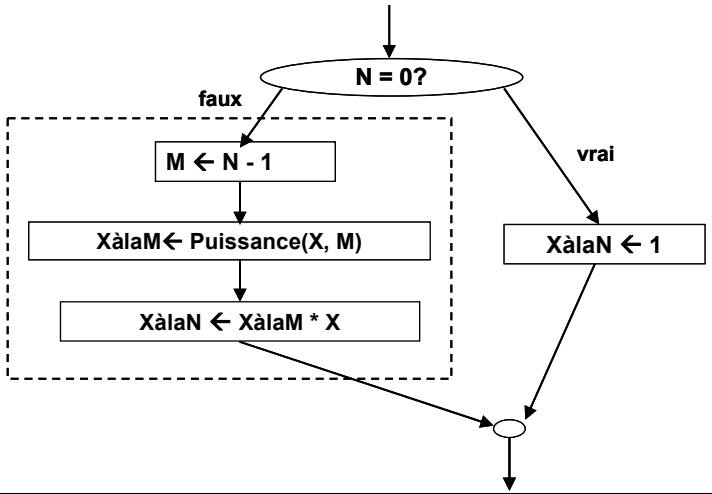

```
public static int sommeRec(int [] t,
                           int n)
{
   int sPartielle; // INTERMÉDIAIRE
   int s; // RÉSULTAT
   if (n == 1)
   {
      s = t[0];
   }
   else
   {
      sPartielle = sommeRec(t, n - 1);
      s = sPartielle  + t[n - 1];
   }
   return s;
}
```

UCT

Modèle algorithmique	Java

Exercice 8-3 (a) - Trouver X^N où X et N sont des entiers et $N \geq 0$, $X \geq 1$. (Version 1)

DONNÉES: X (base)
N (entier, puissance)
RÉSULTAT: XàlaN (X à la puissance N)
INTERMÉDIAIRES:
M (vaut N-1; plus petit!)
XàlaM (résultat partiel)
EN-TÊTE: XàlaN ← Puissance (X,N)
MODULE:

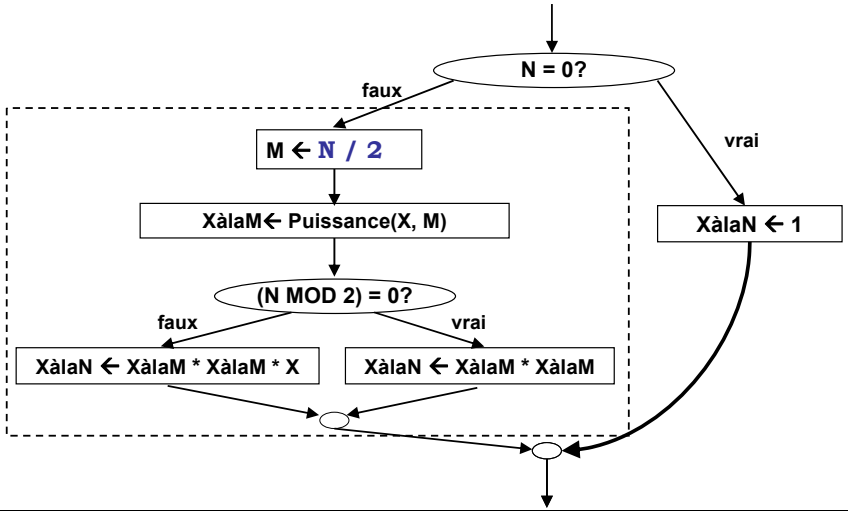



```
// Méthode puissance: trouver X à la puissance N
public static int puissance(int x, int n)
{
   // DÉCLARATION DE VARIABLES
      int m;      // INTERMÉDIAIRE: valeur réduite
      int xÀlaM;  // INTERMÉDIAIRE: résultat partiel
      int xÀlaN;  // RÉSULTAT: résultat recherché

   // MODULE DE L'ALGORITHME
   if (n == 0)
   {
      xÀlaN = 1;
   }
   else
   {
      m = n-1;
      xÀlaM = puissance(x, m);
      xÀlaN = xÀlaM * x;
   }

   // RETOURNE RÉSULTAT
   return xÀlaN;
}
```

