Fonction sortedDiviser pour régner

Fonction sorted Diviser pour régner

Christophe Viroulaud

Teverinie - NSI

Algo 01

Fonction **sorted**Diviser pour régner

Christophe Viroulaud

Terminale - NSI

Algo 01

Fonction sorted Diviser pour régner

Kapper : des algorithmes de tris déjà connus

> par sélection par insertion

louvelle approche Résoudre de petits

our solutionner un gros blème iser nour régner : un

iser pour régner : un prithme récursif

rformances du

tri fusion

Mesure de la durée
d'exécution

Complexité

imsort

Fonction sortedDiviser pour régner

En tant que langage de haut niveau, Python offre des méthodes permettant d'effectuer efficacement certaines La méthode sort trie en place un tableau.

tāches courantes.

tri en place ou nouveau tableau

En tant que langage de haut niveau, Python offre des méthodes permettant d'effectuer efficacement certaines tâches courantes.

La méthode sort trie en place un tableau. .

Fonction sorted Diviser pour régner

Quel algorithme de tri est implémenté dans la méthode

Quel algorithme de tri est implémenté dans la méthode sort?

Fonction sorted Diviser pour régner



Sommaire

- 1. Rappel : des algorithmes de tris déjà connus

Mesure de la durée

Fonction sorted

Diviser pour régner

algorithmes de tris déjà connus

Rappel: des

4 / 56

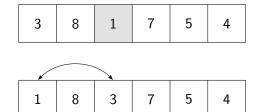
Tri par sélection (en place) Pour chaque élément du tableau Trouver le plus petit élément dans la partie Échanger cet élément avec le premier de la 3 8 1 7 5 4

Tri par sélection (en place)

Pour chaque élément du tableau

Trouver le plus petit élément dans la partie non triée.

Échanger cet élément avec le premier de la 3 partie non triée.



Fonction sorted Diviser pour régner

Tri par sélection

Tri par sélection (en place)

Pour chaque élément du tableau

Trouver le plus petit élément dans la partie non triée.

Échanger cet élément avec le premier de la partie non triée.

Activité 1 :

3

Tri par sélection (en place)

Pour chaque élément du tableau

Trouver le plus petit élément dans la partie non triés. Échanger cet élément avec le prenier de la

Construire par compréhension un tableau de 10 entiers aléatoires compris entre 1 et 100.
 Écrire la fonction tri_selection(tab: list)
 → None qui trie le tableau en place.

- 1. Construire par compréhension un tableau de 10 entiers aléatoires compris entre 1 et 100.
- 2. Écrire la fonction tri_selection(tab: list)
 - $\rightarrow\,$ None qui trie le tableau en place.

Fonction sorted Diviser pour régner

Rappel : des algorithmes de tris

Tri par sélection

Tri par insertion

louvelle approche

Résoudre de petits problèmes

roblème liviser pour régner : un

Diviser pour régner : un algorithme récursif Étape de la fusion

erformances du

Mesure de la durée d'exécution

Timsort

2021-09-27

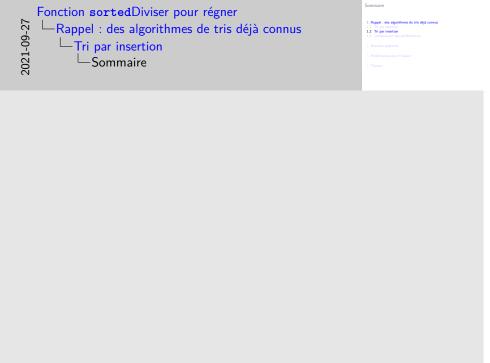
```
tab = [randint(1, 100) for in range(10)]
def tri_melection(tab: list) -> None:
    for i in range(len(tab))
         i_mini = i
        for j in range(i+1, len(tab)):
   if tab[i] < tab[i_mini]:</pre>
         tab[i], tab[i_mini] = tab[i_mini], tab[i]
```

Correction

```
tab = [randint(1, 100) for _ in range(10)]
  def tri selection(tab: list) -> None:
      for i in range(len(tab)):
3
           # trouver le mini
           i mini = i
4
           for j in range(i+1, len(tab)):
5
               if tab[j] < tab[i_mini]:</pre>
6
                   i_mini = j
8
           # échanger
9
           tab[i], tab[i_mini] = tab[i_mini], tab[i]
```

