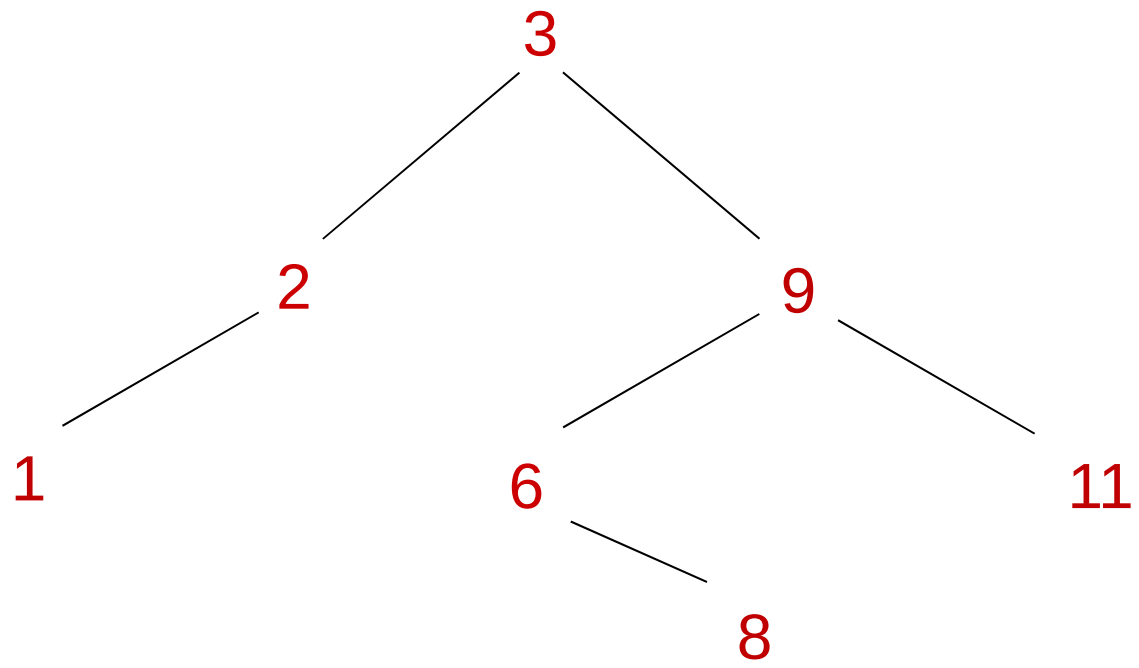
Cours 5 : Les arbres AVL

Arbres binaires équilibrés

Rappel sur les arbres binaires de recherche

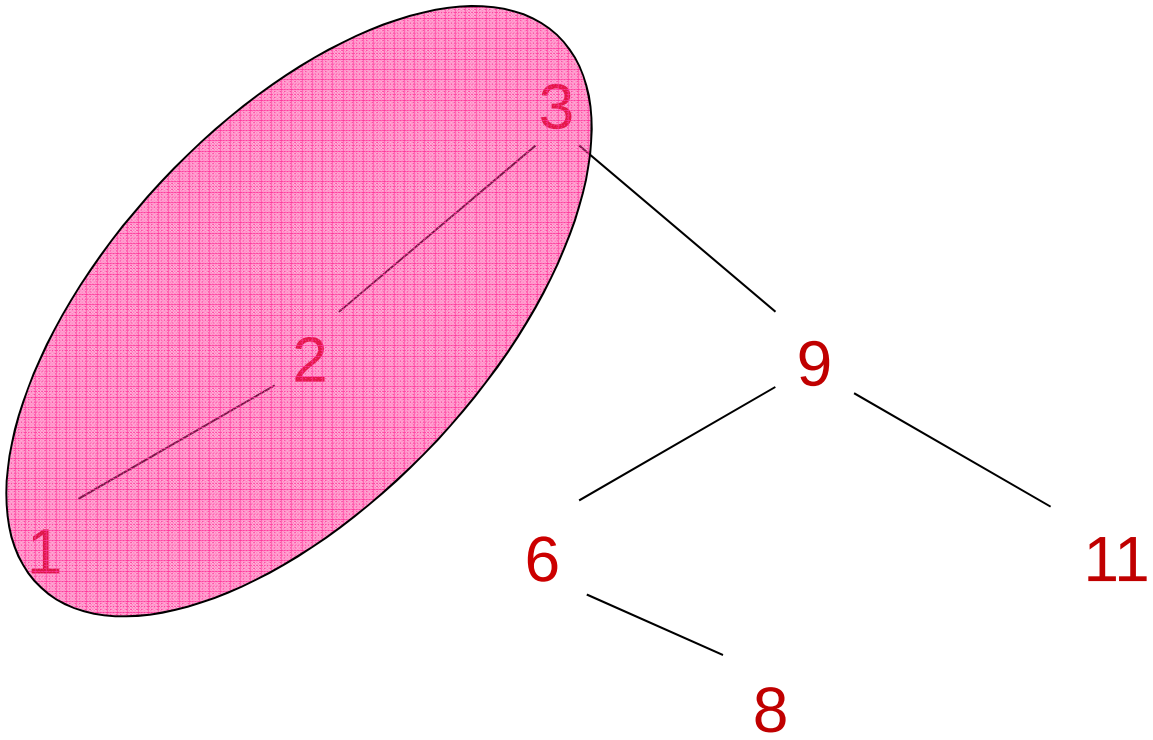
- « Ordre » sur les nœuds :
  - Les plus petits à gauche
  - Les plus grands à droite
- Pour accélérer les recherches d'un élément dans l'arbre
- Objectif : dichotomie (on ne parcourt que la moitié de l'arbre)