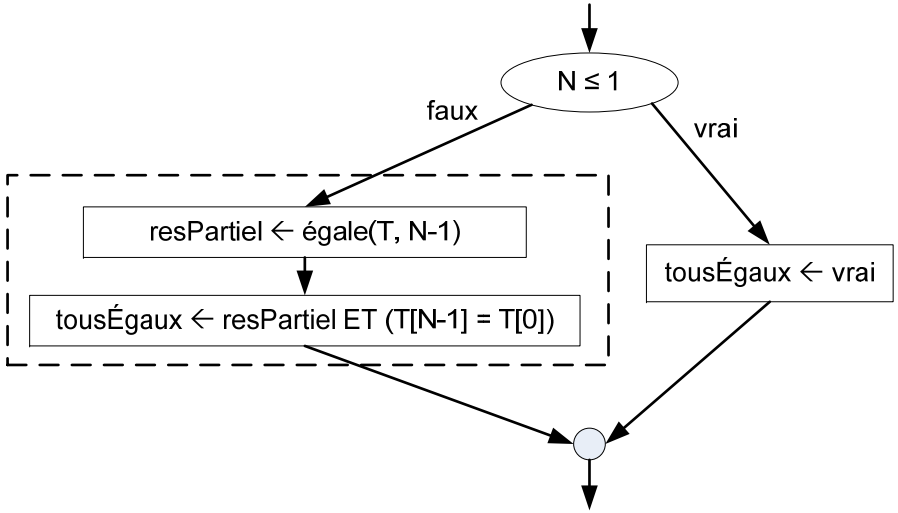
Modèle algorithmique	Java
Exercice 8-3 (b) - Trouver X^N où X et N sont des entiers et $N \geq 0$, $X \geq 1$. (Version 2 – efficace)	

DONNÉES: X (base)
N (entier, puissance)
RÉSULTAT: XàlaN (X à la puissance de N)
INTERMÉDIAIRES:
M (vaut N-1; plus petit!)
XàlaM (résultat partiel)
EN-TÊTE: XàlaN ← Puissance (X,N)
MODULE:
 
 	
	`// MÉTHODE puissance: trouver X à la puissance N`
	`public static int puissance(int x, int n)`
	`{`
	`  // DÉCLARATION DE VARIABLES`
	`  int m;  // INTERMÉDIAIRE: valeur réduite`
	`  int xÀlaM;  // INTERMÉDIAIRE: résultat partiel`
	`  int xÀlaN;  // RÉSULTAT: résultat recherché`
	`  // MODULE D'ALGORITHME`
	`  if (n == 0)`
	`  {`
	`  xÀlaN = 1;`
	`  }`
	`  else`
	`  {`
	`  m = n / 2;`
	`  xÀlaM = puissance(x, m);`
	`  if ( n%2 == 0)`
	`  {`
	`  xÀlaN = xÀlaM * xÀlaM;`
	`  }`
	`  else`
	`  {`
	`  xÀlaN = xÀlaM * xÀlaM * x;`
	`  }`
	`  }`
	`  // RETOURNE RÉSULTAT`
	`  return xÀlaN;`
	`}`

