

# PARTIE I

## LE TYPE ABSTRAIT FILE DE PRIORITÉ

# RAPPEL : TYPES ABSTRAITS / STRUCTURES DE DONNÉES

TYPES ABSTRAITS	STRUCTURES DE DONNÉES
PILE ENSEMBLE FILE DICTIONNAIRE LISTE PYTHON <u>FILE DE PRIORITÉ</u> ?	tableau vecteur tableau dynamique table de hachage liste chaînée arbre binaire
spécification formelle ensemble de données & opérations possibles ≈ interface	manière concrète d'organiser les données ≈ mécanisme

— implémentation —

OBJECTIF : Trouver la structure de données qui optimise la complexité asymptotique de chaque opération

C'EST QUOI UNE FILE DE PRIORITÉ ?

# EXEMPLE MOTIVATIONNEL

Objectif: Développer un client pour télécharger des torrents

→ on veut choisir rapidement les utilisateurs qui ont le  $\oplus$  de bande passante

The screenshot shows the BitTorrent 7.8.0 interface. The top panel lists three torrents: 'WORDPRESS-PREMIUM-PLUGINS.rar', 'Gravity Forms for WordPress - Premium Plugin', and 'TorrentFreak Premium Enterprise 12.0.0.000 - Setup - Patch - ...'. The third torrent is selected. The bottom panel shows the 'Peers' tab for this torrent, displaying a list of peers with columns for IP, Name, Flags, % (Progress), Down Speed, Up Speed, Ratio, Up/Down Ratio, and Peer ID.

IP	Name	Flags	%	Down Speed	Up Speed	Ratio	Up/Down Ratio	Peer ID
82.77.130.197 [uTP]	BitTorrent 7.10	D+P	100.0	0.2 KB/s	29 [0]			1.57 KB/s
100.1.165.235 [uTP]	µTorrent 3.5	D+P	100.0	0.3 KB/s	33 [0]			1.51 KB/s
101.192.104.141 [uTP]	µTorrent 3.5.12	D+P	100.0	0.5 KB/s	9 [0]			89 KB/s
101.99.120.151-127 [uTP]	BitTorrent 7.8.7	D+P	100.0	0.2 KB/s	3 [0]			256 KB/s
117.74.224.50 [uTP]	BitTorrent 7.8.0	D+P	100.0	0.4 KB/s	3 [0]			176 KB/s
107.50.138.45 [uTP]	µTorrent 3.5	D+P	100.0	0.3 KB/s	2 [0]			120 KB/s
104.167.8-dynamic	µTorrent 3.5	D+P	100.0	0.3 KB/s	2 [0]			112 KB/s
200.84.62.14 [uTP]	µTorrent 3.5	D+P	100.0	0.5 KB/s	3 [0]			96.0 KB/s
100.180.254.80 [uTP]	µTorrent 3.5	D+P	100.0	0.4 KB/s	2 [0]			64.0 KB/s
117.197.128.232 [uTP]	µTorrent 3.5	D+P	100.0	0.3 KB/s	2 [0]			32.0 KB/s
40.146.145.240 [uTP]	µTorrent 3.5	D+P	100.0	0.2 KB/s	3 [0]			16.0 KB/s
176.63.207.115	µTorrent 3.5	D+H	100.0	0.5 KB/s	3 [0]			16.0 KB/s
188.10.211.194 [uTP]	µTorrent 3.5	D+P	100.0	0.5 KB/s	3 [0]			16.0 KB/s
188.236.93.177 [uTP]	µTorrent 3.5	D+P	100.0	0.2 KB/s	1 [0]			16.0 KB/s

Peers

106.198.254.80: 100 KO/s  
200.84.62.14 : 251 KO/s  
...

## Opérations:

- Utilisateur bande passante max?
- Ajouter/supprimer un utilisateur
- Changer valeur bande passante utilisateur

# FILE DE PRIORITÉ (BASIQUE)

On ne s'embêtera pas avec des clés/valeurs

## OPÉRATION 1

Créer une file de priorité vide



## OPÉRATION 2

Ajouter un entier dans la file



## OPÉRATION 3

Renvoyer le minimum



On peut remplacer "minimum" par "maximum"

## OPÉRATION 4

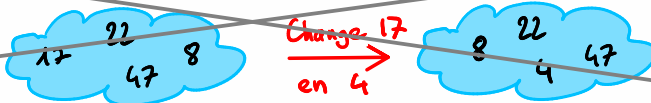
Supprimer le minimum



## OPÉRATION 5

Modifier la valeur d'un entier

(enlevé par souci de simplicité)



# FILE DE PRIORITÉ "ENRICHIE"

On ne verra pas ce cours-ci :

