Université de Lille

FACULTÉ DES SCIENCES ET TECHNOLOGIES
Département Informatique

Projet – L3 – S2

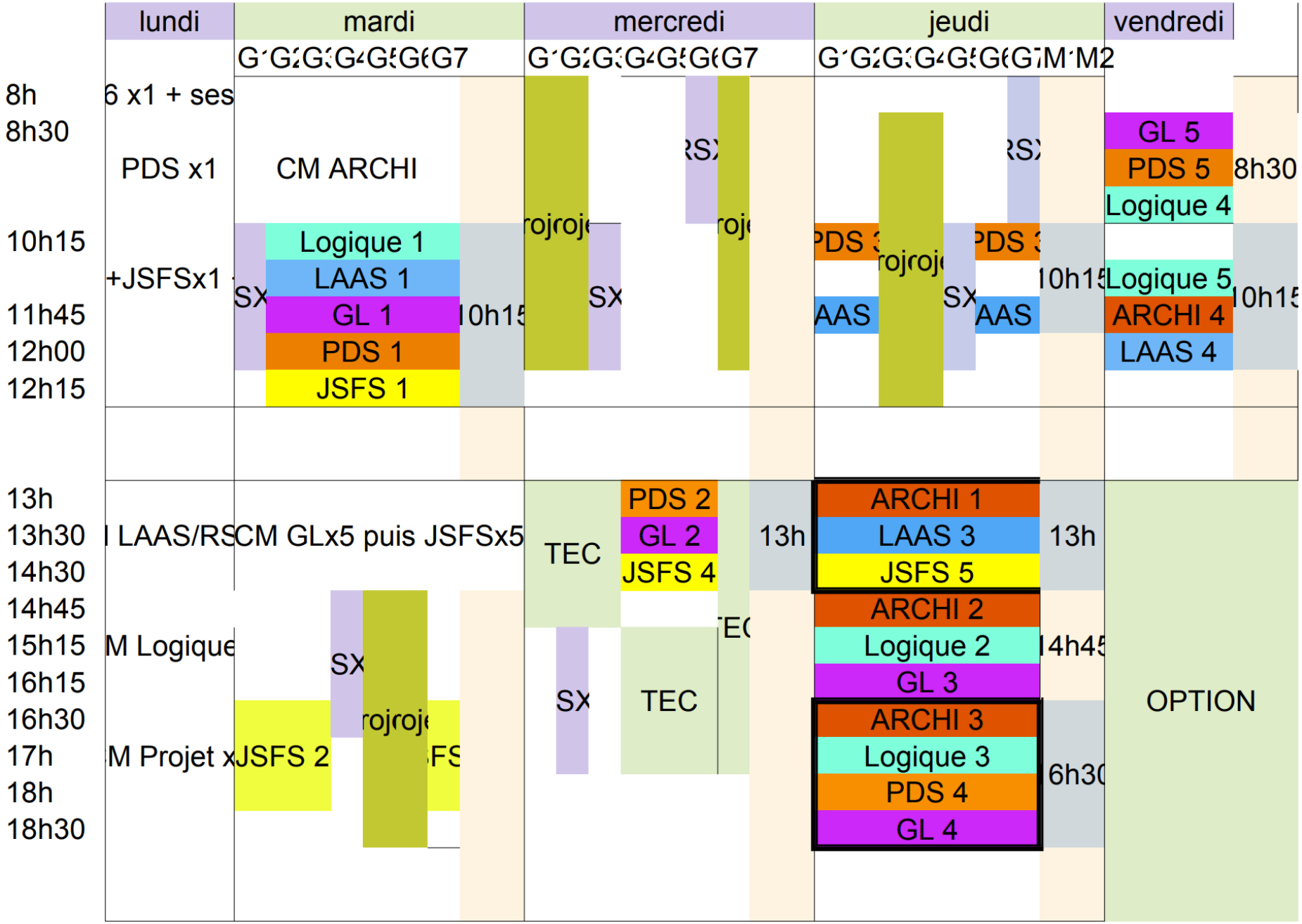
2021-2022

Organisation

- Les cours auront lieu les 3, 10 et 17 janvier 2021.
- TP (42 h TP/groupe)
  - A partir de la semaine du 10 janvier 2021 (mardi après-midi et mercredi matin et jeudi matin)
  - 2 groupes le mardi 14h45-18h30 (Chaabane (G5) – A11, Pierre Tirilly (G6) – A15)
  - 3 groupes le mercredi 8h-11h45 (Chaabane (G1) – A11, Maxime (G2)- A12, Xavier (G7) – A13)
  - 2 groupes le jeudi 8h-11h45 (Pierre-Allegraud(G3)-A11, Jean-Claude (G4) – A12)
  - 11 séances de 3,8 heures (3h45-3h50)
- Cours (4,5 h) :
  - 3 séance de 1,5 heures
  - 03/01/2021 : 08h30 - 10h (SN1 – Amphi Buffon)
  - 10/01/2021 : 16h30 – 18h (SN1 – Amphi Buffon)
  - 17/01/2021 : 16h30 – 18h (M1 – Amphi Galois)

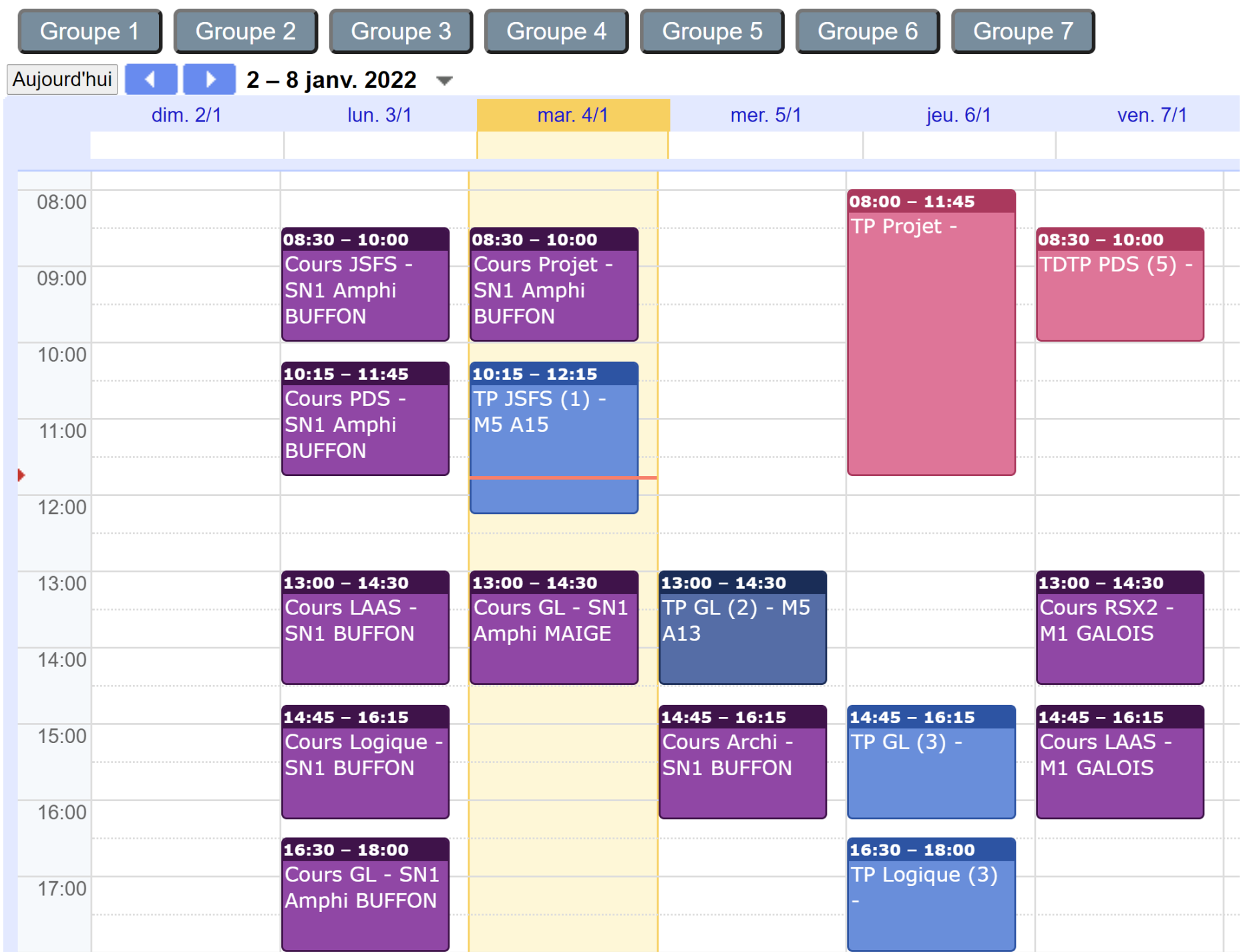
.