Fonction sorted Diviser pour régner

Tri par sélection



Sommaire

- 1. Rappel : des algorithmes de tris déjà connus
 - 1.1 TH par selection
- 1.2 Tri par insertion
- 1.3 Comparaison des performances
- 2. Nouvelle approche
- 2. Nouvelle approche
 - Performances du tri fusio
 - Timcor

Mesure de la durée d'exécution

nplexité

Fonction sorted

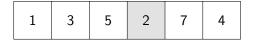
Diviser pour régner

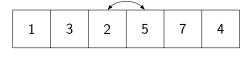
Tri par insertion

nsort

Tri par insertion (en place)

Pour chaque élément du tableau Tant que l'élément précédent est inférieur Permuter les deux éléments







Fonction sorted Diviser pour régner

Tri par insertion

```
Tri par insertion (en place)

1 [Pour chape diseased as tabless
2 | Verification of the control of the control
```

Tri par insertion (en place)

- Pour chaque élément du tableau
- Tant que l'élément précédent est inférieur
- Permuter les deux éléments

Activité 2 :

- 1. Construire par compréhension un tableau de 10 entiers aléatoires compris entre 1 et 100.
- 2. Écrire la fonction tri_insertion(tab: list)
 - \rightarrow None qui trie le tableau en place.

Fonction sorted Diviser pour régner

Rappel : des algorithmes de tris déjà connus

Tri par sélection
Tri par insertion

Comparaison des performances

Nouvelle approche

Résoudre de petits problèmes pour solutionner un gros

problème Diviser pour régner : un

Diviser pour régner : un algorithme récursif

Étape de la fusion

Performances du ri fusion

Mesure de la durée d'exécution Complexité

Timsort

2021-09-27

Correction

```
def tri_insertion(tab: list) -> None:
    for i in range(len(tab)-1):
        j = i+1
        # tant que le précédent est inférieur
    while j > 0 and tab[j] < tab[j-1]:
        # permuter
        tab[j], tab[j-1] = tab[j-1], tab[j]
        j -= 1</pre>
```

Fonction sorted Diviser pour régner

algorithmes de tris déjà connus

Tri par sélection
Tri par insertion

Comparaison des

ouvelle approche

esoudre de petits oblèmes

problème Diviser pour régner : un

Diviser pour régner : un algorithme récursif

Étape de la fusion

erformances du i fusion

Mesure de la durée d'exécution Complexité

imsort



Sommaire

- 1. Rappel : des algorithmes de tris déjà connus

- 1.3 Comparaison des performances

Mesure de la durée

Fonction sorted

Diviser pour régner

Comparaison des performances

créer un nouveau tableau pour chaque fonction!!

Comparaison des performances

Activité 3:

Comparaison des performances

. Construire par compréhension un tableau de 10000

entiers compris entre 1 et 10000. Mesurer la durée d'exécution de la méthode sort et des deux fonctions précédentes.

Activité 3 :

- 1. Construire par compréhension un tableau de 10000 entiers compris entre 1 et 10000.
- 2. Mesurer la durée d'exécution de la méthode sort et des deux fonctions précédentes.

Fonction sorted Diviser pour régner

Comparaison des performances

```
Fonction sortedDiviser pour régner
   Rappel : des algorithmes de tris déjà connus
      -Comparaison des performances
         -Correction
```

```
deb = time()
tri_melection(tab)
fin = time()
print(fin-deb)
              Code 1 - tri par sélection
>>> mélection 4.557838678359985
 >>> mort 0.0019469251169433594
                Code 2 - Résultata
```

Correction

```
tab = [randint(1, 10000) for in range
   (10000)]
deb = time()
tri selection(tab)
fin = time()
print(fin-deb)
```

Code 1 – tri par sélection

```
>>> sélection 4.557838678359985
>>> insertion 3.959839105606079
>>> sort 0.0019469261169433594
```

Code 2 – Résultats

Fonction sorted Diviser pour régner

Comparaison des

performances

Fonction sortedDiviser pour régner Rappel : des algorithmes de tris déjà connus Comparaison des performances Évolution du nombre d'itérations

