

Arbres quaternaires

Algorithmes et structures de données, 2025-2026

P. Albuquerque (B410) et O. Malaspinas (A401), ISC, HEPIA
2026-04-15

En partie inspiré des supports de cours de P. Albuquerque

Rappel sur les arbres quaternaires

Définition ?

Rappel sur les arbres quaternaires

Définition ?

- Arbre dont chaque nœud a 4 enfants ou aucun.

Utilisation dans ce cours ?

Rappel sur les arbres quaternaires

Définition ?

- Arbre dont chaque nœud a 4 enfants ou aucun.

Utilisation dans ce cours ?

- Stockage/compression d'image
- Chaque pixel correspond à une feuille
- Des portions de l'image peuvent être compressées sans/avec perte

Pseudo-code ?

Structure de données

Pseudo-code ?

```
struct node
    info
    node sup_gauche, sup_droit,
        inf_gauche, inf_droit
```

En C ?

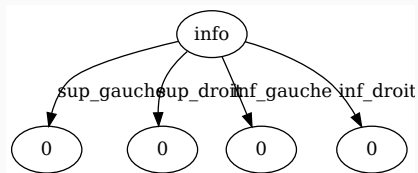


Figure 1 – Un nœud d'arbre quaternaire.

Structure de données

Pseudo-code ?

```
struct node
    info
    node sup_gauche, sup_droit,
        inf_gauche, inf_droit
```

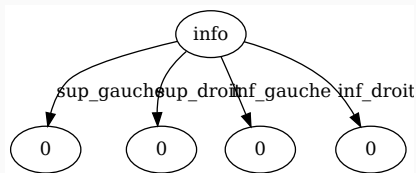


Figure 1 – Un nœud d'arbre quaternaire.

En C ?

```
struct _node {
    int info;
    struct _node *sup_left;
    struct _node *sup_right;
    struct _node *inf_left;
    struct _node *inf_right;
};
```

- Pourquoi le * est important ?

Structure de données

Pseudo-code ?

```
struct node
    info
    node sup_gauche, sup_droit,
        inf_gauche, inf_droit
```

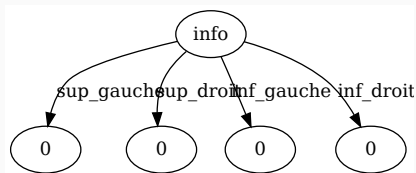


Figure 1 – Un nœud d'arbre quaternaire.

En C ?

```
struct _node {
    int info;
    struct _node *sup_left;
    struct _node *sup_right;
    struct _node *inf_left;
    struct _node *inf_right;
};
```

- Pourquoi le * est important ?
- Type récursif => taille inconnue à la compilation.

Une fonctionnalité simple

La fonction `est_feuille(noeud)`

- Problème avec cette implémentation ?

```
bool est_feuille(noeud)
    retourne
        est_vide(sup_gauche(noeud)) &&
        est_vide(sup_droit(noeud)) &&
        est_vide(inf_gauche(noeud)) &&
        est_vide(inf_droit(noeud))
```

Une fonctionnalité simple

La fonction `est_feuille(noeud)`

- Problème avec cette implémentation ?

```
bool est_feuille(noeud)
    retourne
        est_vide(sup_gauche(noeud)) &&
        est_vide(sup_droit(noeud)) &&
        est_vide(inf_gauche(noeud)) &&
        est_vide(inf_droit(noeud))
```

- Inutile d'avoir 4 conditions (soit 4 enfants soit aucun !)
- Facile d'en oublier un !
- Comment changer la structure pour que ça soit moins terrible ?

Une fonctionnalité simple

La fonction `est_feuille(noeud)`

- Problème avec cette implémentation ?

```
bool est_feuille(noeud)
    retourne
        est_vide(sup_gauche(noeud)) &&
        est_vide(sup_droit(noeud)) &&
        est_vide(inf_gauche(noeud)) &&
        est_vide(inf_droit(noeud))
```

- Inutile d'avoir 4 conditions (soit 4 enfants soit aucun !)
- Facile d'en oublier un !
- Comment changer la structure pour que ça soit moins terrible ?

```
struct node
    info
    node enfant[4]
```

En C ?

Structure de données

En C ?

```
typedef struct _node {  
    int info;  
    struct _node *child[4];  
} node;
```

Fonction `is_leaf(node *tree)` ?

