File prioritaire (priority queue)

1. Mise en situation

Dans le transport aérien, lorsque l'aéroport est congestionné à l'arrivée des appareils, les contrôleurs aériens les placent avant l'atterrissage dans un circuit d'attente pour les faire patienter.

Ce circuit aussi appelé hippodrome, permet de préparer l'avion à l'atterrissage et de séparer les avions à l'arrivée dans les différents terminaux. On trouve donc toujours dans un aéroport un ou plusieurs circuits d'attentes où plusieurs appareils peuvent être "stockés". L'espacement vertical minimum est de 1000 pieds, d'où l'importance de bien respecter l'altitude assignée.

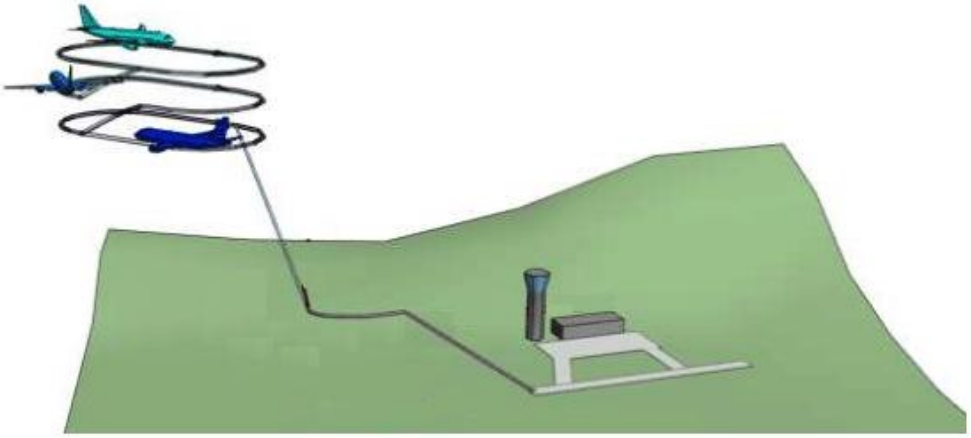
Les avions placés en attente réalisent des trajectoires en forme d'hippodromes, chaque avion avec une altitude différente. Lorsque les conditions d'atterrissage deviennent possibles (piste disponible), le contrôleur demande aux pilotes de l'avion ayant la plus faible altitude de sortir de la file pour atterrir.

Les situations peuvent aussi devenir prioritaires avec des conditions météorologiques dégradées ou des avions présentant des urgences (un avion avec un niveau bas de carburant par exemple).

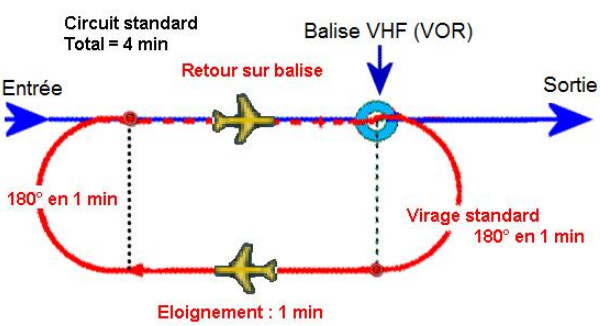
Données concernant le circuit d'attente

Temps moyen d'un tour d'attente	4 minutes
Temps moyen pour sortir de la file, atterrir et dégager la piste	10 minutes
Nombre d'avions maximums dans une file d'attente	10

Trois appareils dans le circuit d'attente avant l'atterrissage

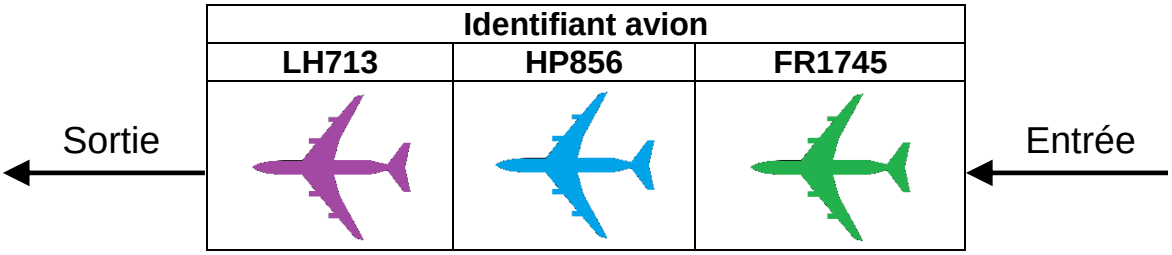


L'appareil autorisé par le contrôleur aérien sort du circuit d'attente et peut atterrir

