Tris, comparaisons et efficacité

Vincent Jugé

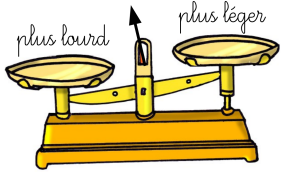
16/08/2022

Le contexte...

Pendant ses vacances à Valbonne, Martin le numismate a acheté n pièces de monnaie, qu'il a alignées de gauche à droite sur une table. Elles se ressemblent beaucoup, mais sont toutes de poids légèrement différents. Il souhaite les trier par ordre croissant de poids.

Le contexte...

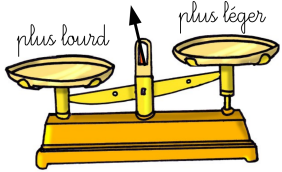
Pendant ses vacances à Valbonne, Martin le numismate a acheté n pièces de monnaie, qu'il a alignées de gauche à droite sur une table. Elles se ressemblent beaucoup, mais sont toutes de poids légèrement différents. Il souhaite les trier par ordre croissant de poids.



Dans ses bagages, Martin dispose d'une balance Roberval très précise, qui lui permet de comparer deux pièces et de décider laquelle est la plus lourde. Toute autre manière de peser ou de comparer les pièces sera trop imprécise pour être exploitable.

Le contexte...

Pendant ses vacances à Valbonne, Martin le numismate a acheté n pièces de monnaie, qu'il a alignées de gauche à droite sur une table. Elles se ressemblent beaucoup, mais sont toutes de poids légèrement différents. Il souhaite les trier par ordre croissant de poids.

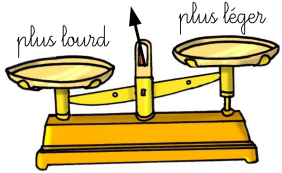


Dans ses bagages, Martin dispose d'une balance Roberval très précise, qui lui permet de comparer deux pièces et de décider laquelle est la plus lourde. Toute autre manière de peser ou de comparer les pièces sera trop imprécise pour être exploitable.

Combien de pesées Martin devra-t-il effectuer ?

Le contexte...

Pendant ses vacances à Valbonne, Martin le numismate a acheté n pièces de monnaie, qu'il a alignées de gauche à droite sur une table. Elles se ressemblent beaucoup, mais sont toutes de poids légèrement différents. Il souhaite les trier par ordre croissant de poids.



Dans ses bagages, Martin dispose d'une balance Roberval très précise, qui lui permet de comparer deux pièces et de décider laquelle est la plus lourde. Toute autre manière de peser ou de comparer les pièces sera trop imprécise pour être exploitable.

Combien de pesées Martin devra-t-il effectuer ?

☛ Pour de nombreux usages, $n \approx 10^6$.

Quelques opérations pas trop méchantes...

Avec sa balance, Martin peut, en $n-1$ pesées :

Quelques opérations pas trop méchantes...

Avec sa balance, Martin peut, en $n-1$ pesées :

- vérifier si une pièce est plus légère que toutes les autres
  - ☛ il la compare aux $n-1$ autres pièces

Quelques opérations pas trop méchantes...

Avec sa balance, Martin peut, en $n-1$ pesées :

- vérifier si une pièce est plus légère que toutes les autres
  ☛ il la compare aux $n-1$ autres pièces
- vérifier si ses pièces sont triées dans l'ordre croissant
  ☛ il vérifie que chaque pièce est plus légère que sa voisine de droite

Quelques opérations pas trop méchantes...

Avec sa balance, Martin peut, en $n-1$ pesées :

- vérifier si une pièce est plus légère que toutes les autres
  ☛ il la compare aux $n-1$ autres pièces
- vérifier si ses pièces sont triées dans l'ordre croissant
  ☛ il vérifie que chaque pièce est plus légère que sa voisine de droite
- trouver la pièce la plus légère
  ☛ il compare la pièce la plus légère qu'il a trouvée avec une pièce pas encore testée

Algorithme de tri n°1 : le bogosort

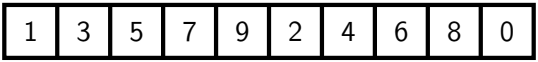
Recette à suivre :

1. Martin mélange ses pièces dans un ordre aléatoire.
2. Il vérifie qu'elles sont bien triées.
3. Si oui, il a gagné. Sinon, il revient à l'étape 1.

Algorithme de tri n°1 : le bogosort

Recette à suivre :

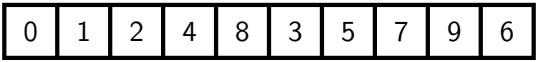
1. Martin mélange ses pièces dans un ordre aléatoire.
2. Il vérifie qu'elles sont bien triées.
3. Si oui, il a gagné. Sinon, il revient à l'étape 1.



Algorithme de tri n°1 : le bogosort

Recette à suivre :

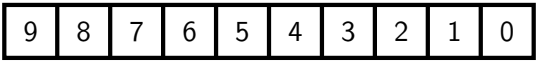
1. Martin mélange ses pièces dans un ordre aléatoire.
2. Il vérifie qu'elles sont bien triées.
3. Si oui, il a gagné. Sinon, il revient à l'étape 1.



Algorithme de tri n°1 : le bogosort

Recette à suivre :

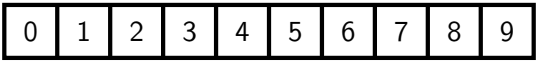
1. Martin mélange ses pièces dans un ordre aléatoire.
2. Il vérifie qu'elles sont bien triées.
3. Si oui, il a gagné. Sinon, il revient à l'étape 1.



Algorithme de tri n°1 : le bogosort

Recette à suivre :

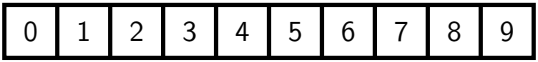
1. Martin mélange ses pièces dans un ordre aléatoire.
2. Il vérifie qu'elles sont bien triées.
3. Si oui, il a gagné. Sinon, il revient à l'étape 1.



Algorithme de tri n°1 : le bogosort

Recette à suivre :

1. Martin mélange ses pièces dans un ordre aléatoire.
2. Il vérifie qu'elles sont bien triées.
3. Si oui, il a gagné. Sinon, il revient à l'étape 1.