Evolution du nombre d'itérations

Le company de la company

15000 éléments \rightarrow 100 millions d'itérations

Évolution du nombre d'itérations

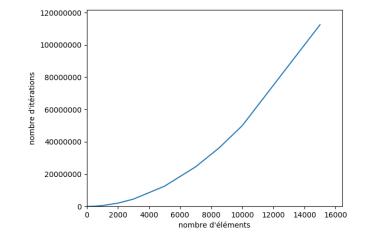


FIGURE 1 – Tri par sélection : complexité **quadratique**

Fonction sorted Diviser pour régner

Rappel : des algorithmes de tris déjà connus

Tri par insertion

Comparaison des performances

Vouvelle approcl

Résoudre de petits problèmes ...pour solutionner un gro problème

Diviser pour régner : ur algorithme récursif

Performances du tri fusion

Mesure de la durée d'exécution Complexité

Timsort



Sommaire

2. Nouvelle approche

- 2.4 Étape de la fusion

Mesure de la durée

Fonction sorted

Diviser pour régner

Nouvelle approche

16 / 56

Fonction sortedDiviser pour régner

Nouvelle approche
Résoudre de petits problèmes
Résoudre de petits problèmes



La propriété triviale suivante va nous permettre de construire une nouvelle méthode de tri :

Résoudre de petits problèmes

Observation

Une liste qui contient 0 ou 1 élément est triée.

8 5

FIGURE 2 – Deux listes triées

Diviser pour régner

Fonction sorted

algorithmes de tris déjà connus

Tri par sélection

problèmes

erformances

nuvelle approche

Nouvelle approche Résoudre de petits

..pour solutionner un gro problème

Diviser pour régner : un

Diviser pour regner : un algorithme récursif

Étape de la fusion

Performances du ri fusion

Mesure de la durée d'exécution Complexité

Timsort

Fonction sortedDiviser pour régner —Nouvelle approche Résoudre de petits problèmes

Deux listes d'un élément chacune peuvent être fusionnées en 8 5 5 8 FIGURE 3 - Fusionner 2 listes de 1 élément

> Deux listes d'un élément chacune peuvent être fusionnées en une liste triée de deux éléments.

5 8

FIGURE 3 – Fusionner 2 listes de 1 élément

Fonction sorted

Diviser pour régner

Résoudre de petits

problèmes

Principe

En résolvant des petits problèmes, nous pouvons remonter à des problèmes plus importants en appliquant le même principe.

Fonction sorted Diviser pour régner

Rappel : des algorithmes de tris déià connus

Tri par sélection

Comparaison des

Nouvelle approche
Résoudre de petits

problèmes

problème

Diviser pour régner : un algorithme récursif

algorithme récursif

Performances du

Mesure de la durée d'exécution

Timsort

Sommaire 1. Regard on apporthess de tre deja comun. 2. Neuvelle approche 22. Accountle approche 22. per velocities en gray spetities 22. per velocities en gray spetities 23. per velocities en a programme visual 24. Europ de la facilie 3. Performance de 16 facilie 4. Timost

Sommaire

- . Rappel : des algorithmes de tris déjà connus
- 2. Nouvelle approche
- 2.1 Résoudre de petits problèmes
- 2.2 ...pour solutionner un gros problème
- 2.3 Diviser pour régner : un algorithme récursif
- 2.4 Étape de la fusion
- 3. Performances du tri fusio
 - E1......

Mesure de la durée d'exécution Complexité

...pour solutionner un gros

Fonction sorted

Diviser pour régner

imsort

TISOFL

...pour solutionner un gros problème

essayons de nous ramener à de petits problèmes

.. pour solutionner un gros problème

FIGURE 4 - Un gros problème : trier une liste

9

FIGURE 4 – Un gros problème : trier une liste

Fonction sorted Diviser pour régner

...pour solutionner un gros

problème

Fonction sortedDiviser pour régner —Nouvelle approche ...pour solutionner un gros problème