Exemple



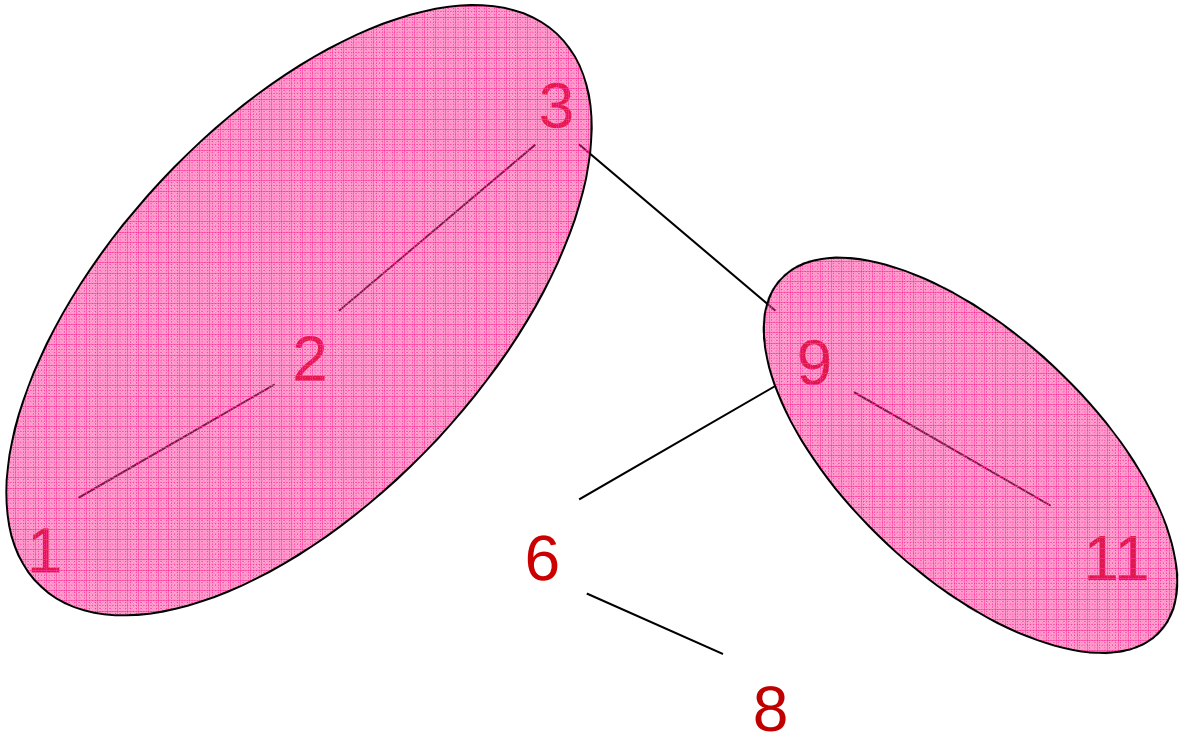
- Si on cherche 7
  - 3 < 7 : pas besoin de chercher dans le sous-arbre gauche

Exemple (2)