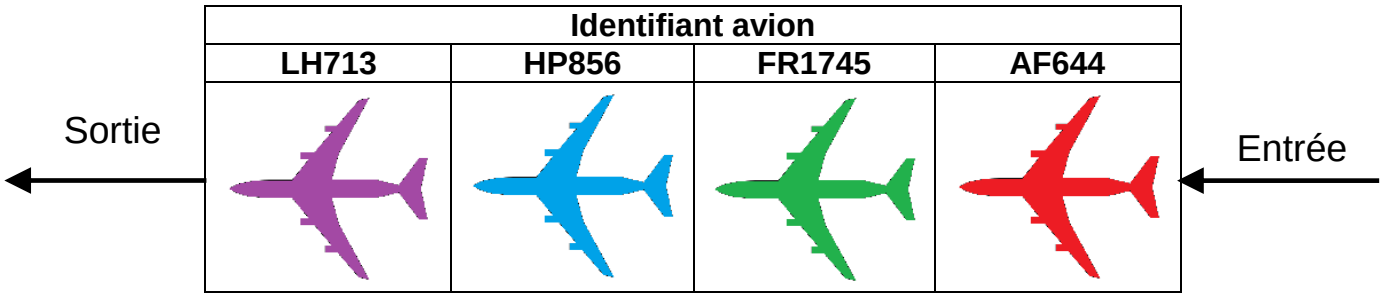


2. Partie 1 : File d'attente sans priorité

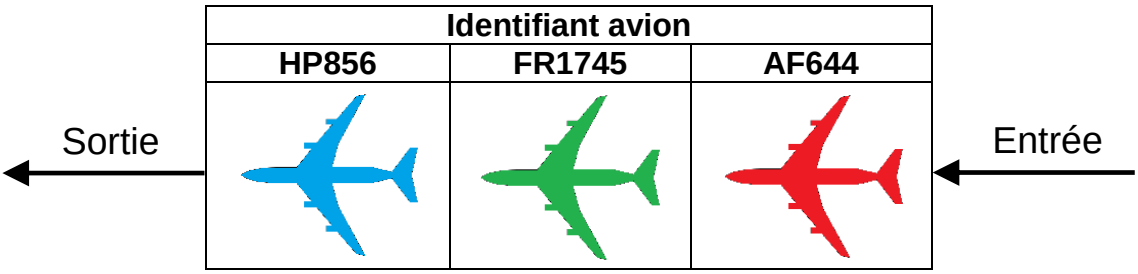
Supposons qu'il y ait 3 avions en attente d'atterrissage.



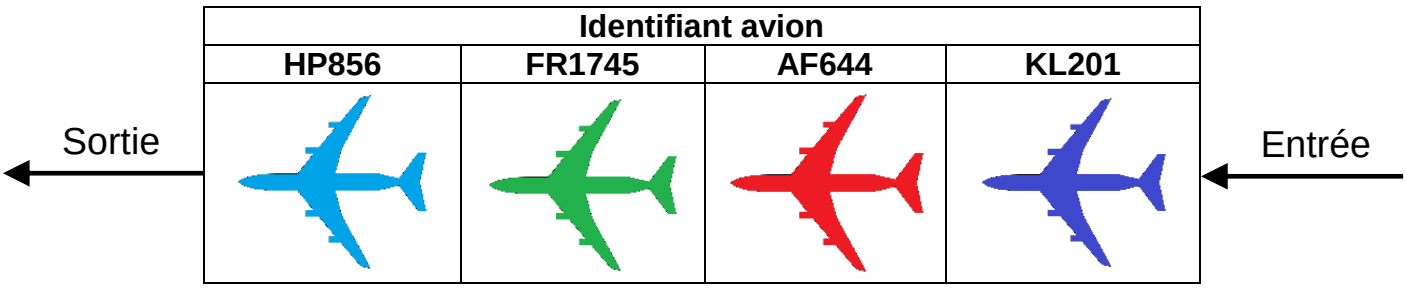
Un nouvel avion AF644 arrive en provenance de Hong-Kong. Celui -ci entre dans la file d'attente. L'opération effectuée en algorithme est enfiler('AF644').



