Algorithmes de tri

Développeur Python

Sommaire

1. Généralités.
2. Algorithmes itératifs.
3. Algorithmes récursifs.

1. Généralités.

1. Généralités.

But des algorithmes de tri

- À partir d'une liste de données numériques, réordonner ces valeurs par ordre croissant.

- La liste en question sera donc un paramètre lu et modifié par ces procédures de tri.

- Pour ce faire on n'utilisera pas de listes intermédiaires, les tris se feront "sur place".

1. Généralités.

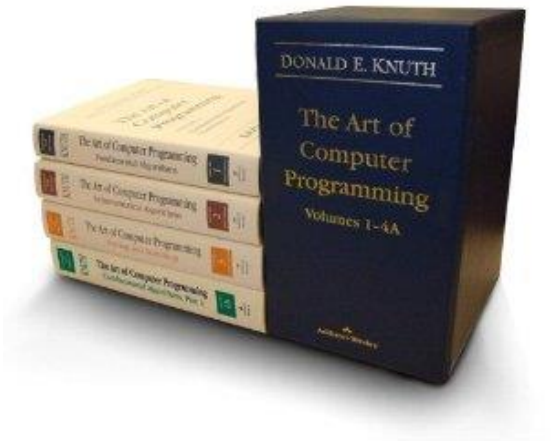
Remarques

- Il s’agit d’un des plus problèmes les plus classiques de l’algorithmique.
- De nombreuses résolutions en sont possibles avec des méthodes radicalement différentes.
- Nous comparerons l’efficacité de certains algorithmes de tri dans le chapitre consacré à la complexité du cours de seconde année.

1. Généralités.

Note bibliographique

- On pourra consulter la série de livres de Donald Knuth “The Art of Computer Programming” où plus d’une trentaine d’algorithmes de tri sont étudiés en détail.



1. Généralités.

Deux types d'algorithmes étudiés dans la suite

- Algorithmes itératifs.

- Algorithmes récursifs, basés sur le principe "diviser c'est régner", c'est-à-dire dans lesquels le tri d'une liste s'effectue en la divisant plusieurs fois par deux jusqu'à obtention de sous-listes ne comportant qu'une seule valeur.

1. Généralités.

2. Algorithmes itératifs.

2. Algorithmes itératifs.

Tri par sélection : principe

- Le tri par sélection est assez intuitif du point de vue mathématique, puisqu'il consiste dans un premier temps à mettre à la première place le plus petit élément de la liste, puis à la seconde place le deuxième plus petit élément, etc.

2. Algorithmes itératifs.

Tri par sélection : description

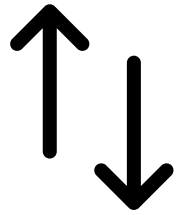
1. Rechercher dans la liste la plus petite valeur et la permuter avec le premier élément de la liste.
2. Rechercher ensuite la plus petite valeur à partir de la deuxième case et la permuter avec le second élément de la liste.
3. Et ainsi de suite jusqu'à avoir parcouru toute la liste.

2. Algorithmes itératifs.

Tri par sélection : exemple

- On considère la liste suivante de 5 entiers :

5	8	2	9	5



2. Algorithmes itératifs.

Tri par sélection : exemple, première itération

- On détermine le minimum des éléments de la liste :

- Et on le permute avec le premier élément de la liste :

2. Algorithmes itératifs.

Tri par sélection : exemple, deuxième itération

- On détermine le minimum des éléments de la liste à partir de la deuxième case :

- Et on le permute avec le second élément de la liste :

2. Algorithmes itératifs.

Tri par sélection : exemple, troisième itération

- On détermine le minimum des éléments de la liste à partir de la troisième case :

- Et on le permute avec le troisième élément de la liste :

2. Algorithmes itératifs.

Tri par sélection : exemple, quatrième itération

- On détermine le minimum des éléments de la liste à partir de la quatrième case :

- Et on le permute avec le quatrième élément de la liste :