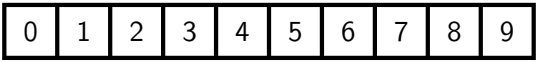
Combien de pesées Martin va-t-il effectuer ?

1. Dans le meilleur cas : $n - 1$ pesées

Algorithme de tri n°1 : le bogosort

Recette à suivre :

1. Martin mélange ses pièces dans un ordre aléatoire.
2. Il vérifie qu'elles sont bien triées.
3. Si oui, il a gagné. Sinon, il revient à l'étape 1.



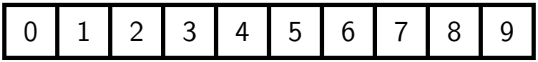
Combien de pesées Martin va-t-il effectuer ?

1. Dans le meilleur cas : $n - 1$ pesées
2. Dans le pire cas : ∞ pesées

Algorithme de tri n°1 : le bogosort

Recette à suivre :

1. Martin mélange ses pièces dans un ordre aléatoire.
2. Il vérifie qu'elles sont bien triées.
3. Si oui, il a gagné. Sinon, il revient à l'étape 1.



Combien de pesées Martin va-t-il effectuer ?

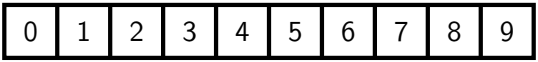
1. Dans le meilleur cas : $n - 1$ pesées
2. Dans le pire cas : ∞ pesées
3. Dans le cas moyen : $\approx e \times n!$ pesées

☛ $n! = 1 \times 2 \times \cdots \times n$: c'est très grand !

Algorithme de tri n°1 : le bogosort

Recette à suivre :

1. Martin mélange ses pièces dans un ordre aléatoire.
2. Il vérifie qu'elles sont bien triées.
3. Si oui, il a gagné. Sinon, il revient à l'étape 1.



Combien de pesées Martin va-t-il effectuer ?

1. Dans le meilleur cas : $n - 1$ pesées
2. Dans le pire cas : ∞ pesées
3. Dans le cas moyen : $\approx e \times n!$ pesées

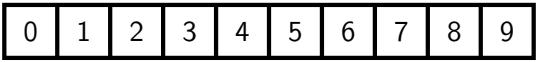
6235300 pesées pour $n = 10$

☛ $n! = 1 \times 2 \times \cdots \times n$: c'est très grand !

Algorithme de tri n°1 : le bogosort

Recette à suivre :

1. Martin mélange ses pièces dans un ordre aléatoire.
2. Il vérifie qu'elles sont bien triées.
3. Si oui, il a gagné. Sinon, il revient à l'étape 1.



Combien de pesées Martin va-t-il effectuer ?

1. Dans le meilleur cas : $n-1$ pesées
2. Dans le pire cas : ∞ pesées
3. Dans le cas moyen : $\approx e \times n!$ pesées

$\approx 136 \times 10^{5565707}$ pesées pour $n = 10^6$

☛ $n! = 1 \times 2 \times \cdots \times n$: c'est très grand !

Algorithme de tri n°2 : le tri par sélection

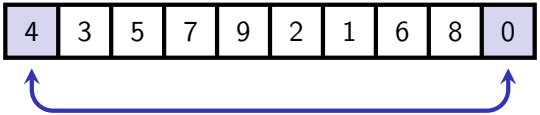
Recette à suivre :

1. Martin identifie la pièce la plus légère.
2. Il la place en première position.
3. Il trie les $n-1$ pièces de droite.

Algorithme de tri n°2 : le tri par sélection

Recette à suivre :

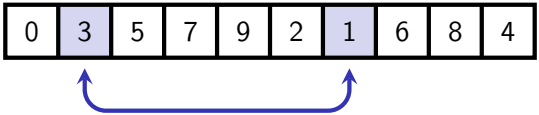
1. Martin identifie la pièce la plus légère.
2. Il la place en première position.
3. Il trie les $n-1$ pièces de droite.



Algorithme de tri n°2 : le tri par sélection

Recette à suivre :

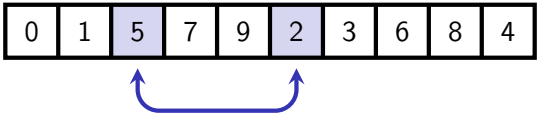
1. Martin identifie la pièce la plus légère.
2. Il la place en première position.
3. Il trie les $n-1$ pièces de droite.



Algorithme de tri n°2 : le tri par sélection

Recette à suivre :

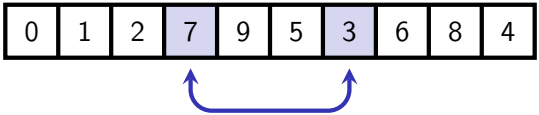
1. Martin identifie la pièce la plus légère.
2. Il la place en première position.
3. Il trie les $n-1$ pièces de droite.



Algorithme de tri n°2 : le tri par sélection

Recette à suivre :

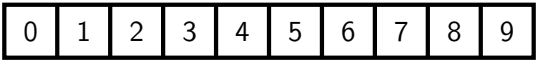
1. Martin identifie la pièce la plus légère.
2. Il la place en première position.
3. Il trie les $n-1$ pièces de droite.



Algorithme de tri n°2 : le tri par sélection

Recette à suivre :

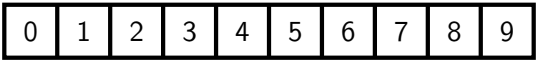
1. Martin identifie la pièce la plus légère.
2. Il la place en première position.
3. Il trie les $n-1$ pièces de droite.



Algorithme de tri n°2 : le tri par sélection

Recette à suivre :

1. Martin identifie la pièce la plus légère.
2. Il la place en première position.
3. Il trie les $n-1$ pièces de droite.



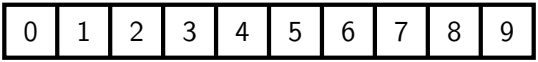
Combien de comparaisons Martin va-t-il effectuer ?

1. Dans le meilleur cas : $(n-1)+(n-2)+\cdots+1+0 = n(n-1)/2$ pesées

Algorithme de tri n°2 : le tri par sélection

Recette à suivre :

1. Martin identifie la pièce la plus légère.
2. Il la place en première position.
3. Il trie les $n-1$ pièces de droite.



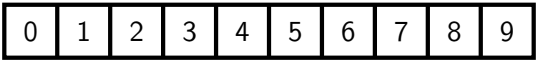
Combien de comparaisons Martin va-t-il effectuer ?