L'avion LH713 est autorisé à atterrir. Celui-ci sort de la file. L'opération effectuée en algorithme est defiler('LH713').



Un nouvel avion KL201 arrive en provenance de Hambourg. Celui-ci entre dans la file d'attente. L'opération effectuée en algorithme est enfiler('KL201').



Le contrôleur de vol doit connaitre le nombre d'avions dans la file d'attente. L'opération effectuée en algorithme est longueur() qui retourne le nombre d'avions dans la file.

Le contrôleur de vol peut également savoir s'il n'y a plus d'avions dans la file d'attente. L'opération effectuée en algorithme est estVide() qui retourne Vrai si la file est vide.

En informatique, une file d'attente (en anglais queue) est une structure de donnée abstraite de type file FIFO.

Définition du type abstrait file FIFO (First In, First Out : premier entré, premier sorti)

- **Représentation : <'a', 'b', 'c' <**

L'élément 'a' est le premier de la file, l'élément 'c' est le dernier. Ainsi, les premiers éléments ajoutés à la file seront les premiers à en être retirés.

- **Opérations :**

- créer(F) : création de la file F vide
  - → File
- enfiler(F,avion) : ajoute l'élément avion en dernier dans F
  - File x avion → File
- defiler(F) : retire l'élément avion en premier dans F
  - File → File // précondition : F n'est pas vide (avion présent)
- longueur(F) : nombre d'avions dans F
  - File → Entier
- estVide : retourne Vrai si F est vide
  - File → Booléen

➢ **Décontextualisation :** Les avions sont repérés par un identifiant unique. La file d'attente est une file de chaines de caractères tous distincts.

➢ **Spécification :**

– Entrée : File d'avions enAttente ; Avion à ajouter ou retirer

– Sortie : File d'avions enAttente modifiée

– Rôle : Ajouter ou retirer un avion de la file enAttente

– Précondition : L'avion à retirer est présent dans la file enAttente

➢ **Principe de l'algorithme :** On ajoute ou on extrait les avions un à un de la file enAttente

Algorithme	Script Python

Fonction créer() → File
enAttente ← [ ]
retourner enAttente
 	
Fonction longueur(File enAttente) → Entier
retourner taille(enAttente)
 	
Fonction enfiler(File enAttente, Chaine caractère avion) → File
enfiler(avion)
 	
Fonction defiler(File enAttente) → File
retourner defiler(enAttente)
 	
Fonction estVide(File enAttente) → Booléen
vide ← faux
si enAttente = [ ] alors
vide ← vrai
finsi
retourner vide
	`def creer():`
	`  enAttente=[]`
	`  return enAttente`
	
	`def longueur(enAttente):`
	`  return len(enAttente)`
	
	`def enfiler(enAttente,avion):`
	`  enAttente.append(avion)`
	
	`def defiler(enAttente):`
	`  return enAttente.pop(0)`
	
	`def estVide(enAttente):`
	`  vide=False`
	`  if enAttente==[]:`
	`  vide=True`
	`  return vide`

Tests	Résultats affichés dans la console

`enAttente = creer()`
`enfiler(enAttente,'LH713')`
`enfiler(enAttente,'HP856')`
`enfiler(enAttente,'FR1745')`
`enfiler(enAttente,'AF644')`
`print(enAttente)`
`print('La liste est vide ?',estVide(enAttente))`
`print('La longueur de la liste est :'`
`,longueur(enAttente))`
`print(defiler(enAttente))`
`print(defiler(enAttente))`
`print(defiler(enAttente))`
`print(defiler(enAttente))`
`print('La liste est vide ?',estVide(enAttente))`
`print('La longueur de la liste est :'`
`,longueur(enAttente))`
	['LH713', 'HP856', 'FR1745', 'AF644']
	La liste est vide ? False
	La longueur de la liste est : 4
	LH713
	HP856
	FR1745
	AF644
	La liste est vide ? True
	La longueur de la liste est : 0

Vous trouverez en annexe un script Python utilisant la programmation orienté objet.