Pour se ramener à un problème plus petit, séparons la liste en deux listes



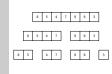


FIGURE 5 – Séparer la liste en deux listes plus petites

Fonction sorted Diviser pour régner

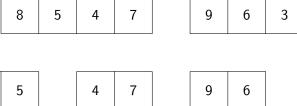
Comparaison des

...pour solutionner un gros





Q		1	7	٥	6	2	l
0)	4	'	9	U)	l
							ı



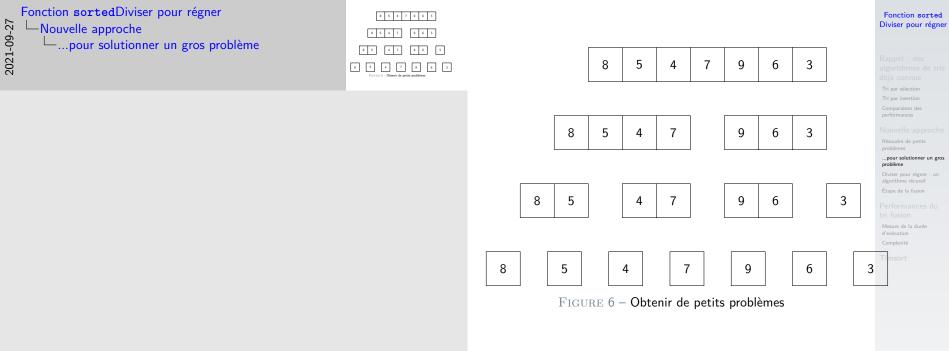
Mesure de la durée

...pour solutionner un gros problème

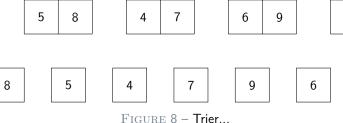
Fonction sorted

Diviser pour régner

Comparaison des







Fonction sorted

Diviser pour régner

...pour solutionner un gros problème

Diviser pour régner : un algorithme récursif

Étape de la fusion
Performances du



3 6 9

5 8

4 7

FIGURE 9 – …et remonter

6 9

problème

Diviser pour régner : us algorithme récursif
Étape de la fusion

Comparaison des

Étape de la fusion
Performances du

...pour solutionner un gros

Fonction sorted

Diviser pour régner

Performances du tri fusion

Mesure de la durée d'exécution Complexité

Timsort

i iiiisoi t

Fonction sortedDiviser pour régner └─Nouvelle approche L...pour solutionner un gros problème





4	5	7	8	

FIGURE 10 – Le tri se termine

Fonction sorted Diviser pour régner

Comparaison des

...pour solutionner un gros problème

Sommaire

- 2. Nouvelle approche
 - 1. Pássudus de metito muchièm
- 2.1 Resoudre de petits problèmes
- 2.3 Diviser pour régner : un algorithme récursif
- 2.3 Diviser pour regner : un algorithme recursit 2.4 Étape de la fusion
- 0.0.0
 - -

Mesure de la durée d'exécution

Diviser pour régner : un

algorithme récursit

Fonction sorted

Diviser pour régner

ort

imsort

Fonction sortedDiviser pour régner

—Nouvelle approche

-Diviser pour régner : un algorithme récursif

Diviser pour régner : un algorithme récursif

➤ cas limite: la liste est de taille minimale
> sinon
> on coope la liste en 2.
> featonner les listes les de la remontée d'appel
Code 3 - Tri featon

Diviser pour régner : un algorithme récursif

Diviser pour régner : un algorithme récursif

- **cas limite**: la liste est de taille minimale
- sinon
 - on coupe la liste en 2,
 - **appel récursif** sur chaque liste
 - ▶ fusionner les listes lors de la remontée d'appel

Code 3 – Tri fusion

Fonction sorted Diviser pour régner

algorithmes de tris déjà connus

Tri par insertion

Comparaison des performances

Résoudre de petits

...pour solutionner un gros problème

Diviser pour régner : un

algorithme récursif

Étape de la fusion

erformances du ri fusion

Mesure de la durée d'exécution Complexité

Timsort

Fonction sortedDiviser pour régner —Nouvelle approche

└─Diviser pour régner : un algorithme récursif

► cas limite : la liste est de taille minimale

 appel récursif sur chaque liste ► fusionner les listes lors de la remontée d'appel

Activité 4 : Soit la fonction fusionner(tab: list, deb: int. fin: int) → None out trie les éléments do tab entre les indices dab et fin Écrire la fonction tri fusion(tab: list. deb: int, fin: int) -> None qui trie le tableau en place.

- **cas limite**: la liste est de taille minimale
- sinon
 - on coupe la liste en 2,
 - **appel récursif** sur chaque liste
 - **fusionner** les listes lors de la remontée d'appel

Code 4 – Tri fusion

Activité 4 : Soit la fonction fusionner (tab: list, deb: int, fin: int) \rightarrow None qui trie les éléments de tab entre les indices deb et fin.

Écrire la fonction tri fusion(tab: list, deb: int, fin: int) \rightarrow None qui trie le tableau *en place*.