1. Dans le meilleur cas : $(n-1)+(n-2)+\cdots+1+0 = n(n-1)/2$ pesées
2. Dans le pire cas : $n(n-1)/2$ pesées
3. Dans le cas moyen : $n(n-1)/2$ pesées

Algorithme de tri n°2 : le tri par sélection

Recette à suivre :

1. Martin identifie la pièce la plus légère.
2. Il la place en première position.
3. Il trie les $n-1$ pièces de droite.



Combien de comparaisons Martin va-t-il effectuer ?

1. Dans le meilleur cas : $(n-1)+(n-2)+\cdots+1+0 = n(n-1)/2$ pesées
2. Dans le pire cas : $n(n-1)/2$ pesées
3. Dans le cas moyen : $n(n-1)/2$ pesées

$499\,999\,500\,000$ pesées pour $n = 10^6$

Algorithme de tri n°3 : le tri rapide

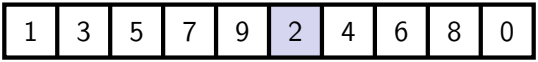
Recette à suivre :

1. Martin choisit une pièce au hasard (il l'appelle la pièce pivot).
2. Il la compare à toutes les autres pièces, qu'il sépare en pièces légères et lourdes.
3. Il met les pièces légères à gauche, puis la pièce pivot, puis les pièces lourdes.
4. Il trie les pièces légères et les pièces lourdes.

Algorithme de tri n°3 : le tri rapide

Recette à suivre :

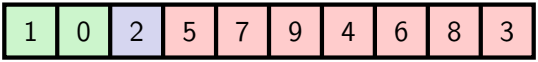
1. Martin choisit une pièce au hasard (il l'appelle la pièce pivot).
2. Il la compare à toutes les autres pièces, qu'il sépare en pièces légères et lourdes.
3. Il met les pièces légères à gauche, puis la pièce pivot, puis les pièces lourdes.
4. Il trie les pièces légères et les pièces lourdes.



Algorithme de tri n°3 : le tri rapide

Recette à suivre :

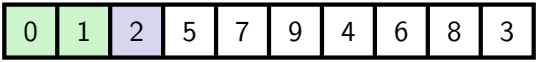
1. Martin choisit une pièce au hasard (il l'appelle la pièce pivot).
2. Il la compare à toutes les autres pièces, qu'il sépare en pièces légères et lourdes.
3. Il met les pièces légères à gauche, puis la pièce pivot, puis les pièces lourdes.
4. Il trie les pièces légères et les pièces lourdes.



Algorithme de tri n°3 : le tri rapide

Recette à suivre :

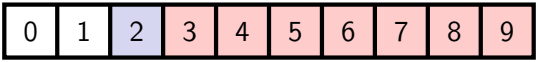
1. Martin choisit une pièce au hasard (il l'appelle la pièce pivot).
2. Il la compare à toutes les autres pièces, qu'il sépare en pièces légères et lourdes.
3. Il met les pièces légères à gauche, puis la pièce pivot, puis les pièces lourdes.
4. Il trie les pièces légères et les pièces lourdes.



Algorithme de tri n°3 : le tri rapide

Recette à suivre :

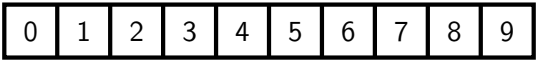
1. Martin choisit une pièce au hasard (il l'appelle la pièce pivot).
2. Il la compare à toutes les autres pièces, qu'il sépare en pièces légères et lourdes.
3. Il met les pièces légères à gauche, puis la pièce pivot, puis les pièces lourdes.
4. Il trie les pièces légères et les pièces lourdes.



Algorithme de tri n°3 : le tri rapide

Recette à suivre :

1. Martin choisit une pièce au hasard (il l'appelle la pièce pivot).
2. Il la compare à toutes les autres pièces, qu'il sépare en pièces légères et lourdes.
3. Il met les pièces légères à gauche, puis la pièce pivot, puis les pièces lourdes.
4. Il trie les pièces légères et les pièces lourdes.



Combien de comparaisons Martin va-t-il effectuer ?

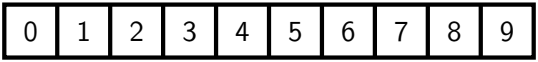
1. Dans le meilleur cas : $\approx n \log_2(n)$ pesées

☛ $x = 2^{\log_2(x)}$

Algorithme de tri n°3 : le tri rapide

Recette à suivre :

1. Martin choisit une pièce au hasard (il l'appelle la pièce pivot).
2. Il la compare à toutes les autres pièces, qu'il sépare en pièces légères et lourdes.
3. Il met les pièces légères à gauche, puis la pièce pivot, puis les pièces lourdes.
4. Il trie les pièces légères et les pièces lourdes.



Combien de comparaisons Martin va-t-il effectuer ?

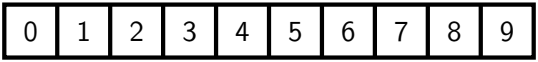
1. Dans le meilleur cas : $\approx n \log_2(n)$ pesées
2. Dans le pire cas : $n(n-1)/2$ pesées

☛ $x = 2^{\log_2(x)}$

Algorithme de tri n°3 : le tri rapide

Recette à suivre :

1. Martin choisit une pièce au hasard (il l'appelle la pièce pivot).
2. Il la compare à toutes les autres pièces, qu'il sépare en pièces légères et lourdes.
3. Il met les pièces légères à gauche, puis la pièce pivot, puis les pièces lourdes.
4. Il trie les pièces légères et les pièces lourdes.



Combien de comparaisons Martin va-t-il effectuer ?

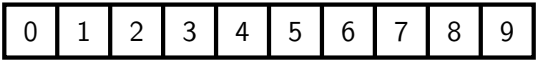
1. Dans le meilleur cas : $\approx n \log_2(n)$ pesées
2. Dans le pire cas : $n(n-1)/2$ pesées
3. Dans le cas moyen : $\approx 2n \ln(n)$ pesées

☛ $x = 2^{\log_2(x)} = e^{\ln(x)}$, avec $e \approx 2,718281828\ldots$

Algorithme de tri n°3 : le tri rapide

Recette à suivre :

1. Martin choisit une pièce au hasard (il l'appelle la pièce pivot).
2. Il la compare à toutes les autres pièces, qu'il sépare en pièces légères et lourdes.
3. Il met les pièces légères à gauche, puis la pièce pivot, puis les pièces lourdes.
4. Il trie les pièces légères et les pièces lourdes.