2. Algorithmes itératifs.

Tri par sélection : exemple, fin de l'algorithme

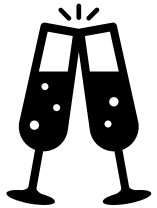
- Le cinquième et dernier élément la liste est de fait à sa place, le tri est terminé :

2	5	5	8	9

2. Algorithmes itératifs.

Tri à bulles : principe

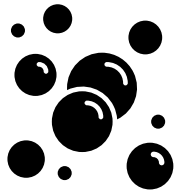
- Le tri à bulles consiste à parcourir toute la liste en comparant chaque élément avec son successeur, puis en les remettant dans le “bon“ ordre si nécessaire. Et à recommencer si besoin est.



2. Algorithmes itératifs.

Tri à bulles : description

1. Parcourir la liste en comparant chaque élément avec son successeur.
2. Si ce dernier est le plus petit des deux, les permuter.
3. Si lors du parcours de la liste une permutation au moins a été effectuée, recommencer un nouveau parcours.

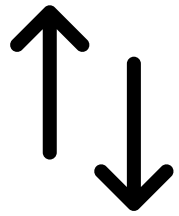


2. Algorithmes itératifs.

Tri à bulles : exemple

- On considère la liste suivante de 5 entiers :

5	8	2	9	5



2. Algorithmes itératifs.

Tri à bulles : exemple, première itération

- On compare 5 et 8 :

- Ils sont dans l'ordre donc on ne les permute pas :

2. Algorithmes itératifs.

Tri à bulles : exemple, première itération

- On compare 8 et 2 :

- On les permute pour les mettre dans l'ordre :

2. Algorithmes itératifs.

Tri à bulles : exemple, première itération

- On compare 8 et 9 :

- Ils sont dans l'ordre donc on ne les permute pas :

2. Algorithmes itératifs.

Tri à bulles : exemple, première itération

- On compare 9 et 5 :

- On les permute pour les mettre dans l'ordre :

2. Algorithmes itératifs.

Tri à bulles : exemple, deuxième itération

- On compare 5 et 2 :

- On les permute pour les mettre dans l'ordre :

2. Algorithmes itératifs.

Tri à bulles : exemple, deuxième itération

- On compare 5 et 8 :

- Ils sont dans l'ordre donc on ne les permute pas :

2. Algorithmes itératifs.

Tri à bulles : exemple, deuxième itération

- On compare 8 et 5 :

- On les permute pour les mettre dans l'ordre :

2. Algorithmes itératifs.

Tri à bulles : exemple, troisième itération, fin de l'algorithme

- On parcourt de nouveau la liste, mais cette fois ci on constate qu'il n'y a plus de permutations à effectuer.

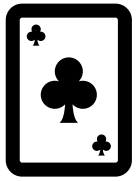
- Le tri est donc terminé.

2	5	5	8	9

2. Algorithmes itératifs.

Tri par insertion : principe

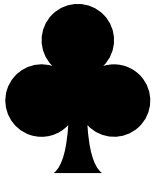
- Le tri par insertion est celui naturellement utilisé par les joueurs de cartes pour trier leur jeu. Il consiste à insérer les éléments un par un en s'assurant que lorsque l'on rajoute un nouvel élément, les éléments déjà insérés restent triés.



2. Algorithmes itératifs.

Tri par insertion : description

1. Considérer le premier élément de la liste. À lui tout seul il constitue une liste triée.
2. Insérer ensuite le second élément de telle sorte que les deux premiers éléments soient triés.
3. Continuer ainsi en insérant successivement chaque élément à sa “bonne” place dans la partie déjà triée de la liste.