S6 INFO


lundi
mardi
mercredi
jeudi
vendredi
8h
8h30
10h15
11h45
12h00
12h15
13h
13h30
14h30
14h45
15h15
16h15
16h30
17h
18h
18h30
6 x1 + ses
PDS x1
+JSFSx1
CM ARCHI
Logique 1
LAAS 1
GL 1
PDS 1
JSFS 1
10h15
SX
GL 5
PDS 5
Logique 4
Logique 5
ARCHI 4
LAAS 4
8h30
10h15
AAS
PDS 2
GL 2
JSFS 4
TEC
13h
ARCHI 1
LAAS 3
JSFS 5
ARCHI 2
Logique 2
GL 3
ARCHI 3
Logique 3
PDS 4
GL 4
14h45
6h30
I LAAS/RS
CM GLx5 puis JSFSx5
M Logique
M Projet x
JSFS 2
OPTION


Groupe 1
Groupe 2
Groupe 3
Groupe 4
Groupe 5
Groupe 6
Groupe 7
Aujourd'hui
2 – 8 janv. 2022
dim. 2/1
lun. 3/1
mar. 4/1
mer. 5/1
jeu. 6/1
ven. 7/1
08:00
09:00
10:00
11:00
12:00
13:00
14:00
15:00
16:00
17:00
08:30 – 10:00 Cours JSFS - SN1 Amphi BUFFON
08:30 – 10:00 Cours Projet - SN1 Amphi BUFFON
08:00 – 11:45 TP Projet - M5 A11
08:30 – 10:00 TP GL (5) -
10:15 – 11:45 Cours PDS - SN1 Amphi BUFFON
10:15 – 11:45 TD PDS (1) - M5 A2
10:15 – 11:45 TDTP LAAS (2) -
13:00 – 14:30 Cours LAAS - SN1 BUFFON
13:00 – 14:30 Cours GL - SN1 Amphi MAIGE
13:00 – 14:30 Cours RSX2 - M1 GALOIS
14:45 – 16:15 Cours Logique - SN1 BUFFON
14:45 – 16:15 Cours Archi - SN1 BUFFON
14:45 – 16:15 TP GL (3) -
14:45 – 16:15 Cours LAAS - M1 GALOIS
16:30 – 18:00 Cours GL - SN1 Amphi BUFFON
16:30 – 18:00 TP JSFS (2) - M5 A12
16:30 – 18:00 TDTP PDS (4) -


Groupe 1
Groupe 2
Groupe 3
Groupe 4
Groupe 5
Groupe 6
Groupe 7
Aujourd'hui
2 – 8 janv. 2022
dim. 2/1
lun. 3/1
mar. 4/1
mer. 5/1
jeu. 6/1
ven. 7/1
08:00
09:00
10:00
11:00
12:00
13:00
14:00
15:00
16:00
17:00
08:30 – 10:00
Cours JSFS - SN1 Amphi BUFFON
08:30 – 10:00
Cours Projet - SN1 Amphi BUFFON
08:30 – 10:00
TP GL (5) -
10:15 – 11:45
Cours PDS - SN1 Amphi BUFFON
10:15 – 1
TP Logic (1) -M5
10:15 – 1
TP JSFS (1) - M5 A15
10:15 – 11:45
TDTP PDS (3) -
10:15 – 1
TP ARCH -
10:15 – 1
TDTP LAAS (4) -
13:00 – 14:30
Cours LAAS - SN1 BUFFON
13:00 – 14:30
Cours GL - SN1 Amphi MAIGE
13:00 – 1
TDTP PD (2) - M5
13:00 – 1
TP JSFS (4) -
13:00 – 1
TDTP LA (3) -
13:00 – 1
TP ARCHI (1) -
13:00 – 14:30
Cours RSX2 - M1 GALOIS
14:45 – 16:15
Cours Logique - SN1 BUFFON
14:45 – 18:00
TP Projet -
14:45 – 16:15
Cours Archi - SN1 BUFFON
14:45 – 16:15
TP ARCHI (2) -
14:45 – 16:15
Cours LAAS - M1 GALOIS
16:30 – 18:00
Cours GL - SN1 Amphi BUFFON
16:30 – 1
TP GL (4
16:30 – 1
TP Logique (3) -


Groupe 1
Groupe 2
Groupe 3
Groupe 4
Groupe 5
Groupe 6
Groupe 7
Aujourd'hui
2 – 8 janv. 2022
dim. 2/1
lun. 3/1
mar. 4/1
mer. 5/1
jeu. 6/1
ven. 7/1
08:00
09:00
10:00
11:00
12:00
13:00
14:00
15:00
16:00
17:00
08:30 – 10:00 Cours JSFS - SN1 Amphi BUFFON
08:30 – 10:00 Cours Projet - SN1 Amphi BUFFON
08:00 – 11:45 TP Projet -
08:30 – 10:00 TDTP PDS (5) -
10:15 – 11:45 Cours PDS - SN1 Amphi BUFFON
10:15 – 12:15 TP JSFS (1) - M5 A15
13:00 – 14:30 Cours LAAS - SN1 BUFFON
13:00 – 14:30 Cours GL - SN1 Amphi MAIGE
13:00 – 14:30 TP GL (2) - M5 A13
13:00 – 14:30 Cours RSX2 - M1 GALOIS
14:45 – 16:15 Cours Logique - SN1 BUFFON
14:45 – 16:15 Cours Archi - SN1 BUFFON
14:45 – 16:15 TP GL (3) -
14:45 – 16:15 Cours LAAS - M1 GALOIS
16:30 – 18:00 Cours GL - SN1 Amphi BUFFON
16:30 – 18:00 TP Logique (3) -

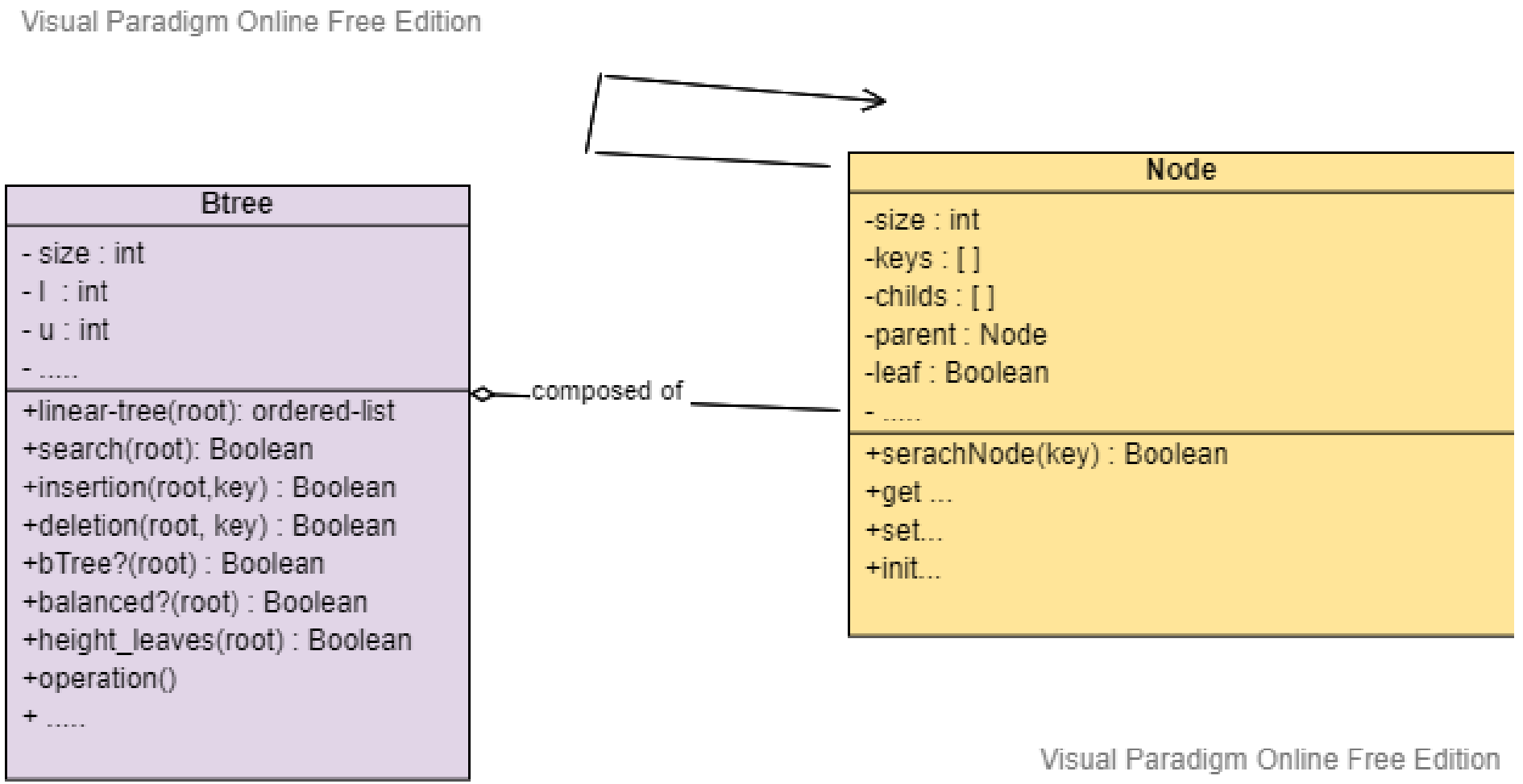
Structure de données : Arbre-B

- Arbre n-aire équilibré
- Recherche en Log(n). n : nombre de clés.
- Populaire dans les bases de données
- Inventé par Rudolf Bayer et Edward Meyers McCreight, 1970