➢ **Complexité de l'algorithme :**

– Il termine car les opérations sur la file enAttente terminent.

– Il est correct car les avions sont bien ajoutés ou retirés de la file enAttente selon le principe du premier arrivé, premier sorti.

– Il prend un temps O(nombre d'avions en attente) car les opérations sur la file sont en temps O(1) pour l'ajout d'un avion et en O(n) pour le retrait.

Opération enfiler(F,avion)	append	O(1)
Opération defiler(F)	pop(0)	O(n)

https://wiki.python.org/moin/TimeComplexity

3. Partie 2 : file d'attente avec priorité

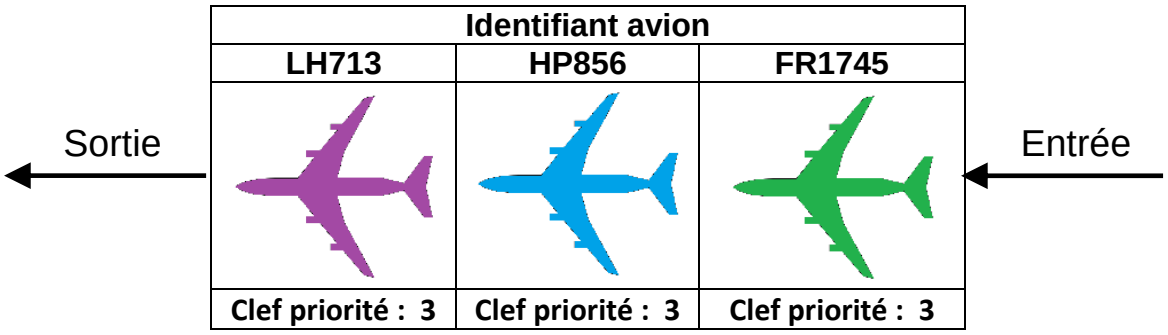
Dans le cas d'une file d'attente avec priorité, les avions ont maintenant un ordre de priorité. En effet, il se peut qu'un avion ait peu de carburant restant par exemple et doive atterrir avant les autres.

Le niveau de priorité est défini par une clef où la priorité la plus importante correspond à la valeur la plus petite.

Les nouvelles contraintes de la file sont donc les suivantes :

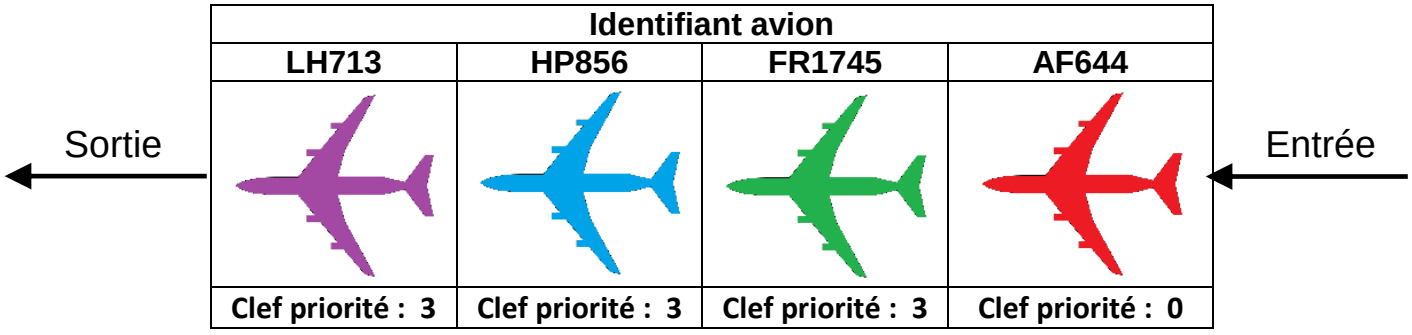
	Clef de priorité	**Information**
Tout va bien dans l'avion, il reste 1H de carburant	3	Avion non prioritaire
Il reste 30 minutes de carburant	2	
Il reste 15 minutes de carburant	1	
Panne sèche, l'avion plane	0	Avion prioritaire devant tous les autres

Supposons qu'il y a 3 avions en attente d'atterrissage de même priorité dans la file d'attente.



