Fusion en place

Francois Schwarzentruber[1]

ENS Rennes

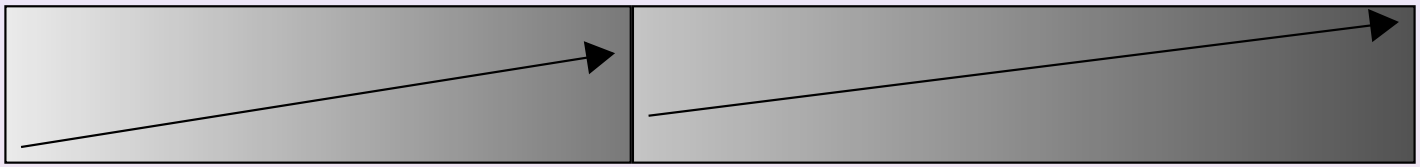
January 31, 2013

[1] adapté de l'article 'Practical In-place merging' de Bing-Chao Huang et Michael A. Langston

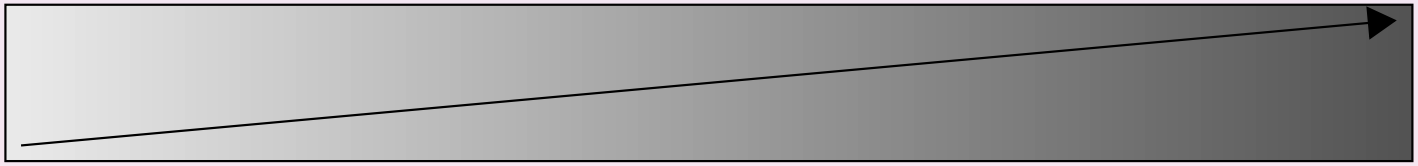
Outline

Spécification du problème

Entrée : un tableau T de taille n, concaténation de 2 séquences triées



Sortie : une permutation triée de T



Souhait

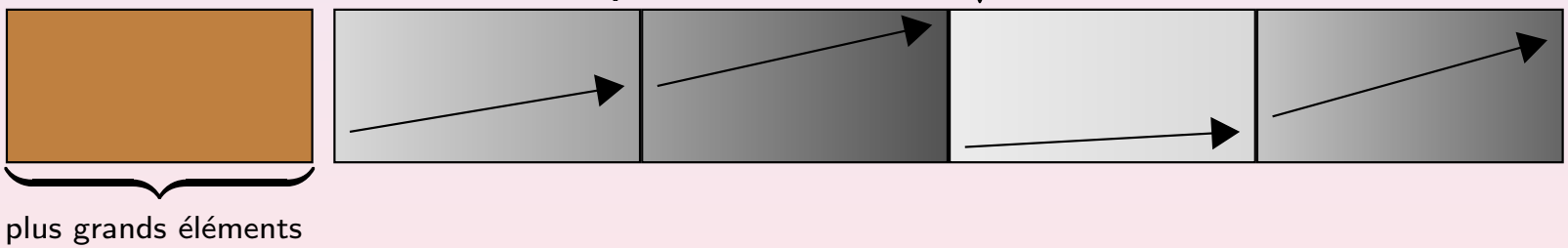
Algorithme en temps $O(n)$ où n est la taille de T
$O(1)$ espace mémoire supplémentaire

Outline

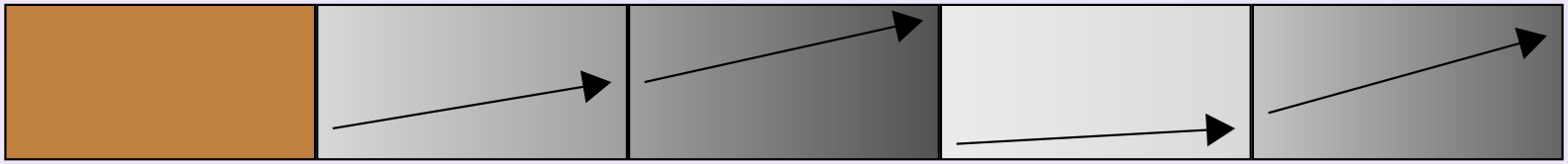
Algorithme principal

On suppose n est un carré parfait.