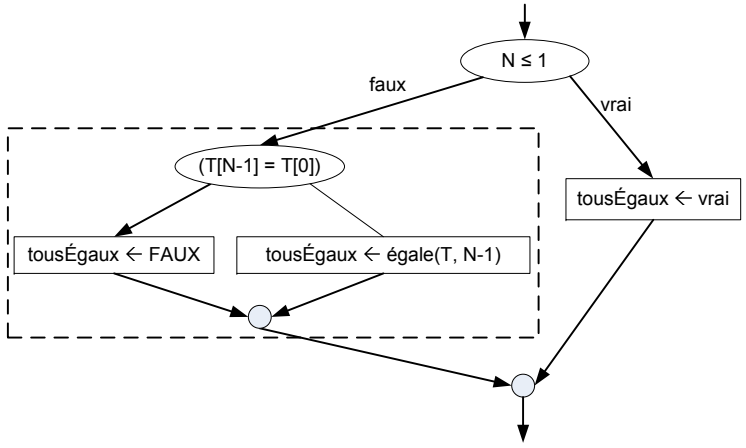
Modèle algorithmique	Java

Exercice 8-4 - Dans un tableau T de plus de N nombres, retourner VRAI si tous les nombres aux positions 0…N-1 de T sont égaux, sinon FAUX.

DONNÉES: T (Référence à un tableau d'entiers)
N (Nombre d'éléments à vérifier)
RÉSULTAT: tousÉgaux (Booléen: VRAI si toutes les premiers N éléments ont la même valeur)
INTERMÉDIAIRE:
resPartiel (résultat partiel)
EN-TÊTE: tousÉgaux ← égale(A, N)
MODULE:

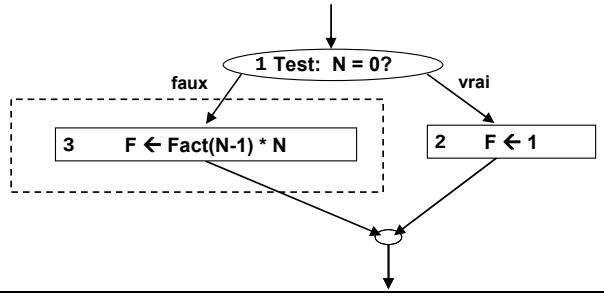


Version efficace

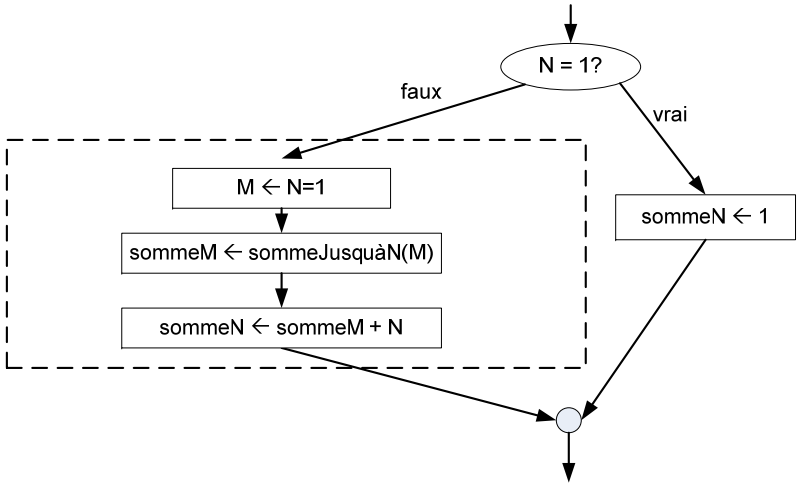



```java
// MÉTHODE égal: Détermine si les N premiers éléments de T
// sont égaux.
public static boolean égal (int [] t, int n)
{
   // DÉCLARATION DES VARIABLES / DICTIONNAIRE DE DONNÉES
   boolean resPartiel; // INTERMÉDIAIRE: Résultat partiel
   boolean tousÉgaux;   // RÉSULTAT: Résultat recherché
   // MODULE DE L'ALGORITHME
   if (n <= 1)
   {
      tousÉgaux = true;
   }
   else
   {
      resPartiel = égal(t, n-1);  // m=n-1 implicite
      tousÉgaux = resPartiel && (t[n-1] == t[0]);
   }
   // RÉSULTAT RETOURNÉ
   return tousÉgaux;
}
// MÉTHODE Égal: Détermine si les N premiers éléments de T
// sont égaux.
public static boolean égal (int [] t, int n)
{
   // DÉCLARATION DES VARIABLES / DICTIONNAIRE DE DONNÉES
   boolean tousÉgaux;   // RÉSULTAT: Résultat recherché
   // MODULE DE L'ALGORITHME
   if (n <= 1)
   {
      tousÉgaux = true;
   }
   else
   {
      if (t[n-1] == t[0])
      {
         tousÉgaux = égal(t, n-1);
      }
      else
      {
         tousÉgaux = false; // inutile de faire l'appel réc.!
      }
   }
   return tousÉgaux;   // RÉSULTAT RETOURNÉ
}
```

