Tri par piles :
Un algorithme polynomial de décision

Adeline Pierrot

Institute of Discrete Mathematics and Geometry, TU Wien

Aléa 2014

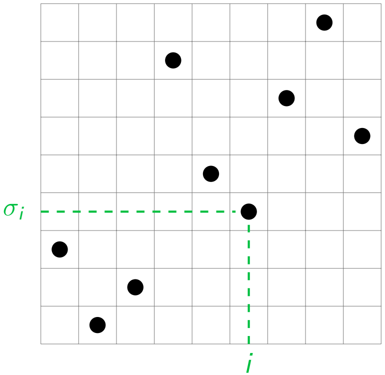
Travail en collaboration avec Dominique Rossin, effectué durant ma thèse au LIAFA

Plan

1. Introduction au tri par piles
2. Cas du tri par sas
3. Cas général

Permutations et motifs

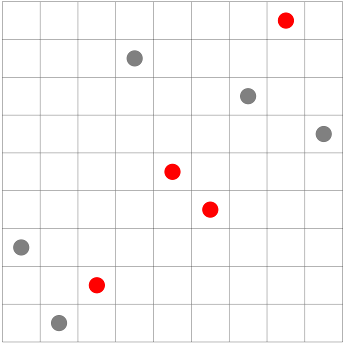
Permutation de taille n : Arrangement des éléments de $[1..n]$
Exemple : $\sigma = 3\,1\,2\,8\,5\,4\,7\,9\,6$

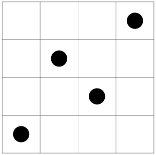


Permutations et motifs

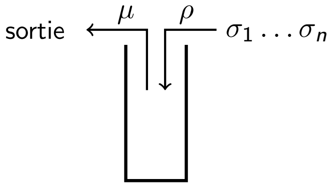
Permutation de taille n : Arrangement des éléments de $[1..n]$
Exemple : $\sigma = 3\,1\,2\,8\,5\,4\,7\,9\,6$

Motif : sous-permutation (cf sous-mot)
Exemple : $1\,3\,2\,4 \preccurlyeq 3\,1\,2\,8\,5\,4\,7\,9\,6$ car $2\,5\,4\,9 \equiv 1\,3\,2\,4$.



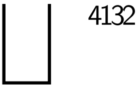


Tri avec une pile

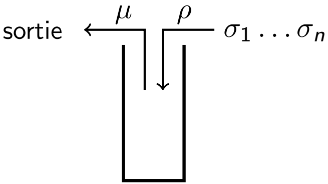


Knuth 1968
(*The Art of computer programming*)

Exemple :

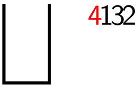


Tri avec une pile

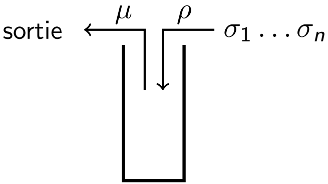


Knuth 1968
(*The Art of computer programming*)

Exemple :

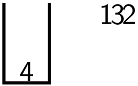


Tri avec une pile

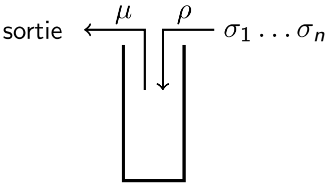


Knuth 1968
(*The Art of computer programming*)

Exemple :

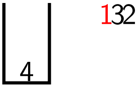


Tri avec une pile

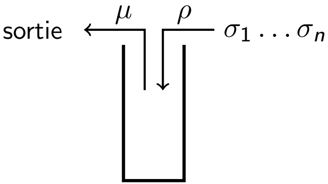


Knuth 1968
(*The Art of computer programming*)

Exemple :

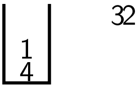


Tri avec une pile



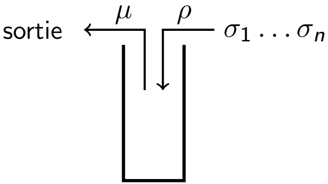
Knuth 1968
(*The Art of computer programming*)

Exemple :

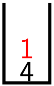


Tri avec une pile

Knuth 1968
(*The Art of computer programming*)



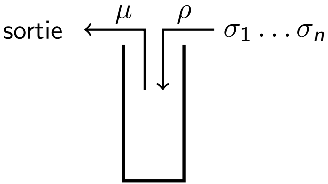
Exemple :



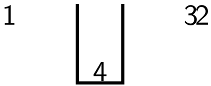
32

Tri avec une pile

Knuth 1968
(*The Art of computer programming*)

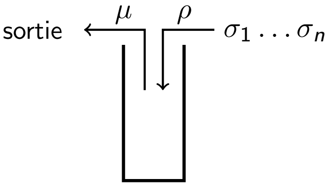


Exemple :

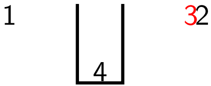


Tri avec une pile

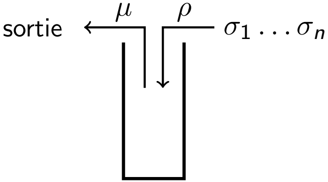
Knuth 1968
(*The Art of computer programming*)



Exemple :



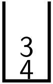
Tri avec une pile



Knuth 1968
(*The Art of computer programming*)

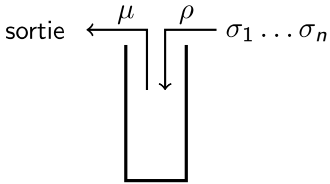
Exemple :

1



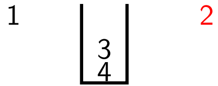
2

Tri avec une pile

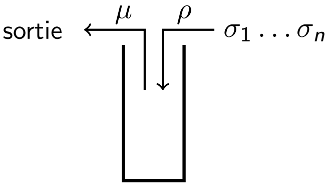


Knuth 1968
(*The Art of computer programming*)

Exemple :

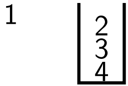


Tri avec une pile

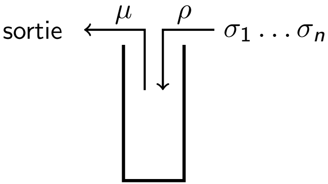


Knuth 1968
(*The Art of computer programming*)

Exemple :

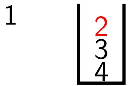


Tri avec une pile

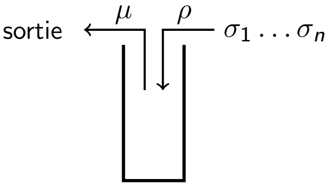


Knuth 1968
(*The Art of computer programming*)

Exemple :



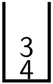
Tri avec une pile



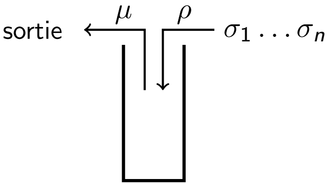
Knuth 1968
(*The Art of computer programming*)

Exemple :

12



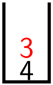
Tri avec une pile



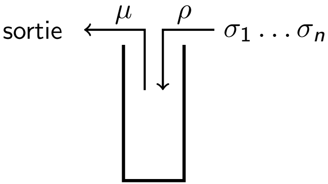
Knuth 1968
(*The Art of computer programming*)

Exemple :

12

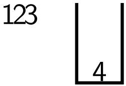


Tri avec une pile

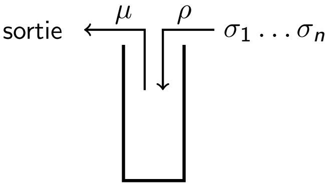


Knuth 1968
(*The Art of computer programming*)

Exemple :

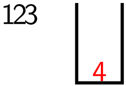


Tri avec une pile

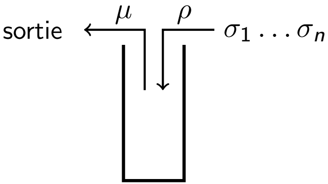


Knuth 1968
(*The Art of computer programming*)

Exemple :

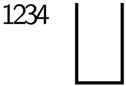


Tri avec une pile



Knuth 1968
(*The Art of computer programming*)

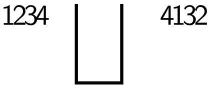
Exemple :



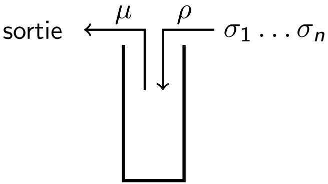
Tri avec une pile

Knuth 1968
(*The Art of computer programming*)