FILE DE PRIORITÉ	
<u>Opérations :</u>	<ul style="list-style-type: none"><li>- Créer une file de priorité vide</li><li>- Ajouter un couple clé/valeur</li><li>- Renvoyer la clé de valeur minimale</li><li>- Supprimer le couple clé/valeur de valeur minimale</li><li>- Modifier la valeur d'une clé</li></ul>

## Application en théorie des graphes :

- Algorithme de Dijkstra
- Algorithme de Prim

Comment on ferait : Ajouter un dictionnaire en plus de la structure qu'on verra aujourd'hui

## PARTIE ②

IMPLÉMENTATIONS  
SOUS - OPTIMALES

# PREMIÈRE IDÉE : TABLEAU DYNAMIQUE

tableau

15	33	3	90	18	21
----	----	---	----	----	----

Opérations	Complexité
Créer une file de priorité vide	$O(1)$
Ajouter un entier dans la file	$O(1)^*$
Renvoyer le minimum Il faut parcourir le tableau!	$O(n)$
Supprimer le minimum  On échange avec l'élément final et on supprime	$O(n)$  $O(1)$ si on connaît la position du minimum

$n$  = nombre d'éléments

\* : complexité amortie

## DEUXIÈME IDÉE : TABLEAU DYNAMIQUE TRIÉ (À L'ENVERS)

tableau

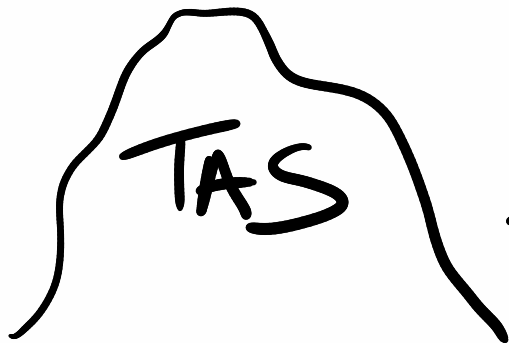
90	33	21	18	15	3
----	----	----	----	----	---

Opérations	Complexité
Créer une file de priorité vide	$O(1)$
Ajouter un entier dans la file Il faut décaler les entiers comme dans un tri insertion	$O(n)$
Renvoyer le minimum C'est le dernier!	$O(1)$
Supprimer le minimum	$O(1)$

$n$  = nombre  
d'éléments



# PARTIE III



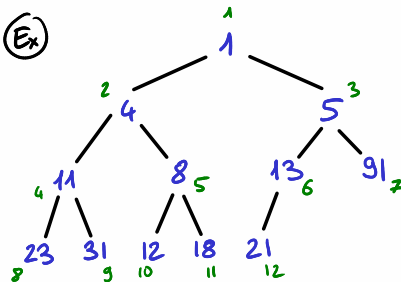
BINAIRES

## TAS BINAIRE : DÉFINITION

### Definition

tas binaire min = arbre binaire

- presque complet (rempli à tous les niveaux sauf éventuellement le dernier niveau, rempli de la gauche jusqu'à un certain point)
- où chaque nœud a une valeur plus petite que celles de ses enfants

$$(E_x)$$


En machine, sous forme de tableau

	1	2	3	4	5	6	7	8	9	10	11	12
tableau	1	4	5	11	8	13	9	23	31	12	18	21

## Astuce

Si les indices du tableau vont de 1 à  $n$ ,

- un noeud en position  $i$  a son enfant gauche en position  $2i$
- un noeud en position  $i$  a son enfant droit en position  $2i+1$
- un noeud en position  $i$  a son parent en position  $\lfloor i/2 \rfloor$

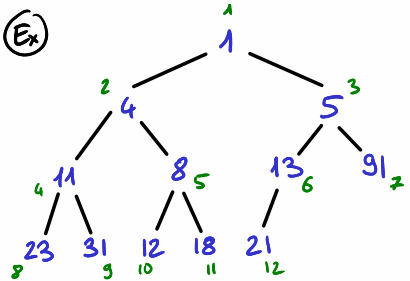
# TAS BINAIRE : DÉFINITION

## Définition

tas binaire min = arbre binaire

- presque complet (rempli à tous les niveaux sauf éventuellement le dernier niveau, rempli de la gauche jusqu'à un certain point)
- où chaque nœud a une valeur plus petite que celles de ses enfants

(Ex)



Vo qu'il est presque complet, la hauteur de l'arbre vaut  $\lfloor \log_2 (\text{nb d'éléments}) \rfloor$