Structure de données : Arbre-B

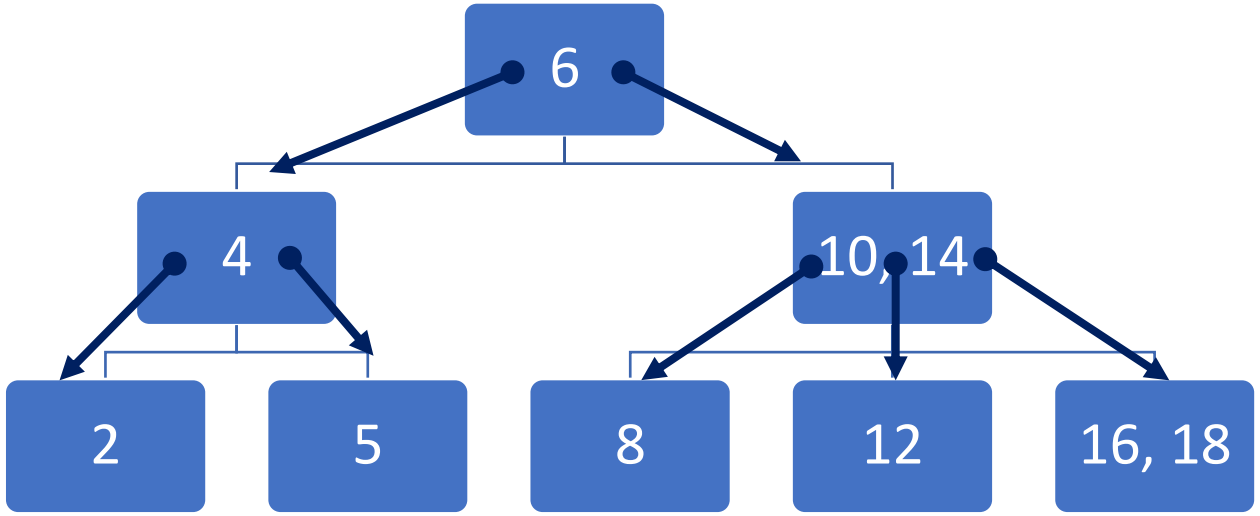
- Arbre-B définit par 2 paramètres : L et U. U est le nombre de nœuds fils max et L le nombre de nœuds fils min. Arbre-B (L, U).
- n = nombre de clés contenues dans le nœud x
- n clefs notées c[1], …, c[n]
- Un booléen indiquant si x est une feuille ou non
- n+1 fils : f[1], …, f[n+1]. Une feuille ne contient pas de fils
- L'arbre-B vérifie les propriété suivantes :
  - Toutes les feuilles ont la même profondeur, à savoir la hauteur h de l'arbre
  - Si x n'est pas une feuille :
    - pour 2<=i<=n, pour toute clef x du fils[i] : cles[i-1] <= x <=cles[i]
    - Pour toute clef x du fils[1] : x <= cles[1]
    - Pour toute clef x du fils[n+1]: cles[n] <= x
  - Si x n'est pas la racine, n est compris entre L-1 et U-1.

Structure de données : Arbre-B /Btree



arbreB? (root) / bTree?(root) : boolean

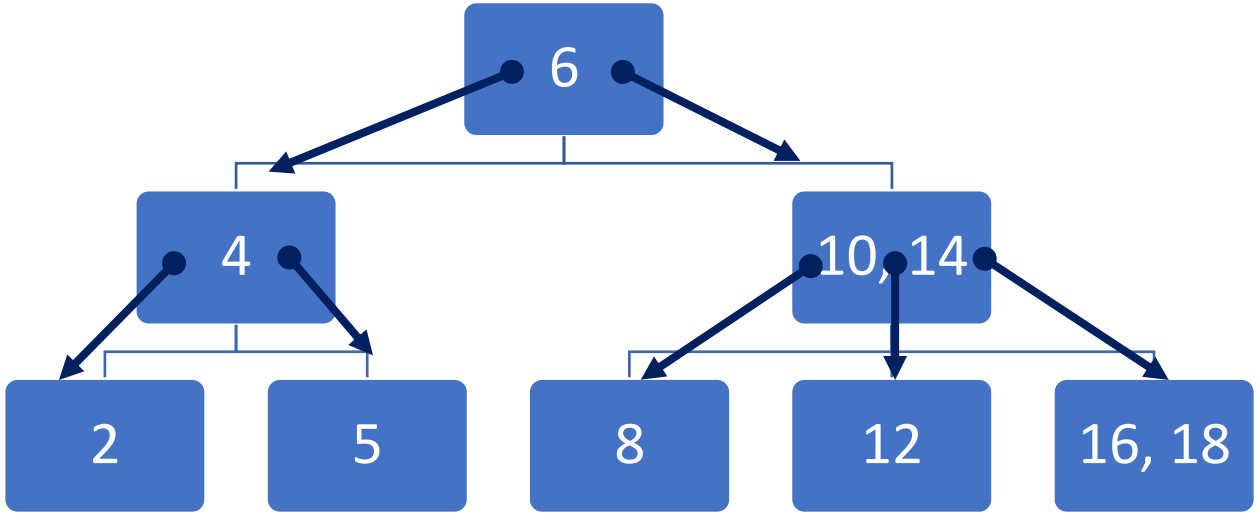
- Exemples :
  - b=bTree(root, L=2, U = 3)
  - b.bTree?(root) -> True



- Tests unitaires / post-conditions
  - && sorted(b.linear-tree(root)) == True
  - && b.balanced(root) == True
  - && b.coverageRate(root) == True

Recherche (clé) / search (key) : boolean

- Exemples :
  - b=Btree(root, L=3, U = 5)
  - B.search(key, root) -> True/False
  - b.search(6, root) -> True
  - b.search(16,root) -> True
  - b.search(13, root) -> False
  - b.search(1,root) -> False
- Tests unitaires / post-conditions
  - key in b.linear-tree(root) == search(key, root)



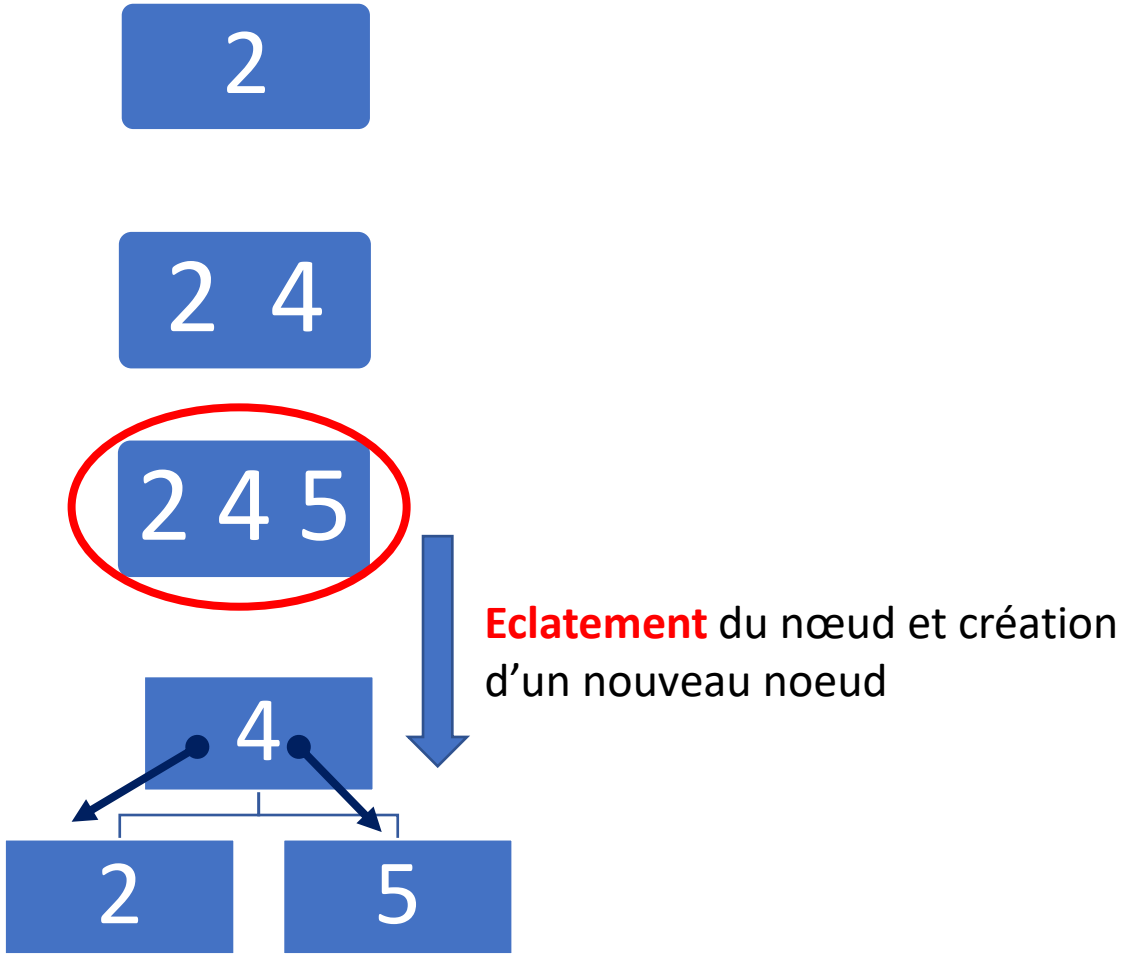
Insertion (clé) / insertion (key)

- Exemples :
  - b=Btree(root, L=2, U = 3)
  - b.insertionL(root, key) -> True/False
  - b.insertionL(root, l=[2, 4, 5, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 7, 9, 11, 13])
  - b.insertion(root, key) -> True/False
  - b.insertion(root, 2) ….
- Tests unitaires / post-conditions
  - b.search(key, root) == True
  - linearListAfterInsertion == linearListBeforeInsertion + key
  - b.bTree?(root)==True

