Généralités

Les algorithmes de tris sont fondamentaux en informatique, ils prennent en entrée une suite de valeurs (nombres, mots, ...) qu'on peut *comparer* entre eux (relation supérieur ou égal pour les nombres, ordre alphabétique pour les mots, ...) et fournissent en sortie cette liste de valeurs ordonnées.

Généralités

Les algorithmes de tris sont fondamentaux en informatique, ils prennent en entrée une suite de valeurs (nombres, mots, ...) qu'on peut *comparer* entre eux (relation supérieur ou égal pour les nombres, ordre alphabétique pour les mots, ...) et fournissent en sortie cette liste de valeurs ordonnées.

⚠ Mutabilité

- Les listes de Python sont mutables, par conséquent lorsqu'on passe une liste en argument à une fonction alors cela modifiera cette liste. Car les modifications sur le paramètre affectera toutes les références de cette liste et donc la liste initiale.

Généralités

Les algorithmes de tris sont fondamentaux en informatique, ils prennent en entrée une suite de valeurs (nombres, mots, ...) qu'on peut *comparer* entre eux (relation supérieur ou égal pour les nombres, ordre alphabétique pour les mots, ...) et fournissent en sortie cette liste de valeurs ordonnées.

⚠ Mutabilité

- Les listes de Python sont mutables, par conséquent lorsqu'on passe une liste en argument à une fonction alors cela modifiera cette liste. Car les modifications sur le paramètre affectera toutes les références de cette liste et donc la liste initiale.
- On peut donc choisir pour une fonction de tri entre :

Généralités

Les algorithmes de tris sont fondamentaux en informatique, ils prennent en entrée une suite de valeurs (nombres, mots, ...) qu'on peut *comparer* entre eux (relation supérieur ou égal pour les nombres, ordre alphabétique pour les mots, ...) et fournissent en sortie cette liste de valeurs ordonnées.

⚠ Mutabilité

- Les listes de Python sont mutables, par conséquent lorsqu'on passe une liste en argument à une fonction alors cela modifiera cette liste. Car les modifications sur le paramètre affectera toutes les références de cette liste et donc la liste initiale.
- On peut donc choisir pour une fonction de tri entre :
  - modifier la liste passée en argument *sans en créer une nouvelle*, on n'a donc pas besoin de l'instruction `return`. On dira qu'on fait un tri en place.

Généralités

Les algorithmes de tris sont fondamentaux en informatique, ils prennent en entrée une suite de valeurs (nombres, mots, ...) qu'on peut *comparer* entre eux (relation supérieur ou égal pour les nombres, ordre alphabétique pour les mots, ...) et fournissent en sortie cette liste de valeurs ordonnées.

⚠ Mutabilité

- Les listes de Python sont mutables, par conséquent lorsqu'on passe une liste en argument à une fonction alors cela modifiera cette liste. Car les modifications sur le paramètre affectera toutes les références de cette liste et donc la liste initiale.
- On peut donc choisir pour une fonction de tri entre :
  - modifier la liste passée en argument *sans en créer une nouvelle*, on n'a donc pas besoin de l'instruction `return`. On dira qu'on fait un tri en place.
  - créer une nouvelle liste *sans modifier celle donnée en argument* et il faut alors renvoyer cette liste avec `return`

Principe de l'algorithme

Un premier algorithme simple de tri est le tri par sélection aussi appelé tri par recherche itérée du minimum. Il consiste à :

- Rechercher le plus petit élément de la liste à partir de l'indice 0

Principe de l'algorithme

Un premier algorithme simple de tri est le tri par sélection aussi appelé tri par recherche itérée du minimum. Il consiste à :

- Rechercher le plus petit élément de la liste à partir de l'indice 0
- Echanger cet élément avec le premier de la liste

Principe de l'algorithme

Un premier algorithme simple de tri est le tri par sélection aussi appelé tri par recherche itérée du minimum. Il consiste à :

- Rechercher le plus petit élément de la liste à partir de l'indice 0
- Echanger cet élément avec le premier de la liste
- Rechercher le plus petit élément de la liste à partir de l'indice 1