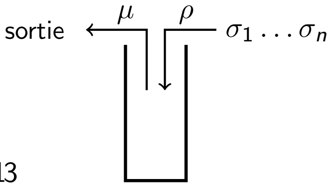
Exemple :



4132 est triable

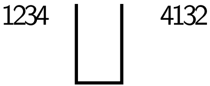


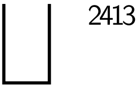
Tri avec une pile



Knuth 1968
(*The Art of computer programming*)

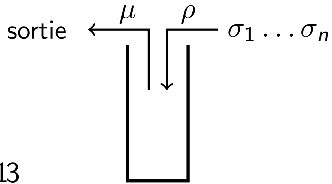
Exemple :





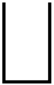
4132 est triable

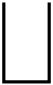
Tri avec une pile



Knuth 1968
(*The Art of computer programming*)

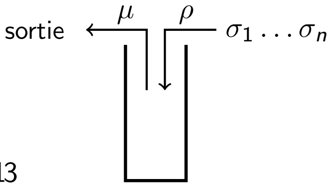
Exemple :

1234  4132

 2413

4132 est triable

Tri avec une pile



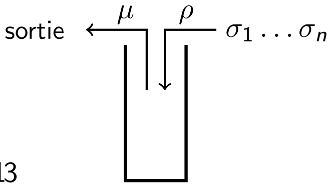
Knuth 1968
(*The Art of computer programming*)

Exemple :

4132 est triable

Tri avec une pile

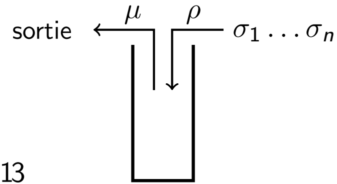
Knuth 1968
(*The Art of computer programming*)



Exemple :

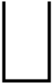
4132 est triable

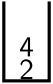
Tri avec une pile



Knuth 1968
(*The Art of computer programming*)

Exemple :

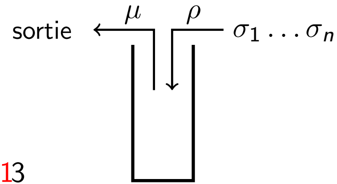
1234  4132



13

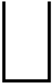
4132 est triable

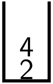
Tri avec une pile



Knuth 1968
(*The Art of computer programming*)

Exemple :

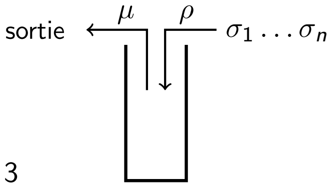
1234  4132



13

4132 est triable

Tri avec une pile

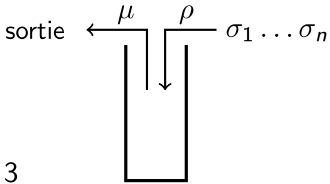


Knuth 1968
(*The Art of computer programming*)

Exemple :

4132 est triable

Tri avec une pile



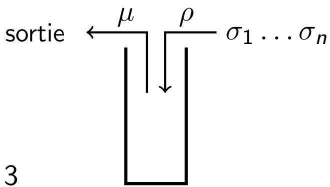
Knuth 1968
(*The Art of computer programming*)

Exemple :

1234   4132   1 4 2   3

4132 est triable

Tri avec une pile



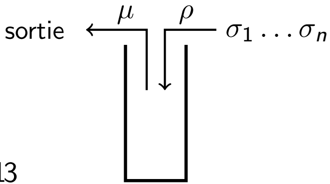
Knuth 1968
(*The Art of computer programming*)

Exemple :

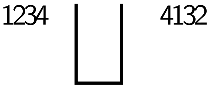
4132 est triable

Tri avec une pile

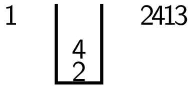
Knuth 1968
(*The Art of computer programming*)



Exemple :

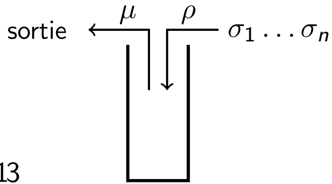


4132 est triable



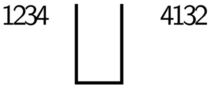
2413 n'est pas triable

Tri avec une pile

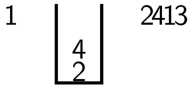


Knuth 1968
(*The Art of computer programming*)

Exemple :



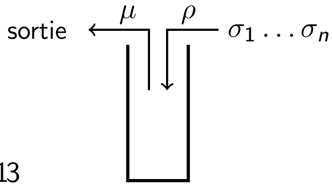
4132 est triable



2413 n'est pas triable

Décision : Au plus une façon de trier une permutation

Tri avec une pile



Knuth 1968
(*The Art of computer programming*)

Exemple :

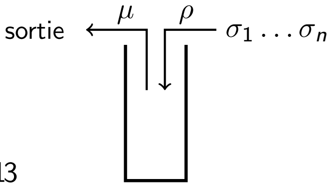
4132 est triable

2413 n'est pas triable

Décision : Au plus une façon de trier une permutation

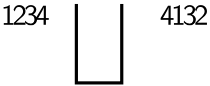
$\rightarrow$ Algorithme glouton linéaire

Tri avec une pile

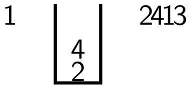


Knuth 1968
(*The Art of computer programming*)

Exemple :



4132 est triable



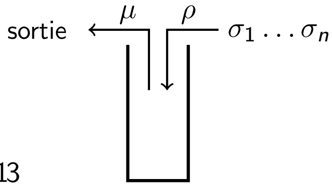
2413 n'est pas triable

Décision : Au plus une façon de trier une permutation

$\rightarrow$ Algorithme glouton linéaire

Caractérisation : σ triable $\Leftrightarrow$ σ évite 231

Tri avec une pile



Knuth 1968
(*The Art of computer programming*)

Exemple :

4132 est triable

2413 n'est pas triable

Décision : Au plus une façon de trier une permutation

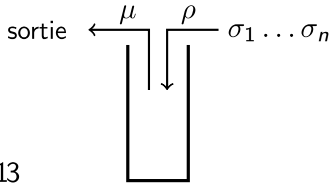
$\rightarrow$ Algorithme glouton linéaire

Caractérisation : σ triable $\Leftrightarrow$ σ évite 231

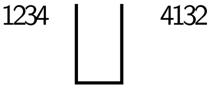
Nombre : $\frac{1}{n+1}\binom{2n}{n} \sim 4^n \;<<\; n! \sim n^n$

Tri avec une pile

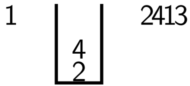
Knuth 1968
(*The Art of computer programming*)



Exemple :



4132 est triable



2413 n'est pas triable

Décision : Au plus une façon de trier une permutation

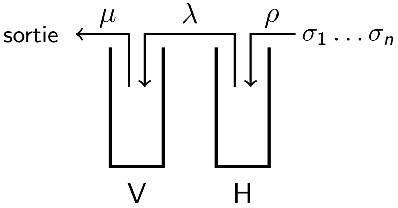
$\rightarrow$ Algorithme glouton linéaire

Caractérisation : σ triable $\Leftrightarrow$ σ évite 231

Nombre : $\frac{1}{n+1}\binom{2n}{n} \sim 4^n \;<<\; n! \sim n^n$

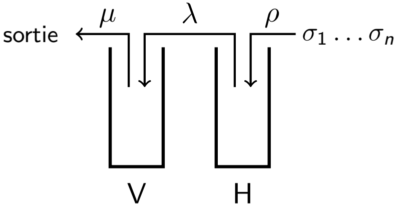
Généralisé par Tarjan, Pratt...

Tri avec deux piles en série



Knuth 1973

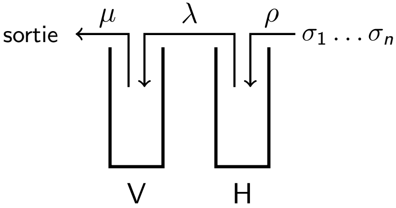
Tri avec deux piles en série



Knuth 1973

Question : Décider si σ est triable.

Tri avec deux piles en série

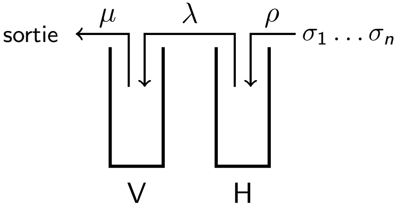


Knuth 1973

Question : Décider si σ est triable.