Insertion (clé) / insertion (key)

L = 2 et U = 3
=> Fils : 2 min et 3 max
⇒ clés : 1 min et 2 max
⇒ Eclatement du noeud

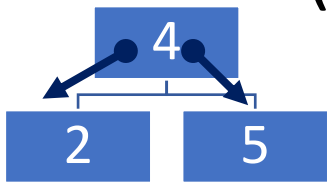
- b.insertion(root, 2)
- b.insertion(root, 4)
- b.insertion(root, 5)

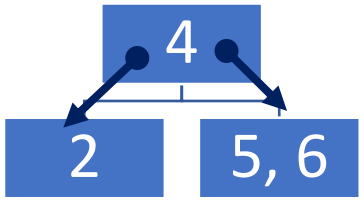


Insertion (clé) / insertion (key)

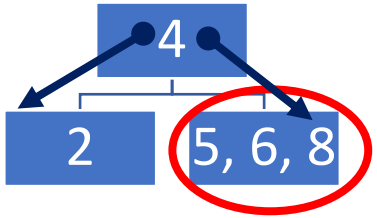
L = 2 et U = 3
=> Fils : 2 min et 3 max
⇒ clés : 1 min et 2 max
⇒ Eclatement du noeud

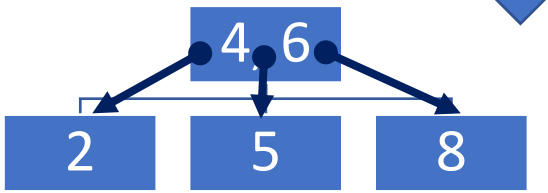
- B.insertion(root, 6)





- b.insertion(root, 8)

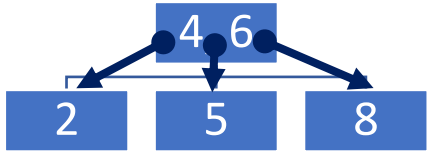




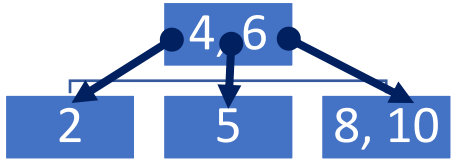
Eclatement du nœud et remonter une clé au parent

Insertion (clé) / insertion (key)

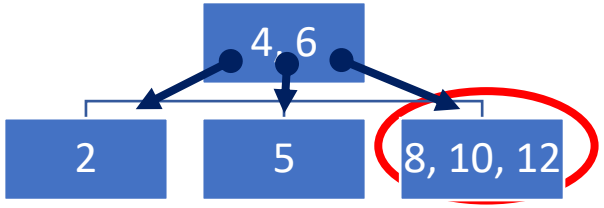
L = 2 et U = 3
=> Fils : 2 min et 3 max
⇒ clés : 1 min et 2 max
⇒ Eclatement du noeud



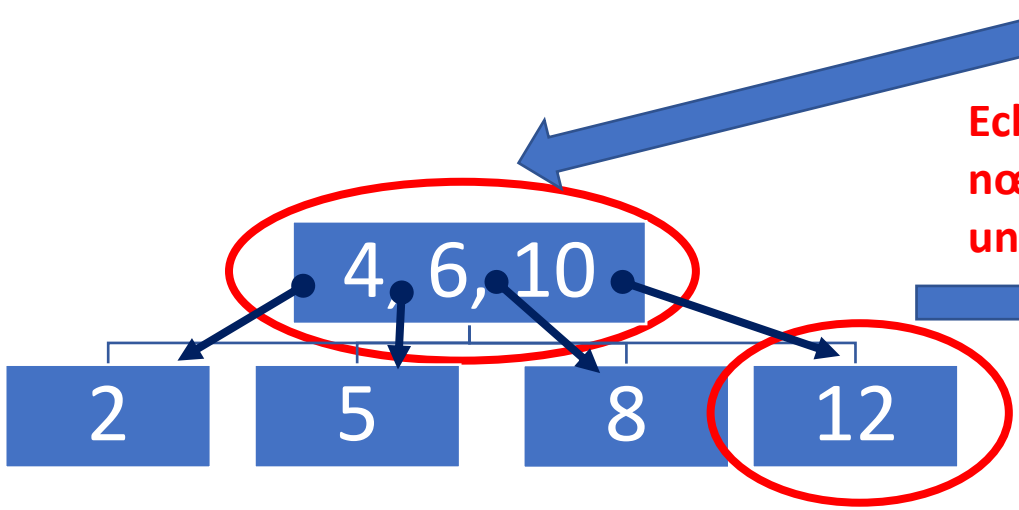
- b.insertion(root, 10)



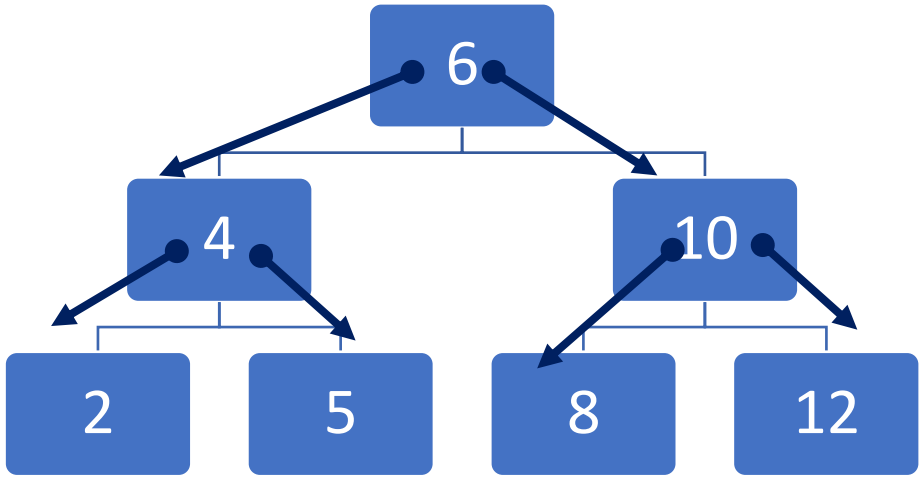
- b.insertion(root, 12)



Eclatement du nœud et remonter une clé au parent

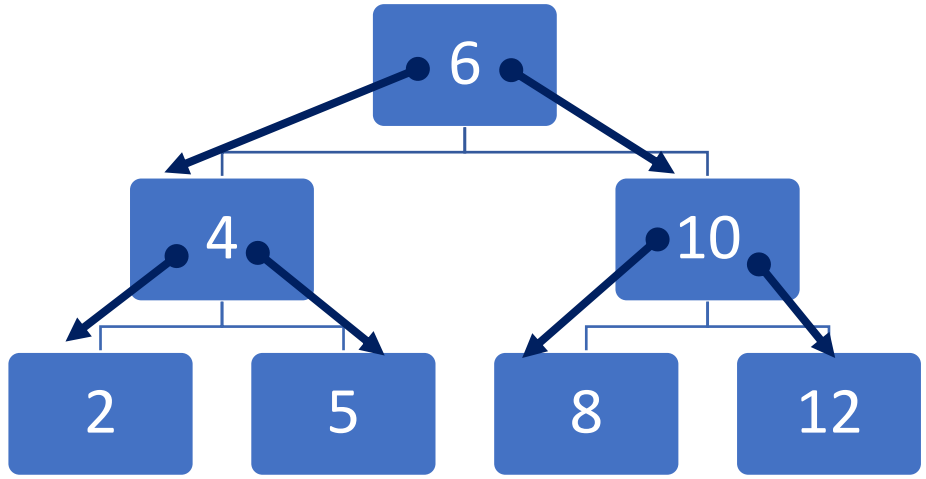


Eclatement du nœud et remonter une clé au parent

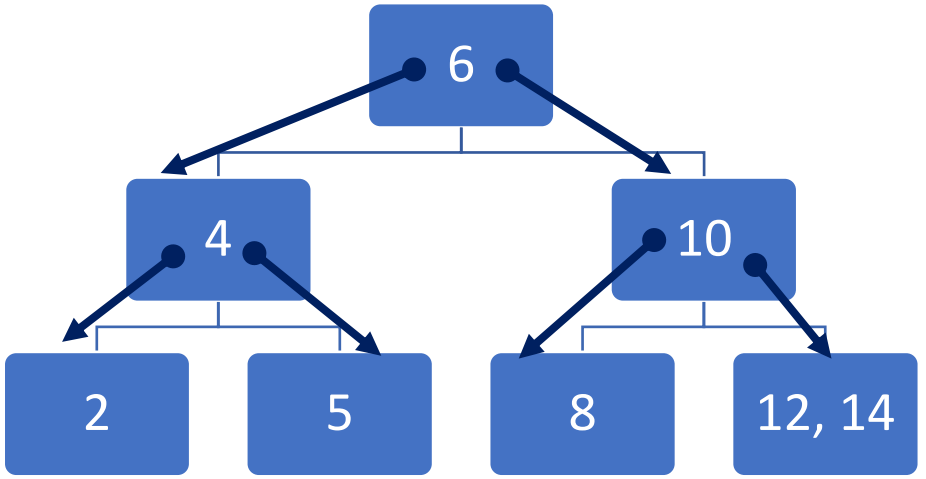


Insertion (clé) / insertion (key)

L = 2 et U = 3
=> Fils : 2 min et 3 max
⇒ clés : 1 min et 2 max
⇒ Eclatement du noeud