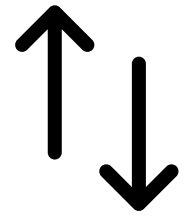


2. Algorithmes itératifs.

Tri par insertion : exemple

- On considère la liste suivante de 5 entiers :

5	3	1	4	2



2. Algorithmes itératifs.

Tri par insertion : exemple, insertion du second élément vis-à-vis du premier

- On considère la valeur 3 :

- Que l'on "retire" provisoirement de la liste :

- On décale le 5 :

- Et on réinsère la valeur 3 :

2. Algorithmes itératifs.

Tri par insertion : exemple, insertion du troisième élément vis-à-vis des premiers

- On considère la valeur 1 :

- Que l'on "retire" provisoirement de la liste :

- On décale le 3 et le 5 :

- Et on réinsère la valeur 1 :

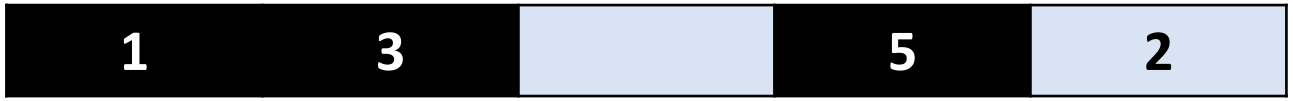
2. Algorithmes itératifs.

Tri par insertion : exemple, insertion du quatrième élément vis-à-vis des premiers

- On considère la valeur 4 :

- Que l'on "retire" provisoirement de la liste :

- On décale le 5 :



- Et on réinsère la valeur 4 :

2. Algorithmes itératifs.

Tri par insertion : exemple, insertion du cinquième élément vis-à-vis des premiers

- On considère la valeur 2 :

- Que l'on "retire" provisoirement de la liste :

- On décale le 3, le 4 et le 5 :

- Et on réinsère la valeur 2 :

2. Algorithmes itératifs.

Tri par insertion : exemple, fin de l'algorithme

- Chaque élément a été inséré à sa place, le tri est terminé :

1	2	3	4	5

2. Algorithmes itératifs.

3. Algorithmes récursifs.

3. Algorithmes récursifs.

Tri fusion : principe

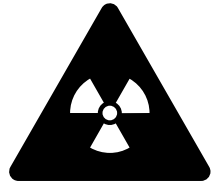
- Le tri fusion consiste à trier récursivement les deux moitiés de la liste, puis à fusionner ces deux sous-listes triées en une seule. La condition d'arrêt à la récursivité sera l'obtention d'une liste à un seul élément, car une telle liste est évidemment déjà triée.

- C'est donc bien un algorithme de type "Diviser pour Régner".

3. Algorithmes récursifs.

Tri fusion : description

1. Diviser la liste en deux sous-listes de même taille (à un élément près) en la “coupant“ par la moitié.
2. Trier récursivement chacune de ces deux sous-listes. Arrêter la récursion lorsque les listes n'ont plus qu'un seul élément.
3. Fusionner les deux sous-listes triées en une seule.

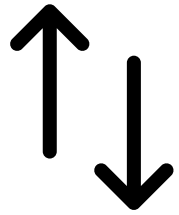


3. Algorithmes récursifs.

Tri fusion : exemple

- On considère la liste suivante de 7 entiers :