Structure de données

En C ?

```
typedef struct _node {  
    int info;  
    struct _node *child[4];  
} node;
```

Fonction `is_leaf(node *tree)` ?

```
bool is_leaf(node *tree) {  
    return (NULL == tree->child[0]); // only first matters  
}
```

Problème à résoudre

- Construire un arbre quaternaire à partir d'une image :
 - Créer l'arbre (allouer la mémoire pour tous les nœuds)
 - Remplir l'arbre avec les valeurs des pixels
- Compression de l'image :
 - Si les pixels sont les mêmes dans le quadrant on supprime le sous-arbre (sans perte)
 - Si les pixels ne dévient pas trop, on supprime le quadrant (avec perte)

Création de l'arbre

Comment créer un arbre de profondeur prof (3min) ?

Création de l'arbre

Comment créer un arbre de profondeur prof (3min) ?

```
arbre creer_arbre(prof)
    n = nouveau_noeud() # alloue la mémoire
    si prof > 0
        pour i = 0 à 3
            n.enfant[i] = creer_arbre(prof-1)
    retourne n
```

En C (3 min, matrix) ?

Création de l'arbre

Comment créer un arbre de profondeur prof (3min) ?

```
arbre creer_arbre(prof)
    n = nouveau_noeud() # alloue la mémoire
    si prof > 0
        pour i = 0 à 3
            n.enfant[i] = creer_arbre(prof-1)
    retourne n
```

En C (3 min, matrix) ?

```
node *qt_create(int depth) {
    node *n = calloc(1, sizeof(node));
    if (depth > 0) {
        for (int i = 0; i < 4; ++i) {
            n->child[i] = qt_create(depth-1);
        }
    }
    return n;
}
```

Le nombre de nœuds ?

Comment implémenter la fonction (pseudo-code, 5min, matrix) ?

Le nombre de nœuds ?

Comment implémenter la fonction (pseudo-code, 5min, matrix) ?

```
entier nombre_nœuds(arbre)
  si est_feuille(arbre)
    retourne 1
  sinon
    somme = 1
    pour i de 0 à 3
      somme += nombre_nœuds(arbre.enfant[i])
    retourne somme
```

Le nombre de nœuds ?

Comment implémenter la fonction en C (3min, matrix) ?

Le nombre de nœuds ?

Comment implémenter la fonction en C (3min, matrix) ?

```
int size(node *qt) {  
    if (is_leaf(qt)) {  
        return 1;  
    } else {  
        int sum = 1;  
        for (int i = 0; i < 4; ++i) {  
            sum += size(qt->child[i]);  
        }  
        return sum;  
    }  
}
```

La profondeur en C ?

Implémentation (5min, matrix)

La profondeur en C ?

Implémentation (5min, matrix)

```
int max(int x, int y) {
    return (x >= y ? x : y);
}

int max_depth(int depths[4]) {
    int m = depths[0];
    for (int i = 1; i < 4; ++i) {
        m = max(m, depths[i]);
    }
    return m;
}

int depth(node *qt) {
    int depths[] = {0, 0, 0, 0};
    if (is_leaf(qt)) {
        return 0;
    } else {
        for (int i = 0; i < 4; ++i) {
            depths[i] = depth(qt->child[i]);
        }
        return 1 + max_depth(depths);
    }
}
```


Fonctions utiles (1/4)

Comment remplir un arbre depuis une matrice ?

| | | | |
|-------|--|------|--------|
| SG=0 | | SD=1 | |
| 21 | | 12 | 4 4 |
| 9 | | 7 | 4 4 |
| ----- | | | |
| 1 | | 1 | 0 31 |
| 1 | | 1 | 3 27 |
| IG=2 | | ID=3 | |

Quel arbre cela représente ?

Fonctions utiles (1/4)

Comment remplir un arbre depuis une matrice ?

SG=0 | SD=1

21 | 12 | 4 | 4

9 | 7 | 4 | 4

1 | 1 | 0 | 31

1 | 1 | 3 | 27

IG=2 | ID=3

Quel arbre cela représente ?