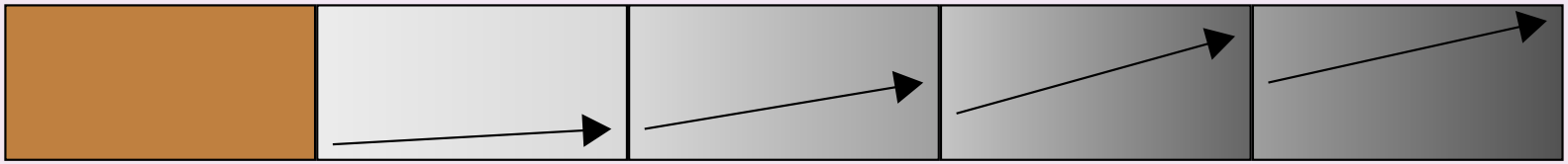
Situation de départ :
on suppose que les $\sqrt{n}$ plus grands éléments devant en décallant le reste. Le reste est découpé en bloc de $\sqrt{n}$ éléments



Algorithme principal



Étape 1 : on trie les blocs par dernier élément croissant.



On utilise le tri sélection :

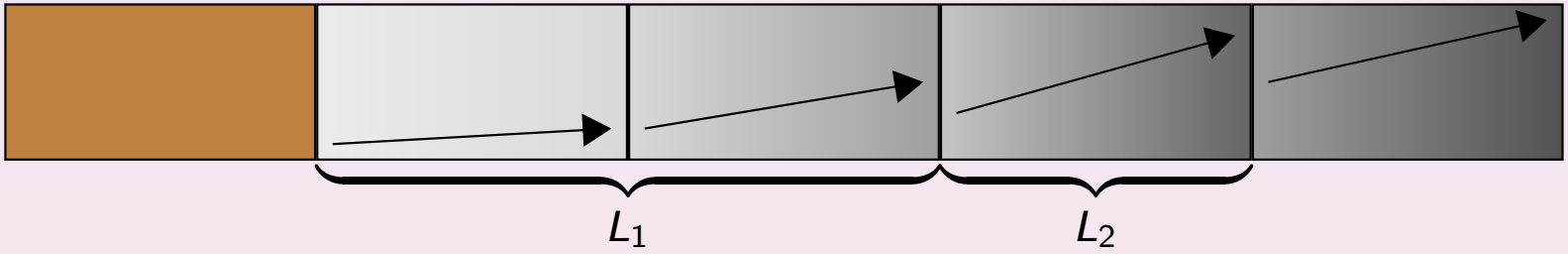
- $O((\sqrt{n})^2) = O(n)$ comparaisons de blocs.
  Une comparaison coûte $O(1)$;
- $O(\sqrt{n})$ échanges de blocs. Un échange coûte $O(\sqrt{n})$.

L'étape 1 coûte donc $O(n)$.

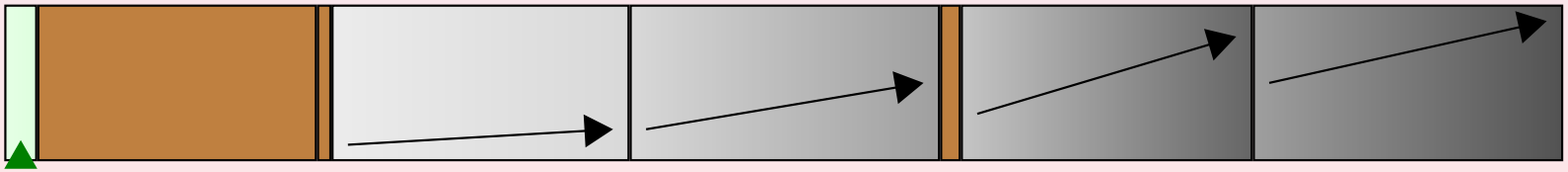
Algorithme principal

Étape 2 :

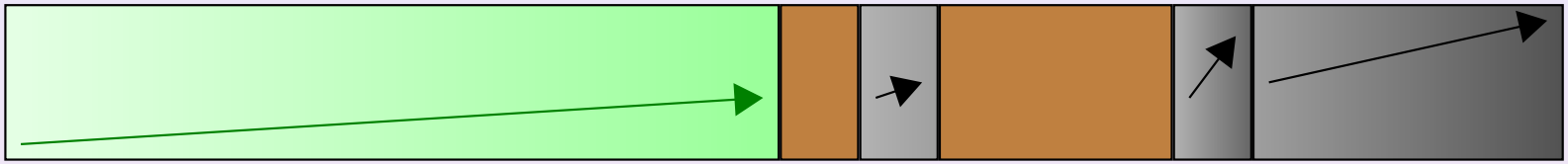
- on regroupe ensemble le plus grand nombre de blocs pour former une séquence croissante, notée L_1.
- Le bloc suivant est L_2.