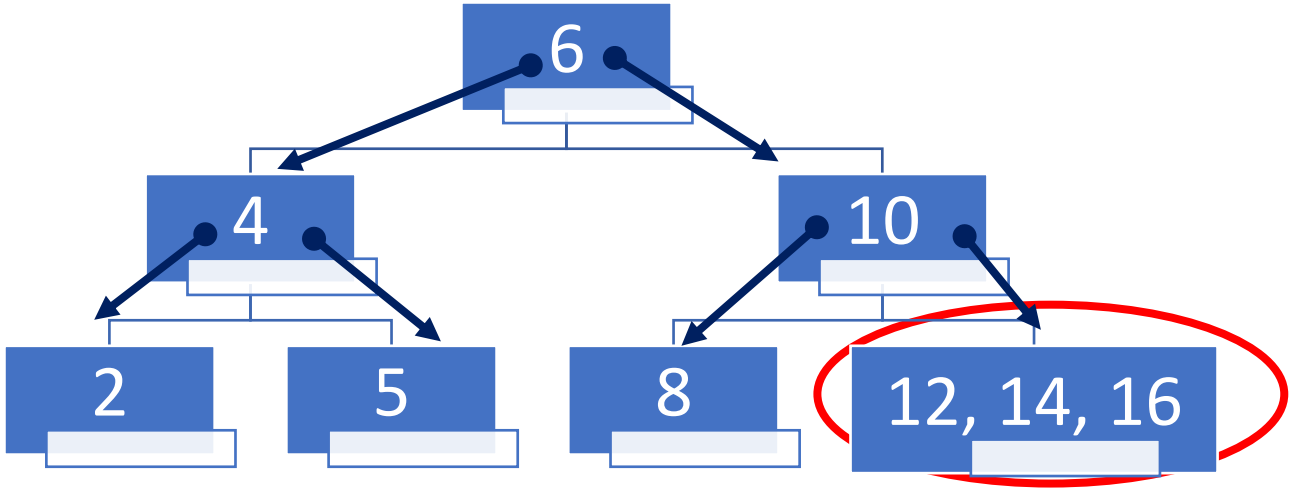
- b.insertion(root, 14)

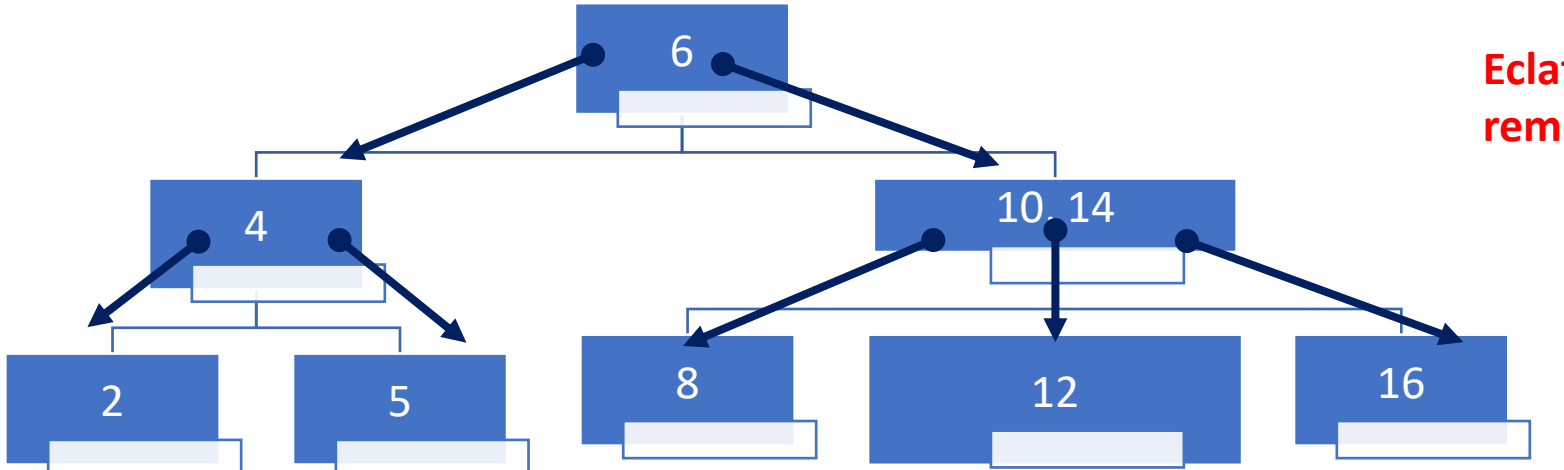


Insertion (clé) / insertion (key)

L = 2 et U = 3
=> Fils : 2 min et 3 max
⇒ clés : 1 min et 2 max
⇒ Eclatement du noeud

- b.insertion(root, 16)



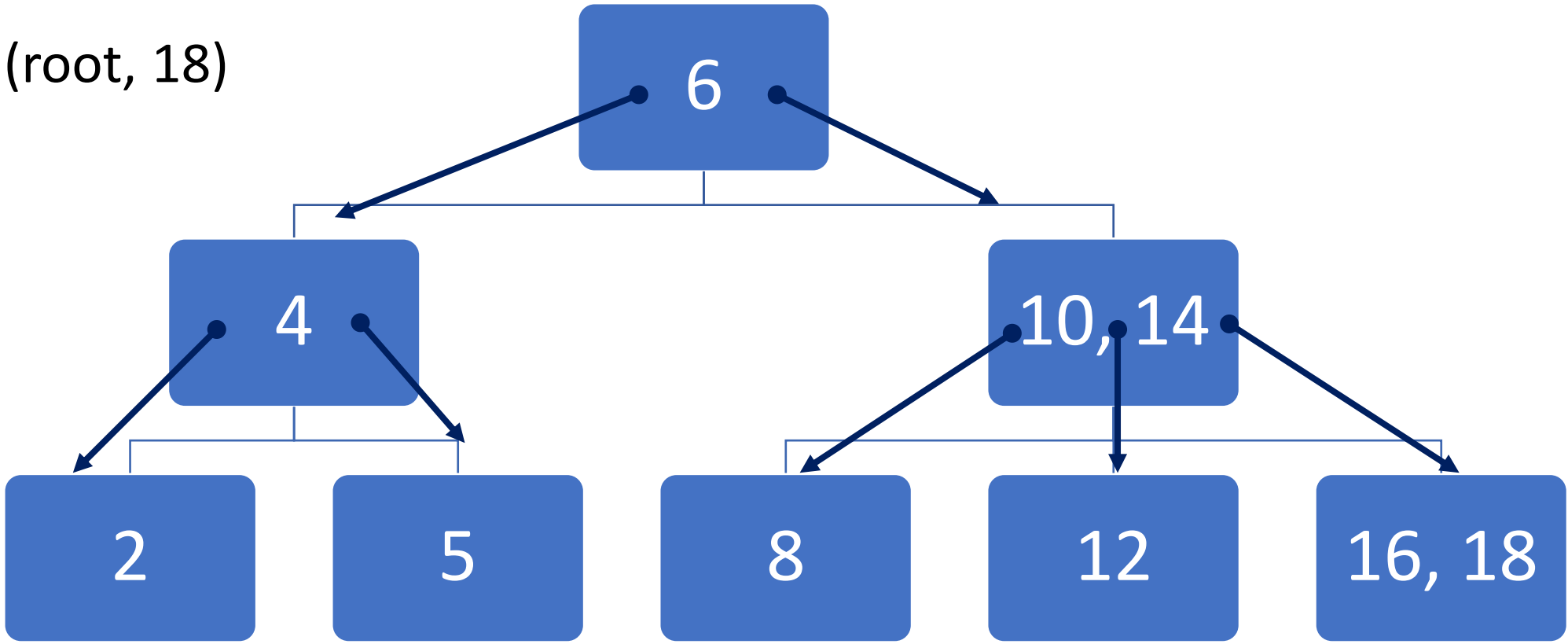


Eclatement du nœud et remonter une clé au parent

Insertion (clé) / insertion (key)

L = 2 et U = 3
=> Fils : 2 min et 3 max
⇒ clés : 1 min et 2 max
⇒ Eclatement du noeud

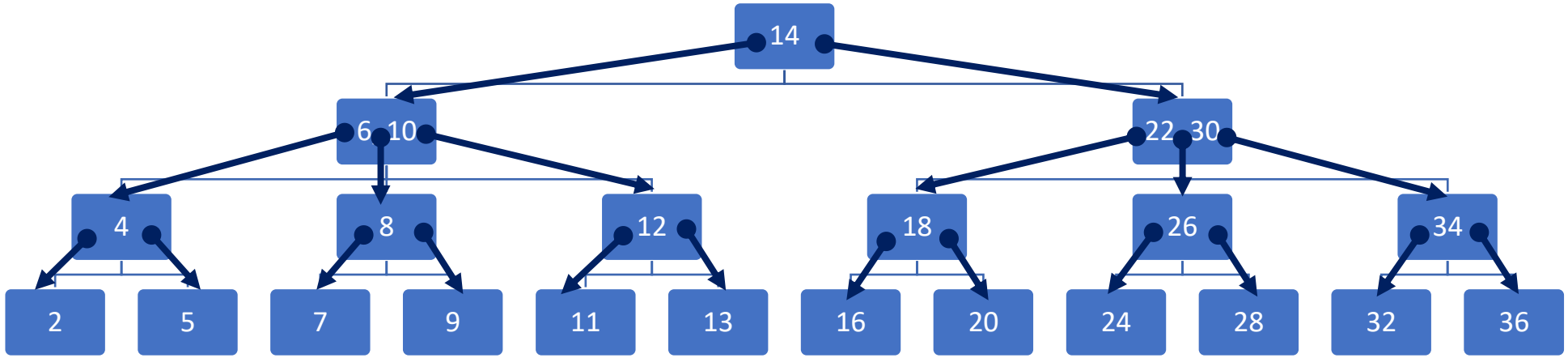
- b.insertion(root, 18)