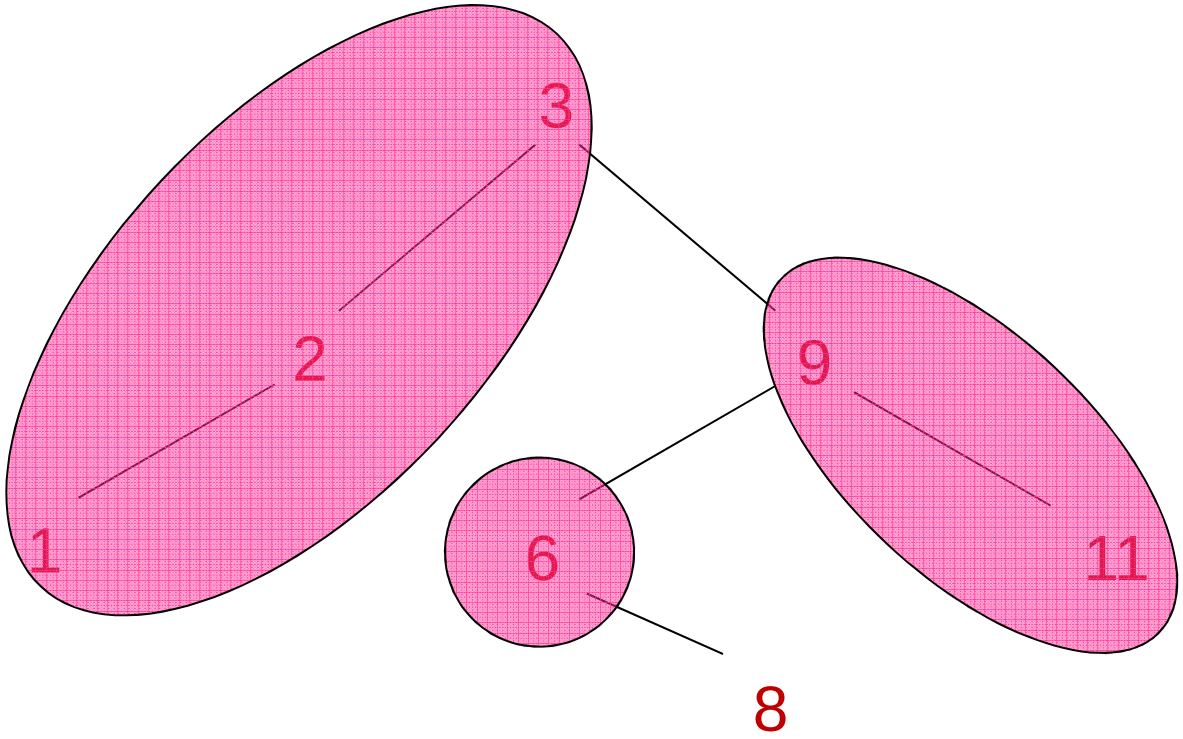
- 7 < 9 : pas besoin de chercher dans le sous-arbre droit

Exemple (3)



- 6 < 7 : on cherche a droit

Exemple (4)

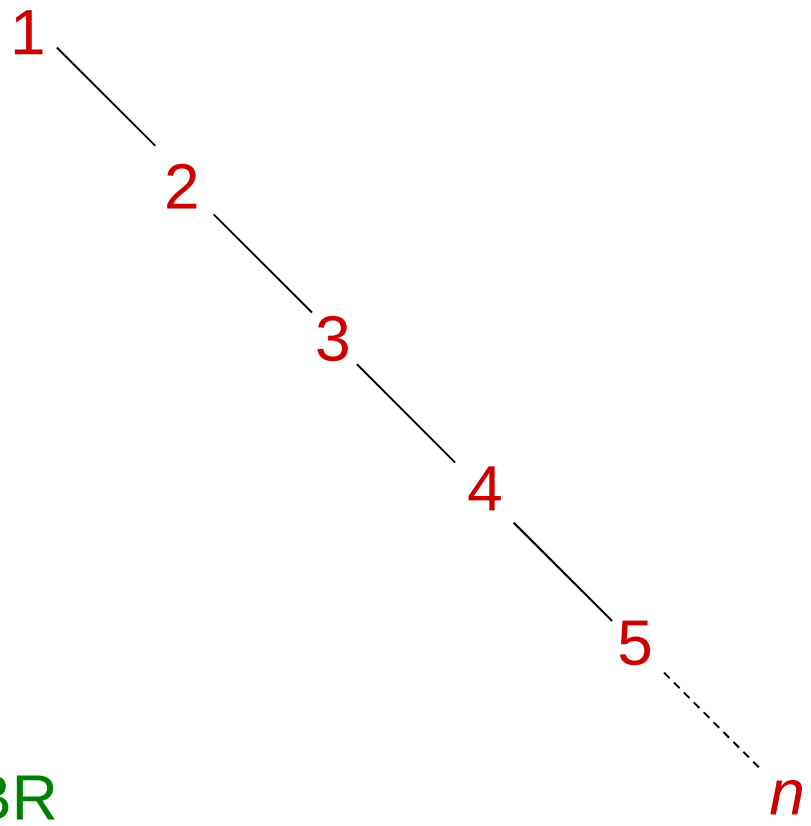


- 7 < 8 et le nœud 8 n’a pas de sous-arbres donc la recherche a échoué

Rappel sur les ABR (suite)