Algorithme naïf : Tester tous les processus de $\{\rho, \lambda, \mu\}^{3n}$
$\rightarrow$ exponentiel (3^{3n} tests).

Tri avec deux piles en série



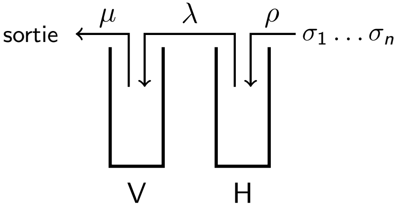
Knuth 1973

Question : Décider si σ est triable.

Algorithme naïf : Tester tous les processus de $\{\rho, \lambda, \mu\}^{3n}$
$\rightarrow$ exponentiel (3^{3n} tests).

Un algorithme polynomial existe-t-il ?

Tri avec deux piles en série



Knuth 1973

Question : Décider si σ est triable.

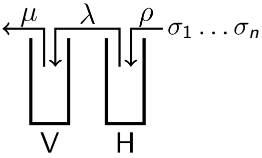
Algorithme naïf : Tester tous les processus de $\{\rho, \lambda, \mu\}^{3n}$
$\rightarrow$ exponentiel (3^{3n} tests).

Un algorithme polynomial existe-t-il ?

Conjecturé NP-complet dans la littérature
[Atkinson, Murphy, Ruskuc (2002)], [Bona (2003)], [Albert, Atkinson, Linton (2010)]

Un tri canonique ?

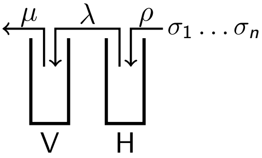
- Tri non unique



Un tri canonique ?

- Tri non unique

Exemple : μ et ρ commutent

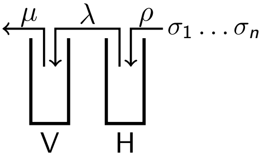


Un tri canonique ?

- Tri non unique

Exemple : μ et ρ commutent

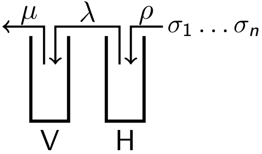
- Tri canonique ?



Un tri canonique ?

- Tri non unique

Exemple : μ et ρ commutent



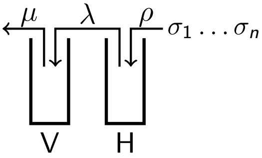
- Tri canonique ?

$\mu \Leftrightarrow$ le sommet de V est le prochain élément qui doit sortir

Un tri canonique ?

- Tri non unique

Exemple : μ et ρ commutent

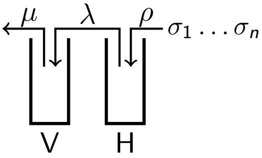


- Tri canonique ?

$\mu \Leftrightarrow$ le sommet de V est le prochain élément qui doit sortir

Certaines permutations ont un nombre exponentiel de tels tri :
$n\ (n-1)\ \ldots\ 1$ admet $2^{(n-1)}$ tris.

Un tri canonique ?



- Tri non unique

Exemple : μ et ρ commutent

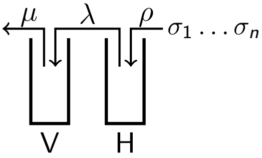
- Tri canonique ?

$\mu \Leftrightarrow$ le sommet de V est le prochain élément qui doit sortir

Certaines permutations ont un nombre exponentiel de tels tri : $n\ (n-1)\ \dots\ 1$ admet $2^{(n-1)}$ tris.

Aucun moyen de décider entre λ et ρ

Un tri canonique ?



- Tri non unique

Exemple : μ et ρ commutent

- Tri canonique ?

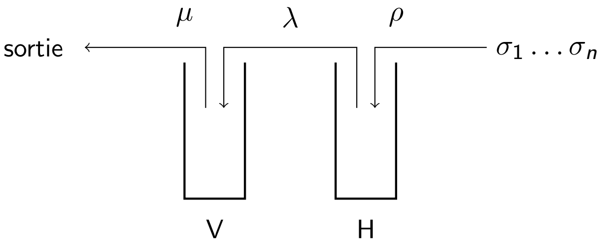
$\mu \Leftrightarrow$ le sommet de V est le prochain élément qui doit sortir

Certaines permutations ont un nombre exponentiel de tels tri :
$n\ (n-1)\ \dots\ 1$ admet $2^{(n-1)}$ tris.

Aucun moyen de décider entre λ et ρ

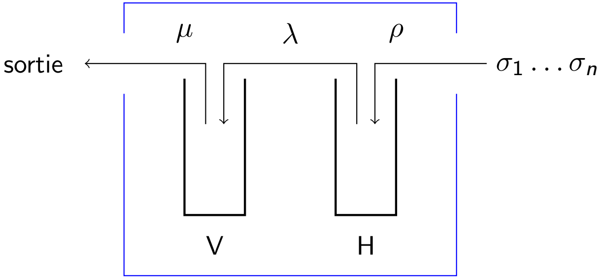
De nombreuses variantes plus faibles étudiées dans la littérature :
Algorithme glouton (West 93), Piles croissantes (Murphy 02)...

Tri par sas



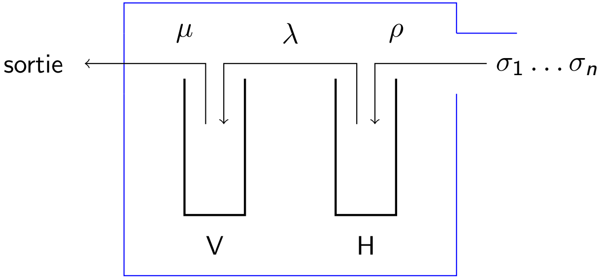
Tri en 2 étapes : la première $\in \{\rho, \lambda\}^*$, la seconde $\in \{\lambda, \mu\}^*$

Tri par sas



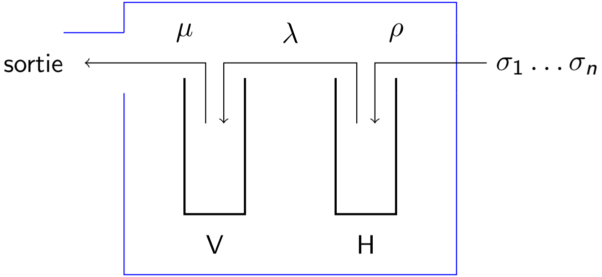
Tri en 2 étapes : la première $\in \{\rho, \lambda\}^*$, la seconde $\in \{\lambda, \mu\}^*$

Tri par sas



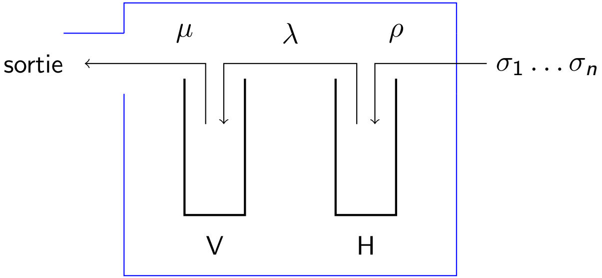
Tri en 2 étapes : la première $\in \{\rho, \lambda\}^*$, la seconde $\in \{\lambda, \mu\}^*$

Tri par sas



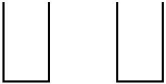
Tri en 2 étapes : la première $\in \{\rho, \lambda\}^*$, la seconde $\in \{\lambda, \mu\}^*$

Tri par sas

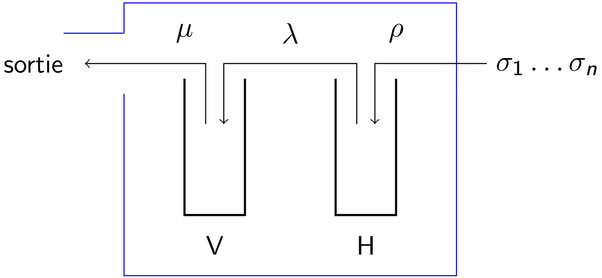


Tri en 2 étapes : la première $\in \{\rho, \lambda\}^*$, la seconde $\in \{\lambda, \mu\}^*$

Exemple : 2 4 1 3 triable par sas :