Insertion (clé) / insertion (key)

L = 2 et U = 3
=> Fils : 2 min et 3 max
⇒ clés : 1 min et 2 max
⇒ Eclatement du noeud

- …..
- b.insertionL(root, l=[20, 22, 24, 26, 28, 30, 32, 34, 36, 7, 9, 11, 13]) -> True



- www.youtube.com/watch?v=coRJrcIYbF4

Suppression (clé) / deletion (key)

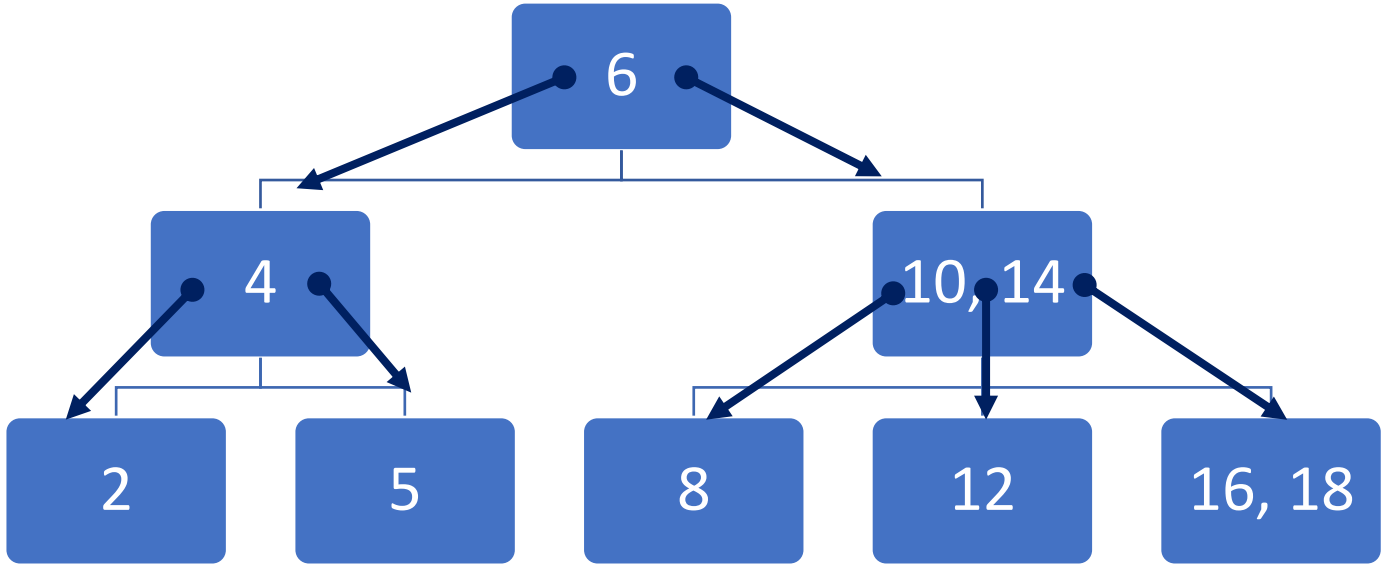
- Exemples :
  - b=Btree(root, L=2, U = 3)
  - b.deletionL(root,l) -> True/False
  - b.deletion(root, key) -> True/False
  - b.deletionL(root, l=[14, 10, 20, 18, 16, 24, 6]) -> True
  - b.deletion(root, 14) …. b.deletion(root, 6)

- Tests unitaires / post-conditions
  - b.search(root, key) == False
  - && linearListAfterDeletion== linearListBeforedeletion - key
  - b.Btree?(root)==True

deletion (clé) / deletion (key)

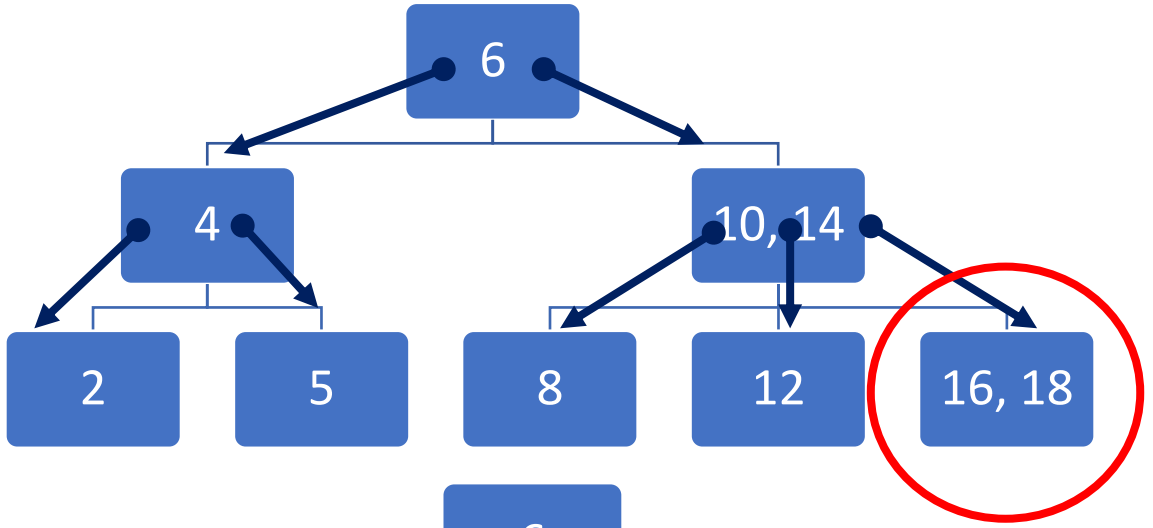
L = 2 et U = 3
=> Fils : 2 min et 3 max
⇒ clés : 1 min et 2 max
⇒ Eclatement du noeud

- b.deletionL(root, , l=[18, 10, 20, 18, 16, 24, 6])

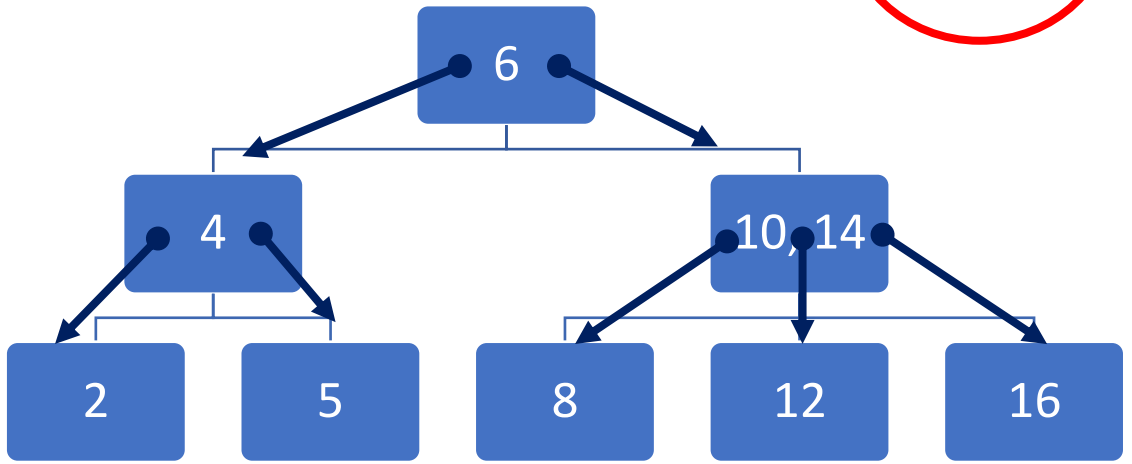


deletion (clé) / deletion (key)

L = 2 et U = 3
=> Fils : 2 min et 3 max
⇒ clés : 1 min et 2 max
⇒ Eclatement du noeud

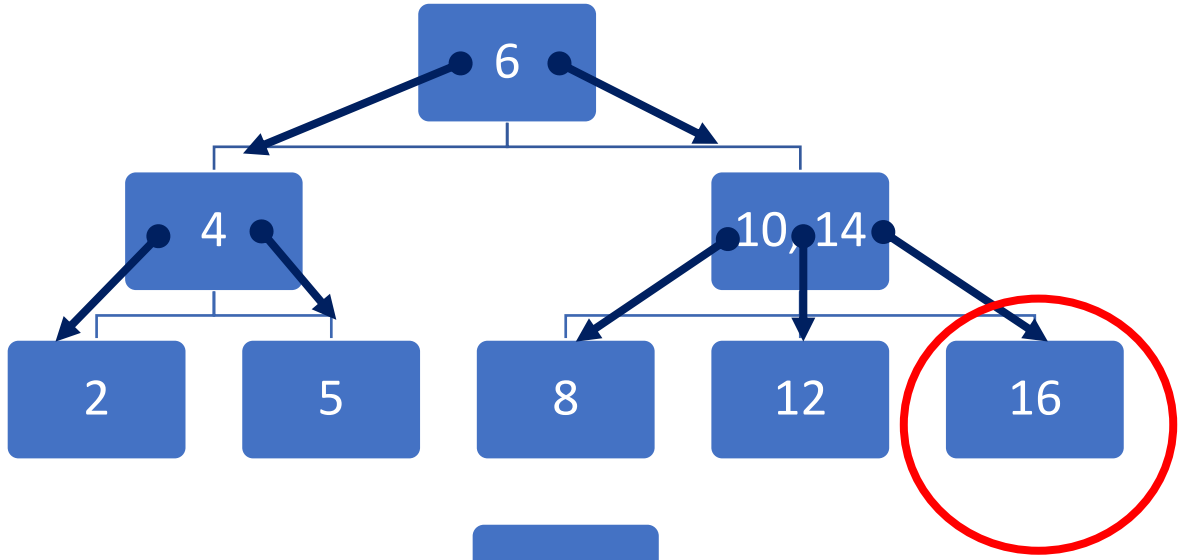


- b.deletion(root, 18)