Principe de l'algorithme

Un premier algorithme simple de tri est le tri par sélection aussi appelé tri par recherche itérée du minimum. Il consiste à :

- Rechercher le plus petit élément de la liste à partir de l'indice 0
- Echanger cet élément avec le premier de la liste
- Rechercher le plus petit élément de la liste à partir de l'indice 1
- Echanger cet élément avec le second de la liste

Principe de l'algorithme

Un premier algorithme simple de tri est le tri par sélection aussi appelé tri par recherche itérée du minimum. Il consiste à :

- Rechercher le plus petit élément de la liste à partir de l'indice 0
- Echanger cet élément avec le premier de la liste
- Rechercher le plus petit élément de la liste à partir de l'indice 1
- Echanger cet élément avec le second de la liste
- Et ainsi de suite jusqu'à ce que la liste soit entièrement triée

Exemple

On considère la liste `[12,10,18,15,14]` décrire les étapes d'un tri par sélection sur cette liste

Exemple

On considère la liste `[12,10,18,15,14]` décrire les étapes d'un tri par sélection sur cette liste

1. Sélection du plus petit élément depuis l'indice 0 : `[12,10,18,15,14]`

Exemple

On considère la liste `[12,10,18,15,14]` décrire les étapes d'un tri par sélection sur cette liste

1. Sélection du plus petit élément depuis l'indice 0 : `[12,10,18,15,14]`
2. Placement en première position de liste : `[10,12,18,15,14]`

Exemple

On considère la liste `[12,10,18,15,14]` décrire les étapes d'un tri par sélection sur cette liste

1. Sélection du plus petit élément depuis l'indice 0 : `[12,10,18,15,14]`
2. Placement en première position de liste : `[10,12,18,15,14]`
3. Sélection du plus petit élément depuis l'indice 1 : `[10,12,18,15,14]`

Exemple

On considère la liste `[12,10,18,15,14]` décrire les étapes d'un tri par sélection sur cette liste

1. Sélection du plus petit élément depuis l'indice 0 : `[12,10,18,15,14]`
2. Placement en première position de liste : `[10,12,18,15,14]`
3. Sélection du plus petit élément depuis l'indice 1 : `[10,12,18,15,14]`
4. Placement en deuxième position de liste : `[10,12,18,15,14]`

Exemple

On considère la liste `[12,10,18,15,14]` décrire les étapes d'un tri par sélection sur cette liste

1. Sélection du plus petit élément depuis l'indice 0 : `[12,10,18,15,14]`
2. Placement en première position de liste : `[10,12,18,15,14]`
3. Sélection du plus petit élément depuis l'indice 1 : `[10,12,18,15,14]`
4. Placement en deuxième position de liste : `[10,12,18,15,14]`
5. Sélection du plus petit élément depuis l'indice 2 : `[10,12,18,15,14]`

Exemple

On considère la liste `[12,10,18,15,14]` décrire les étapes d'un tri par sélection sur cette liste

1. Sélection du plus petit élément depuis l'indice 0 : `[12,10,18,15,14]`
2. Placement en première position de liste : `[10,12,18,15,14]`
3. Sélection du plus petit élément depuis l'indice 1 : `[10,12,18,15,14]`
4. Placement en deuxième position de liste : `[10,12,18,15,14]`
5. Sélection du plus petit élément depuis l'indice 2 : `[10,12,18,15,14]`
6. Placement en 3^e de liste : `[10,12,14,15,18]`

Exemple

On considère la liste `[12,10,18,15,14]` décrire les étapes d'un tri par sélection sur cette liste

1. Sélection du plus petit élément depuis l'indice 0 : `[12,10,18,15,14]`
2. Placement en première position de liste : `[10,12,18,15,14]`
3. Sélection du plus petit élément depuis l'indice 1 : `[10,12,18,15,14]`
4. Placement en deuxième position de liste : `[10,12,18,15,14]`
5. Sélection du plus petit élément depuis l'indice 2 : `[10,12,18,15,14]`
6. Placement en 3^e de liste : `[10,12,14,15,18]`
7. Sélection du plus petit élément depuis l'indice 3 : `[10,12,14,15,18]`