Fonction sorted Diviser pour régner

Diviser pour régner : un algorithme récursif

```
Fonction sortedDiviser pour régner
 —Nouvelle approche
    —Diviser pour régner : un algorithme récursif
        -Correction
```

```
def tri_fusion(tab: list, deb: int, fin: int) ->
 if deb < fin:
milieu = (deb*fin)//2
    tri_fumion(tab, deb, milieu)
    tri_fumion(tab, milieu+1, fin)
    fusionner(tab, deb, fin)
                  Code 5 - Tri fusion
```

Correction

```
def tri_fusion(tab: list, deb: int, fin: int) ->
     None:
  if deb < fin:</pre>
      milieu = (deb+fin)//2
3
      tri fusion(tab, deb, milieu)
4
      tri fusion(tab, milieu+1, fin)
5
      fusionner(tab, deb, fin)
6
```

Code 5 – Tri fusion

Fonction sorted Diviser pour régner

Diviser pour régner : un algorithme récursif

Sommaire

- 1. Rappel : des algorithmes de tris déjà connus
- 2. Nouvelle approche
- 2. Nouvelle approche
- 2.1 Resouare de petits problemes
- 2.2 ...pour solutionner un gros problème
- 2.3 Diviser pour régner : un algorithme récursif
- 2.4 Étape de la fusion
- 3. Performances du tri fusio
 - E----

Mesure de la durée d'exécution

Étape de la fusion

exité

Fonction sorted

Diviser pour régner

ort

33 / 56



Étape de la fusion

Lors de la remontée d'appel, la fusion assemble deux tableaux triés en un seul.

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

4 5 7 8 3	6
-----------	---

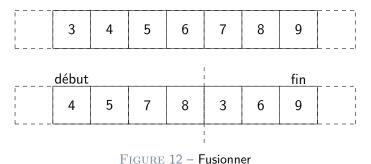
FIGURE 11 – Fusionner

Fonction sorted Diviser pour régner

Diviser pour régner : un

Étape de la fusion

En pratique, nous effectuons un tri place.



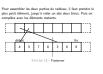
Fonction sorted

Diviser pour régner

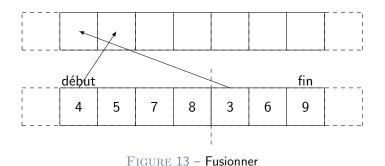
Comparaison des

Étape de la fusion

Fonction sortedDiviser pour régner —Nouvelle approche –Étape de la fusion



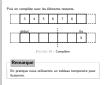
Pour assembler les deux parties du tableau, il faut prendre le plus petit élément, jusqu'à vider un des deux blocs. Puis on complète avec les éléments restants.



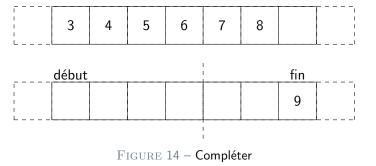
Fonction sorted Diviser pour régner

Étape de la fusion

Fonction sortedDiviser pour régner
Nouvelle approche
Étape de la fusion



Puis on complète avec les éléments restants.



Remarque

En pratique nous utiliserons un tableau temporaire pour fusionner.

Fonction sorted Diviser pour régner

Rappel : des algorithmes de tris déjà connus

Fri par sélection

Nouvelle approche Résoudre de petits

...pour solutionner un gro problème

Diviser pour régner : un

Diviser pour régner : un algorithme récursif

Étape de la fusion

tri fusion

Mesure de la durée

Complexité

Fonction sortedDiviser pour régner —Nouvelle approche —Étape de la fusion

Activité 5 : Écrire la fonction fusionner (tab: list, deb: int, fin: int) → None qui assemble les éléments de tab entre les indices deb et fin. Les éléments seront d'abord stockés dans un tableau temporaire qui viendra ensuite écraser la partie de tab.

> Activité 5 : Écrire la fonction fusionner (tab: list, deb: int, fin: int) \rightarrow None qui assemble les éléments de tab entre les indices deb et fin. Les éléments seront d'abord stockés dans un tableau temporaire qui viendra ensuite écraser la partie de tab.