- En général, la recherche dans un ABR coute $\Theta(h)$, où h est l'hauteur de l'arbre
- La hauteur minimale pour un arbre binaire avec n nœuds est $\log(n)$
- Donc dans le meilleur de cas, le cout de la recherche dans un ABR est $\log(n)$ où n est le nombre des nœuds de l'arbre

Problème



Est un ABR

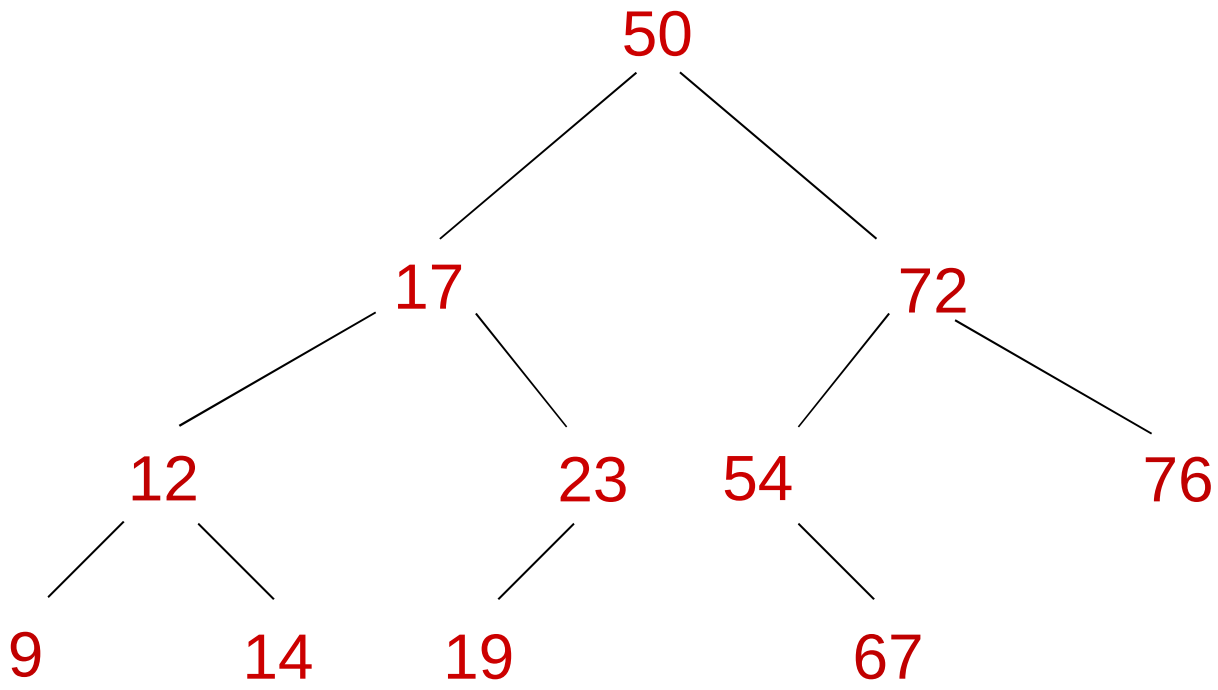
Identification du problème

- On ne gagne rien au niveau de la recherche
  - On est obligé de chercher dans le s.-a. droit
  - Recherche en $\Theta(n)$ forcément
- Solution
  - Obliger l'arbre à être relativement symétrique
  - Hauteur du s.-a. gauche proche de la hauteur du s.-a. droit

Arbres AVL

- Arbres de recherche équilibrés
- Principe :
  - Pour *chaque nœud*, les hauteurs du s.-a. gauche et du s.-a. droit différent au plus de 1
- Modèle proposé par G.M. Adelson-Velsky et E.M. Landis (d'où son nom)
- Notion de *facteur d'équilibrage* d'un nœud
  - Différence entre les hauteurs des sag et sad
  - Un arbre est AVL si tous les nœuds ont un facteur de -1, 0 ou 1

Exemple



Les changements

- Cette fois, on a systématiquement la moitié, ou près de la moitié de l'arbre de chaque côté de la racine
- Et ceci pour tous les nœuds
- Chaque choix entre s.-a. gauche et s.-a. droit élimine la moitié des nœuds restants
- On a un vrai parcours dichotomique
- Complexité $\Theta(\log(n))$ dans le pire des cas

Implémentation habituelle

- Rajouter un attribut a l'arbre
  - Sa profondeur

    ou
  - Son facteur d'équilibrage
- A mettre a jour à chaque modification (ajout ou suppression)

Problématique de l’ajout

- On ajoute un élément
- L’arbre (ou un de ses sous-arbres) peut devenir déséquilibré
  - Facteur < -1 ou >1
- Principe :
  - On fait l’ajout normalement
  - On remonte en mettant les profondeurs ou facteurs à jour jusqu'à rencontrer un arbre déséquilibré