Exemple

On considère la liste `[12,10,18,15,14]` décrire les étapes d'un tri par sélection sur cette liste

1. Sélection du plus petit élément depuis l'indice 0 : `[12,10,18,15,14]`
2. Placement en première position de liste : `[10,12,18,15,14]`
3. Sélection du plus petit élément depuis l'indice 1 : `[10,12,18,15,14]`
4. Placement en deuxième position de liste : `[10,12,18,15,14]`
5. Sélection du plus petit élément depuis l'indice 2 : `[10,12,18,15,14]`
6. Placement en 3^e de liste : `[10,12,14,15,18]`
7. Sélection du plus petit élément depuis l'indice 3 : `[10,12,14,15,18]`
8. Placement en 4^e position de liste : `[10,12,14,15,18]`

Implémentation en Python

Ecrire les fonctions Python suivantes :

Implémentation en Python

Ecrire les fonctions Python suivantes :

- la fonction `ind_min` qui renvoie l'indice du plus petit des éléments à partir de l'indice `ind`,

Implémentation en Python

Ecrire les fonctions Python suivantes :

- la fonction `ind_min` qui renvoie l'indice du plus petit des éléments à partir de l'indice `ind`,
- la fonction `echange` qui intervertit les éléments de la liste situés aux indices `ind` et `ind_min`.

Implémentation en Python

Ecrire les fonctions Python suivantes :

- la fonction `ind_min` qui renvoie l'indice du plus petit des éléments à partir de l'indice `ind`,
- la fonction `echange` qui intervertit les éléments de la liste situés aux indices `ind` et `ind_min`.

En déduire la fonction `tri_selection` qui trie *en place* la liste donnée en argument.

Correction

- Recherche de l'indice du minimum depuis un indice donné

```
1  def indice_min(liste, idx):
2      '''Renvoie l'indice du plus petit élément de la liste à
   ↪   partir de l'indice idx'''
3      imin = idx
4      for i in range(idx+1, len(liste)):
5          if liste[i] < liste[imin]:
6              imin = i
7      return imin
```

- Echange deux éléments dans une liste en donnant leur indice

```
1  def echange(liste, i, j):
2      '''Echange les éléments d'indices i et j dans la liste'''
3      liste[i], liste[j] = liste[j], liste[i]
```

Implémentation du tri par sélection en Python

```
1  def tri_selection(liste):
2      '''Trie la liste  par sélection'''
3      for i in range(len(liste)):
4          imin = indice_min(liste, i)
5          echange(liste, i, imin)
```

Principe de l'algorithme

Le principe est de considérer qu'une partie située en début de liste est déjà triée (cette partie est initialement vide), ensuite on parcourt le reste de la liste et on insère chaque élément qu'on rencontre dans la partie déjà triée.

- on parcourt la liste à partir du premier élément

Principe de l'algorithme

Le principe est de considérer qu'une partie située en début de liste est déjà triée (cette partie est initialement vide), ensuite on parcourt le reste de la liste et on insère chaque élément qu'on rencontre dans la partie déjà triée.

- on parcourt la liste à partir du premier élément
- chaque élément rencontré est inséré à la bonne position en début de liste

Principe de l'algorithme

Le principe est de considérer qu'une partie située en début de liste est déjà triée (cette partie est initialement vide), ensuite on parcourt le reste de la liste et on insère chaque élément qu'on rencontre dans la partie déjà triée.

- on parcourt la liste à partir du premier élément
- chaque élément rencontré est inséré à la bonne position en début de liste
- Cette insertion peut se faire en échangeant cet élément avec son voisin de gauche tant qu'il lui est supérieur

Exemple

On considère la liste `[12,10,18,15,14]` voici les étapes d'un tri par insertion sur cette liste, où on, a indiqué par le séparateur | la frontière entre la partie déjà triée et celle à trier. L'élément qui va être inséré dans la partie trié est en rouge

- `[|12, 10, 18, 15, 14]`.