Fonction sorted Diviser pour régner

Étape de la fusion

Mesure de la durée

Correction

```
def fusionner(tab: list, deb: int, fin: int) ->
    None:
    res = [0 for _ in range(fin-deb+1)]
    milieu = (deb+fin)//2
    i = deb
    j = milieu+1
    k = 0
```

Code 6 – initialiser

Fonction sorted Diviser pour régner

algorithmes de tris déjà connus

Tri par insertion

Nouvelle approch

ésoudre de petits roblèmes pour solutionner un gros

Diviser pour régner : un algorithme récursif

algorithme récursif Étape de la fusion

> erformances du i fusion

Mesure de la durée d'exécution Complexité

Correction

```
while i <= milieu and j <= fin:
    if tab[i] < tab[j]:
        res[k] = tab[i]
        i += 1

else:
        res[k] = tab[j]
        j += 1
        k += 1</pre>
```

Code 7 – assembler

Fonction sorted Diviser pour régner

Kapper : des algorithmes de tris déjà connus

Fri par sélection Fri par insertion

Comparaison des performances

ouvelle approche

Résoudre de petits problèmes ...pour solutionner un gro

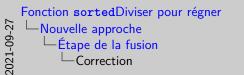
Diviser pour régner : un

algorithme récursif Étane de la fusion

Étape de la fusion

Performances du tri fusion Mesure de la durée

Complexité



```
for il in range(i, milieu+1):
res[k] = tab[il]
      k += 1
4 for j1 in range(j, fin+1):
5    res[k] = tab[j1]
6    k += 1
                             Code 8 - compléter
```

Correction

```
1 for i1 in range(i, milieu+1):
      res[k] = tab[i1]
      k += 1
  for j1 in range(j, fin+1):
      res[k] = tab[j1]
5
6
      k += 1
```

Code 8 – compléter

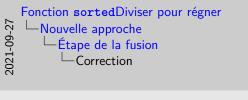
Fonction sorted Diviser pour régner

Comparaison des

Diviser pour régner : un

Étape de la fusion

Mesure de la durée



```
# remplacement tab par res
for k in range(fin-deb+1):
      tab[deb+k] = rez[k]
```

Code 9 - remplacer

```
Correction
```

```
# remplacement tab par res
  for k in range(fin-deb+1):
      tab[deb+k] = res[k]
3
```

Code 9 - remplacer

Diviser pour régner

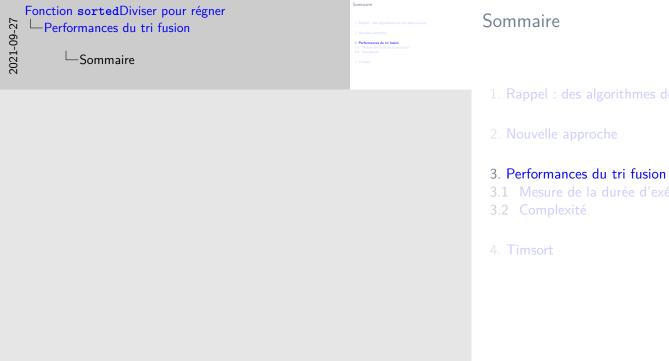
Fonction sorted

Comparaison des

Diviser pour régner : un

Étape de la fusion

Mesure de la durée



Fonction sorted

Diviser pour régner

Comparaison des

Performances du tri fusion Mesure de la durée

43 / 56

Fonction sortedDiviser pour régner -Performances du tri fusion └─Mesure de la durée d'exécution -Mesure de la durée d'exécution

Mesure de la durée d'exécution

Activité 6:

Mesure de la durée d'exécution

. Construire par compréhension un tableau de 10000

entiers compris entre 1 et 10000. Mesurer la durée d'exécution de la fonction tri fusion et des deux fonctions précédentes.

Activité 6 :

- 1. Construire par compréhension un tableau de 10000 entiers compris entre 1 et 10000.
- 2. Mesurer la durée d'exécution de la fonction tri_fusion et des deux fonctions précédentes.

Fonction sorted Diviser pour régner

Mesure de la durée

d'exécution

Fonction sortedDiviser pour régner -Performances du tri fusion └─Mesure de la durée d'exécution -Correction

```
(10000)]
deb = time()
tri fusion(tab2, 0, len(tab)=1)
fin = time()
print("fusion ", fin-deb)
 >>> selection 4.557838678359985
  >>> insertion 3.959839105606079
 >>> mort 0.0019469261169433594
 >>> fusion 0.1485743522644043
```