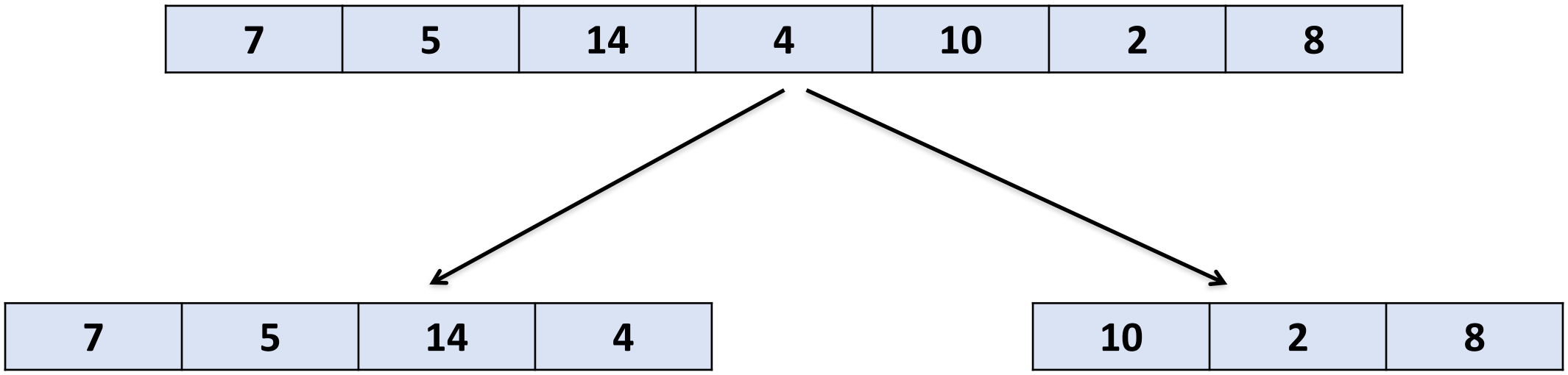
7	5	14	4	10	2	8



3. Algorithmes récursifs.

Tri fusion : exemple

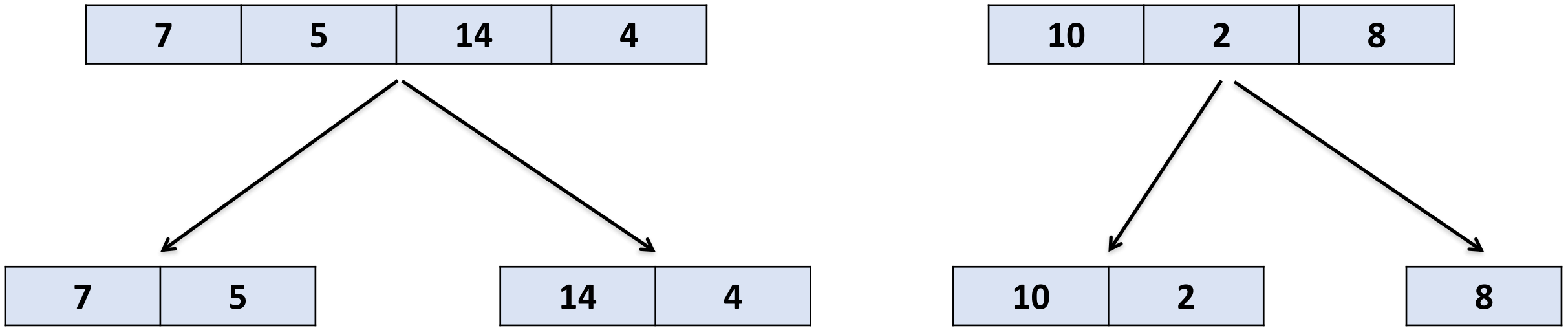
- On la scinde en deux sous-listes :



3. Algorithmes récursifs.

Tri fusion : exemple

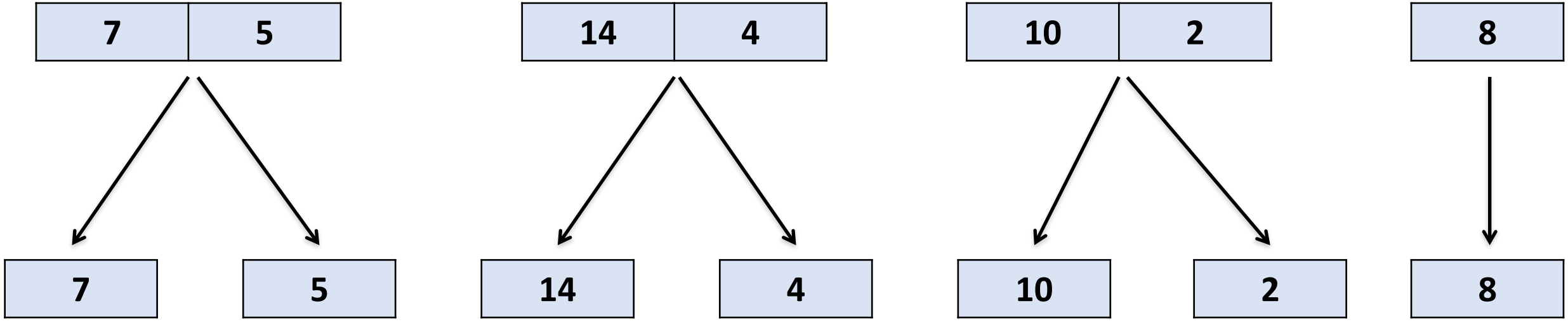
- Sous-listes que l'on scinde à leur tour en deux :