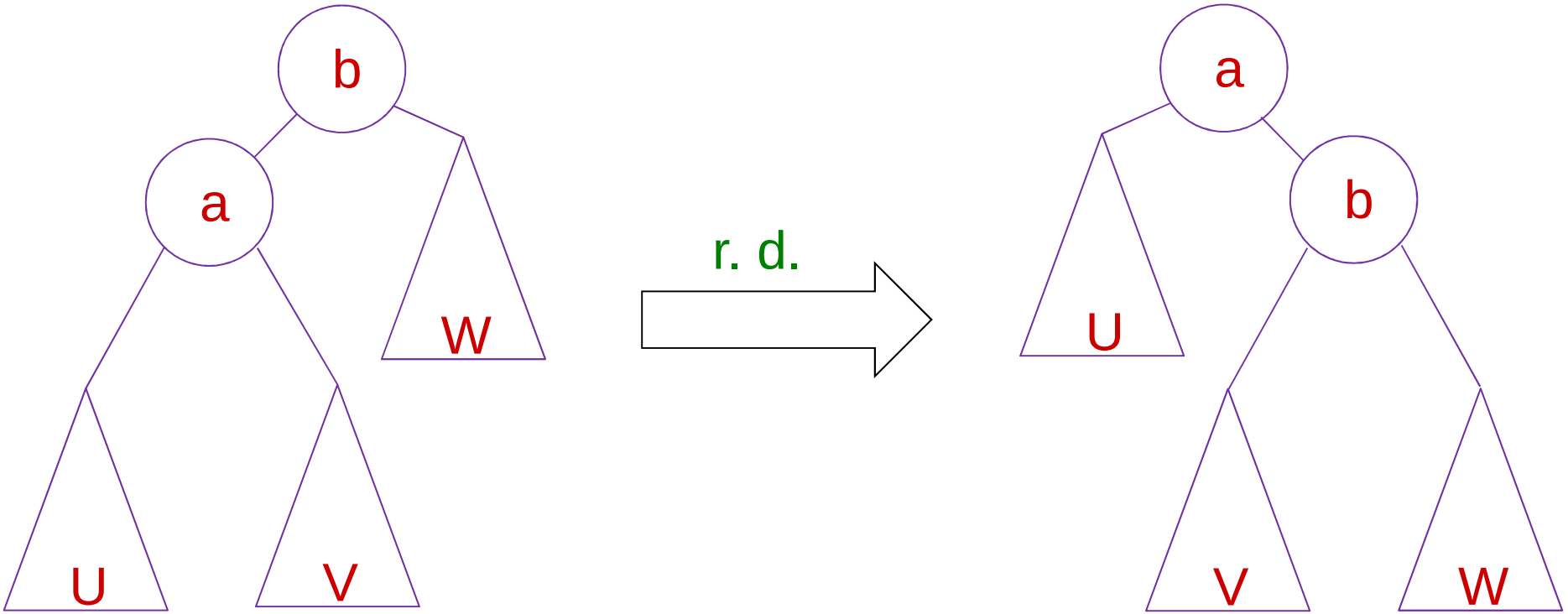
Problématique de l’ajout (2)

- Si on ne rencontre aucun arbre déséquilibré, parfait
- Sinon, on rééquilibre le premier arbre qui devient déséquilibré lors de la remontée
  - Il redevient équilibré
  - Avec la même profondeur qu’avant l’ajout
  - Donc inutile de remonter davantage