Correction

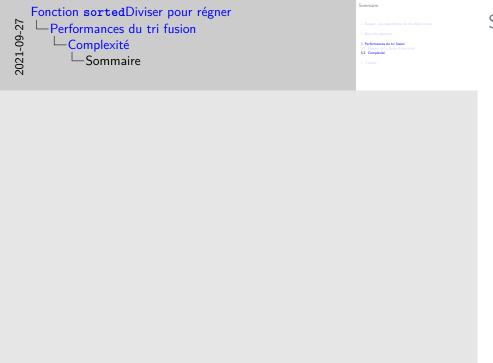
```
tab = [randint(1, 10000) for _ in range
   (10000)]
deb = time()
tri fusion(tab2, 0, len(tab)-1)
fin = time()
print("fusion ", fin-deb)
```

```
>>> sélection 4.557838678359985
>>> insertion 3.959839105606079
>>> sort 0.0019469261169433594
>>> fusion 0.1485743522644043
```

Fonction sorted Diviser pour régner

Mesure de la durée

d'exécution



Sommaire

3.2 Complexité

Fonction sorted

Diviser pour régner

Comparaison des

Diviser pour régner : un

Mesure de la durée Complexité

46 / 56

3. Performances du tri fusion

Complexité du découpage

Complexité du découpage

À chaque appel de la fonction tri_fusion nous divisons la liste en deux. Fonction sorted Diviser pour régner

> gorithmes de tris ijà connus ii par sélection

Tri par insertion

Comparaison des
performances

louvelle approche Résoudre de petits

oroblèmes ...pour solutionner un gro oroblème

Diviser pour régner : un algorithme récursif

formances du

Mesure de la durée d'exécution Complexité

Timeort

Complexité du découpage

 À chaque appel de la fonction tri_fusion nous divisons la liste en deux.

Combien de fois faut-il couper la liste en deux pour obtenir des listes d'un élément? Fonction sorted

Diviser pour régner

- divisons la liste en deux.
- ► Combien de fois faut-il couper la liste en deux pour obtenir des listes d'un élément ?

▶ À chaque appel de la fonction tri_fusion nous

► Combien de fois faut-il couper la liste en deux pour obtenir des listes d'un élément?

Complexité du découpage

► À chaque appel de la fonction tri fusion nous divisons la liste en deux.

► Combien de fois faut-il couper la liste en deux pour obtenir des listes d'un élément?

Fonction sorted Diviser pour régner

Mesure de la durée Complexité

Mathématiquement nous cherchons a tel que :

$$\frac{n}{2a} = 1$$

n gros

Fonction sorted

Diviser pour régner

Mesure de la durée d'exécution Complexité

Fonction sortedDiviser pour régner —Performances du tri fusion —Complexité

Le logarithme base 2 moté deg. se définit $\log_2(2^n) = x$. $\frac{p}{p} = 1$ $\Longleftrightarrow n - 2^n$ $\Longleftrightarrow \log_2 n - \log(2^n)$ $\Longleftrightarrow \log_2 n - x$ A rectange $n - \log_2(2^n)$ and n - x is a complete de découpage en sons défine en logarithmique.

Le logarithme base 2 noté log_2 se définit : $log_2(2^x) = x$.

$$\frac{n}{2^{a}} = 1$$

$$\iff n = 2^{a}$$

$$\iff \log_{2} n = \log_{2}(2^{a})$$

$$\iff \log_{2} n = a$$

À retenir

La complexité du découpage en sous-listes est **logarithmique**. Fonction sorted Diviser pour régner

Rappel : des algorithmes de tris déjà connus Tri par sélection

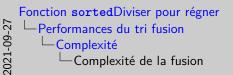
Nouvelle approche

problèmes
...pour solutionner un gros
problème
Diviser pour régner : un

Diviser pour régner : un algorithme récursif Étape de la fusion

erformances du ri fusion

Mesure de la durée d'exécution Complexité



Complexité de la fusion

qq soit le niveau, pour fusionner on se contente de parcourir une fois un tableau

Complexité de la fusion

assembler deux listes de taille -

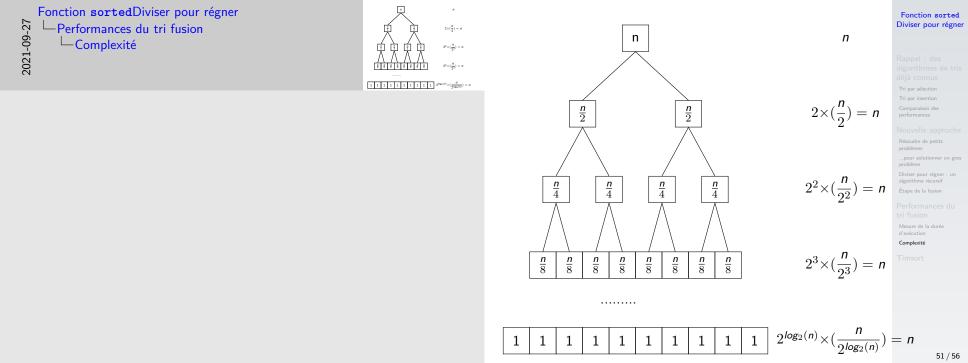
La fonction fusionner réalise n comparaisons pour