3. Algorithmes récursifs.

Tri fusion : exemple

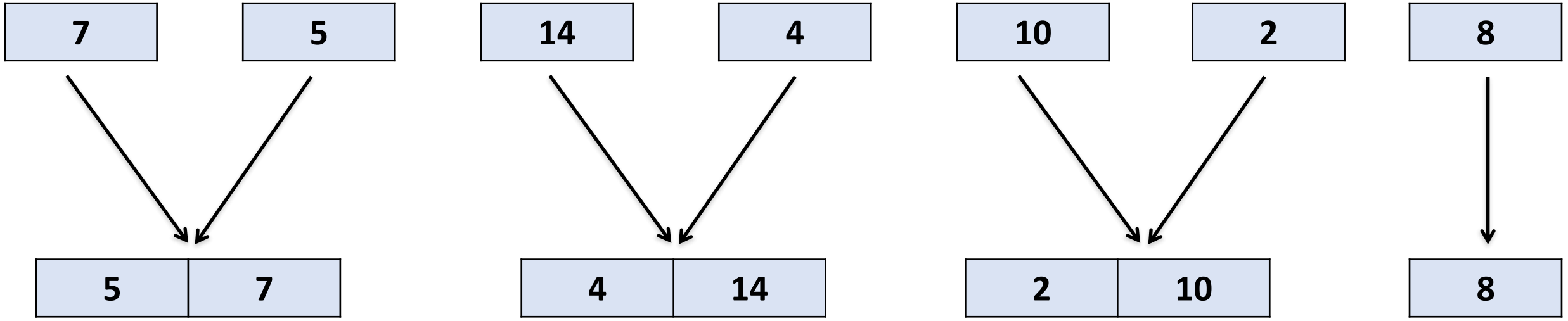
- Sous-listes que l'on scinde à leur tour en deux :



3. Algorithmes récursifs.

Tri fusion : exemple

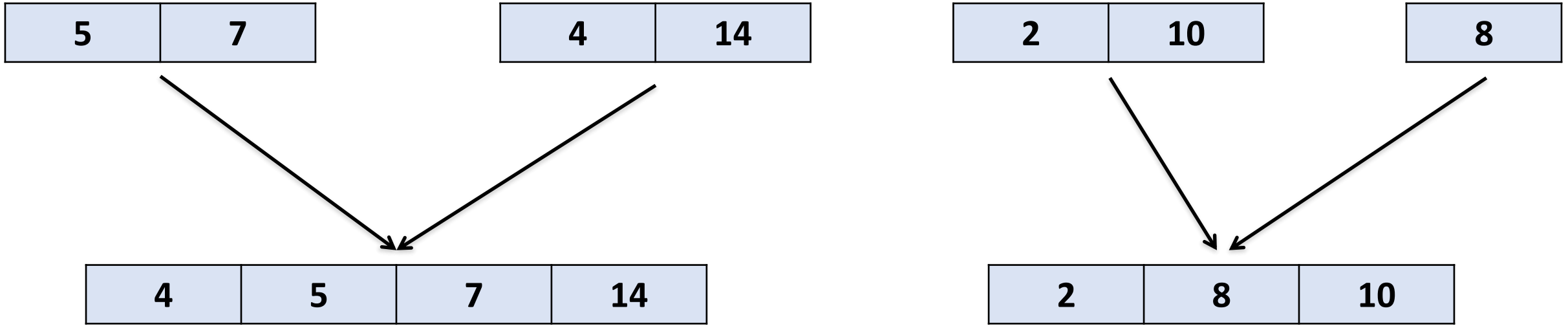
- Ces sous-listes sont triées car elles n'ont qu'un élément. On va maintenant les fusionner deux par deux en de nouvelles sous-listes triées :