Modèle algorithmique	Java
Exercice 8-5 - Calcule N !.	

DONNÉES: N *(entier)*
RÉSULTAT: F *(entier, N factoriel)*
INTERMÉDIAIRE: (aucune)
EN-TÊTE: F ← Fact(N)
MODULE:
 
 	
	`// Method fact`
	`// Given: n, an integer`
	`public static int fact(int n)`
	`{`
	`  int f;  // RÉSULTAT`
	`  if (n == 0)`
	`  {`
	`  f = 1;`
	`  }  // base case`
	`  else`
	`  {`
	`  f = fact(n-1) * n;`
	`  }`
	`  return f;`
	`}`

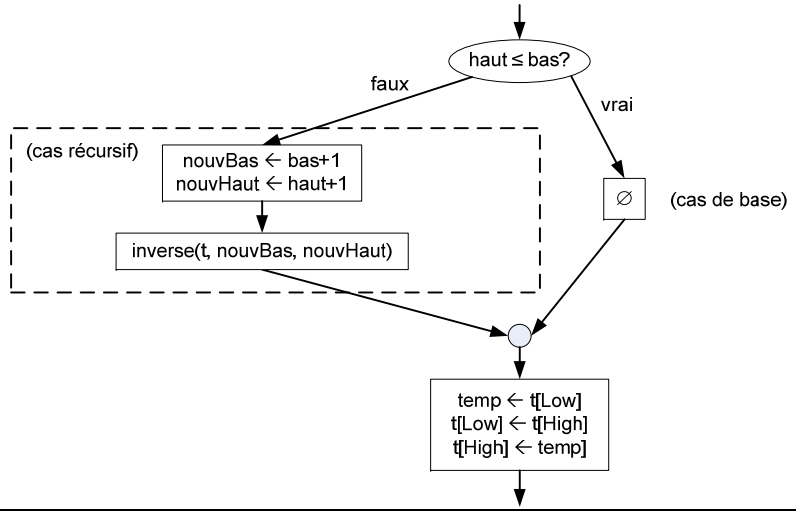
Exercice 8-6 – Trouver la somme de 1+2+…+N.	

DONNÉES: N (un entier)
RÉSULTAT: sommeN (somme des entiers de 1 a N)
INTERMÉDIAIRE:
M (vaur N-1; plus petit!)
sommeM (somme des entiers de 1 à M)
EN-TÊTE: sommeN ← sommeJusquàN(N)
MODULE:
 
 	
	`public static int sommeJusquàN(int n)`
	`{`
	`  // Déclarations de variables`
	`  int sommeN;  // RÉSULTAT : la somme de 1 à N`
	`  int m;  // INTERMÉDIAIRE : vaut n-1`
	`  int sommeM;  // INTERMÉDIAIRE : la somme de 1 à M`
	`  if(n == 1)`
	`  {`
	`  sommeN = 1;  // cas de base`
	`  }`
	`  else`
	`  {`
	`  m = n-1;`
	`  sommeM = sommeJusquàN(M);  // appel récursif`
	`  sommeN = sommeM + 1;`
	`  }`
	`  // Retourne résultat`
	`  return(sommeN);`
	`}`

Modèle algorithmique	Java

Exercice 8-7 - Inverser l'ordre des caractères dans un tableau A de N caractères.