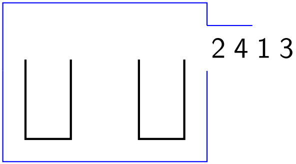


Tri par sas

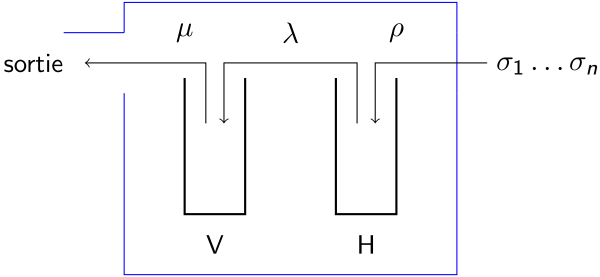


Tri en 2 étapes : la première $\in \{\rho, \lambda\}^*$, la seconde $\in \{\lambda, \mu\}^*$

Exemple : 2 4 1 3 triable par sas :

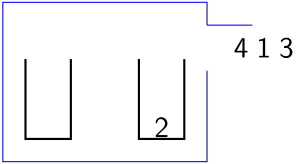


Tri par sas

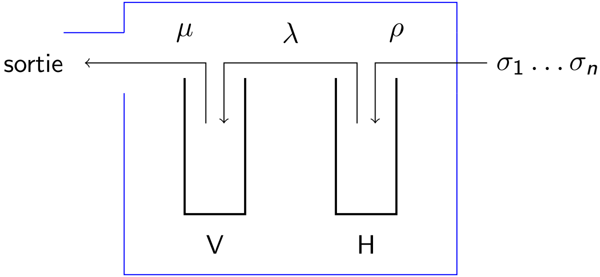


Tri en 2 étapes : la première $\in \{\rho, \lambda\}^*$, la seconde $\in \{\lambda, \mu\}^*$

Exemple : 2 4 1 3 triable par sas :

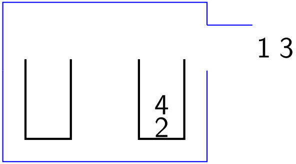


Tri par sas

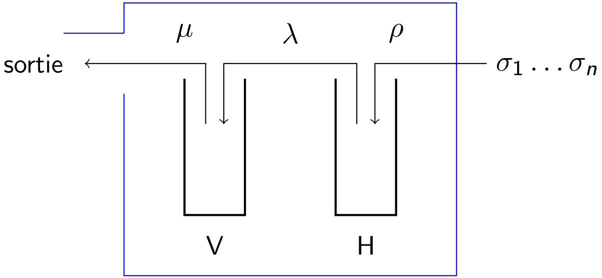


Tri en 2 étapes : la première $\in \{\rho, \lambda\}^*$, la seconde $\in \{\lambda, \mu\}^*$

Exemple : 2 4 1 3 triable par sas :

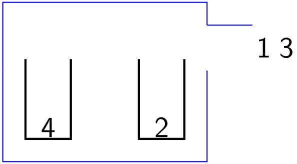


Tri par sas

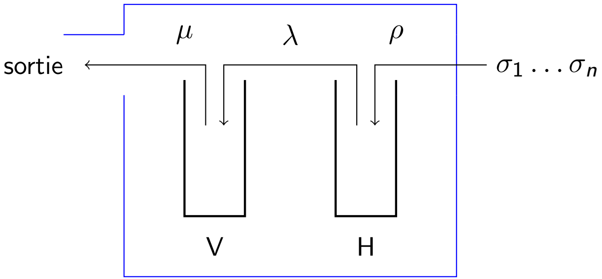


Tri en 2 étapes : la première $\in \{\rho, \lambda\}^*$, la seconde $\in \{\lambda, \mu\}^*$

Exemple : 2 4 1 3 triable par sas :

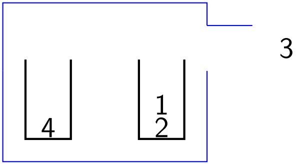


Tri par sas

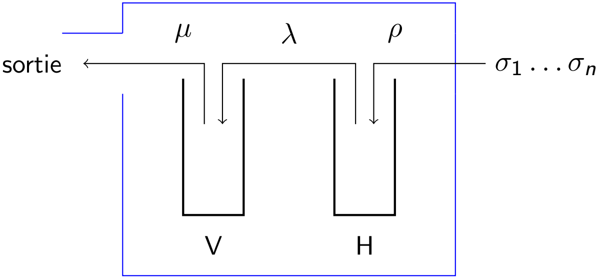


Tri en 2 étapes : la première $\in \{\rho, \lambda\}^*$, la seconde $\in \{\lambda, \mu\}^*$

Exemple : 2 4 1 3 triable par sas :

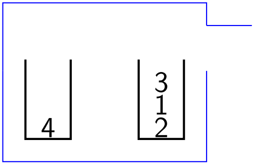


Tri par sas

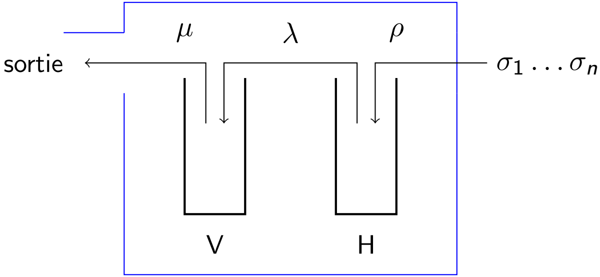


Tri en 2 étapes : la première $\in \{\rho, \lambda\}^*$, la seconde $\in \{\lambda, \mu\}^*$

Exemple : 2 4 1 3 triable par sas :

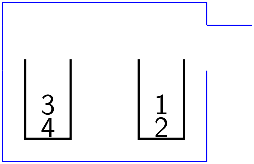


Tri par sas

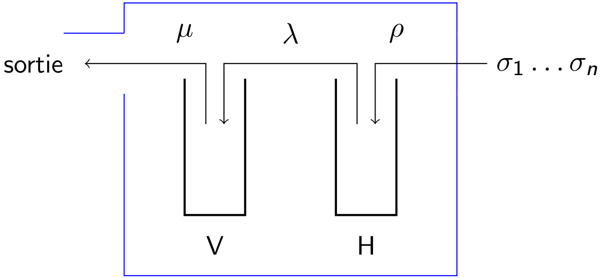


Tri en 2 étapes : la première $\in \{\rho, \lambda\}^*$, la seconde $\in \{\lambda, \mu\}^*$

Exemple : 2 4 1 3 triable par sas :

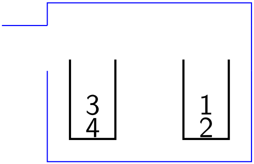


Tri par sas

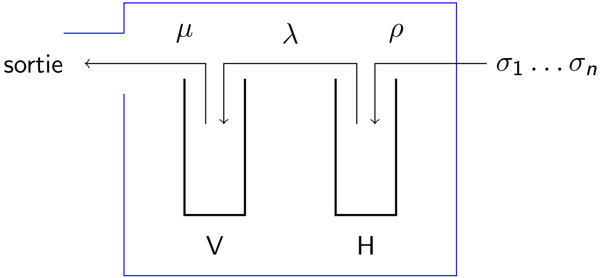


Tri en 2 étapes : la première $\in \{\rho, \lambda\}^*$, la seconde $\in \{\lambda, \mu\}^*$

Exemple : 2 4 1 3 triable par sas :

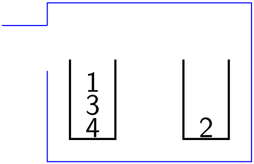


Tri par sas

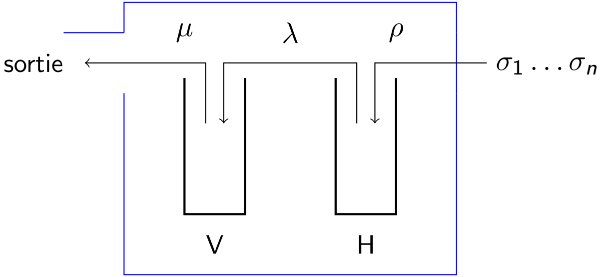


Tri en 2 étapes : la première $\in \{\rho, \lambda\}^*$, la seconde $\in \{\lambda, \mu\}^*$

Exemple : 2 4 1 3 triable par sas :