Le 1er avion autorisé à atterrir dans ce cas sera le numéro LH713. En effet, c'est le premier de la file FIFO (premier entré, premier sorti).

Si un nouvel avion AF644 arrive en provenance de Hong-Kong avec une priorité maximale, il entre dans la file d'attente avec une clef de priorité 0. L'opération est enfiler(0,'AF644').



Dans ce cas, le 1er avion autorisé à atterrir sera maintenant le numéro AF644 en raison de sa clef de priorité 0. L'opération defiler(file) consiste alors à rechercher l'avion ayant la clef de priorité la plus petite et à le retirer de la file.

Seule l'opération defiler() est différente entre la file sans priorité et la file avec priorité.

L'élément avion à enfiler sera par exemple un tuple pour associer la clef de priorité avec l'identifiant de l'avion ⇒ avion=(3,'LH713')

avion[0] : clef de priorité ⇒ ici 3

avion[1] : identifiant de l'avion ⇒ ici LH713

Définition du type abstrait file de priorité

- **Opérations :**

- créer(F) : création de la file F vide
  - → File
- enfiler(F,avion) : ajoute l'élément avion en dernier dans F
  - File x avion → File
- defiler(F) : retire l'élément avion qui a la clef de priorité la plus faible dans F
  - File → avion // précondition : F n'est pas vide (avion présent)
- longueur(F) : nombre d'avions dans F
  - File → Entier
- estVide : retourne Vrai si F est vide
  - File → Booléen

➢ **Décontextualisation :** Le 1er avion autorisé à atterrir est l'avion avec la clef de priorité la plus faible. Il faut donc rechercher l'avion ayant la clef de priorité la plus petite et le retirer de la file.

➢ **Spécification :**

– Entrée : File d'avions enAttente ; Avion à ajouter ou retirer

– Sortie : File d'avions enAttente modifiée

– Rôle : Ajouter un avion ou retirer un avion qui a la clef de priorité la plus petite dans la file enAttente

– Précondition : L'avion à retirer est présent dans la file enAttente

➢ **Principe de l'algorithme :** On ajoute un avion dans la file enAttente avec sa clef de priorité et son identifiant. Pour extraire un avion, on recherche la clef de priorité minimale dans la file puis on extrait l'avion de la file enAttente. En cas d'égalité des clefs de priorité, on retire l'avion en premier dans la file.

Algorithme de la fonction defiler(enAttente)

```
Fonction defiler (file enAttente) → tuple
    avion ← None
    si estVide(enAttente) = faux alors
        indicePrioritaire ← 0
        pour indice de 0 à longueur(enAttente) -1 pas de 1 faire
            si enAttente [indice][0]  < enAttente [indicePrioritaire][0]
                indicePrioritaire ← indice
            fin si
        fin pour
        avion ← enAttente [indicePrioritaire]
        defiler (enAttente [indicePrioritaire])
    fin si
    retourner avion
```

Script Python de la fonction defiler(enAttente)

```
def defiler(enAttente):
    avion=None
    if estVide(enAttente)==False:
        indicePrioritaire=0
        for indice in range(0,len(enAttente)):
            if enAttente[indice][0] < enAttente[indicePrioritaire][0]:
                indicePrioritaire = indice
        avion = enAttente[indicePrioritaire]
        enAttente.pop(indicePrioritaire)
    return avion
```

Tests	Résultats affichés dans la console

`enAttente = initialisation()`
`enfiler(enAttente,(3,'LH713'))`
`enfiler(enAttente,(3,'HP856'))`
`enfiler(enAttente,(3,'FR1745'))`
`enfiler(enAttente,(0,'AF644'))`
`print(enAttente)`
`print('La liste est vide ?'`
`,estVide(enAttente))`
`print('La longueur de la liste est :'`
`,longueur(enAttente))`
`print(defiler(enAttente))`
`print(defiler(enAttente))`
`print(defiler(enAttente))`
`print(defiler(enAttente))`
`print('La liste est vide ?'`
`,estVide(enAttente))`
`print('La longueur de la liste est :`
`,longueur(enAttente))`
	[(3, 'LH713'), (3, 'HP856'), (3, 'FR1745'), (0, 'AF644')]
	La liste est vide ? False
	La longueur de la liste est : 4
	(0, 'AF644')
	(3, 'LH713')
	(3, 'HP856')
	(3, 'FR1745')
	La liste est vide ? True
	La longueur de la liste est : 0