Preuve Un arbre presque complet de hauteur  $h$  avec  $n$  nœuds satisfait

$$\begin{aligned}
 1 + 2 + 2^2 + \dots + 2^{h-1} &< n \leq 1 + 2 + 2^2 + \dots + 2^h \\
 \frac{2^h - 1}{2^h} &< n \leq \frac{2^{h+1} - 1}{2^h} \\
 \text{soit} & \\
 \text{ou} & \\
 \text{et donc} & \quad h \leq \log_2(n) < h+1.
 \end{aligned}$$

# OPÉRATION 1 :

Dans notre tête



# CRÉATION D'UNE FILE VIDE

Dans la mémoire de l'ordinateur

tableau



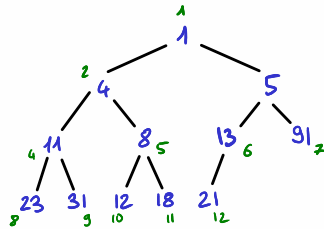
Création d'une file de priorité vide :

- Allouer un tableau dynamique vide

Complexité :  $O(1)$

# OPÉRATION 2 : AJOUT D'UN ÉLÉMENT

Dans notre tête



Dans la mémoire de l'ordinateur

tableau

1	2	3	4	5	6	7	8	9	10	11	12
1	4	5	11	8	13	91	23	31	12	18	21

Ajout de  
2

tableau

1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	4	2	11	8	5	91	23	31	12	18	21	13	2

Annotations: An arrow labeled ③ points from index 13 to index 5. An arrow labeled ① points from index 14 to index 5. A handwritten '2' and '15' are above index 6. A handwritten '14' is to the right of index 13.

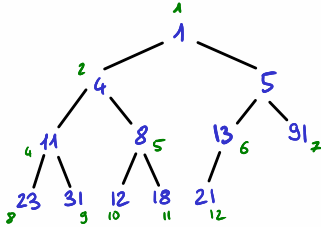
Instructions :

- Placer  $x$  à la fin de tableau
- Si  $x$  est plus grand que son parent, échanger les 2 nombres
- Ainsi de suite, jusqu'à que  $x$  soit à la racine ou plus petit que son parent

Complexité (amortie) :  $O(\log(\text{nb éléments}))$

# OPÉRATION 3 : MINIMUM

Dans notre tête



Dans la mémoire de l'ordinateur

tableau

1	2	3	4	5	6	7	8	9	10	11	12
1	4	5	11	8	13	31	23	31	12	18	21

↑  
C'est ici!

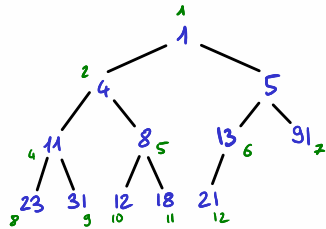
Instructions :

- Renvoyer le premier élément du tableau

Complexité :  $O(1)$

# OPÉRATION 4 : SUPPRIMER LE MINIMUM

Dans notre tête



Dans la mémoire de l'ordinateur

tableau

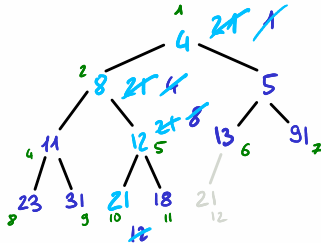
1	2	3	4	5	6	7	8	9	10	11	12
1	4	5	11	8	13	9	23	31	12	18	21

Suppression  
du minimum



tableau

1	2	3	4	5	6	7	8	9	10	11
4	8	5	11	12	13	9	23	31	21	18



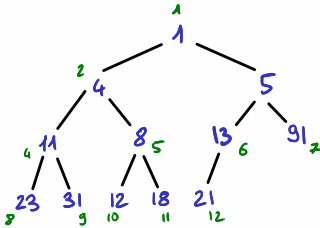
## Instructions :

- Écarter la première valeur du tableau par la dernière (on la notera  $y$ )
- Supprimer le dernier élément du tableau
- Tant que  $y$  est plus grand que ses enfants, le descendre d'un cran en l'échangeant avec son plus petit enfant.

Complexité :  $O(\log(\text{nb éléments}))$

# EN RÉSUMÉ

Dans notre tête



Dans la mémoire de l'ordinateur

tableau

1	2	3	4	5	6	7	8	9	10	11	12
1	4	5	11	8	13	31	23	31	12	18	21

Opérations	Complexité
Créer une file de priorité vide	$O(1)$
Ajouter un entier dans la file On le rajoute comme une feuille et on le fait remonter dans l'arbre	$O(\log(n))^*$
Renvoyer le minimum	$O(1)$
Supprimer le minimum On remplace la valeur de la racine par celle de la dernière feuille et on la fait descendre	$O(\log(n))$

$n$  = nombre d'éléments

\* : complexité amortie