Combien de comparaisons Martin va-t-il effectuer ?

1. Dans le meilleur cas : $\approx n \log_2(n)$ pesées
2. Dans le pire cas : $n(n-1)/2$ pesées
3. Dans le cas moyen : $\approx 2n \ln(n)$ pesées $\approx 24\,785\,482$ pesées pour $n = 10^6$

☛ $x = 2^{\log_2(x)} = e^{\ln(x)}$, avec $e \approx 2,718281828\ldots$

Algorithme de tri n°3 : le tri rapide – Pourquoi ça marche ?

Preuve :

1. Le pivot qu'a choisi Martin est la $k^{\text{ème}}$ pièce.

Algorithme de tri n°3 : le tri rapide – Pourquoi ça marche ?

Preuve :

1. Le pivot qu'a choisi Martin est la $k^{\text{ème}}$ pièce.
2. En moyenne, il effectue $n - 1 + \text{coût}(k) + \text{coût}(n - 1 - k)$ pesées.

Algorithme de tri n°3 : le tri rapide – Pourquoi ça marche ?

Preuve :

1. Le pivot qu'a choisi Martin est la $k^{\text{ème}}$ pièce.
2. En moyenne, il effectue $n-1+\text{coût}(k)+\text{coût}(n-1-k)$ pesées.
3. Par conséquent,

$$\text{coût}(n) = n-1+\frac{1}{n}\sum_{k=0}^{n-1}\text{coût}(k)+\text{coût}(n-1-k) = n-1+\frac{2}{n}\sum_{k=0}^{n-1}\text{coût}(k).$$

Algorithme de tri n°3 : le tri rapide – Pourquoi ça marche ?

Preuve :

1. Le pivot qu'a choisi Martin est la $k^{\text{ème}}$ pièce.
2. En moyenne, il effectue $n-1+\text{coût}(k)+\text{coût}(n-1-k)$ pesées.
3. Par conséquent,

$$\text{coût}(n) = n-1+\frac{1}{n}\sum_{k=0}^{n-1}\text{coût}(k)+\text{coût}(n-1-k) = n-1+\frac{2}{n}\sum_{k=0}^{n-1}\text{coût}(k).$$

4. On peut alors montrer par récurrence fastidieuse sur n que

$$2(n+1)\ln(n+1)-4n \leqslant \text{coût}(n) \leqslant 2n\ln(n).$$

Algorithme de tri n°4 : le tri encore plus rapide

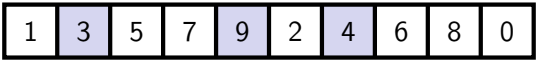
Recette à suivre :

1. Martin prend trois pièces au hasard puis choisit la pièce médiane comme pivot.
2. Il la compare à toutes les autres pièces, qu'il sépare en pièces légères et lourdes.
3. Il met les pièces légères à gauche, puis la pièce pivot, puis les pièces lourdes.
4. Il trie les pièces légères et les pièces lourdes.

Algorithme de tri n°4 : le tri encore plus rapide

Recette à suivre :

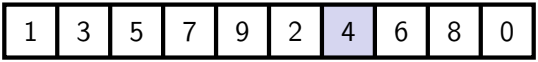
1. Martin prend trois pièces au hasard puis choisit la pièce médiane comme pivot.
2. Il la compare à toutes les autres pièces, qu'il sépare en pièces légères et lourdes.
3. Il met les pièces légères à gauche, puis la pièce pivot, puis les pièces lourdes.
4. Il trie les pièces légères et les pièces lourdes.



Algorithme de tri n°4 : le tri encore plus rapide

Recette à suivre :

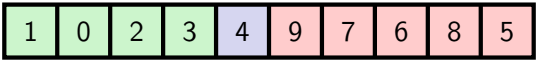
1. Martin prend trois pièces au hasard puis choisit la pièce médiane comme pivot.
2. Il la compare à toutes les autres pièces, qu'il sépare en pièces légères et lourdes.
3. Il met les pièces légères à gauche, puis la pièce pivot, puis les pièces lourdes.
4. Il trie les pièces légères et les pièces lourdes.



Algorithme de tri n°4 : le tri encore plus rapide

Recette à suivre :

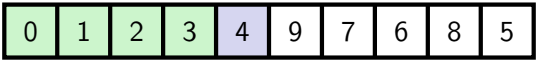
1. Martin prend trois pièces au hasard puis choisit la pièce médiane comme pivot.
2. Il la compare à toutes les autres pièces, qu'il sépare en pièces légères et lourdes.
3. Il met les pièces légères à gauche, puis la pièce pivot, puis les pièces lourdes.
4. Il trie les pièces légères et les pièces lourdes.



Algorithme de tri n°4 : le tri encore plus rapide

Recette à suivre :

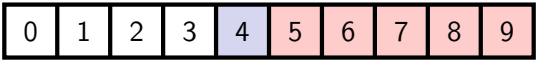
1. Martin prend trois pièces au hasard puis choisit la pièce médiane comme pivot.
2. Il la compare à toutes les autres pièces, qu'il sépare en pièces légères et lourdes.
3. Il met les pièces légères à gauche, puis la pièce pivot, puis les pièces lourdes.
4. Il trie les pièces légères et les pièces lourdes.



Algorithme de tri n°4 : le tri encore plus rapide

Recette à suivre :

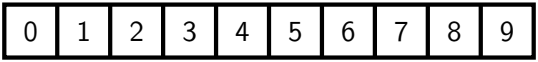
1. Martin prend trois pièces au hasard puis choisit la pièce médiane comme pivot.
2. Il la compare à toutes les autres pièces, qu'il sépare en pièces légères et lourdes.
3. Il met les pièces légères à gauche, puis la pièce pivot, puis les pièces lourdes.
4. Il trie les pièces légères et les pièces lourdes.



Algorithme de tri n°4 : le tri encore plus rapide

Recette à suivre :

1. Martin prend trois pièces au hasard puis choisit la pièce médiane comme pivot.
2. Il la compare à toutes les autres pièces, qu'il sépare en pièces légères et lourdes.
3. Il met les pièces légères à gauche, puis la pièce pivot, puis les pièces lourdes.
4. Il trie les pièces légères et les pièces lourdes.