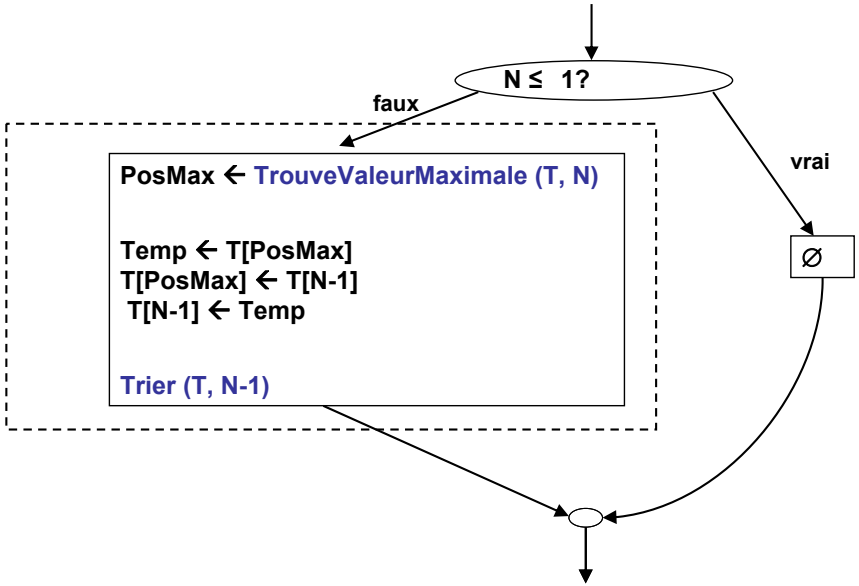
DONNÉES: t (référence à un tableau de caractère à inverser)
bas (index bas)
haut (index haut)
RÉSULTATS: (aucun – tableau référé est modifié)
MODIFIED: A (réfère au tableau inversé)
INTERMÉDIAIRES:
nouvHaut (nouveau index haut)
nouvBas (nouveau index bas)
temp (temporaire pour faire l'échange)
EN-TÊTE inverse(t, bas, haut)
MODULE:




```java
// MÉTHODE Échange: Inverser les caractères dans un tableau
// référé par t de taille N.
// À invoquer initialement avec Échange(t, 0, N-1).
public static void inverse (char [] t, int bas, int haut)
{
   // DÉCLARATION DES VARIABLES / DICTIONNAIRE DE DONNÉES
   int nouvHaut;  // INTERMÉDIAIRE: Haut plus petit
   int nouvBas;   // INTERMÉDIAIRE: Bas 'plus petit'
                  //                    (plus grand!)
   char temp;     // INTERMÉDIAIRE: Caractère tampon
   // MODULE DE L'ALGORITHME
   if (haut - bas <= 1)
   {
      ; // cas de base: ne rien faire
   }
   else
   {
      nouvBas = bas + 1;
      nouvHaut = haut - 1;  // Deux variables à rendre
                            // « plus petites »!
      inverse(t, nouvBas, nouvHaut);
   }
   // échange t[bas] et t[haut]
   temp = t[bas];
   t[bas] = t[haut];
   t[haut] = temp;
   // RÉSULTAT RETOURNÉ: Le tableau 't' est déjà modifié!
}
```

Exercice 8-8 - Trier un tableau A de N nombres en ordre croissant.