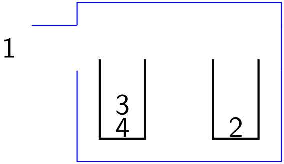


Tri par sas

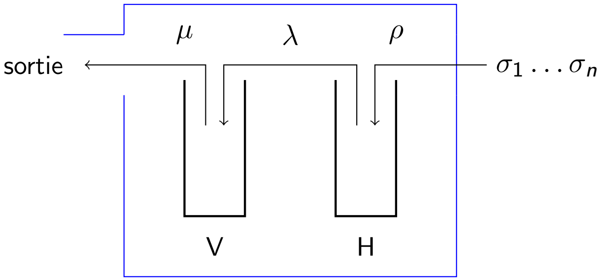


Tri en 2 étapes : la première $\in \{\rho, \lambda\}^*$, la seconde $\in \{\lambda, \mu\}^*$

Exemple : 2 4 1 3 triable par sas :

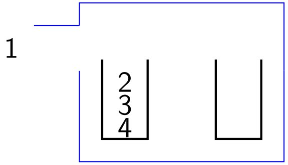


Tri par sas

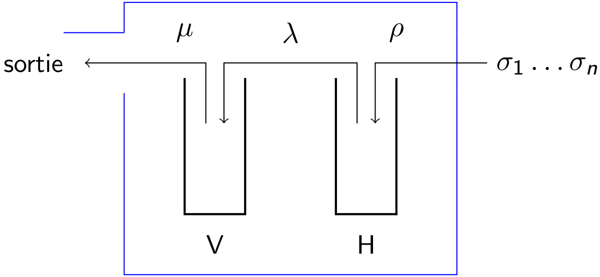


Tri en 2 étapes : la première $\in \{\rho, \lambda\}^*$, la seconde $\in \{\lambda, \mu\}^*$

Exemple : 2 4 1 3 triable par sas :

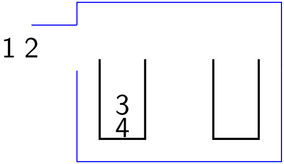


Tri par sas

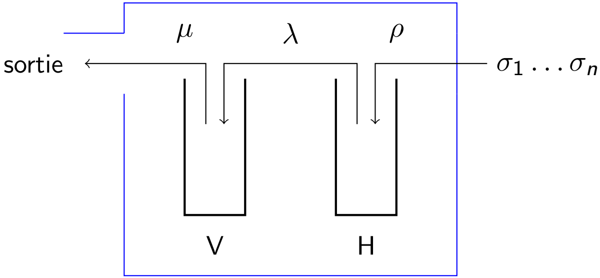


Tri en 2 étapes : la première $\in \{\rho, \lambda\}^*$, la seconde $\in \{\lambda, \mu\}^*$

Exemple : 2 4 1 3 triable par sas :

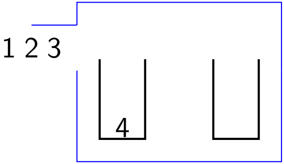


Tri par sas

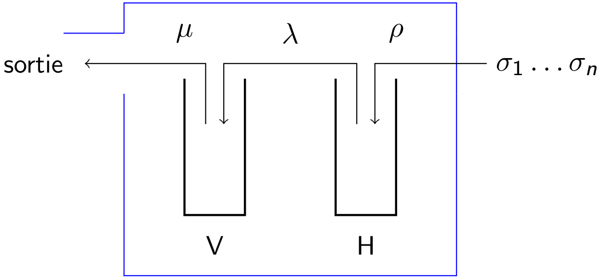


Tri en 2 étapes : la première $\in \{\rho, \lambda\}^*$, la seconde $\in \{\lambda, \mu\}^*$

Exemple : 2 4 1 3 triable par sas :

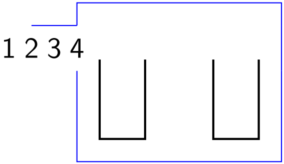


Tri par sas

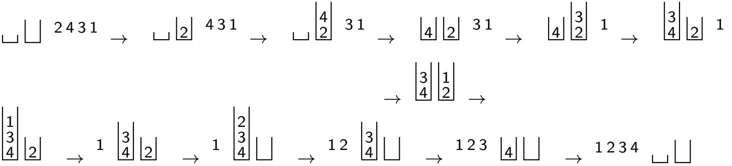


Tri en 2 étapes : la première $\in \{\rho, \lambda\}^*$, la seconde $\in \{\lambda, \mu\}^*$

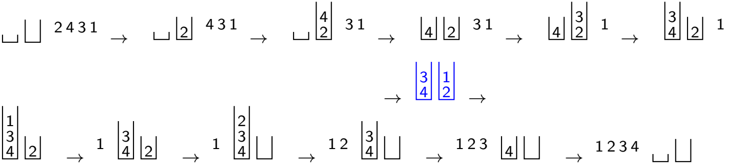
Exemple : 2 4 1 3 triable par sas :



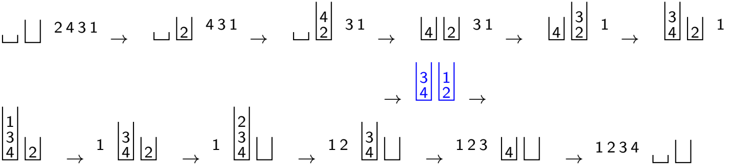
Codage d'un processus de tri



Codage d'un processus de tri

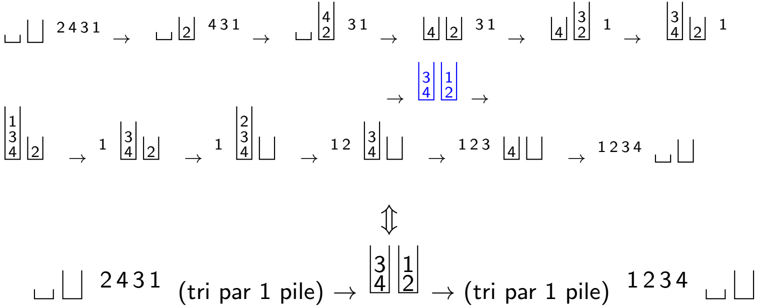


Codage d'un processus de tri



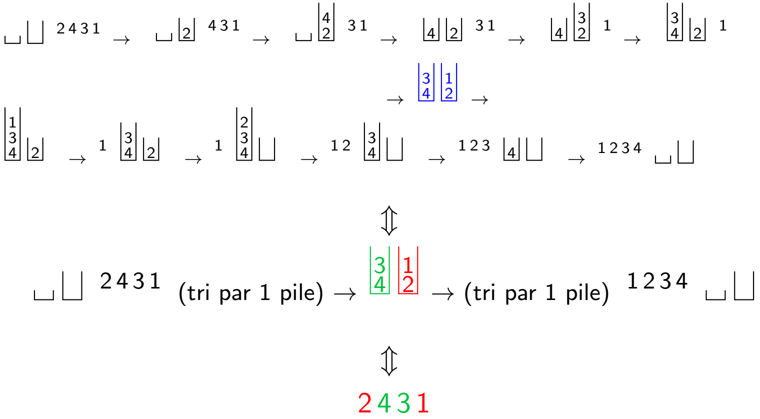
configuration totale

Codage d'un processus de tri



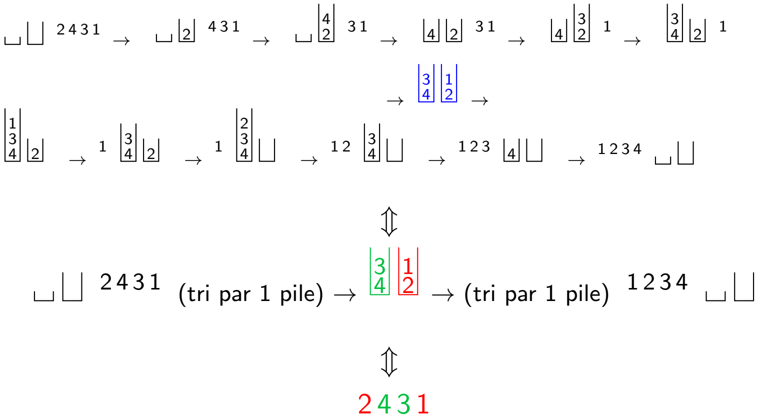
Processus de tri par sas $\Leftrightarrow$ configuration totale