3. Algorithmes récursifs.

Tri fusion : exemple

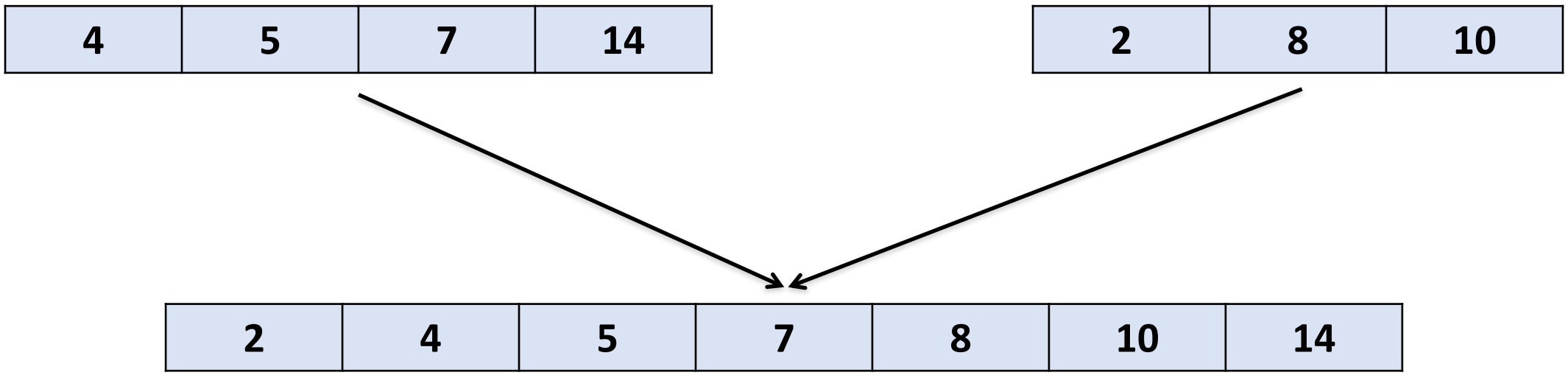
- De nouveau une étape de fusionnement :



3. Algorithmes récursifs.

Tri fusion : exemple

- Une dernière fusion :



3. Algorithmes récursifs.

Tri fusion : exemple, fin de l'algorithme

- On a fusionné toutes les sous-listes obtenues lors des appels récursifs, le tri est terminé.

2	4	5	7	8	10	14

3. Algorithmes récursifs.

Tri rapide : principe

- Le tri rapide consiste à positionner un par un par les éléments à leur place définitive en ne laissant à leur gauche que des éléments plus petits et à leur droite que des éléments plus grands.

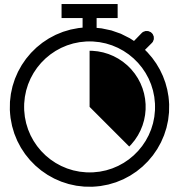
- Les deux sous-listes obtenues par ce positionnement sont ensuite traitées de la même façon par récursivité.

- La condition d’arrêt à la récursivité sera l’obtention d'une liste à un seul élément, car cet unique élément est nécessairement au "bon" endroit.

3. Algorithmes récursifs.

Tri rapide : description

1. Considérer le premier élément de la liste et le positionner à sa place définitive, avec à sa gauche une sous-liste constituée d'éléments qui lui sont inférieurs ou et égaux et à sa droite une sous-liste constituée d'éléments qui lui sont strictement supérieurs.
2. Appliquer récursivement ce même traitement aux deux sous-listes ainsi obtenues. Arrêter la récursion lorsque les listes n'ont plus qu'un seul élément.
3. Pas de résultats à combiner.