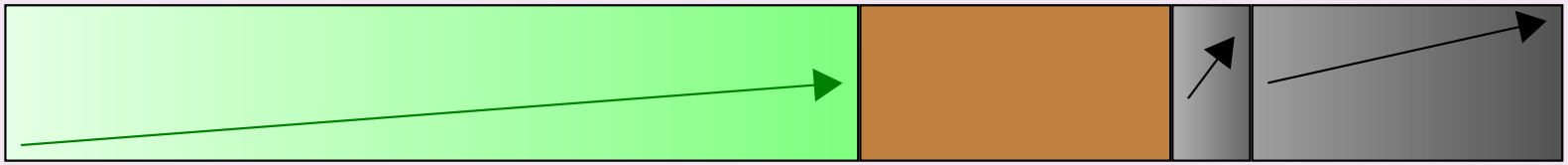
Étape 3 : on fusionne L_1 et L_2 jusqu'à avoir traité le plus grand élément de L_1, en utilisant le buffer comme tampon.



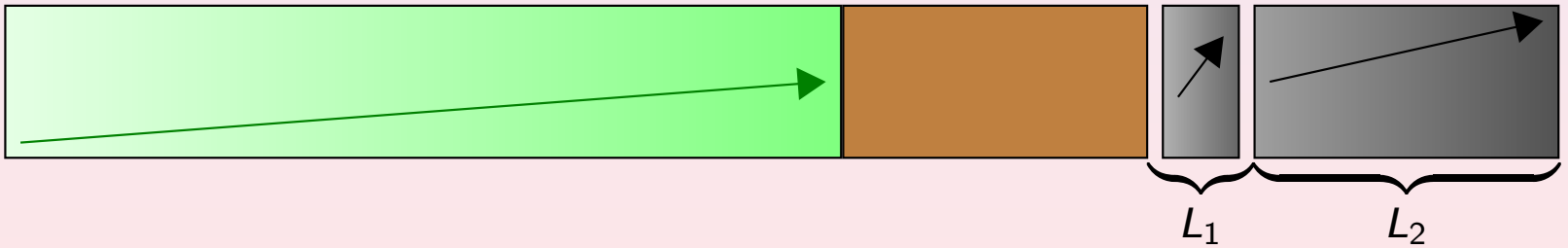
Algorithme principal



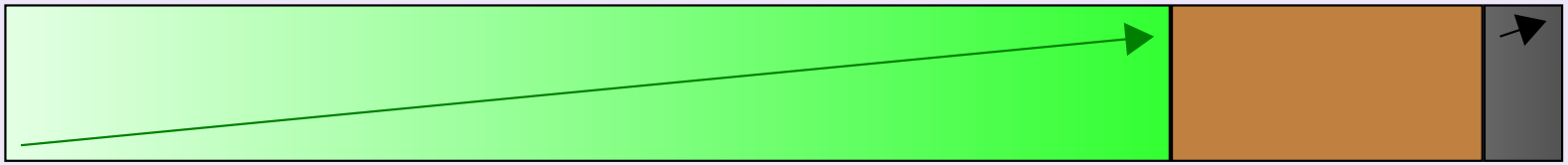
A la fin :



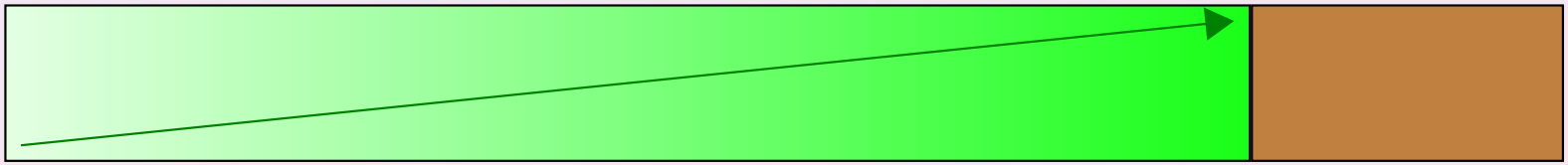
Puis on refait des fusions, etc. :