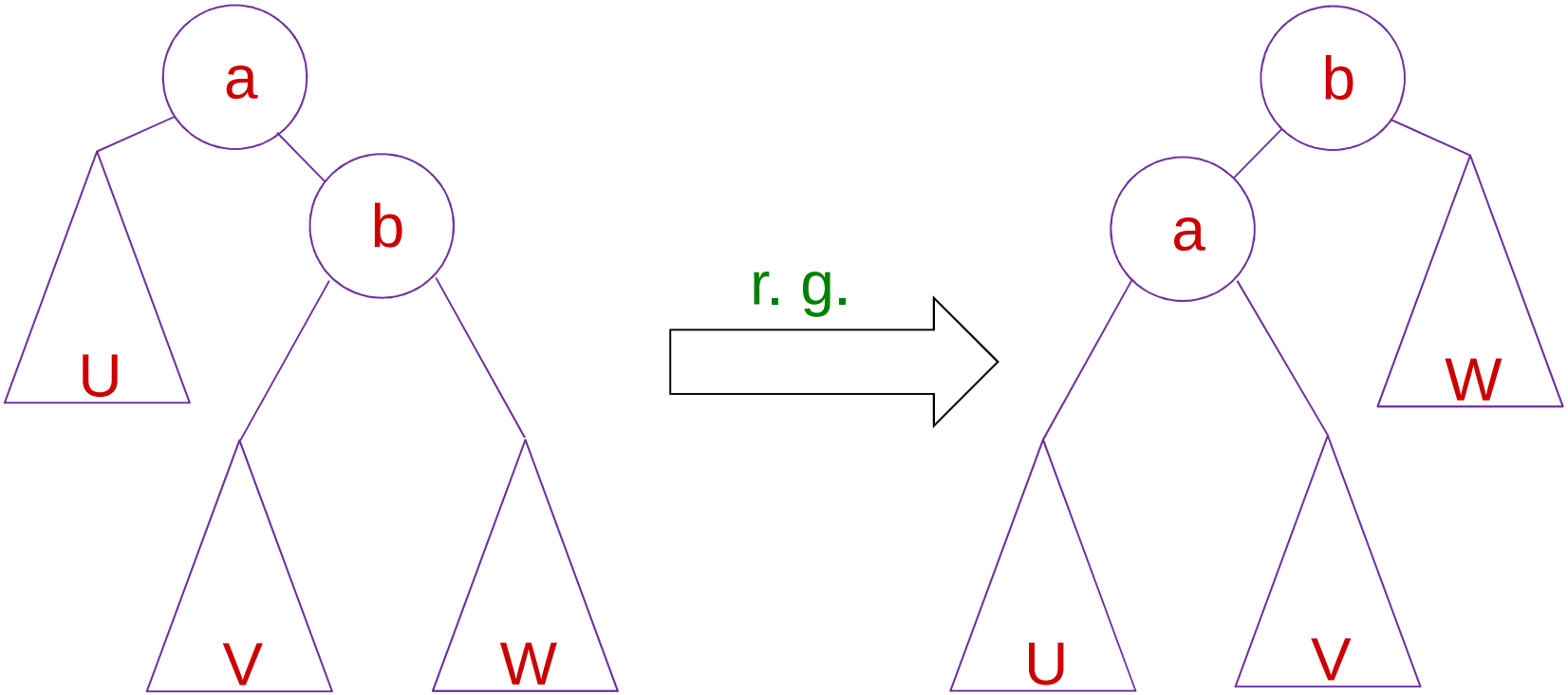
Le rééquilibrage

- Un principe : la rotation
- En fait, selon le facteur de déséquilibrage de l'arbre et celui de ses sous-arbres, on va devoir faire une ou deux rotations (je ne detaille pas)
- Deux actions
  - Rotation à gauche
  - Rotation à droite