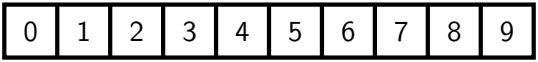
Combien de comparaisons Martin va-t-il effectuer ?

1. Dans le meilleur cas : $\approx n \log_2(n)$ pesées

Algorithme de tri n°4 : le tri encore plus rapide

Recette à suivre :

1. Martin prend trois pièces au hasard puis choisit la pièce médiane comme pivot.
2. Il la compare à toutes les autres pièces, qu'il sépare en pièces légères et lourdes.
3. Il met les pièces légères à gauche, puis la pièce pivot, puis les pièces lourdes.
4. Il trie les pièces légères et les pièces lourdes.



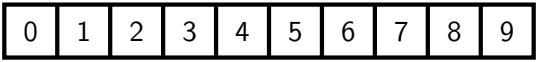
Combien de comparaisons Martin va-t-il effectuer ?

1. Dans le meilleur cas : $\approx n \log_2(n)$ pesées
2. Dans le pire cas : $(n+7)(n-1)/4$ pesées

Algorithme de tri n°4 : le tri encore plus rapide

Recette à suivre :

1. Martin prend trois pièces au hasard puis choisit la pièce médiane comme pivot.
2. Il la compare à toutes les autres pièces, qu'il sépare en pièces légères et lourdes.
3. Il met les pièces légères à gauche, puis la pièce pivot, puis les pièces lourdes.
4. Il trie les pièces légères et les pièces lourdes.



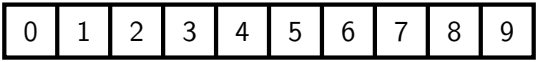
Combien de comparaisons Martin va-t-il effectuer ?

1. Dans le meilleur cas : $\approx n \log_2(n)$ pesées
2. Dans le pire cas : $(n+7)(n-1)/4$ pesées
3. Dans le cas moyen : $\approx 12n \ln(n)/7$ pesées

Algorithme de tri n°4 : le tri encore plus rapide

Recette à suivre :

1. Martin prend trois pièces au hasard puis choisit la pièce médiane comme pivot.
2. Il la compare à toutes les autres pièces, qu'il sépare en pièces légères et lourdes.
3. Il met les pièces légères à gauche, puis la pièce pivot, puis les pièces lourdes.
4. Il trie les pièces légères et les pièces lourdes.



Combien de comparaisons Martin va-t-il effectuer ?

1. Dans le meilleur cas : $\approx n \log_2(n)$ pesées
2. Dans le pire cas : $(n+7)(n-1)/4$ pesées
3. Dans le cas moyen : $\approx 12n\ln(n)/7$ pesées $\approx 22\,428\,000$ pesées pour $n = 10^6$

☛ On vient d'économiser $\approx 10\%$ de pesées !

Algorithme de tri n°4 : le tri encore plus rapide – Pourquoi ça marche ?

Idées :

1. En moyenne, Martin effectue $n - 1 + \text{coût}(k) + \text{coût}(n - 1 - k)$ pesées.

Algorithme de tri n°4 : le tri encore plus rapide – Pourquoi ça marche ?

Idées :

1. En moyenne, Martin effectue $n - 1 + \text{coût}(k) + \text{coût}(n - 1 - k)$ pesées.

Algorithme de tri n°4 : le tri encore plus rapide – Pourquoi ça marche ?

Idées :

1. En moyenne, Martin effectue $n - 1 + \text{coût}(k) + \text{coût}(n - 1 - k)$ pesées.
2. Or, $\text{coût}(n) \approx \alpha n \ln(n)$ pesées est une fonction (à peu près) convexe : $\text{coût}(k) + \text{coût}(n - 1 - k)$ est minimal lorsque $k = n - 1 - k \approx (n - 1)/2$.

Algorithme de tri n°4 : le tri encore plus rapide – Pourquoi ça marche ?

Idées :

1. En moyenne, Martin effectue $n-1+\text{coût}(k)+\text{coût}(n-1-k)$ pesées.
2. Or, $\text{coût}(n) \approx \alpha n \ln(n)$ pesées est une fonction (à peu près) convexe : $\text{coût}(k)+\text{coût}(n-1-k)$ est minimal lorsque $k = n-1-k \approx (n-1)/2$.
3. Martin doit donc choisir son pivot aussi près de la médiane que possible !

Algorithme de tri n°5 : le tri fusion

Recette à suivre :

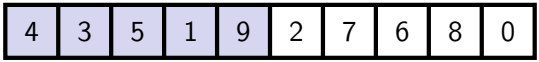
1. Martin trie la moitié gauche puis la moitié droite.
2. Il identifie la plus petite pièce et la met à gauche d'une nouvelle file.
   ☛ Disposer de deux files en tout suffit, mais est indispensable.
3. Puis il identifie la plus petite pièce restante à l'ajoute à droite de sa nouvelle file, etc.

Algorithme de tri n°5 : le tri fusion

Recette à suivre :

1. Martin trie la moitié gauche puis la moitié droite.
2. Il identifie la plus petite pièce et la met à gauche d'une nouvelle file.

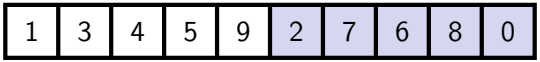
   ☛ Disposer de deux files en tout suffit, mais est indispensable.
3. Puis il identifie la plus petite pièce restante à l'ajoute à droite de sa nouvelle file, etc.



Algorithme de tri n°5 : le tri fusion

Recette à suivre :

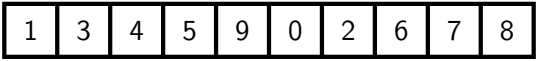
1. Martin trie la moitié gauche puis la moitié droite.
2. Il identifie la plus petite pièce et la met à gauche d'une nouvelle file.
   ☛ Disposer de deux files en tout suffit, mais est indispensable.
3. Puis il identifie la plus petite pièce restante à l'ajoute à droite de sa nouvelle file, etc.