Codage d'un processus de tri



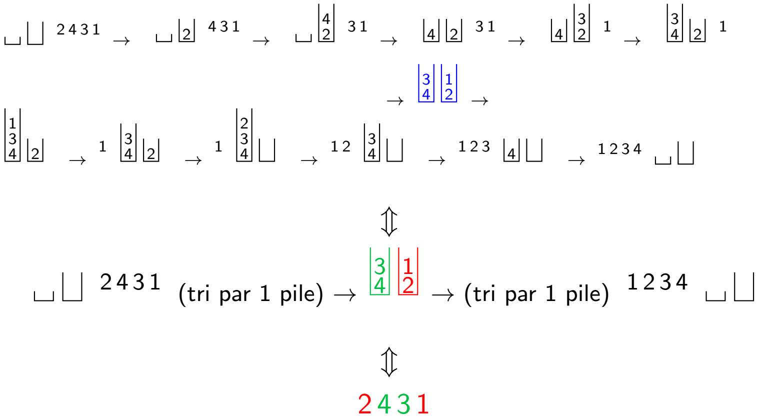
Processus de tri par sas $\Leftrightarrow$ configuration totale

Codage d'un processus de tri



Processus de tri par sas $\Leftrightarrow$ configuration totale $\Leftrightarrow$ coloriage valide

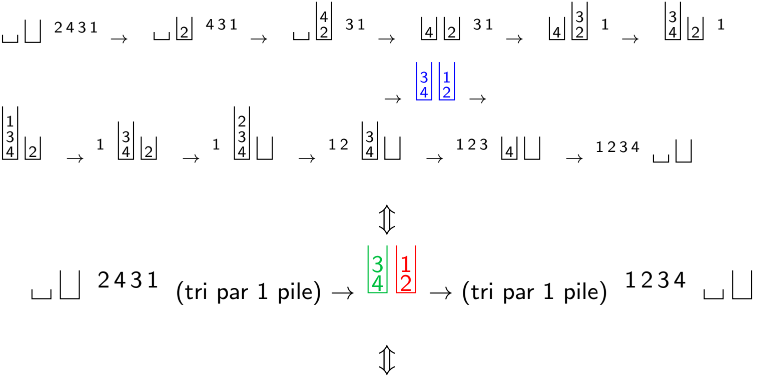
Codage d'un processus de tri



Processus de tri par sas $\Leftrightarrow$ configuration totale $\Leftrightarrow$ coloriage valide

$\rightarrow$ Test en temps linéaire si coloriage valide.

Codage d'un processus de tri



2 4 3 1

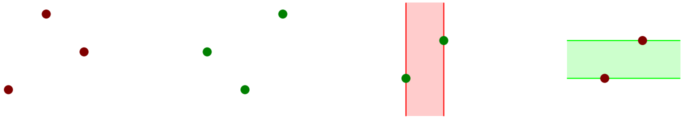
Processus de tri par sas $\Leftrightarrow$ configuration totale $\Leftrightarrow$ coloriage valide

$\rightarrow$ Test en temps linéaire si coloriage valide.
2^n coloriages à tester $\rightarrow$ réduire ce nombre.

Coloriages valides

Coloriage valide : coloriage de σ avec deux couleurs G et R t.q.

- aucun motif 132 dans R
- aucun motif 213 dans G
- aucun point de R situé verticalement entre un motif 12 de G
- aucun point de G situé horizontalement entre un motif 12 de R

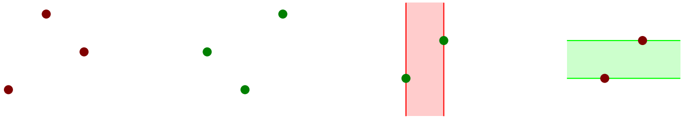


$\Rightarrow$ coloriage avec motifs interdits 132, 213, 1X2 et 2/13

Coloriages valides

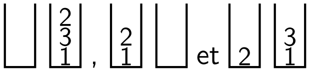
Coloriage valide : coloriage de σ avec deux couleurs G et R t.q.

- aucun motif 132 dans R
- aucun motif 213 dans G
- aucun point de R situé verticalement entre un motif 12 de G
- aucun point de G situé horizontalement entre un motif 12 de R



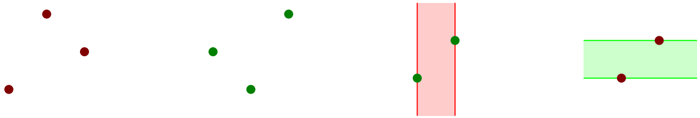
$\Rightarrow$ coloriage avec motifs interdits 132, 213, 1X2 et 2/13

Preuve : R = pile de droite et G = pile de gauche.

Correspondance avec les motifs de pile 

Décomposition

Motifs colorés interdits :

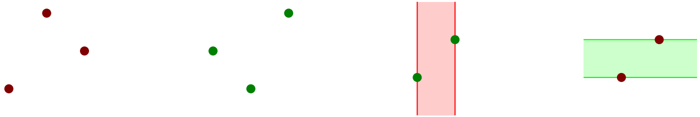


$Col(\sigma)$ = ensemble des coloriages valides de σ

$\sharp Col(n\,(n{-}1)\,\ldots 1) = 2^n$

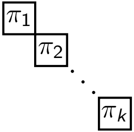
Décomposition

Motifs colorés interdits :



$Col(\sigma)$ = ensemble des coloriages valides de σ

$\sharp Col(n\,(n{-}1)\,\ldots 1) = 2^n$

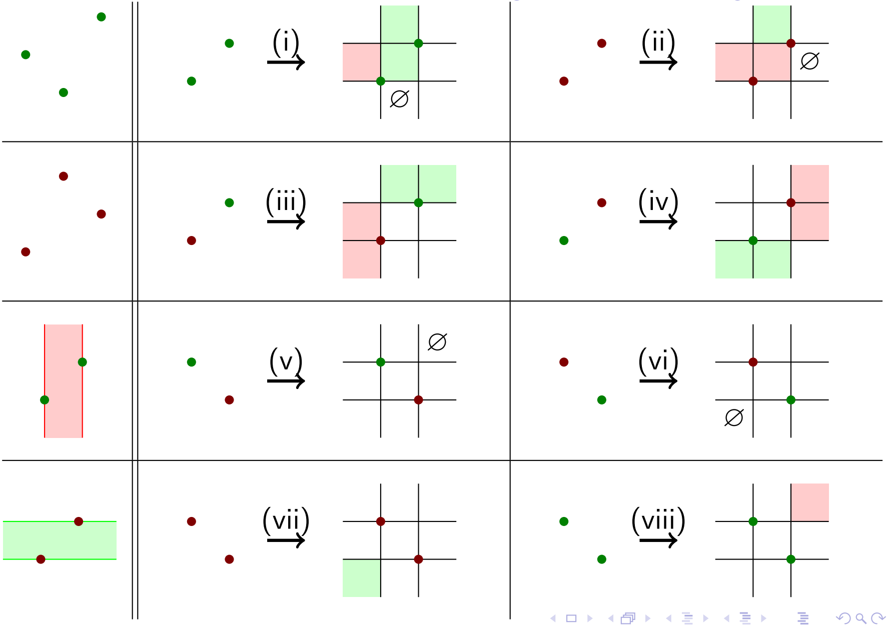
$\ominus[\pi_1, \ldots, \pi_k] =$ 

Exemple : $\ominus[1, \ldots, 1] = n\,(n{-}1)\,\ldots 1$

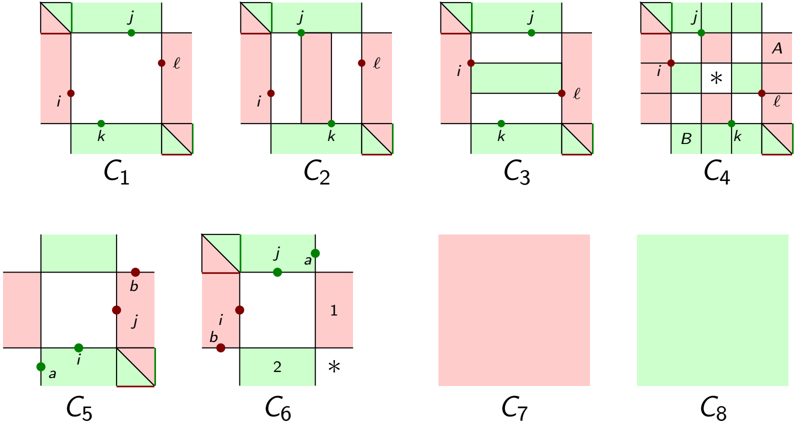
Théorème

$\sigma = \ominus[\pi_1, \ldots, \pi_k] \Rightarrow Col(\sigma) \approx Col(\pi_1) \times \cdots \times Col(\pi_k)$

Motifs colorés interdits $\Rightarrow$ règles de coloriage



8 types de coloriages



Théorème : c coloriage valide de $\sigma \Rightarrow \exists m, p$ t.q. $c = C_m(p)$.