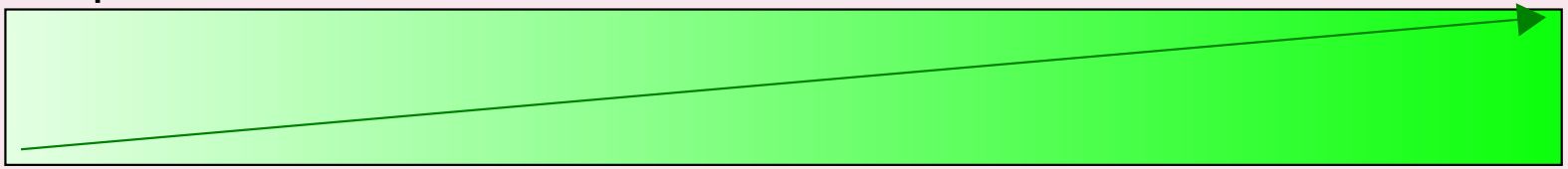
Algorithme principal



Étape 4 : on décale le buffer



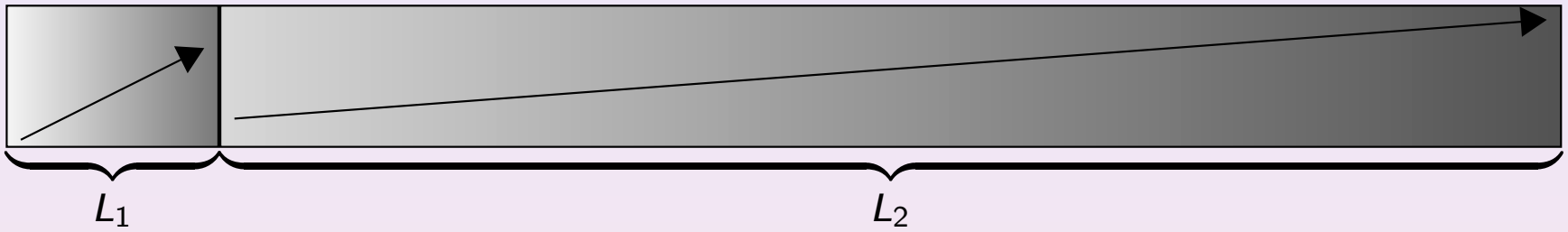
Étape 5 : on trie le buffer



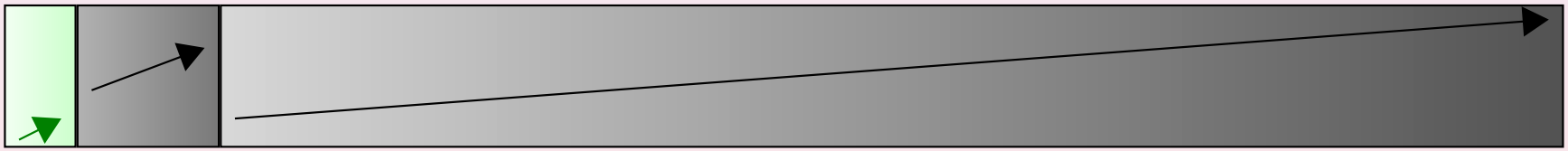
Tri sélection par exemple : $O((\sqrt{n})^2) = O(n)$.