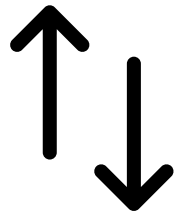


3. Algorithmes récursifs.

Tri rapide : exemple

- On considère la liste suivante de 8 entiers :

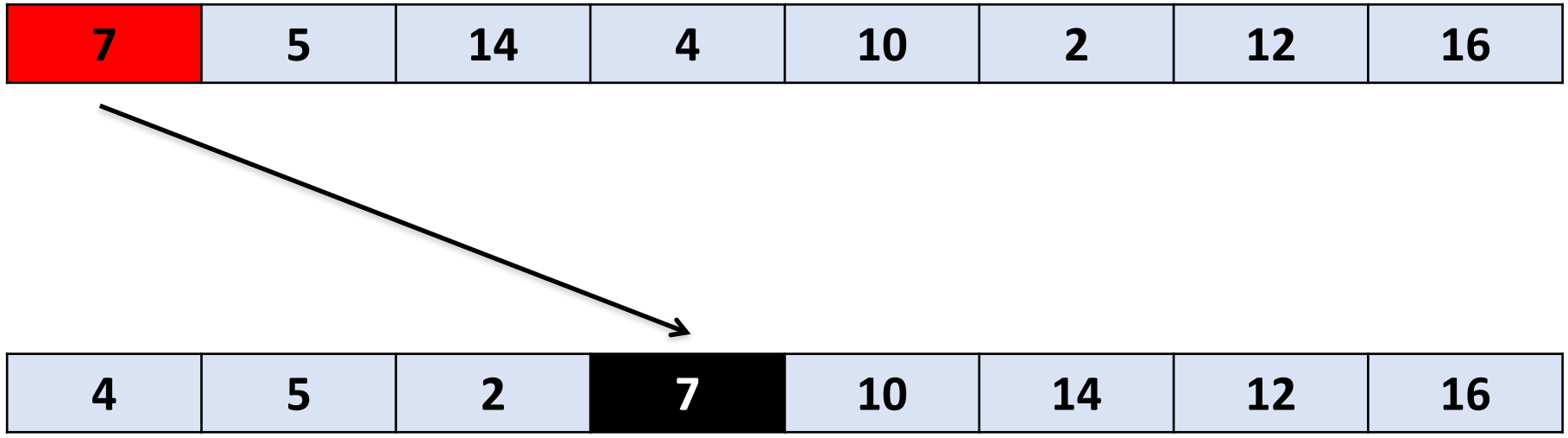
7	5	14	4	10	2	12	16



3. Algorithmes récursifs.

Tri rapide : exemple, premier appel récursif

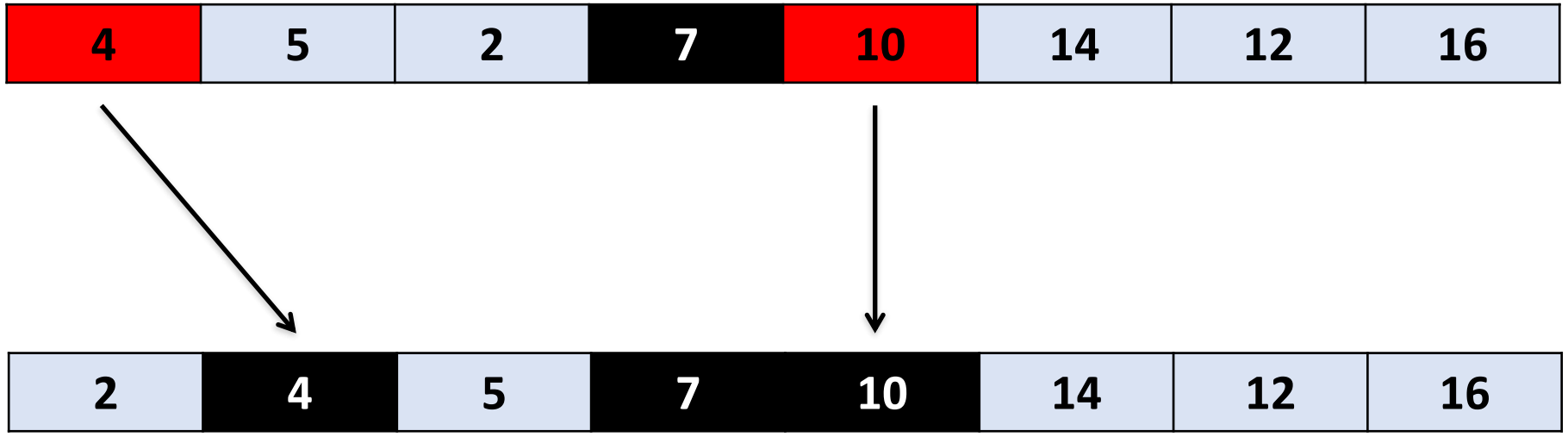
- On positionne le premier élément, *i.e.* 7 à sa place définitive, avec à sa gauche des éléments inférieurs ou égal et à sa droite des éléments supérieurs