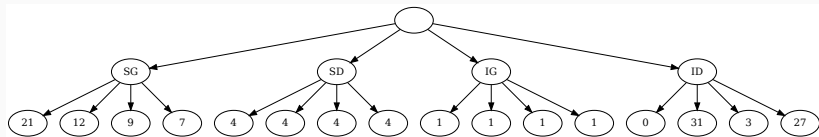


Figure 2 – L'arbre correspondant

Fonctions utiles (2/4)

- On veut transférer la valeur d'une case ligne/colonne dans une feuille.
- Comment ?

Soit ligne=2, colonne=3

Trouver un algorithme

| | | | |
|-------|--|------|--------|
| SG=0 | | SD=1 | |
| 21 | | 12 | 4 4 |
| 9 | | 7 | 4 4 |
| ----- | | | |
| 1 | | 1 | 0 31 |
| 1 | | 1 | 3 27 |
| IG=2 | | ID=3 | |

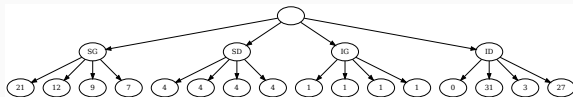


Figure 3 – Déterminer un algorithme.

- Quelle feuille pour 31 (li=2, co=3) ?
- Plus important : quel chemin ?

Fonctions utiles (2/4)

- On veut transférer la valeur d'une case ligne/colonne dans une feuille.
- Comment ?

Soit ligne=2, colonne=3

Trouver un algorithme

| | | | |
|-------|--|------|--------|
| SG=0 | | SD=1 | |
| 21 | | 12 | 4 4 |
| 9 | | 7 | 4 4 |
| ----- | | | |
| 1 | | 1 | 0 31 |
| 1 | | 1 | 3 27 |
| IG=2 | | ID=3 | |

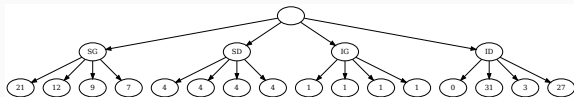


Figure 3 – Déterminer un algorithme.

- Quelle feuille pour 31 (li=2, co=3) ?
- Plus important : quel chemin ?
- $co \rightarrow G/D, li \rightarrow S/I,$
- $2 * (li / 2) + co / 2 \rightarrow 2 * 1 + 1 = 3$
- $2 * ((li \% 2) / 1) + (co \% 2) / 1 \rightarrow$

Fonctions utiles (3/4)

Soit ligne=2, colonne=3

| | | | |
|-------|--|------|--------|
| SG=0 | | SD=1 | |
| 21 | | 12 | 4 4 |
| 9 | | 7 | 4 4 |
| ----- | | | |
| 1 | | 1 | 0 31 |
| 1 | | 1 | 3 27 |
| IG=2 | | ID=3 | |

Trouver un algorithme (prendre plusieurs exemples, 15min, matrix)

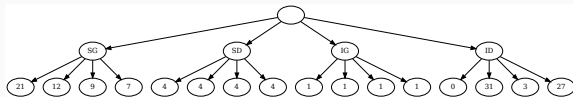


Figure 4 – Déterminer un algorithme.

- Comment généraliser ?

Fonctions utiles (3/4)

Soit ligne=2, colonne=3

| | | | |
|-------|--|------|--------|
| SG=0 | | SD=1 | |
| 21 | | 12 | 4 4 |
| 9 | | 7 | 4 4 |
| ----- | | | |
| 1 | | 1 | 0 31 |
| 1 | | 1 | 3 27 |
| IG=2 | | ID=3 | |

Trouver un algorithme (prendre plusieurs exemples, 15min, matrix)

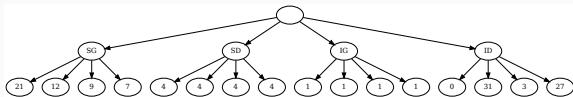


Figure 4 – Déterminer un algorithme.

- Comment généraliser ?

```
noeud position(li, co, arbre)
    d = profondeur(arbre);
    tant_que (d >= 1)
        index = 2 * ((li % 2^d) / 2^(d-1))
                (col % 2^d) / 2^(d-1)
        arbre = arbre.enfant[index]
        d -= 1
    retourne arbre
```

Fonctions utiles (4/4)

Pseudo-code

```
noeud position(li, co, arbre)
    d = profondeur(arbre);
    tant_que (d >= 1)
        index = 2 * ((li % 2d) / 2(d-1)) +
                (col % 2d) / 2(d-1)
        arbre = arbre.enfant[index]
        d -= 1
    retourne arbre
```

Écrire le code C correspondant (5min, matrix)

Remplir l'arbre

A partir d'une matrice (pseudo-code, 5min, matrix) ?