Outline

Quand une des séquences est de longueur $< \sqrt{n}$

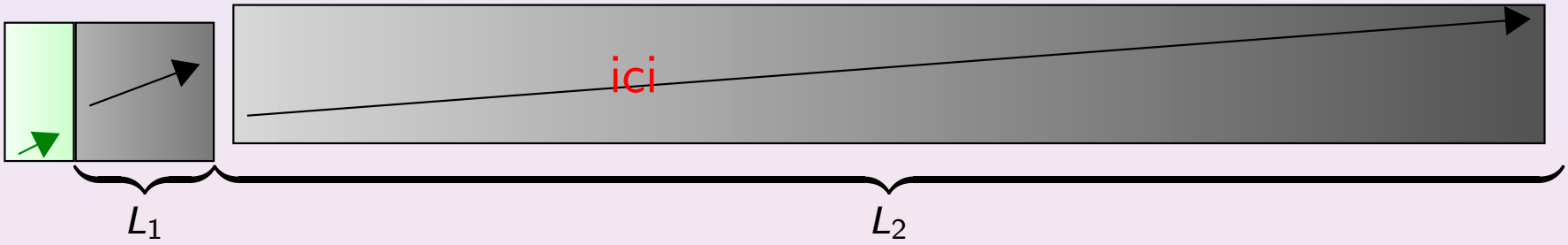


Etape 1 : Repérer les éventuels éléments de L_1 à la bonne place

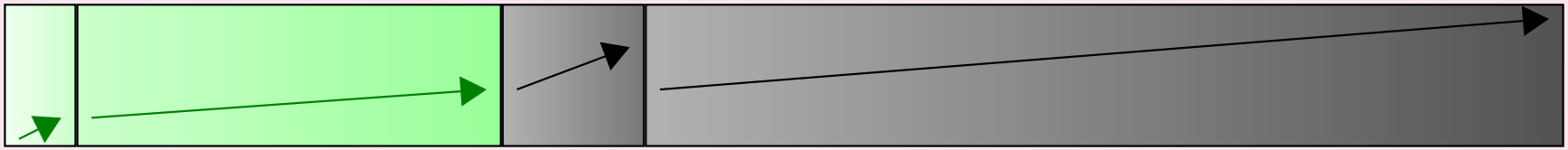


Quand une des séquences est de longueur $< \sqrt{n}$

Etape 2 : repérer où insérer les premiers éléments de L_1 dans L_2

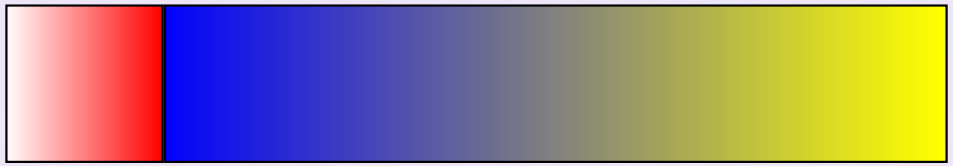


Etape 3 : effectuer une "rotation"



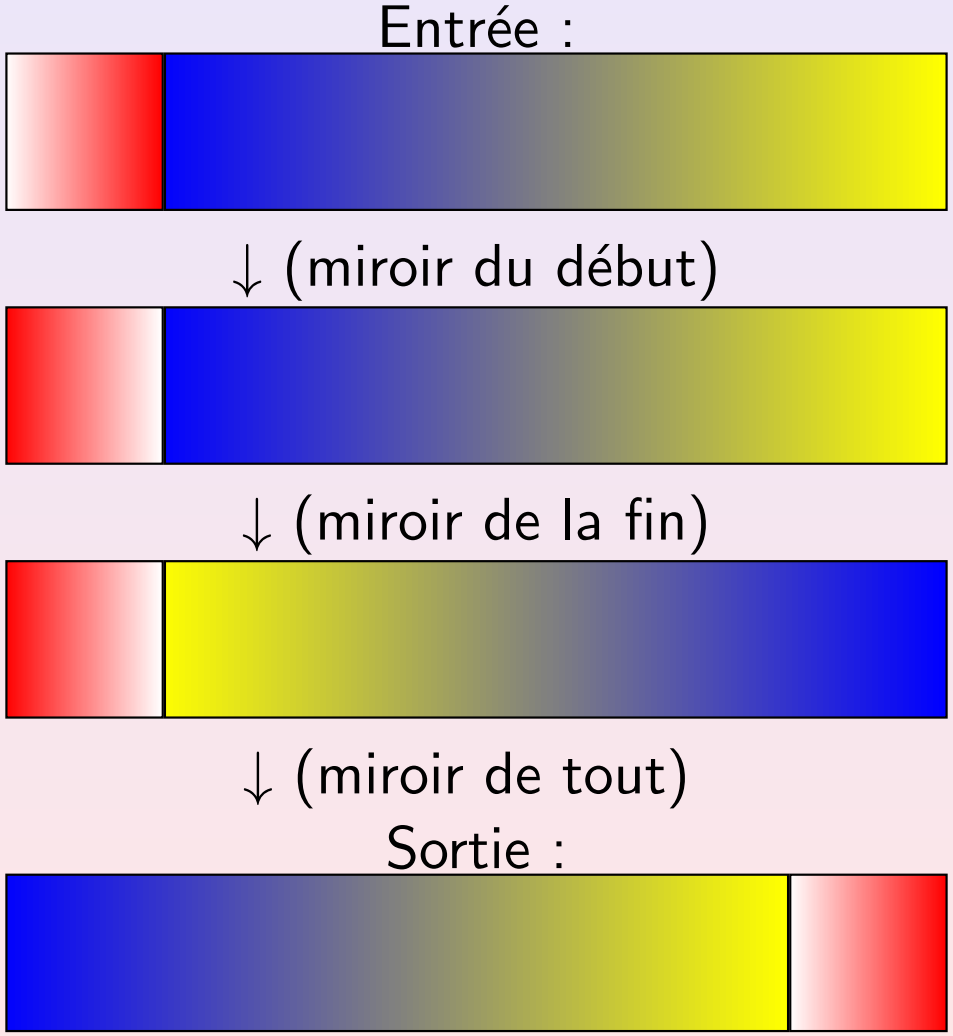
Rotation d'un tableau (en place)