3. Algorithmes récursifs.

Tri rapide : exemple, deuxième appel récursif

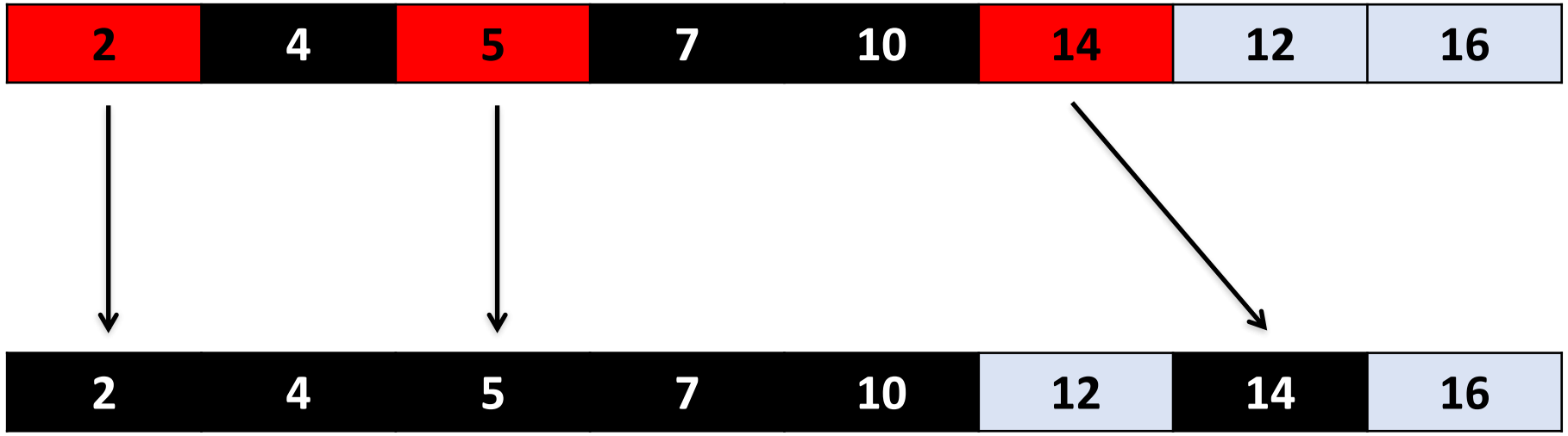
- On va placer les éléments 4 et 10 qui sont les premiers éléments des sous-listes créées lors de l'appel précédent :



3. Algorithmes récursifs.

Tri rapide : exemple, troisième appel récursif

- On va placer les éléments 2, 5 et 14 qui sont les premiers éléments des sous-listes créées lors de l'appel précédent :



3. Algorithmes récursifs.

Tri rapide : exemple, quatrième appel récursif

- On va placer les éléments 12 et 16 qui sont les premiers éléments des sous-listes créées lors de l'appel précédent :

3. Algorithmes récursifs.

Tri rapide : exemple, fin de l'algorithme

- Chaque élément a été positionné à sa place, le tri est terminé :

2	4	5	7	10	12	14	16

3. Algorithmes récursifs.

Tri rapide : remarque

- Dans ce contexte, l’élément que l'on positionne à sa place définitive est appelé pivot.
- Le choix effectué ici est de prendre pour pivot le premier élément de la liste.
- Ce choix est le plus simple mais il n'est pas optimal. On pourra se demander quel autre pivot on aurait pu prendre pour améliorer les performances de cet algorithme.

3. Algorithmes récursifs.