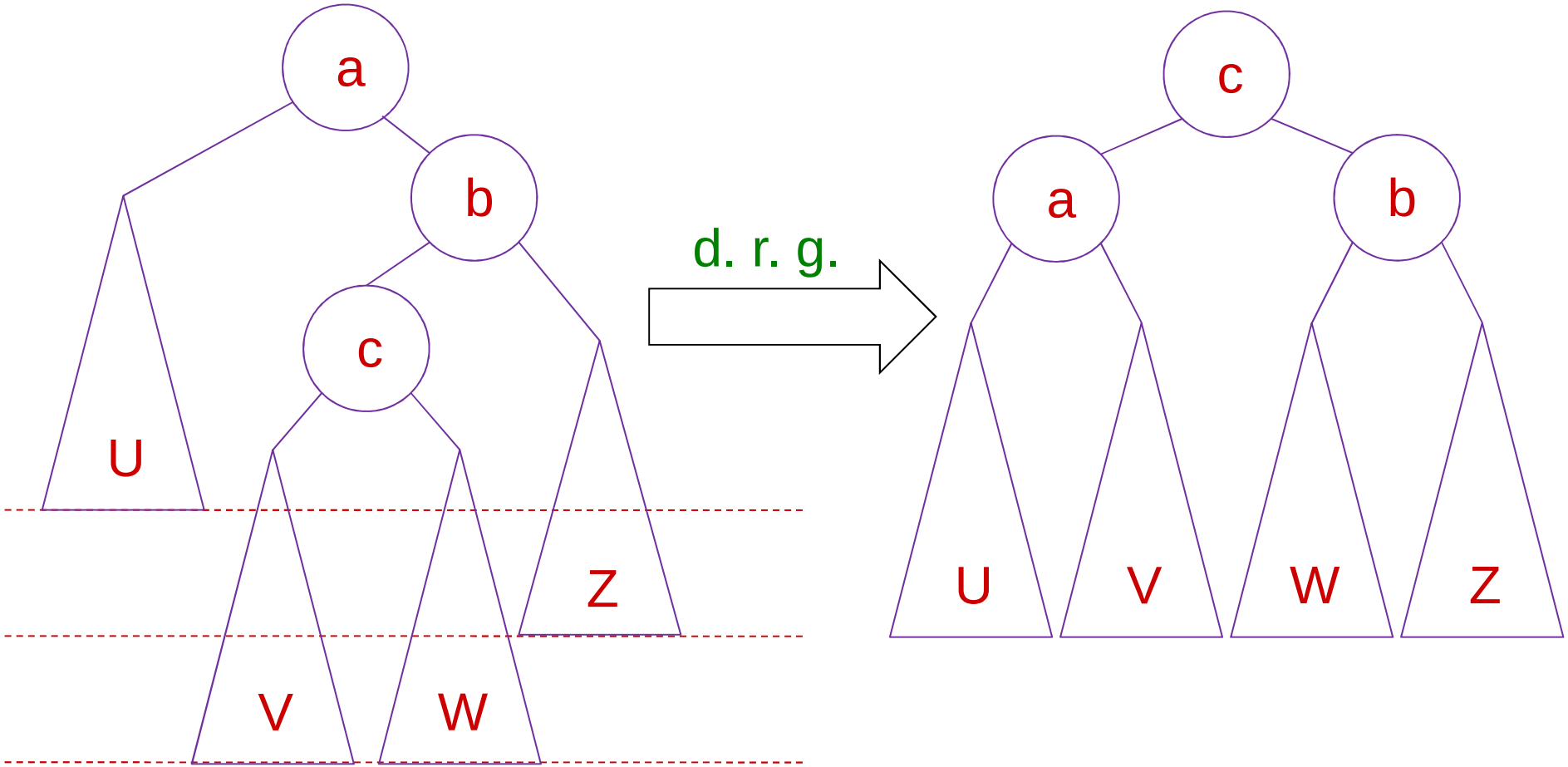
La rotation à droite



La rotation à gauche



La double rotation à gauche



La compression de données

Principes
Exemples
Codage de Huffmann

La compression de données

- La compression de données traite de la manière dont on peut réduire l'espace nécessaire à la représentation d'une certaine quantité d'information
- Deux grandes catégories
  - Sans perte
  - Avec perte
- Manipulée dans tous les secteurs
  - Données quelconques : zip, rar, gz
  - Graphiques : gif, png, jpg
  - Audio : wav, mp3, ogg
  - Video : mpeg2, mpeg4, xvid, mkv

Compression sans perte

- Codage de Huffmann
  - Plus un symbole apparaît, plus son codage est court
- Codage RLE (Run-Length Encoding)
  - *abab* remplacé par 2*ab*
- Codage LZW (Lempel-Ziv-Welch)(→GIF)
  - De type « dictionnaire »
  - Des successions de caractères se retrouvent plus souvent que d'autres
    - On peut donc les remplacer par un nouveau caractère

Compression avec pertes

- Idée : seule une partie des données est utile
- On ne garde que celles-là
- Trois grands types
  - Transformée de Fourier (DCT) : jpeg
  - Compression par ondelettes
  - Compression fractale

Le codage de Huffmann

- Codage de type statistique, analogue au morse
- Idée :
  - Plus un symbole (ici un caractère) revient souvent, plus son code sera court
- On commence par lire le texte et compter le nombre d'occurences de chaque caractère
- Le reste est à base d'arbres binaires