Remplir l'arbre

A partir d'une matrice (pseudo-code, 5min, matrix) ?

```
arbre matrice_vers_arbre(matrice)
    arbre = creer_arbre(profondeur)
    pour li de 0 à nb_lignes(matrice)
        pour co de 0 à nb_colonnes(matrice)
            noeud = position(li, co, arbre)
            noeud.info = matrice[li][co]
    retourne arbre
```

Remplir l'arbre

A partir d'une matrice (pseudo-code, 5min, matrix) ?

```
arbre matrice_vers_arbre(matrice)
    arbre = creer_arbre(profondeur)
    pour li de 0 à nb_lignes(matrice)
        pour co de 0 à nb_colonnes(matrice)
            noeud = position(li, co, arbre)
            noeud.info = matrice[li][co]
    retourne arbre
```

A partir d'une matrice (C, 5min, matrix) ?

Remplir l'arbre

A partir d'une matrice (pseudo-code, 5min, matrix) ?

```
arbre matrice_vers_arbre(matrice)
    arbre = creer_arbre(profondeur)
    pour li de 0 à nb_lignes(matrice)
        pour co de 0 à nb_colonnes(matrice)
            noeud = position(li, co, arbre)
            noeud.info = matrice[li][co]
    retourne arbre
```

A partir d'une matrice (C, 5min, matrix) ?

```
node *matrix_to_qt(int nb_li, int nb_co, int matrix[nb_li][nb_co], int depth) {
    node *qt = qt_create(depth);
    for (int li = 0; li < nb_li; ++li) {
        for (int co = 0; co < nb_co; ++co) {
            node *current = position(li, co, qt);
            current->info = matrix[li][co];
        }
    }
    return qt;
}
```

Remplir la matrice

A partir de l'arbre (pseudo-code, 3min, matrix) ?

Remplir la matrice

A partir de l'arbre (pseudo-code, 3min, matrix) ?

```
matrice arbre_vers_matrice(arbre)
    matrice = creer_matrice(nb_lignes(arbre), nb_colonnes(arbre))
    pour li de 0 à nb_lignes(matrice)
        pour co de 0 à nb_colonnes(matrice)
            noeud = position(li, co, arbre)
            matrice[co][li] = noeud.info
    retourne matrice
```

Remplir la matrice

A partir de l'arbre (pseudo-code, 3min, matrix) ?

```
matrice arbre_vers_matrice(arbre)
    matrice = creer_matrice(nb_lignes(arbre), nb_colonnes(arbre))
    pour li de 0 à nb_lignes(matrice)
        pour co de 0 à nb_colonnes(matrice)
            noeud = position(li, co, arbre)
            matrice[co][li] = noeud.info
    retourne matrice
```

A partir de l'arbre (C, 3min, matrix) ?

Remplir la matrice

A partir de l'arbre (pseudo-code, 3min, matrix) ?

```
matrice arbre_vers_matrice(arbre)
    matrice = creer_matrice(nb_lignes(arbre), nb_colonnes(arbre))
    pour li de 0 à nb_lignes(matrice)
        pour co de 0 à nb_colonnes(matrice)
            noeud = position(li, co, arbre)
            matrice[co][li] = noeud.info
    retourne matrice
```

A partir de l'arbre (C, 3min, matrix) ?

```
void qt_to_matrix(node *qt, int nb_li, int nb_co, int matrix[nb_li][nb_co]) {
    for (int li = 0; li < nb_li; ++li) {
        for (int co = 0; co < nb_co; ++co) {
            node *current = position(li, co, qt);
            matrix[li][co] = current->info;
        }
    }
}
```

Transformations avec un arbre quaternaire

A faire

- Symétrie axiale (horizontale/verticale).
- Rotation quart de cercle (gauche/droite).
- Compression.