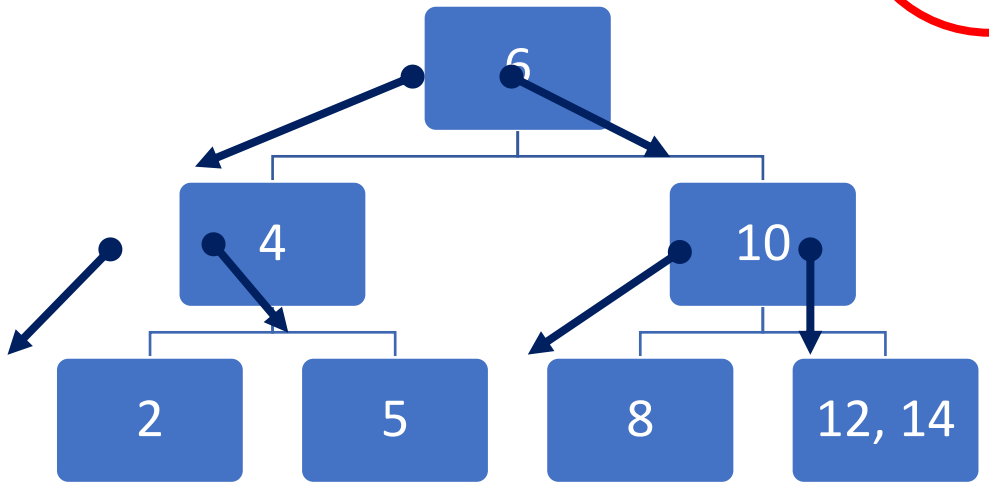


deletion (clé) / deletion (key)

L = 2 et U = 3
=> Fils : 2 min et 3 max
⇒ clés : 1 min et 2 max
⇒ Eclatement du noeud



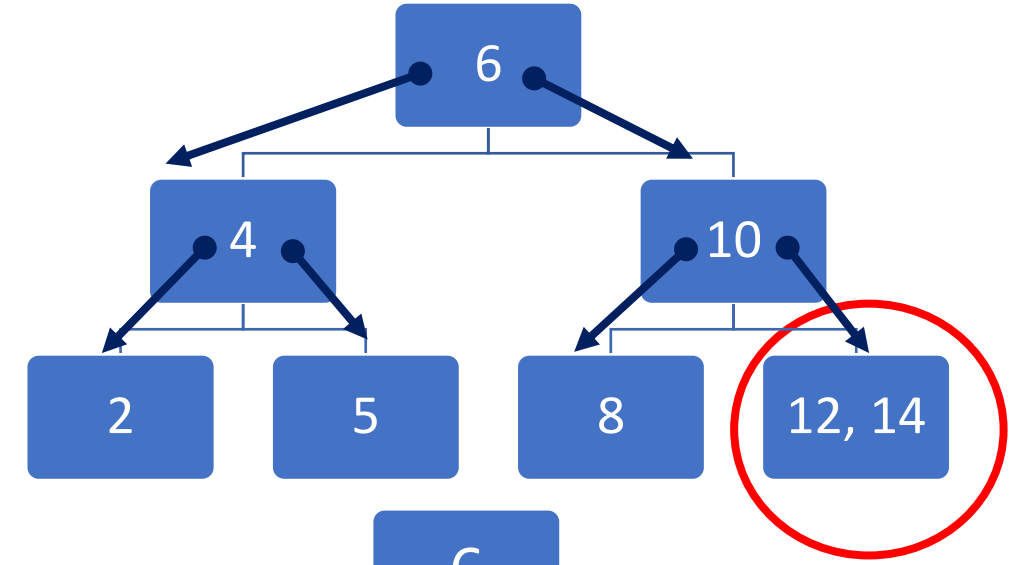
- b.deletion(root, 16)

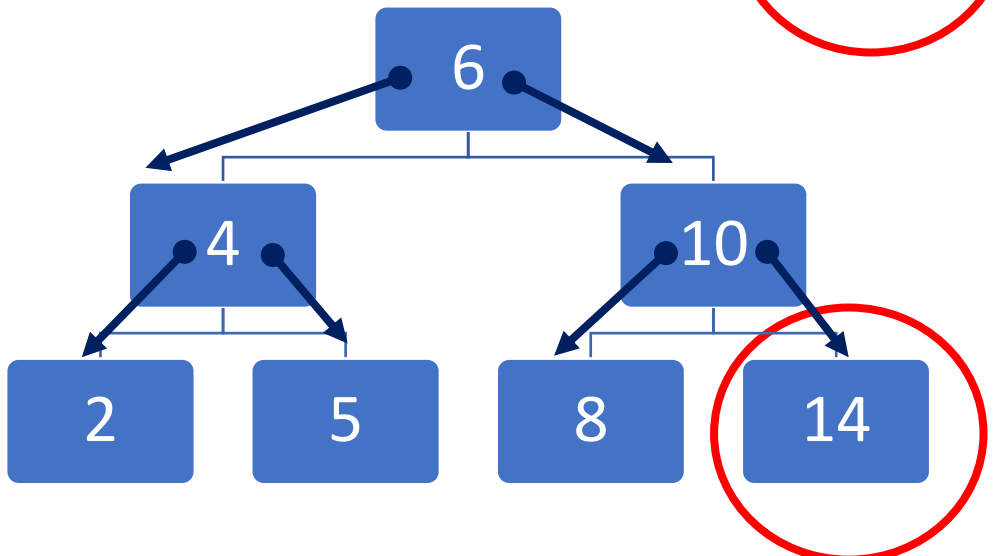


deletion (clé) / deletion (key)

L = 2 et U = 3
=> Fils : 2 min et 3 max
⇒ clés : 1 min et 2 max
⇒ Eclatement du noeud

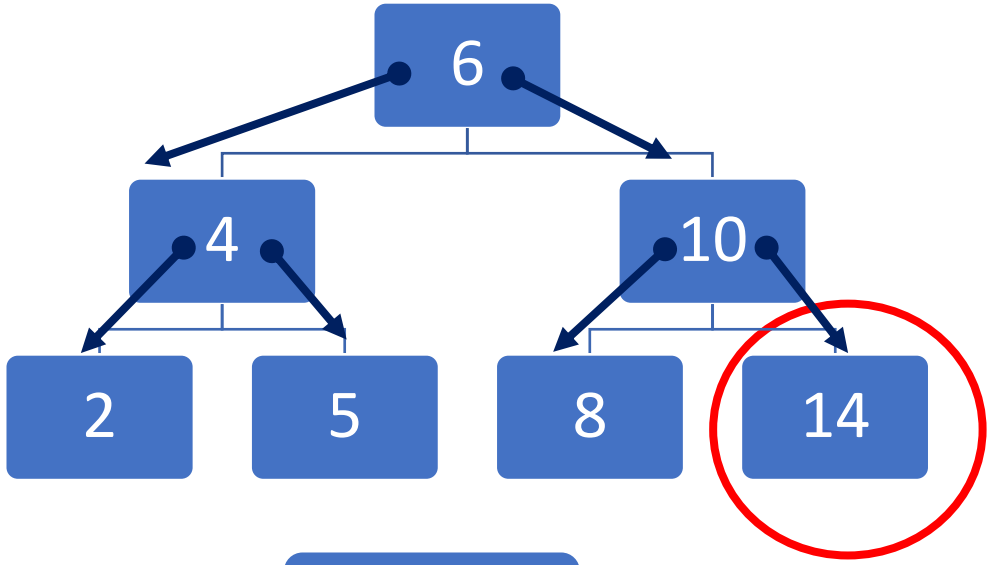
- b.deletion(root, 12)