Algorithme de tri n°5 : le tri fusion

Recette à suivre :

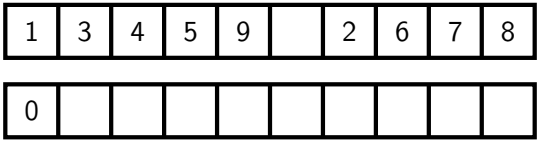
1. Martin trie la moitié gauche puis la moitié droite.
2. Il identifie la plus petite pièce et la met à gauche d'une nouvelle file.
   ☛ Disposer de deux files en tout suffit, mais est indispensable.
3. Puis il identifie la plus petite pièce restante à l'ajoute à droite de sa nouvelle file, etc.



Algorithme de tri n°5 : le tri fusion

Recette à suivre :

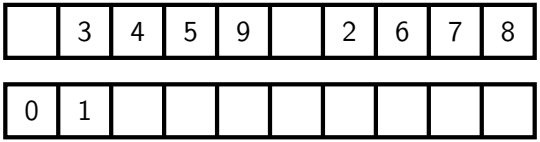
1. Martin trie la moitié gauche puis la moitié droite.
2. Il identifie la plus petite pièce et la met à gauche d'une nouvelle file.
   ☛ Disposer de deux files en tout suffit, mais est indispensable.
3. Puis il identifie la plus petite pièce restante à l'ajoute à droite de sa nouvelle file, etc.



Algorithme de tri n°5 : le tri fusion

Recette à suivre :

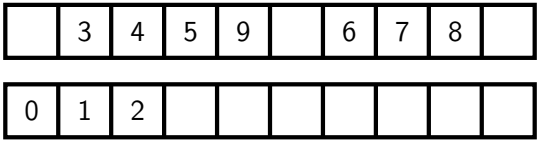
1. Martin trie la moitié gauche puis la moitié droite.
2. Il identifie la plus petite pièce et la met à gauche d'une nouvelle file.
   - ☛ Disposer de deux files en tout suffit, mais est indispensable.
3. Puis il identifie la plus petite pièce restante à l'ajoute à droite de sa nouvelle file, etc.



Algorithme de tri n°5 : le tri fusion

Recette à suivre :

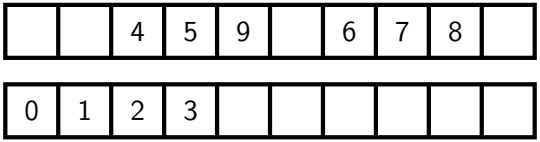
1. Martin trie la moitié gauche puis la moitié droite.
2. Il identifie la plus petite pièce et la met à gauche d'une nouvelle file.
   - ☛ Disposer de deux files en tout suffit, mais est indispensable.
3. Puis il identifie la plus petite pièce restante à l'ajoute à droite de sa nouvelle file, etc.



Algorithme de tri n°5 : le tri fusion

Recette à suivre :

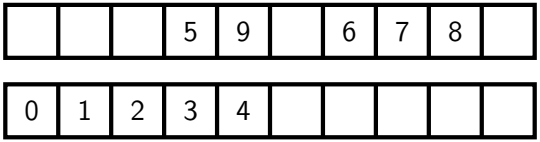
1. Martin trie la moitié gauche puis la moitié droite.
2. Il identifie la plus petite pièce et la met à gauche d'une nouvelle file.
   - ☛ Disposer de deux files en tout suffit, mais est indispensable.
3. Puis il identifie la plus petite pièce restante à l'ajoute à droite de sa nouvelle file, etc.



Algorithme de tri n°5 : le tri fusion

Recette à suivre :

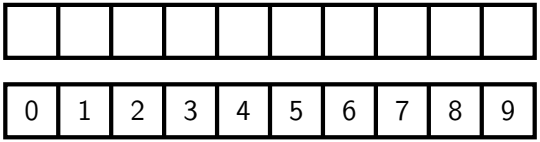
1. Martin trie la moitié gauche puis la moitié droite.
2. Il identifie la plus petite pièce et la met à gauche d'une nouvelle file.
   ☛ Disposer de deux files en tout suffit, mais est indispensable.
3. Puis il identifie la plus petite pièce restante à l'ajoute à droite de sa nouvelle file, etc.



Algorithme de tri n°5 : le tri fusion

Recette à suivre :

1. Martin trie la moitié gauche puis la moitié droite.
2. Il identifie la plus petite pièce et la met à gauche d'une nouvelle file.
   - ☛ Disposer de deux files en tout suffit, mais est indispensable.
3. Puis il identifie la plus petite pièce restante à l'ajoute à droite de sa nouvelle file, etc.

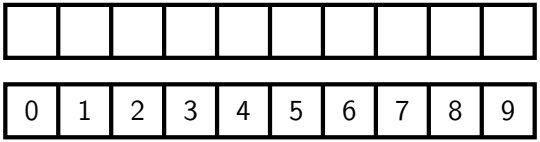


Algorithme de tri n°5 : le tri fusion

Recette à suivre :

1. Martin trie la moitié gauche puis la moitié droite.
2. Il identifie la plus petite pièce et la met à gauche d'une nouvelle file.

   ☛ Disposer de deux files en tout suffit, mais est indispensable.
3. Puis il identifie la plus petite pièce restante à l'ajoute à droite de sa nouvelle file, etc.



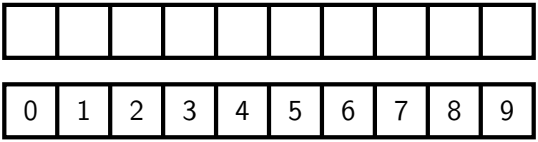
Combien de comparaisons Martin va-t-il effectuer ?

1. Dans le meilleur cas : $\approx n \log_2(n)/2$ pesées

Algorithme de tri n°5 : le tri fusion

Recette à suivre :

1. Martin trie la moitié gauche puis la moitié droite.
2. Il identifie la plus petite pièce et la met à gauche d'une nouvelle file.
   ☛ Disposer de deux files en tout suffit, mais est indispensable.
3. Puis il identifie la plus petite pièce restante à l'ajoute à droite de sa nouvelle file, etc.



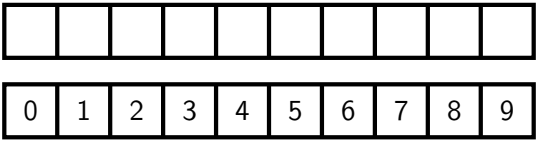
Combien de comparaisons Martin va-t-il effectuer ?