La fonction fusionner réalise n comparaisons pour assembler deux listes de taille $\frac{n}{2}$.

Fonction sorted Diviser pour régner

Mesure de la durée

Complexité



Complexité du tri fusion

À retenir

Chaque niveau de fusion a un coup de n et il y a $log_2(n)$ niveaux.

$$O(n \times log_2(n))$$

Fonction sorted Diviser pour régner

Comparaison des

Mesure de la durée

Complexité

Activité 7 : Sachant que la complexité du tri par sélection est quadratique, la comparer au tri fusion pour un tableau de 100, 1000, 10000, 100000 éléments.

 $log_2(n) = \frac{ln n}{ln 2}$

Activité 7 : Sachant que la complexité du tri par sélection est quadratique, la comparer au tri fusion pour un tableau de 100, 1000, 10000, 100000 éléments.

$$\log_2(n) = \frac{\ln n}{\ln 2}$$

Fonction sorted Diviser pour régner

Kappel : des algorithmes de tris déjà connus Tri par sélection

Comparaison des performances

Résoudre de petits problèmes ...pour solutionner un gros problème

roblème Diviser pour régner : un Igorithme récursif

rformances du

tri fusion

Mesure de la durée
d'exécution

Complexité

Correction

Correction

éléments	tri par sélection	tri fusion
100	10^{4}	664
1000	10^{6}	9966
10000	10^{8}	132877
100000	10^{10}	1660964

Fonction sorted Diviser pour régner

Rappel : des algorithmes de tris déià connus

Tri par sélection

Tri par insertion

Comparaison des

Résoudre de petits problèmes

problème Diviser pour régner : un

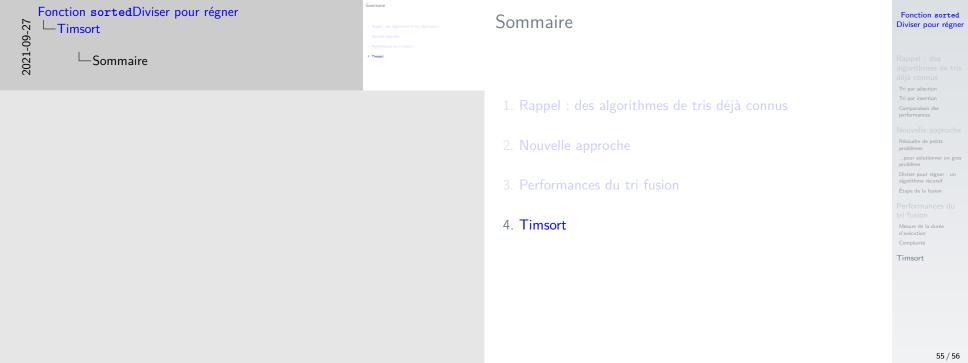
Diviser pour régner : un algorithme récursif

Algorithme recursit Étape de la fusion

Performances du ri fusion

Mesure de la durée d'exécution

Complexité



Fonction sortedDiviser pour régner └─Timsort

-Timsort

La méthode native gort implémente l'algorithme Timsort mis au point par Tim Peters en 2002. C'est un algorithme hybride de plusieurs tris. ► Détailler dans quel cas est utilisé chacun des tris ► En discutant de la complexité, expliquer pour quelle raison le tri par insertion est plus intéressan

Timsort

La méthode native sort implémente l'algorithme **Timsort** mis au point par Tim Peters en 2002. C'est un algorithme hybride de plusieurs tris.

Activité 8 : Recherche :

- Donner les algorithmes de tris utilisés dans Timsort.
- Détailler dans quel cas est utilisé chacun des tris.
- En discutant de la complexité, expliquer pour quelle raison le tri par insertion est plus intéressant que le tri par sélection.

Fonction sorted Diviser pour régner