Entrée :

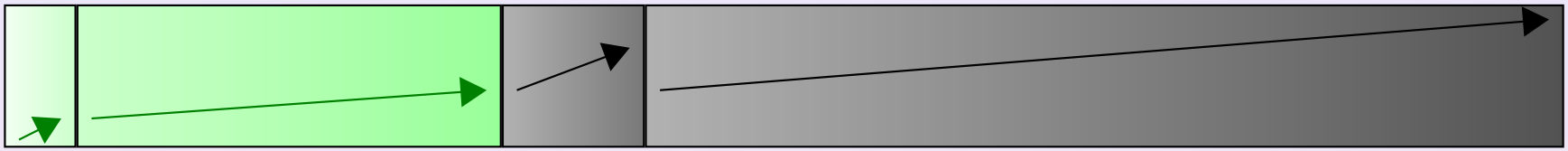


Sortie :

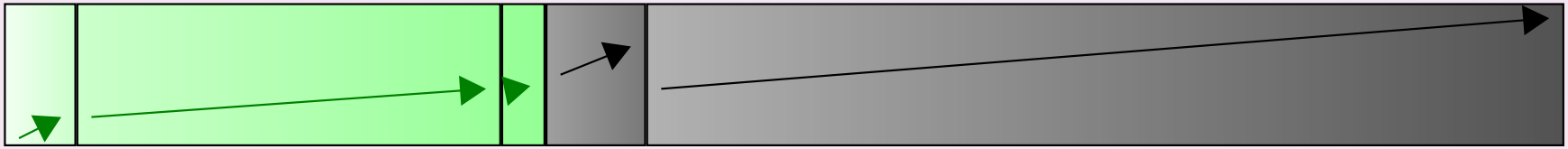
Rotation d'un tableau (en place)



Quand une des séquences est de longueur $< \sqrt{n}$



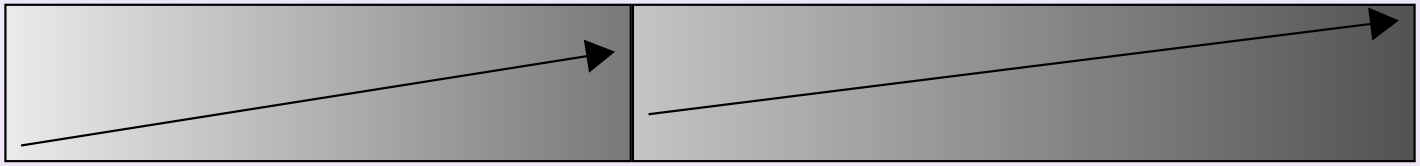
Etape 4 : repérer les éléments de L_1 bien triées (il y en a au moins un)



Bilan :
Les éléments de L_1 sont déplacés au plus $O(\sqrt{n})$ fois.
Les éléments de L_2 sont déplacés au plus 1 fois.
Complexité en $O(\sqrt{n}\sqrt{n} + 1 \times n) = O(n)$.