La symétrie verticale

Que donne la symétrie verticale de

SG=0 | SD=1

21 | 12 | 4 | 4

9 | 7 | 4 | 4

1 | 1 | 0 | 31

1 | 1 | 3 | 27

IG=2 | ID=3

La symétrie verticale

Que donne la symétrie verticale de

SG=0 | SD=1

21 | 12 | 4 | 4

9 | 7 | 4 | 4

1 | 1 | 0 | 31

1 | 1 | 3 | 27

IG=2 | ID=3

SG=0 | SD=1

4 | 4 | 12 | 21

4 | 4 | 7 | 9

31 | 0 | 1 | 1

27 | 3 | 1 | 1

IG=2 | ID=3

La symétrie d'axe vertical

Comment faire sur une matrice (3min, matrix) ?

La symétrie d'axe vertical

Comment faire sur une matrice (3min, matrix) ?

```
matrice symétrie(matrice)
    pour i de 0 à nb_lignes(matrice)
        pour j de 0 à nb_colonnes(matrice)/2
            échanger(matrice[i][j], matrice[i][nb_colonnes(matrice)-1-j])
    retourne matrice
```

La symétrie d'axe vertical

Comment faire sur un arbre ?

- Faire un dessin de l'arbre avant/après (5min, matrix)

| | | | | | | |
|-------|--|------|--|------|-------|------|
| SG=0 | | SD=1 | | SG=0 | | SD=1 |
| 21 | | 12 | | 4 | | 4 |
| 9 | | 7 | | 4 | | 4 |
| ----- | | | | => | ----- | |
| 1 | | 1 | | 0 | | 31 |
| 1 | | 1 | | 3 | | 27 |
| IG=2 | | ID=3 | | IG=2 | | ID=3 |

- Écrire le pseudo-code (3min, matrix)

La symétrie d'axe vertical

Comment faire sur un arbre ?

- Faire un dessin de l'arbre avant/après (5min, matrix)

| SG=0 | | SD=1 | | SG=0 | | SD=1 |
|----------------|--|------|--|------|--|------|
| 21 | | 12 | | 4 | | 4 |
| 9 | | 7 | | 4 | | 4 |
| ----- => ----- | | | | | | |
| 1 | | 1 | | 0 | | 31 |
| 1 | | 1 | | 3 | | 27 |
| IG=2 | | ID=3 | | IG=2 | | ID=3 |

- Écrire le pseudo-code (3min, matrix)

```
arbre symétrie(arbre)
  si !est_feuille(arbre)
    échanger(arbre.enfant[0], arbre.enfant[1])
    échanger(arbre.enfant[2], arbre.enfant[3])
    pour i de 0 à 3
      symétrie(arbre.enfant[i])
  retourne arbre
```

La symétrie d'axe horizontal

- Trivial de faire l'axe horizontal (exercice à la maison)

Rotation d'un quart de cercle

Comment faire sur un arbre ?

- Faire un dessin de l'arbre avant/après (5min, matrix)

| SG=0 | | | | | SD=1 | | | | | SG=0 | | | | | SD=1 | | | |
|-------|--|----|--|---|------|----|--|----|-------|------|--|---|--|----|------|----|--|--|
| 21 | | 12 | | 4 | | 4 | | | | 4 | | 4 | | 31 | | 27 | | |
| 9 | | 7 | | 4 | | 4 | | | | 4 | | 4 | | 0 | | 3 | | |
| ----- | | | | | | | | => | ----- | | | | | | | | | |
| 1 | | 1 | | 0 | | 31 | | | | 12 | | 7 | | 1 | | 1 | | |
| 1 | | 1 | | 3 | | 27 | | | | 21 | | 9 | | 1 | | 1 | | |
| IG=2 | | | | | ID=3 | | | | | IG=2 | | | | | ID=3 | | | |

- Écrire le pseudo-code (3min, matrix)

Rotation d'un quart de cercle

Comment faire sur un arbre ?