DONNÉES: T (Référence au tableau d'entiers à trier)
N (entier, 0 à N-1 pour indexé T)
RÉSULTATS: (aucun)
MODIFIED: T (référence au tableau trié)
INTERMÉDIAIRES:
PosMax (position de la plus grande valeur dans T[0] à T[N-1])
Temp (valeur temporaire pour échange)
EN-TÊTE Trier(T, N)
MODULE:

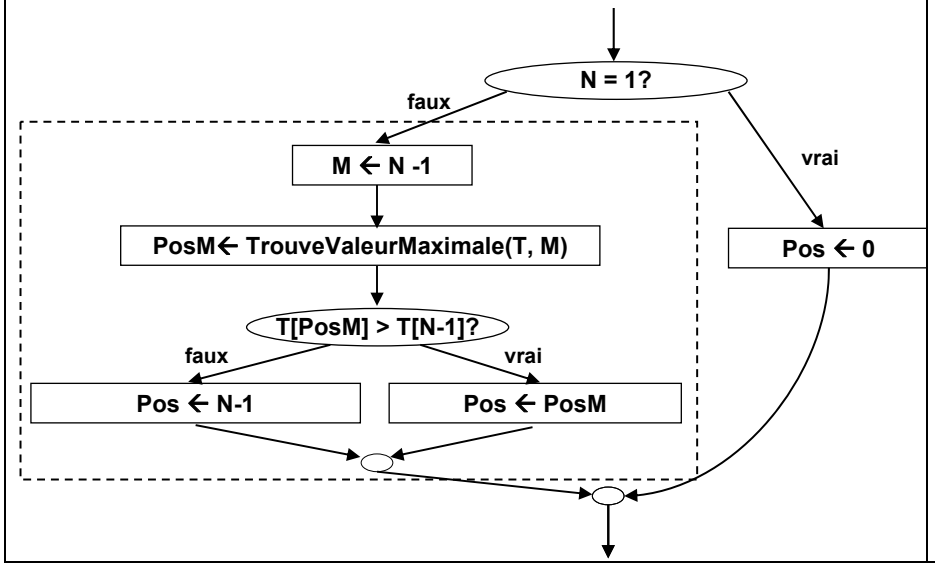



```
public static void trier(int[] t, int n)
{
  // DÉCLARATION DE VARIABLES
  // DONNÉES: t – référence au tableau à trier
  //          n – nombres d'éléments à trier - notez
  //              que t.length NE peut PAS être utilisé.
  // INTERMÉDIAIRES
  int posMax;  // position de la plus grande valeur
  int temp; // utliser pour échange
  // MODULE
  if(n <= 1)
  {
      /* rien faire */ ;
  }
  else
  {
      posMax = trouveValeurMaximale(a, n);
      temp = t[posMax];
      t[posMax] = t[n-1];
      t[n-1] = temp;
      trier(t,n-1);  // trier le reste du tableau
  }
}
```

Modèle algorithmique	Java

Exercice 8-8 - Trier un tableau *A* de N nombres en ordre croissant: – algorithme/méthode **trouveValeurMaximale**

DONNÉES: T (référence à un tableau d'entiers)
N (entier, 0 à N-1 pour indexé T)
RÉSULTATS: Pos (position de l'élément maximal de T[0] à T[N-1])
INTERMÉDIAIRES:
M (entier, intervalle plus petit!)
PosM (position de l'élément maximal dans T[0] à T[M-1)
EN-TÊTE Pos ← TrouveValeurMaximale(T, N)
MODULE :




```
public static void trouveValeurMaximale(int[] t, int n)
{
  // DÉCLARATION DE VARIABLES
  // DONNÉES: t – référence au tableau
  //          n – nombres d'éléments à chercher - notez
  //              que t.length NE peut PAS être utilisé.
  int pos; // RÉSULTAT–position de la plus grande valeur
  // INTERMÉDIAIRES
  int m;    // plus petit
  int posM; // utiliser pour échange

  if(n == 1)
  {
     pos = 0;  // cas de base
  }
  else
  {
     m = n-1;
     posM = trouveValeurMaximale(a, m); //appel récursif
     if(t[posM] > t[n-1])
     {
       pos = posM;
     }
     else
     {
       pos = n-1;
     }
  }
  // RÉSULTAT
  return(pos);
}
```