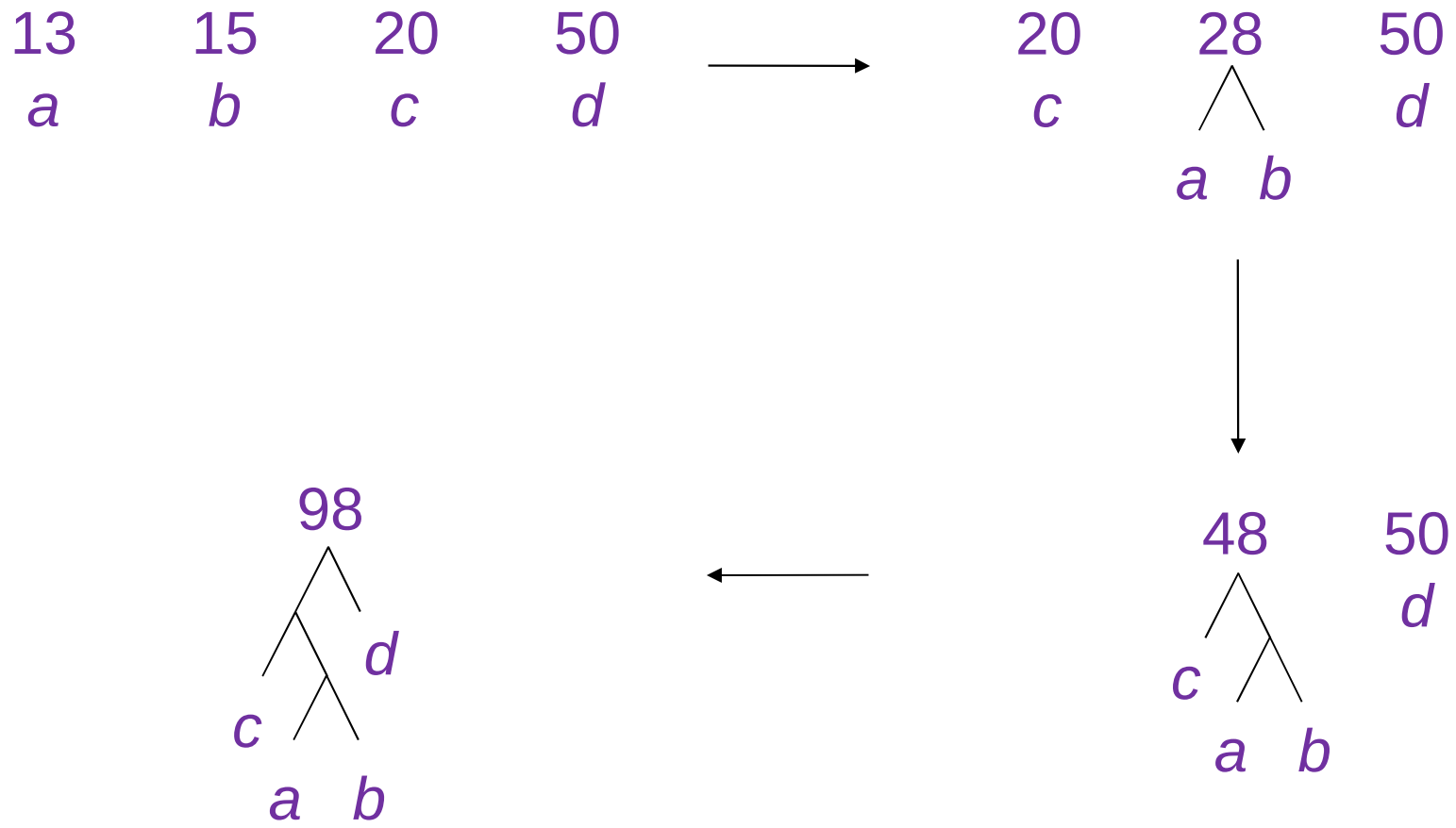
Construction de l’arbre de codage

- Au départ
  - Chaque caractère est racine d’un arbre dont la valeur associée est le nombre d’occurences du caractère
  - On dispose d'un tableau d'arbres (ou autre structure)
    - De préférence trié par valeur

Construction de l'arbre de codage (2)

- On va systématiquement « fusionner » les deux arbres de coût minimal
  - On obtient un arbre dont les s.-a. sont les deux arbres de départ
  - La plus petite valeur à gauche
  - La valeur de l'arbre fusionné est la somme des valeurs des deux arbres de départ
- On supprime les deux arbres du tableau et on ajoute l'arbre fusionné

Exemple



Construction de l’arbre de codage (3)

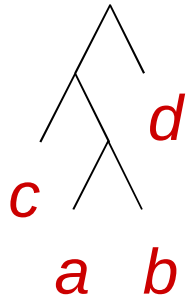
- On continue jusqu’à ce qu’il n’y ait plus qu’un seul arbre dans le tableau
- On a construit l’arbre en « montant »
- On va le « redescendre » pour construire les codes

Construction des codes

- Code = mot binaire (des 0 et des 1)
- Principe :
  - Quand on descend à droite, on ajoute 1 en fin de mot
  - Quand on descend à gauche, on ajoute 0 en fin de mot
  - (Parfaitement arbitraire, on pourrait faire l'inverse)

Reprenons l'exemple

13	15	20	50
a	*b*	*c*	*d*



Le code de *d* est 1
Le code de *c* est 00
Le code de *a* est 010
Le code de *b* est 011

Astuce du code

- Le préfixe d'un code ne peut pas être un code
  - Si 1001 est un code, 1,10 et 100 ne peuvent pas être des codes
- Conséquence :
  - Pas de confusion possible
  - Code parfaitement déterministe

Le fichier codé

- Représentation de l'arbre
  - Il faut choisir une façon simple d'enregistrer l'arbre
- Le texte codé
  - Chaque lettre est remplacée par son code

Décodage

- On recrée l’arbre
- On part de la racine :
  - Si 0, on descend à gauche
  - Si 1, on descend à droite
  - Si on tombe sur une feuille
    - Étiquetée par un caractère
    - On écrit le caractère et on repart de la racine