L'algorithme s'arrête car on a atteint la fin de la liste.

Exemple

On considère la liste `[12,10,18,15,14]` voici les étapes d'un tri par insertion sur cette liste, où on, a indiqué par le séparateur `|` la frontière entre la partie déjà triée et celle à trier. L'élément qui va être inséré dans la partie trié est en rouge

- `[|12, 10, 18, 15, 14]`.
- `[12 |,10 , 18, 15, 14]`

L'algorithme s'arrête car on a atteint la fin de la liste.

Exemple

On considère la liste `[12,10,18,15,14]` voici les étapes d'un tri par insertion sur cette liste, où on, a indiqué par le séparateur `|` la frontière entre la partie déjà triée et celle à trier. L'élément qui va être inséré dans la partie trié est en rouge

- `[|12, 10, 18, 15, 14]`.
- `[12 |,10 , 18, 15, 14]`
- `[10, 12 |,18, 15, 14]`

L'algorithme s'arrête car on a atteint la fin de la liste.

Exemple

On considère la liste `[12,10,18,15,14]` voici les étapes d'un tri par insertion sur cette liste, où on, a indiqué par le séparateur `|` la frontière entre la partie déjà triée et celle à trier. L'élément qui va être inséré dans la partie trié est en rouge

- `[|12, 10, 18, 15, 14]`.
- `[12 |,10 , 18, 15, 14]`
- `[10, 12 |,18, 15, 14]`
- `[10, 12, 18 |,15, 14]`

L'algorithme s'arrête car on a atteint la fin de la liste.

Exemple

On considère la liste `[12,10,18,15,14]` voici les étapes d'un tri par insertion sur cette liste, où on, a indiqué par le séparateur `|` la frontière entre la partie déjà triée et celle à trier. L'élément qui va être inséré dans la partie trié est en rouge

- `[|12, 10, 18, 15, 14]`.
- `[12 |,10 , 18, 15, 14]`
- `[10, 12 |,18, 15, 14]`
- `[10, 12, 18 |,15, 14]`
- `[10, 12, 15, 18 |,14]`

L'algorithme s'arrête car on a atteint la fin de la liste.

Exemple

On considère la liste [12,10,18,15,14] voici les étapes d'un tri par insertion sur cette liste, où on, a indiqué par le séparateur | la frontière entre la partie déjà triée et celle à trier. L'élément qui va être inséré dans la partie trié est en rouge

- [|12, 10, 18, 15, 14].
- [12 |,10 , 18, 15, 14]
- [10, 12 |,18, 15, 14]
- [10, 12, 18 |,15, 14]
- [10, 12, 15, 18 |,14]
- [10, 12, 14, 15, 18 |]

L'algorithme s'arrête car on a atteint la fin de la liste.

Implémentation du tri par insertion en Python

- L'ingrédient essentiel de l'algorithme est la fonction permettant d'insérer la valeur d'indice donné `idx` dans le début de la liste en supposant ce début de liste déjà triée.

Implémentation du tri par insertion en Python

- L'ingrédient essentiel de l'algorithme est la fonction permettant d'insérer la valeur d'indice donné `idx` dans le début de la liste en supposant ce début de liste déjà triée.
- C'est à dire qu'on doit écrire une fonction `insere` qui prend en argument une liste et un indice `i` et insère l'élément `liste[i]` au bon emplacement dans le début de la liste.

Implémentation du tri par insertion en Python

- L'ingrédient essentiel de l'algorithme est la fonction permettant d'insérer la valeur d'indice donné `idx` dans le début de la liste en supposant ce début de liste déjà triée.
- C'est à dire qu'on doit écrire une fonction `insere` qui prend en argument une liste et un indice `i` et insère l'élément `liste[i]` au bon emplacement dans le début de la liste.
- On propose de procéder de la façon suivante : on échange l'élément d'indice `i` avec son voisin de gauche tant qu'on a pas atteint le début de la liste (donc `i>0`) ou que l'élément à gauche est plus petit ou égal .