➢ **Complexité de l'algorithme :**

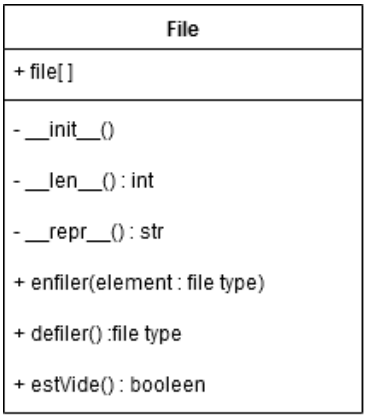
– Il termine car les opérations sur la file enAttente terminent et la boucle pour termine.

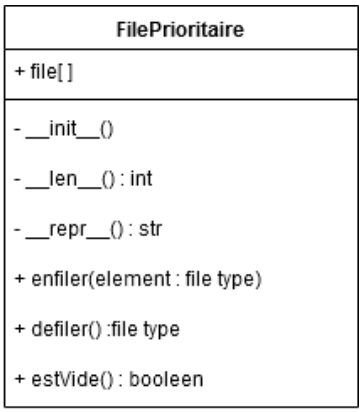
– Il est correct car les avions sont bien ajoutés ou retirés de la file enAttente selon la clef de priorité.

– Il prend un temps O(nombre d'avions en attente) car les opérations sur la file sont en temps O(1) pour l'ajout d'un avion et en 2*O(n) pour le retrait.

Opération enfiler(F,avion)	append	O(1)
Opération defiler(F)	recherche de la clef de priorité minimale	O(n)
	pop(indice)	O(n)

Annexe : Files avec la POO






```
class File:
    def __init__(self):
        self.file = []

    def __len__(self):
        return len(self.file)

    def __repr__(self):
        return ' '.join([str(i) for i in self.file])

    def enfiler(self, avion):
        self.file.append(avion)

    def defiler(self):
        return self.file.pop(0)

    def estVide(self):
        vide=False
        if self.file==[]:
            vide=True
        return vide)
```

```
class FilePrioritaire:
    def __init__(self):
        self.file = []

    def __len__(self):
        return len(self.file)

    def __repr__(self):
        return ' '.join([str(i) for i in self.file])

    def enfiler(self, avion):
        self.file.append(avion)

    def defiler(self):
        avion=None
        indicePrioritaire=0
        if self.estVide()==False:
            for indice in range(0,len(self.file)):
                if self.file[indice][0] <
self.file[indicePrioritaire][0]:
                    indicePrioritaire = indice
            avion = self.file[indicePrioritaire]
            del self.file[indicePrioritaire]
        return avion

    def estVide(self):
        vide=False
        if self.file==[]:
            vide=True
        return vide
```

```
f = File()
f.enfiler('LH713')
f.enfiler('HP856')
f.enfiler('FR1745')
f.enfiler('AF644')
print(f)
print('La liste est vide ?',f.estVide())
print('La longueur de la liste est :',len(f))
print(f.defiler())
print(f.defiler())
print(f.defiler())
print(f.defiler())
print('La liste est vide ?',f.estVide())
print('La longueur de la liste est :',len(f))
```

```
f = FilePrioritaire()
f.enfiler((3,'LH713'))
f.enfiler((3,'HP856'))
f.enfiler((2,'FR1745'))
f.enfiler((0,'AF644'))
print(f.file)
print('la liste est vide ?',f.estVide())
print('la longueur de la liste est :',len(f))
print(f.defiler())
print(f.defiler())
print(f.defiler())
print(f.defiler())
print('la liste est vide ?',f.estVide())
print('la longueur de la liste est :',len(f))
```