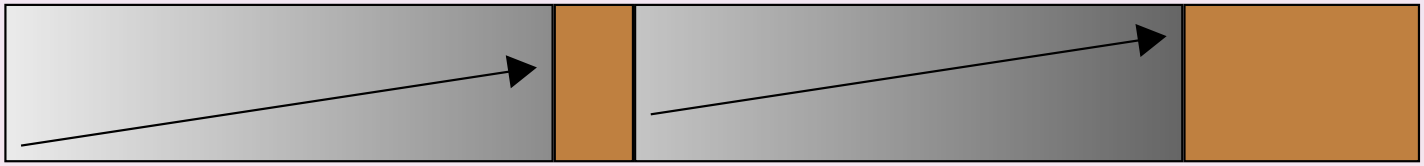
Outline

Détails d'implémentation

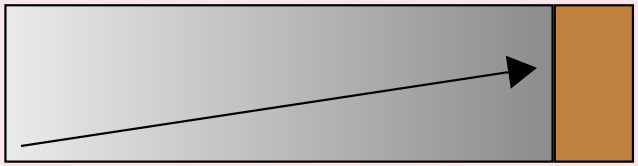


$s = E[\sqrt{n}]$

Étape 1 : identifier les éléments qui vont dans le buffer de taille s

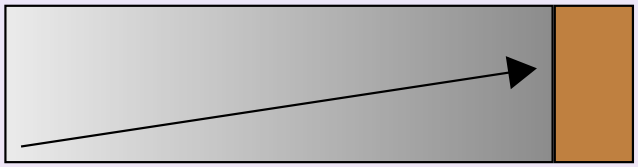


Étape 2 : identifier les éléments restants pour que la liste de droite soient des s-blocs



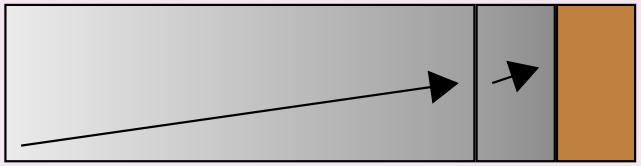
des s-blocs . . .

Détails d'implémentation



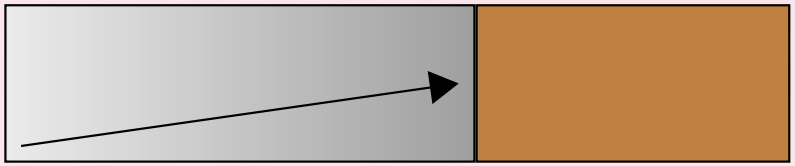
des s-blocs . . .

Étape 3 : identifier pour compléter en un s-bloc dans la liste de gauche

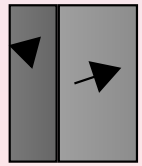


des s-blocs . . .

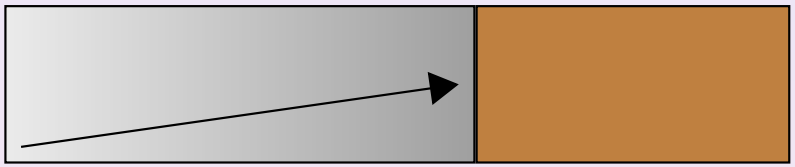
Étape 4 : mettre le buffer ensemble et les autres éléments derrière



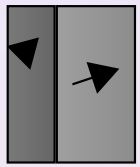
des s-blocs . . .



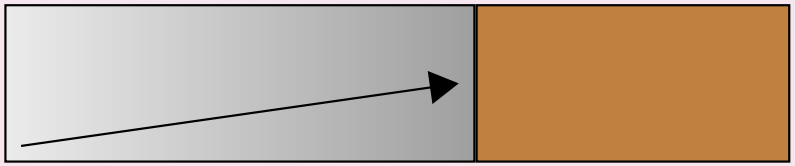
Détails d'implémentation



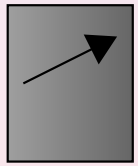
des s-blocs ...