Correction

- Fonction d'insertion

```
1  def insere(liste, idx):
2      '''Insère l'élément d'indice idx dans la liste triée'''
3      while idx>0 and liste[idx]<liste[idx-1]:
4          echange(liste, idx, idx-1)
5          idx -= 1
```

- Tri par insertion

```
1  def tri_insertion(liste):
2      '''Trie la liste  par insertion'''
3      for i in range(len(liste)):
4          insere(liste,i)
```

Principe de l'algorithme

Le tri fusion un algorithme récursif, en effet, pour trier une liste l de taille n,

Remarque

L'algorithme du tri fusion illustre une stratégie algorithmique appelée *diviser pour régner* qui consiste à résoudre un problème en le divisant en sous-problèmes plus petits, puis en combinant les solutions de ces sous-problèmes pour résoudre le problème initial.

Principe de l'algorithme

Le tri fusion un algorithme récursif, en effet, pour trier une liste l de taille n,

- Diviser : on sépare l en deux moitiés (à une unité près) l_1 et l_2.

Remarque

L'algorithme du tri fusion illustre une stratégie algorithmique appelée *diviser pour régner* qui consiste à résoudre un problème en le divisant en sous-problèmes plus petits, puis en combinant les solutions de ces sous-problèmes pour résoudre le problème initial.

Principe de l'algorithme

Le tri fusion un algorithme récursif, en effet, pour trier une liste l de taille n,

- Diviser : on sépare l en deux moitiés (à une unité près) l_1 et l_2.
- Régner : on trie l_1 et l_2 (récursivement)

Remarque

L'algorithme du tri fusion illustre une stratégie algorithmique appelée *diviser pour régner* qui consiste à résoudre un problème en le divisant en sous-problèmes plus petits, puis en combinant les solutions de ces sous-problèmes pour résoudre le problème initial.

Principe de l'algorithme

Le tri fusion un algorithme récursif, en effet, pour trier une liste l de taille n,

- Diviser : on sépare l en deux moitiés (à une unité près) l_1 et l_2.
- Régner : on trie l_1 et l_2 (récursivement)
- Combiner : on fusionne les listes triées afin de construire la solution au problème initial

Remarque

L'algorithme du tri fusion illustre une stratégie algorithmique appelée *diviser pour régner* qui consiste à résoudre un problème en le divisant en sous-problèmes plus petits, puis en combinant les solutions de ces sous-problèmes pour résoudre le problème initial.

Exemple

On considère la liste `[12, 10, 18, 15, 14, 7]` voici les étapes d'un tri fusion sur cette liste

Exemple

On considère la liste `[12, 10, 18, 15, 14, 7]` voici les étapes d'un tri fusion sur cette liste

- Diviser : on sépare la liste en deux moitiés `[12, 10, 18]` et `[15, 14, 7]`

Exemple

On considère la liste `[12, 10, 18, 15, 14, 7]` voici les étapes d'un tri fusion sur cette liste

- Diviser : on sépare la liste en deux moitiés `[12, 10, 18]` et `[15, 14, 7]`
- Régner : on trie récursivement les deux listes