Algorithme quadratique

Algorithme :

Entrée : σ $\ominus$-indécomposable.
Sortie : Tous les coloriages valides de σ:

> Pour i de 1 à 8
>> Pour p de 1 à $n = |\sigma|$
>>> Tester si $C_i(p)$ est un coloriage valide de σ

Algorithme quadratique

Algorithme :

Entrée : σ $\ominus$-indécomposable.
Sortie : Tous les coloriages valides de σ:

Pour i de 1 à 8
    Pour p de 1 à $n = |\sigma|$
        Tester si $C_i(p)$ est un coloriage valide de σ

Complexité :

Tester si coloriage valide $=$ linéaire

Algorithme quadratique

Algorithme :

Entrée : σ $\ominus$-indécomposable.
Sortie : Tous les coloriages valides de σ:

> Pour i de 1 à 8
>> Pour p de 1 à $n = |\sigma|$
>>> Tester si $C_i(p)$ est un coloriage valide de σ

Complexité :

Tester si coloriage valide = linéaire

σ $\ominus$-indécomposable $\Rightarrow |Col(\sigma)| \leq 8|\sigma|$ calculé en $\mathcal{O}(|\sigma|^2)$

Algorithme quadratique

Algorithme :

Entrée : σ $\ominus$-indécomposable.
Sortie : Tous les coloriages valides de σ:

Pour i de 1 à 8
    Pour p de 1 à $n = |\sigma|$
        Tester si $C_i(p)$ est un coloriage valide de σ

Complexité :

Tester si coloriage valide $=$ linéaire

σ $\ominus$-indécomposable $\Rightarrow |Col(\sigma)| \leq 8|\sigma|$ calculé en $\mathcal{O}(|\sigma|^2)$

$\sigma = \ominus[\pi_1, \ldots, \pi_k] \Rightarrow Col(\sigma) \approx Col(\pi_1) \times \cdots \times Col(\pi_k)$

$\rightarrow Col(\sigma)$ décrit par $\big(Col(\pi_1), \ldots, Col(\pi_k)\big)$

Algorithme quadratique

Algorithme :

Entrée : σ $\ominus$-indécomposable.
Sortie : Tous les coloriages valides de σ:

Pour i de 1 à 8
  Pour p de 1 à $n = |\sigma|$
    Tester si $C_i(p)$ est un coloriage valide de σ

Complexité :

Tester si coloriage valide $=$ linéaire

σ $\ominus$-indécomposable $\Rightarrow |Col(\sigma)| \leq 8|\sigma|$ calculé en $\mathcal{O}(|\sigma|^2)$

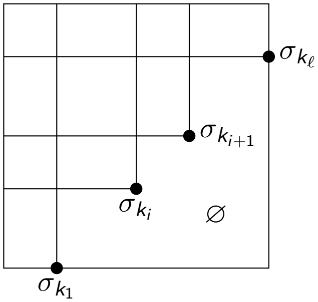
$\sigma = \ominus[\pi_1, \ldots, \pi_k] \Rightarrow Col(\sigma) \approx Col(\pi_1) \times \cdots \times Col(\pi_k)$

$\rightarrow$ $Col(\sigma)$ décrit par $\big(Col(\pi_1), \ldots, Col(\pi_k)\big)$

$\rightarrow$ calculé en temps quadratique : $8|\pi_1|^2 + \cdots + 8|\pi_k|^2 \leq 8|\sigma|^2$.

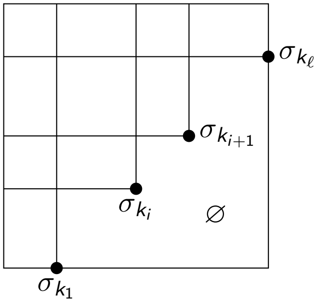
Du tri par sas au tri général

σ_{k_i} = minima droite-gauche de σ



σ_{k_i} = minima droite-gauche de σ

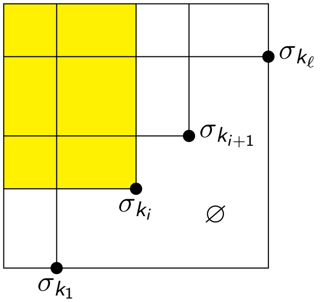
Configuration quand σ_{k_i} entre dans les piles



Du tri par sas au tri général

σ_{k_i} = minima droite-gauche de σ

Configuration quand σ_{k_i} entre dans les piles

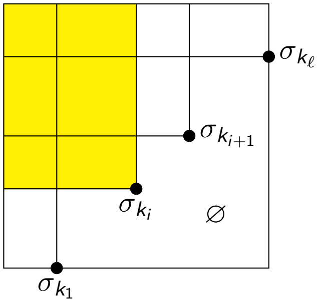


$$\sigma^{(i)} = \{\sigma_j \mid j < k_i \text{ et } \sigma_j > \sigma_{k_i}\}$$

Du tri par sas au tri général

σ_{k_i} = minima droite-gauche de σ

Configuration quand σ_{k_i} entre dans les piles = totale pour $\sigma^{(i)}$

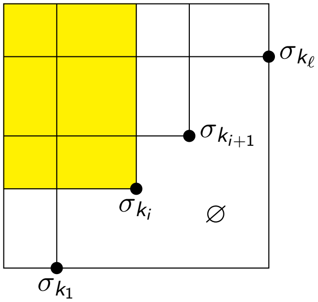


$$\sigma^{(i)} = \{\sigma_j \mid j < k_i \text{ et } \sigma_j > \sigma_{k_i}\}$$

Du tri par sas au tri général

σ_{k_i} = minima droite-gauche de σ

Configuration quand σ_{k_i} entre dans les piles = totale pour $\sigma^{(i)}$



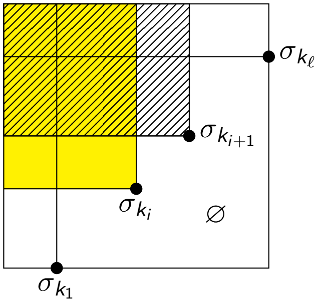
$$\sigma^{(i)} = \{\sigma_j \mid j < k_i \text{ et } \sigma_j > \sigma_{k_i}\}$$

σ triable $\Rightarrow \sigma^{(i)}$ triable par sas $\forall i$

Du tri par sas au tri général

σ_{k_i} = minima droite-gauche de σ

Configuration quand σ_{k_i} entre dans les piles = totale pour $\sigma^{(i)}$



$$\sigma^{(i)} = \{\sigma_j \mid j < k_i \text{ et } \sigma_j > \sigma_{k_i}\}$$

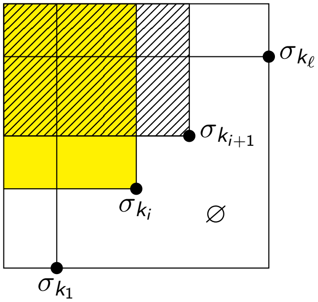
σ triable $\Rightarrow$ $\sigma^{(i)}$ triable par sas $\forall i$

Les tris par sas des $\sigma^{(i)}$ doivent être compatibles

Du tri par sas au tri général

σ_{k_i} = minima droite-gauche de σ

Configuration quand σ_{k_i} entre dans les piles = totale pour $\sigma^{(i)}$



$\sigma^{(i)} = \{\sigma_j \mid j < k_i \text{ et } \sigma_j > \sigma_{k_i}\}$

σ triable $\Rightarrow$ $\sigma^{(i)}$ triable par sas $\forall i$

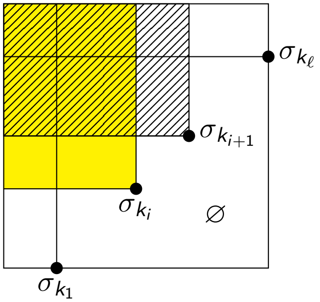
Les tris par sas des $\sigma^{(i)}$ doivent être compatibles

Algorithme récursif

Du tri par sas au tri général

σ_{k_i} = minima droite-gauche de σ

Configuration quand σ_{k_i} entre dans les piles = totale pour $\sigma^{(i)}$



$\sigma^{(i)} = \{\sigma_j \mid j < k_i \text{ et } \sigma_j > \sigma_{k_i}\}$

σ triable $\Rightarrow$ $\sigma^{(i)}$ triable par sas $\forall i$

Les tris par sas des $\sigma^{(i)}$ doivent être compatibles

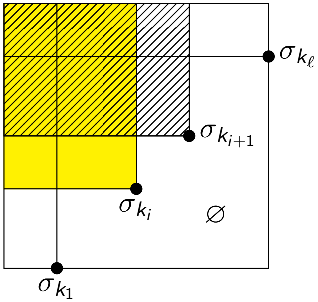
Algorithme récursif

Test de compatibilité = linéaire.