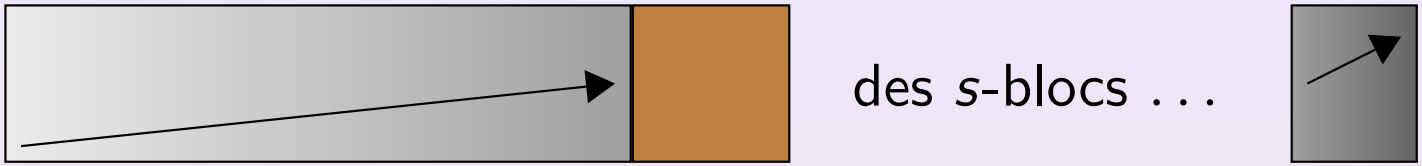
Étape 5 : fusionner les derniers éléments (en utilisant le buffer)



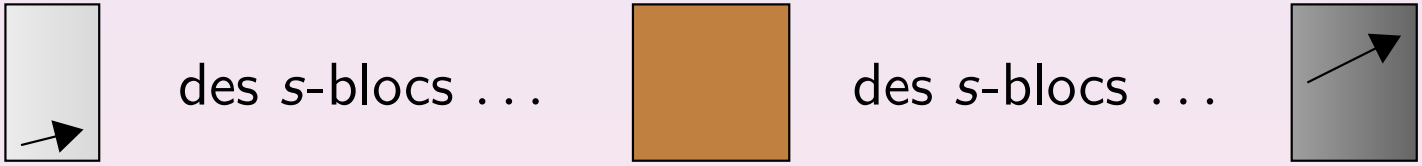
des s-blocs ...



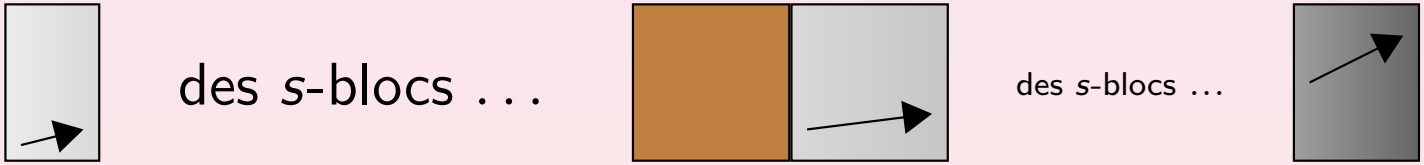
Détails d'implémentation



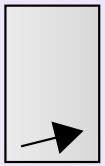
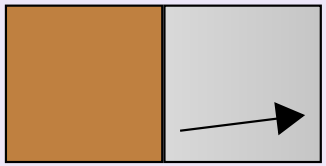
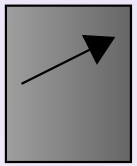
Étape 6 : identifier le début



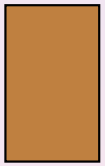
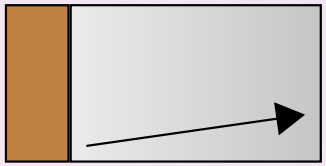
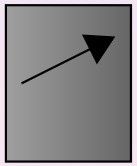
Étape 7 : identifier le premier bloc de la liste de droite



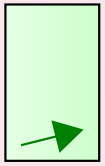
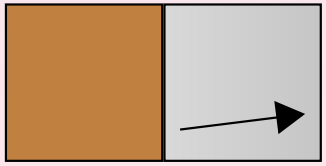
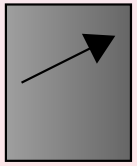
Détails d'implémentation


des *s*-blocs . . .

des *s*-blocs


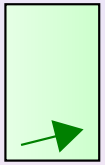
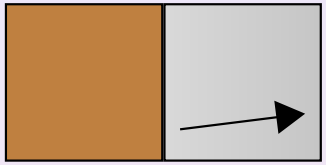
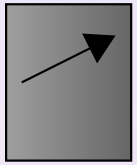
Étape 8 : fusionner les débuts (en utilisant le buffer)


des *s*-blocs . . .

des *s*-blocs


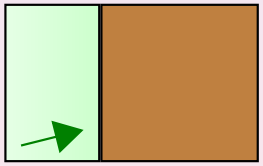
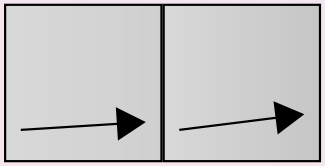
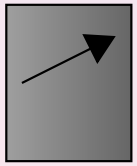
Étape 9 : échanger le début de la fusion la partie du buffer au début


des *s*-blocs . . .

des *s*-blocs


Détails d'implémentation


des s-blocs . . .

des s-blocs . . .


Étape 10 : échanger le buffer et le premier s-bloc de la partie gauche


des s-blocs

des s-blocs . . .


Et maintenant on peut appliquer l'algorithme principal !