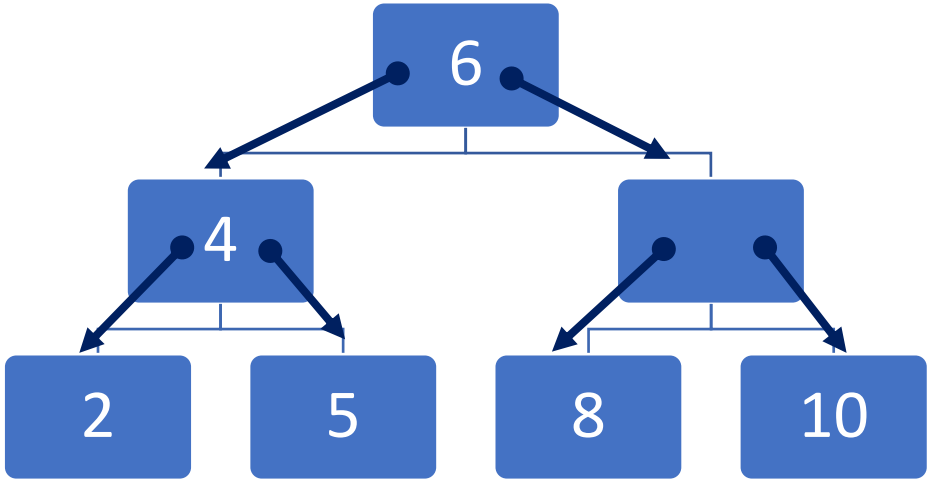


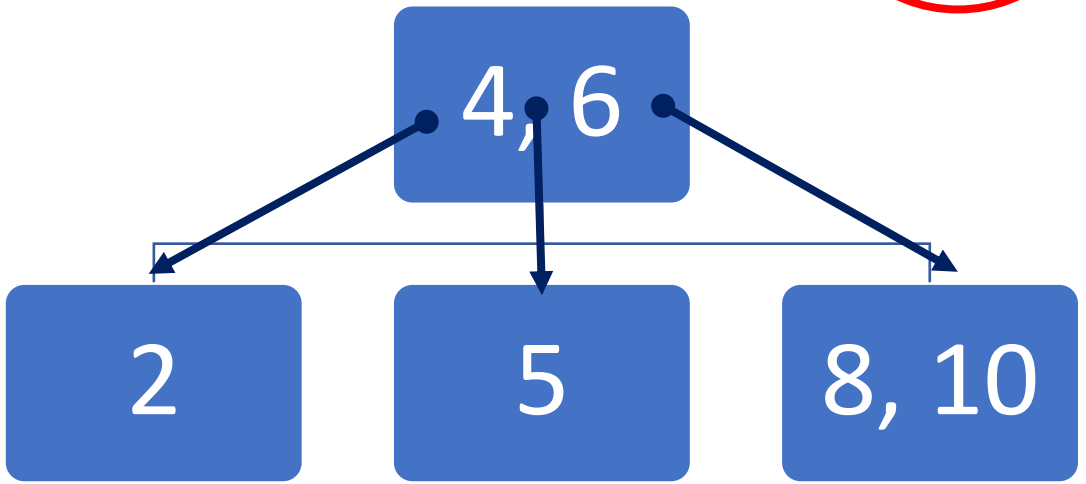
deletion (clé) / deletion (key)

L = 2 et U = 3
=> Fils : 2 min et 3 max
⇒ clés : 1 min et 2 max
⇒ Eclatement du noeud



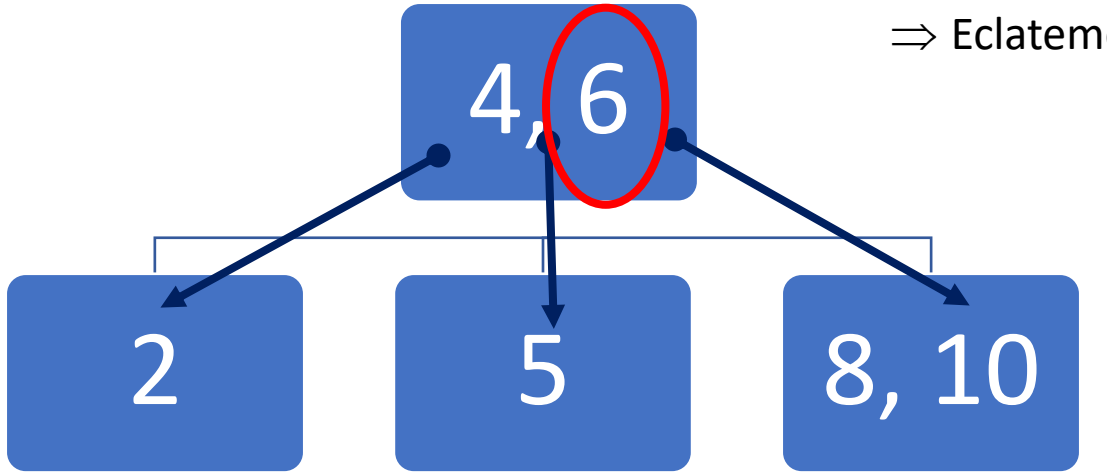
- b.deletion(root, 14)



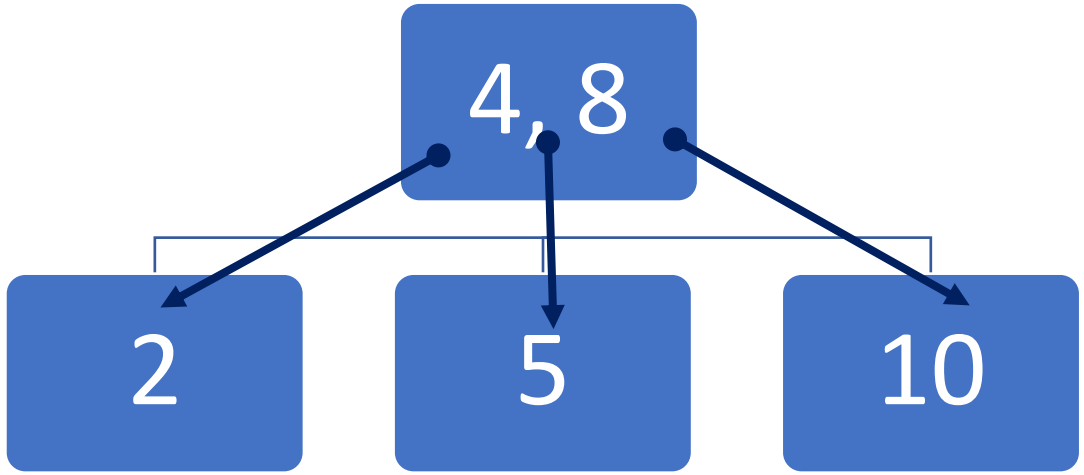


deletion (clé) / deletion (key)

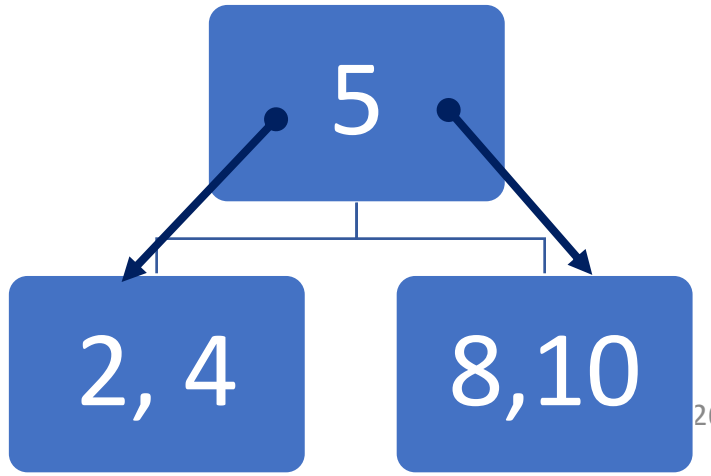
L = 2 et U = 3
=> Fils : 2 min et 3 max
⇒ clés : 1 min et 2 max
⇒ Eclatement du noeud



- b.deletion(root, 6)



OU



Grille d’évaluation

- Livrable = Démonstration + Documentation, à la fin de chaque séance aux cours des deux dernières séances du projet.
- La démonstration est réalisée sur le poste de travail, sur une durée de 30 minutes environ, sur les deux batteries de tests présentées.
- La documentation ne doit pas dépasser 10 pages. Elle doit décrire : l’historique de l’arbre B et les algorithmes (codes Python) commentés : l’algorithme de recherche, l’algorithme d’insertion, l’algorithme de suppression. Chaque algorithme doit comporter la documentation (docstring) : entrée, sortie, exemples, estimation de la complexité temporelle des différents algorithmes.

Test - 1

- (L,U)=(2,3). Le noeud contient 2 clés au maximum et 1 clé au minimum.
- Insertion dans l'ordre : 2, 4, 5, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 7, 9, 11, 13.
- Suppression dans l'ordre : 14, 10, 20, 18, 16, 24, 6
- La vidéo ci-dessous visualise l'état de l'arbre à chaque insertions ou suppression.
- https://www.youtube.com/watch?v=coRJrcIYbF4

Test - 2

- Suite à cette première batterie de tests, il convient de réaliser d'autres tests.
- (L, U) = (6, 11). Le noeud contient 10 clé au maximum et 5 clés au minimum.
- Soit la liste des valeurs suivantes : 10, 20, 30, … 5000, 5, 15, 25, 35, ..4995.
- Suppression dans l'ordre : 10, 20, 30, … 5000, 5, 15, 25, 35, ..4995

- Généralisez les tests en considérant N = (2, 10, 100, 1000, 10000).

Règles de programmation

- Les travail réalisé doit être téléversé sur Gitlab, à la fin de chaque séance.
- Le projet peut être réalisé par une personne ou un binôme.
- Il est possible pour l'étudiant de choisir un projet différent. Le sujet en question doit être proposé et validé par l'enseignant responsable du groupe et l'enseignant responsable de l'UE, avant le 30 janvier 2022.
- Avant de se lancer dans la réalisation et le développement des algorithmes du projet, il est nécessaire de prendre le temps de le comprendre et de le dérouler sur des données tests. Ensuite, organiser développer l'architecture en technologie objet du projet (classes et liens entre les classes).