1. Dans le meilleur cas : $\approx n\log_2(n)/2$ pesées
2. Dans le pire cas : $\approx n\log_2(n)$ pesées

Algorithme de tri n°5 : le tri fusion

Recette à suivre :

1. Martin trie la moitié gauche puis la moitié droite.
2. Il identifie la plus petite pièce et la met à gauche d'une nouvelle file.
   ☛ Disposer de deux files en tout suffit, mais est indispensable.
3. Puis il identifie la plus petite pièce restante à l'ajoute à droite de sa nouvelle file, etc.



Combien de comparaisons Martin va-t-il effectuer ?

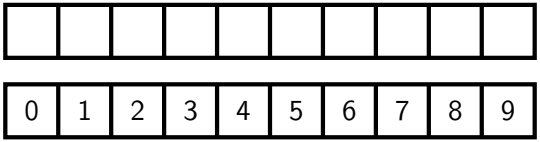
1. Dans le meilleur cas : $\approx n\log_2(n)/2$ pesées
2. Dans le pire cas : $\approx n\log_2(n)$ pesées
3. Dans le cas moyen : $\approx n\log_2(n)$ pesées

Algorithme de tri n°5 : le tri fusion

Recette à suivre :

1. Martin trie la moitié gauche puis la moitié droite.
2. Il identifie la plus petite pièce et la met à gauche d'une nouvelle file.

   ☛ Disposer de deux files en tout suffit, mais est indispensable.
3. Puis il identifie la plus petite pièce restante à l'ajoute à droite de sa nouvelle file, etc.



Combien de comparaisons Martin va-t-il effectuer ?

1. Dans le meilleur cas : $\approx n \log_2(n)/2$ pesées
2. Dans le pire cas : $\approx n \log_2(n)$ pesées
3. Dans le cas moyen : $\approx n \log_2(n)$ pesées

$18\,674\,241$ pesées pour $n = 10^6$

☛ On vient encore d'économiser $\approx 20\%$ de pesées !

Algorithme de tri n°5 : le tri fusion – Pourquoi ça marche

Preuve :

1. Cette fois-ci, Martin découpe sa file en deux parties de même taille (à 1 près).

Algorithme de tri n°5 : le tri fusion – Pourquoi ça marche

Preuve :

1. Cette fois-ci, Martin découpe sa file en deux parties de même taille (à 1 près).
2. L'étape de fusion nécessite
   - $\lfloor n/2 \rfloor$ comparaisons dans le meilleur cas ;
   - $n-1$ comparaisons dans le pire cas ;
   - $C_n = n - 2 + 2/\lfloor (n+3)/2 \rfloor$ comparaisons dans le cas moyen.

Algorithme de tri n°5 : le tri fusion – Pourquoi ça marche

Preuve :

1. Cette fois-ci, Martin découpe sa file en deux parties de même taille (à 1 près).
2. L'étape de fusion nécessite
   - $\lfloor n/2 \rfloor$ comparaisons dans le meilleur cas ;
   - $n-1$ comparaisons dans le pire cas ;
   - $C_n = n - 2 + 2/\lfloor (n+3)/2 \rfloor$ comparaisons dans le cas moyen.
3. Martin effectue donc
   - $\text{coût}(n) = \lfloor n/2 \rfloor + \text{coût}(\lfloor n/2 \rfloor) + \text{coût}(\lceil n/2 \rceil)$ pesées ;
   - $\text{coût}(n) = n - 1 + \text{coût}(\lfloor n/2 \rfloor) + \text{coût}(\lceil n/2 \rceil)$ pesées ;
   - $\text{coût}(n) \geqslant C_n + \text{coût}(\lfloor n/2 \rfloor) + \text{coût}(\lceil n/2 \rceil)$ pesées.

Algorithme de tri n°5 : le tri fusion – Pourquoi ça marche

Preuve :

1. Cette fois-ci, Martin découpe sa file en deux parties de même taille (à 1 près).
2. L'étape de fusion nécessite
   - $\lfloor n/2 \rfloor$ comparaisons dans le meilleur cas ;
   - $n-1$ comparaisons dans le pire cas ;
   - $C_n = n - 2 + 2/\lfloor (n+3)/2 \rfloor$ comparaisons dans le cas moyen.
3. Martin effectue donc
   - $\text{coût}(n) = \lfloor n/2 \rfloor + \text{coût}(\lfloor n/2 \rfloor) + \text{coût}(\lceil n/2 \rceil)$ pesées ;
   - $\text{coût}(n) = n - 1 + \text{coût}(\lfloor n/2 \rfloor) + \text{coût}(\lceil n/2 \rceil)$ pesées ;
   - $\text{coût}(n) \geqslant C_n + \text{coût}(\lfloor n/2 \rfloor) + \text{coût}(\lceil n/2 \rceil)$ pesées.
4. On peut alors montrer par récurrence gentille sur n que

$$n\lfloor \log_2(n) \rfloor - 2n \leqslant \text{coût}(n) \leqslant \text{coût}(n) \leqslant 2\,\text{coût}(n) \leqslant n\lceil \log_2(n) \rceil.$$

Peut-on mieux faire ?

1. Dans le meilleur cas : $n - 1$ pesées

Peut-on mieux faire ?

1. Dans le meilleur cas : $n - 1$ pesées
   - ☛ Le monde est une simulation que contrôle Raphaël, l'allié de Martin.

Peut-on mieux faire ?

1. Dans le meilleur cas : $n - 1$ pesées
   - Le monde est une simulation que contrôle Raphaël, l'allié de Martin.
   - Martin doit quand même vérifier que sa file est triée.

Peut-on mieux faire ?

1. Dans le meilleur cas : $n - 1$ pesées
   - Le monde est une simulation que contrôle Raphaël, l'allié de Martin.
   - Martin doit quand même vérifier que sa file est triée.
   - S'il omet de comparer deux pièces consécutives, elles peuvent être triées dans le mauvais sens.

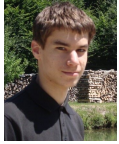
Peut-on mieux faire ?

1. Dans le meilleur cas : $n - 1$ pesées
   - Le monde est une simulation que contrôle Raphaël, l'allié de Martin.
   - Martin doit quand même vérifier que sa file est triée.
   - S'il omet de comparer deux pièces consécutives, elles peuvent être triées dans le mauvais sens.
2. Dans le pire cas : au moins $\log_2(n!) \approx n \log_2(n)$ pesées

Peut-on mieux faire ?