Du tri par sas au tri général

σ_{k_i} = minima droite-gauche de σ

Configuration quand σ_{k_i} entre dans les piles = totale pour $\sigma^{(i)}$



$\sigma^{(i)} = \{\sigma_j \mid j < k_i \text{ et } \sigma_j > \sigma_{k_i}\}$

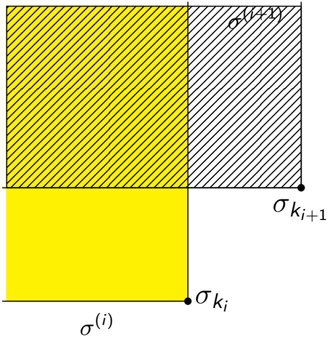
σ triable $\Rightarrow$ $\sigma^{(i)}$ triable par sas $\forall i$

Les tris par sas des $\sigma^{(i)}$ doivent être compatibles

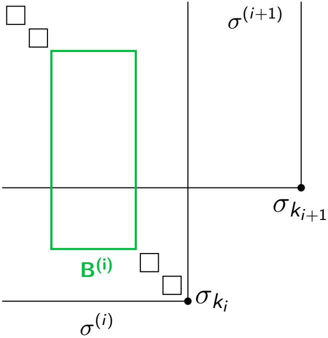
Algorithme récursif

Test de compatibilité = linéaire. Nombre exponentiel de tests?

Réduire le nombre de tests

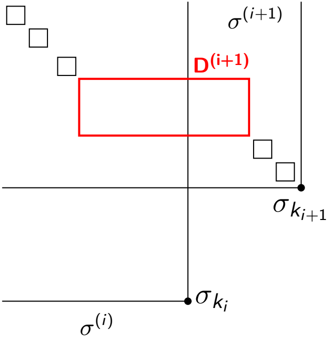


Réduire le nombre de tests



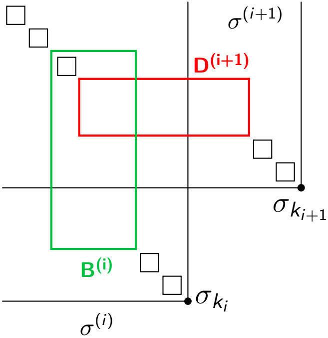
$$Col(\sigma^{(i)}) \approx$$
$$Col(B_1^{(i)}) \times \cdots \times Col(B_k^{(i)})$$

Réduire le nombre de tests



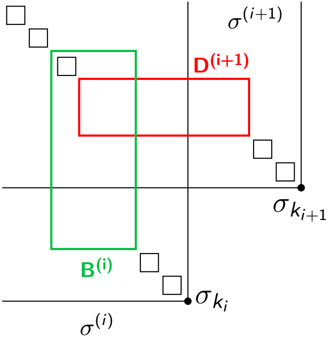
$$Col(\sigma^{(i)}) \approx Col(B_1^{(i)}) \times \cdots \times Col(B_k^{(i)})$$

Réduire le nombre de tests



$$Col(\sigma^{(i)}) \approx$$
$$Col(B_1^{(i)}) \times \cdots \times Col(B_k^{(i)})$$

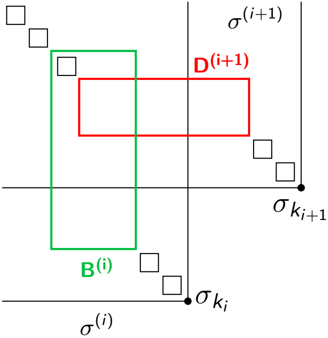
Réduire le nombre de tests



$$Col(\sigma^{(i)}) \approx Col(B_1^{(i)}) \times \cdots \times Col(B_k^{(i)})$$

Suffisant de tester la compatibilité sur $B^{(i)}$ et $D^{(i+1)}$

Réduire le nombre de tests

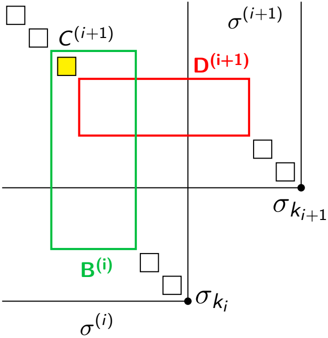


$$Col(\sigma^{(i)}) \approx$$
$$Col(B_1^{(i)}) \times \cdots \times Col(B_k^{(i)})$$

Suffisant de tester la compatibilité sur $B^{(i)}$ et $D^{(i+1)}$

$\rightarrow$ nombre linéaire de tests

Réduire le nombre de tests



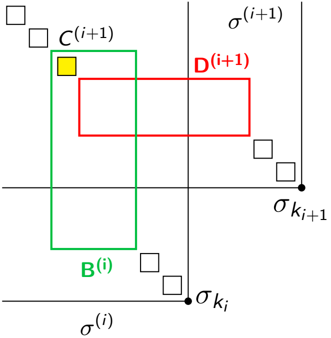
$Col(\sigma^{(i)}) \approx$
$Col(B_1^{(i)}) \times \cdots \times Col(B_k^{(i)})$

Suffisant de tester la compatibilité sur $B^{(i)}$ et $D^{(i+1)}$

$\rightarrow$ nombre linéaire de tests

Configurations de $C^{(i+1)}$ liées avec celles de $D^{(i+1)}$

Réduire le nombre de tests



$Col(\sigma^{(i)}) \approx$
$Col(B_1^{(i)}) \times \cdots \times Col(B_k^{(i)})$

Suffisant de tester la compatibilité sur $B^{(i)}$ et $D^{(i+1)}$

$\rightarrow$ nombre linéaire de tests

Configurations de $C^{(i+1)}$ liées avec celles de $D^{(i+1)}$

$\rightarrow$ graphe de tri

Conclusion

Algorithme polynomial de décision pour 2 piles en série

- Introduction d'une nouvelle notion : le tri par sas
- Caractérisation par bicoloriage à motifs exclus
- Algorithme quadratique optimal calculant tous les tris par sas
- Décomposition suivant les minima droite-gauche
- On obtient tous les tris vérifiant une propriété P.

Perspectives

- Simplification de l'algorithme ?

Perspectives

- Simplification de l'algorithme ?
- Permutations triables par 2 piles en série: caractérisation ? Énumération ?

Perspectives

- Simplification de l'algorithme ?
- Permutations triables par 2 piles en série: caractérisation ? Énumération ?
- Énumération des permutations triables par sas ?

Perspectives

- Simplification de l'algorithme ?
- Permutations triables par 2 piles en série: caractérisation ? Énumération ?
- Énumération des permutations triables par sas ?

- Complexité de la décision pour k piles en série :

Perspectives

- Simplification de l'algorithme ?
- Permutations triables par 2 piles en série: caractérisation ? Énumération ?
- Énumération des permutations triables par sas ?

- Complexité de la décision pour k piles en série :
  - Généralisation pour plus de 2 piles ?

Perspectives

- Simplification de l'algorithme ?
- Permutations triables par 2 piles en série: caractérisation ? Énumération ?
- Énumération des permutations triables par sas ?

- Complexité de la décision pour k piles en série :
  - Généralisation pour plus de 2 piles ?
  - Pour k fixé, le problème est-il toujours polynomial ? Ou existe-t-il un seuil ?

Perspectives

- Simplification de l'algorithme ?
- Permutations triables par 2 piles en série: caractérisation ? Énumération ?
- Énumération des permutations triables par sas ?

- Complexité de la décision pour k piles en série :
  - Généralisation pour plus de 2 piles ?
  - Pour k fixé, le problème est-il toujours polynomial ? Ou existe-t-il un seuil ?

Merci !