k). Pour cela, on parcourt séquentiellement de gauche à droite cette troisième zone. Pour l'élément courant de cette zone, trois cas se présentent :

- c'est un 1 : il suffit d'accroître la zone des 1 ;
- c'est un 2 : le permuter avec l'élément le plus à droite de la zone non triée et accroître la zone des 2 ; et enfin
- c'est un 0 : le permuter avec l'élément le plus à gauche de la zone des 1, accroître la zone des 0 et « décaler » la zone des 1.

Initialement les deux premières zones ainsi que la quatrième sont vides.

1. Écrire une fonction C qui trie le tableau t en utilisant l'algorithme donné ci-dessus.
2. Quelle est la complexité de cette fonction ?

► **Exercice 3. (Tri par distribution)**

Soit Tab un tableau de n caractères entre A et Z . Pour trier le tableau selon l'ordre lexicographique, on utilise un tableau auxiliaire Aux d'entiers tel que $Aux[i]$ indique le nombre de fois que le i -ème caractère apparaît dans le tableau Tab .

1. Écrire une fonction qui crée le tableau Aux à partir du tableau Tab .
2. Le tableau Aux étant créé, l'utiliser pour trier le tableau Tab .
3. Quelle est la complexité de cet algorithme en temps et en espace ?
4. Peut-il être utilisé quel que soit le type des données ?

► **Exercice 4. (Tri de Shell)**

L'une des variantes les plus connue du tri par insertion est celle proposée par Donald L. Shell. Cet algorithme utilise le tri par insertion sur des sous-séquences périodiques afin d'obtenir un tri plus rapide.

L'idée de l'algorithme est de raffiner progressivement la position d'un élément dans le tableau trié en réalisant des pas de plus en plus petits (jusqu'à atteindre des pas de 1). L'étape de base de l'algorithme considère que le tableau initial est composé de h sous-tableaux dont les éléments sont à une distance de h . On va alors trier ces h tableaux en utilisant l'algorithme du tri par insertion. Cette opération est répétée pour des valeurs de h diminuant jusqu'à la valeur 1.

Dans un premier temps, la suite des valeurs de h sera définie par :

$$h_n = \begin{cases} 1 & \text{pour } n = 1 \\ 2 \times h_{n-1} & \text{pour } n \geq 2 \end{cases}$$

1. Donner les étapes nécessaires au tri du tableau

6	5	4	9	1	7	8	3	2
---	---	---	---	---	---	---	---	---

2. Écrire une fonction C qui trie un tableau d'entiers en utilisant l'algorithme donné ci-dessus.

Il faut noter que la qualité du tri de Donald L. Shell dépend fortement de la suite des valeurs de h . La suite définie par :

$$h_n = \begin{cases} 1 & \text{pour } n = 1 \\ 3 \times h_{n-1} + 1 & \text{pour } n \geq 2 \end{cases}$$

est suggérée par Donald E. Knuth, mais on peut imaginer d'autres suites pour la décroissance des pas.

► **Exercice 5. (Tri rapide)**

La méthode consiste à placer un élément du tableau (appelé pivot) à sa place définitive, en permutant tous les éléments de telle sorte que tous ceux qui lui sont inférieurs soient à sa gauche et que tous ceux qui lui sont supérieurs soient à sa droite. Cette opération s'appelle le partitionnement. Pour chacun des sous-tableaux situés à droite et à gauche du pivot, on définit un nouveau pivot et on répète l'opération de partitionnement. Ce processus est répété récursivement, jusqu'à ce que l'ensemble des éléments soit trié.

Généralement, on choisit comme pivot le premier élément de la liste.

1. Trier le tableau suivant en donnant toutes les étapes intermédiaires à l'aide du tri rapide

3	5	2	0	1	4
---	---	---	---	---	---

2. Écrire une fonction C qui ordonne les éléments d'un tableau suivant le tri rapide.