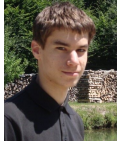
1. Dans le meilleur cas : $n-1$ pesées
   - Le monde est une simulation que contrôle Raphaël, l'allié de Martin.
   - Martin doit quand même vérifier que sa file est triée.
   - S'il omet de comparer deux pièces consécutives, elles peuvent être triées dans le mauvais sens.
2. Dans le pire cas : au moins $\log_2(n!) \approx n \log_2(n)$ pesées
   - Trier la file revient à identifier l'ordre de départ (parmi les $n!$ ordres possibles).

Peut-on mieux faire ?



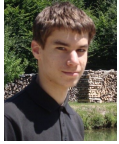
1. Dans le meilleur cas : $n-1$ pesées
   - Le monde est une simulation que contrôle Raphaël, l'allié de Martin.
   - Martin doit quand même vérifier que sa file est triée.
   - S'il omet de comparer deux pièces consécutives, elles peuvent être triées dans le mauvais sens.
2. Dans le pire cas : au moins $\log_2(n!) \approx n \log_2(n)$ pesées
   - Trier la file revient à identifier l'ordre de départ (parmi les $n!$ ordres possibles).
   - Le monde est une simulation que contrôle Théo, l'adversaire de Martin.

Peut-on mieux faire ?



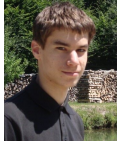
1. Dans le meilleur cas : $n-1$ pesées
   - Le monde est une simulation que contrôle Raphaël, l'allié de Martin.
   - Martin doit quand même vérifier que sa file est triée.
   - S'il omet de comparer deux pièces consécutives, elles peuvent être triées dans le mauvais sens.
2. Dans le pire cas : au moins $\log_2(n!) \approx n \log_2(n)$ pesées
   - Trier la file revient à identifier l'ordre de départ (parmi les $n!$ ordres possibles).
   - Le monde est une simulation que contrôle Théo, l'adversaire de Martin.
   - Théo agit systématiquement de manière à invalider le moins d'ordres possibles.

Peut-on mieux faire ?



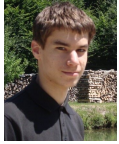
1. Dans le meilleur cas : $n-1$ pesées
   - Le monde est une simulation que contrôle Raphaël, l'allié de Martin.
   - Martin doit quand même vérifier que sa file est triée.
   - S'il omet de comparer deux pièces consécutives, elles peuvent être triées dans le mauvais sens.
2. Dans le pire cas : au moins $\log_2(n!) \approx n\log_2(n)$ pesées
   - Trier la file revient à identifier l'ordre de départ (parmi les $n!$ ordres possibles).
   - Le monde est une simulation que contrôle Théo, l'adversaire de Martin.
   - Théo agit systématiquement de manière à invalider le moins d'ordres possibles : après k pesées, il reste au moins $n!/2^k$ ordres possibles.

Peut-on mieux faire ?



1. Dans le meilleur cas : $n-1$ pesées
   - Le monde est une simulation que contrôle Raphaël, l'allié de Martin.
   - Martin doit quand même vérifier que sa file est triée.
   - S'il omet de comparer deux pièces consécutives, elles peuvent être triées dans le mauvais sens.
2. Dans le pire cas : au moins $\log_2(n!) \approx n\log_2(n)$ pesées
   - Trier la file revient à identifier l'ordre de départ (parmi les $n!$ ordres possibles).
   - Le monde est une simulation que contrôle Théo, l'adversaire de Martin.
   - Théo agit systématiquement de manière à invalider le moins d'ordres possibles : après k pesées, il reste au moins $n!/2^k$ ordres possibles.
3. Dans le cas moyen : au moins $\log_2(n!) \approx n\log_2(n)$ pesées

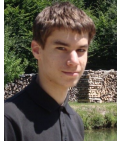
Peut-on mieux faire ?



1. Dans le meilleur cas : $n-1$ pesées
   - Le monde est une simulation que contrôle Raphaël, l'allié de Martin.
   - Martin doit quand même vérifier que sa file est triée.
   - S'il omet de comparer deux pièces consécutives, elles peuvent être triées dans le mauvais sens.
2. Dans le pire cas : au moins $\log_2(n!) \approx n\log_2(n)$ pesées
   - Trier la file revient à identifier l'ordre de départ (parmi les $n!$ ordres possibles).
   - Le monde est une simulation que contrôle Théo, l'adversaire de Martin.
   - Théo agit systématiquement de manière à invalider le moins d'ordres possibles : après k pesées, il reste au moins $n!/2^k$ ordres possibles.
3. Dans le cas moyen : au moins $\log_2(n!) \approx n\log_2(n)$ pesées
   - Le nombre X_k d'ordres possibles après k pesées, même choisies astucieusement au vu des résultats des pesées précédentes, satisfait toujours l'inégalité

$$\mathbb{E}[\log_2(X_k)] \geqslant \log_2(n!) - k.$$

Peut-on mieux faire ?



1. Dans le meilleur cas : $n-1$ pesées
   - Le monde est une simulation que contrôle Raphaël, l'allié de Martin.
   - Martin doit quand même vérifier que sa file est triée.
   - S'il omet de comparer deux pièces consécutives, elles peuvent être triées dans le mauvais sens.
2. Dans le pire cas : au moins $\log_2(n!) \approx n\log_2(n)$ pesées
   - Trier la file revient à identifier l'ordre de départ (parmi les $n!$ ordres possibles).
   - Le monde est une simulation que contrôle Théo, l'adversaire de Martin.
   - Théo agit systématiquement de manière à invalider le moins d'ordres possibles : après k pesées, il reste au moins $n!/2^k$ ordres possibles.
3. Dans le cas moyen : au moins $\log_2(n!) \approx n\log_2(n)$ pesées
   - Le nombre X_k d'ordres possibles après k pesées, même choisies astucieusement au vu des résultats des pesées précédentes, satisfait toujours l'inégalité

$$\mathbb{E}[\log_2(X_k)] \geqslant \log_2(n!) - k.$$

   - La bonne manière d'étudier ce cas consiste à invoquer les notions d'entropie et de quantité d'information.

MERCI POUR VOTRE ATTENTION !
NE POSEZ PAS DE QUESTIONS
TROP DIFFICILES S'IL VOUS PLAÎT !