Exemple

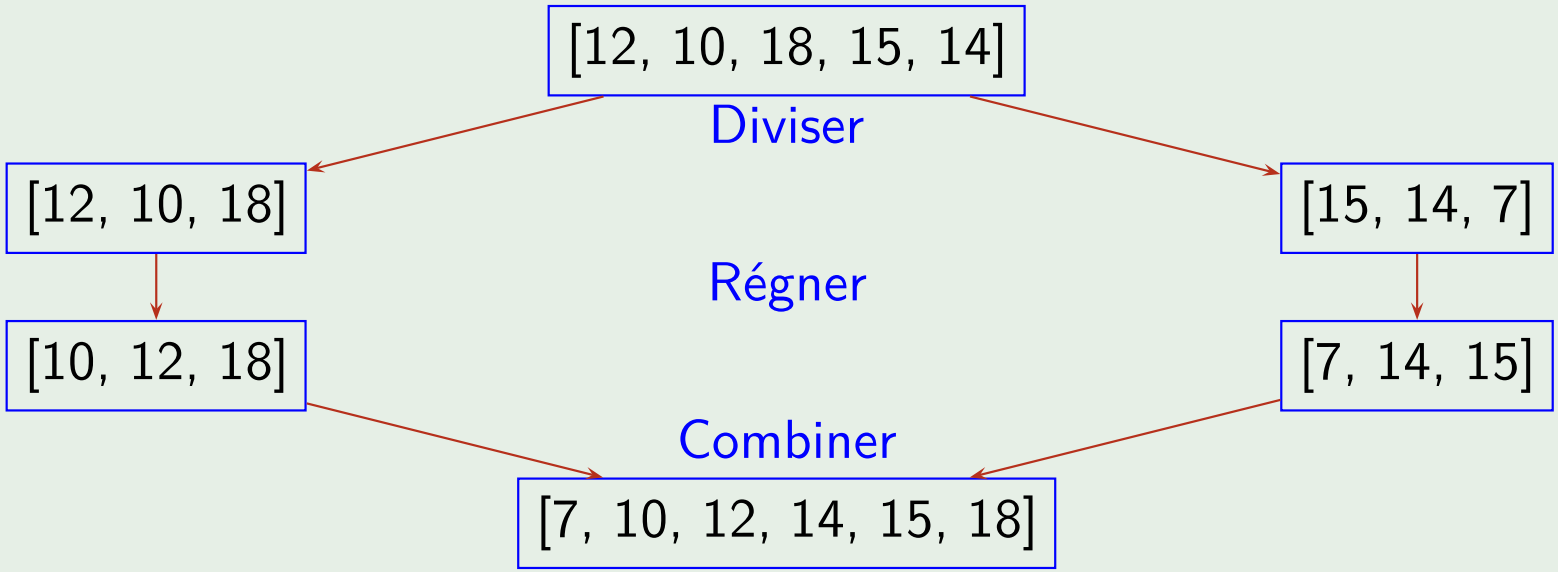
On considère la liste `[12, 10, 18, 15, 14, 7]` voici les étapes d'un tri fusion sur cette liste

- Diviser : on sépare la liste en deux moitiés `[12, 10, 18]` et `[15, 14, 7]`
- Régner : on trie récursivement les deux listes
- Combiner : on fusionne les listes triées `[10, 12, 18]` et `[15, 14, 7]`

Exemple

On considère la liste `[12, 10, 18, 15, 14, 7]` voici les étapes d'un tri fusion sur cette liste

- Diviser : on sépare la liste en deux moitiés `[12, 10, 18]` et `[15, 14, 7]`
- Régner : on trie récursivement les deux listes
- Combiner : on fusionne les listes triées `[10, 12, 18]` et `[15, 14, 7]`



Implémentation du tri fusion en Python

Implémentation du tri fusion en Python

- La première étape est de définir une fonction `fusion` qui prend en argument deux listes triées et les fusionne en une seule liste triée.

Implémentation du tri fusion en Python

- La première étape est de définir une fonction `fusion` qui prend en argument deux listes triées et les fusionne en une seule liste triée.
- Ensuite, on peut définir la fonction `tri_fusion` qui trie une liste en utilisant la méthode du tri fusion.L'étape de séparation peut s'effectuer avec l'extraction de *tranches*

Implémentation en Python

- Fusion

```
1  def fusion(l1,l2):
2      if l1==[]:
3          return l2
4      if l2==[]:
5          return l1
6      if l1[0]<l2[0]:
7          return [l1[0]]+fusion(l1[1:],l2)
8      return [l2[0]]+fusion(l1,l2[1:])
```

- Tri fusion

```
1  def tri_fusion(liste):
2      if len(liste)<=1:
3          return liste
4      l1 = liste[:len(liste)//2]
5      l2 = liste[len(liste)//2:]
6      return fusion(tri_fusion(l1),tri_fusion(l2))
```