- Faire un dessin de l'arbre avant/après (5min, matrix)

| SG=0 | | SD=1 | | SG=0 | | SD=1 | | | | | | | | |
|----------------|--|------|--|------|--|------|--|----|--|---|--|----|--|----|
| 21 | | 12 | | 4 | | 4 | | 4 | | 4 | | 31 | | 27 |
| 9 | | 7 | | 4 | | 4 | | 4 | | 4 | | 0 | | 3 |
| ----- => ----- | | | | | | | | | | | | | | |
| 1 | | 1 | | 0 | | 31 | | 12 | | 7 | | 1 | | 1 |
| 1 | | 1 | | 3 | | 27 | | 21 | | 9 | | 1 | | 1 |
| IG=2 | | ID=3 | | IG=2 | | ID=3 | | | | | | | | |

- Écrire le pseudo-code (3min, matrix)

```
rien rotation_gauche(arbre)
  si !est_feuille(arbre)
    échange_cyclique_gauche(arbre.enfant)
    pour i de 0 à 3
      rotation_gauche(arbre.enfant[i])
```

Rotation d'un quart de cercle

Comment faire sur un arbre ?

| SG=0 | | SD=1 | | SG=0 | | SD=1 |
|-------|--|------|--|------|-------|------|
| 21 | | 12 | | 4 | | 4 |
| 9 | | 7 | | 4 | | 4 |
| ----- | | | | => | ----- | |
| 1 | | 1 | | 0 | | 31 |
| 1 | | 1 | | 3 | | 27 |
| IG=2 | | ID=3 | | IG=2 | | ID=3 |

- Écrire le vrai code (5min, matrix)

Rotation d'un quart de cercle

Comment faire sur un arbre ?

| SG=0 | | SD=1 | | SG=0 | | SD=1 |
|----------------|--|------|--|------|--|------|
| 21 | | 12 | | 4 | | 4 |
| 9 | | 7 | | 4 | | 4 |
| ----- => ----- | | | | | | |
| 1 | | 1 | | 0 | | 31 |
| 1 | | 1 | | 3 | | 27 |
| IG=2 | | ID=3 | | IG=2 | | ID=3 |

- Écrire le vrai code (5min, matrix)

```
void rotate(node *qt) {
    if (!is_leaf(qt)) {
        node *tmp = qt->child[2];
        qt->child[2] = qt->child[0];
        qt->child[0] = qt->child[1];
        qt->child[1] = qt->child[3];
        qt->child[3] = tmp;
        for (int i=0; i<CHILDREN; i++) {
            rotate(qt->child[i]);
        }
    }
}
```

Compression sans perte (1/5)

Idée générale

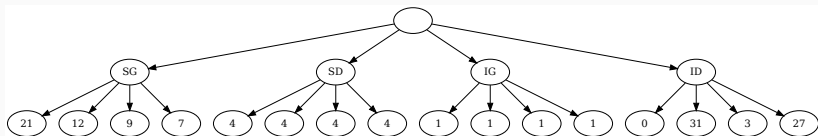
- Regrouper les pixels par valeur

| SG=0 | | SD=1 | | SG=0 | | SD=1 |
|-------|--|------|--|------|-------|------|
| 21 | | 12 | | 4 | | 4 |
| 9 | | 7 | | 4 | | 4 |
| ----- | | | | => | ----- | |
| 1 | | 1 | | 0 | | 31 |
| 1 | | 1 | | 3 | | 27 |
| IG=2 | | ID=3 | | IG=2 | | ID=3 |

- Comment faire ?

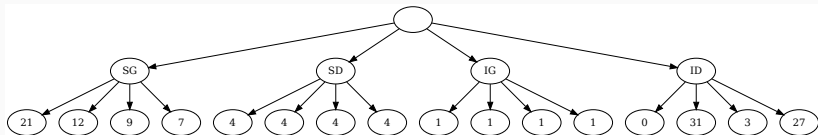
Compression sans perte (2/5)

Que devient l'arbre suivant ?

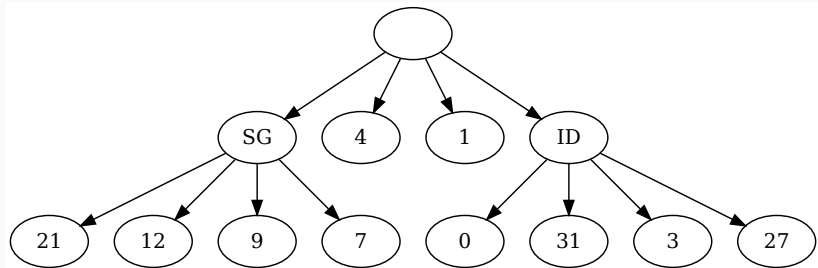


Compression sans perte (2/5)

Que devient l'arbre suivant ?



Arbre compressé



Compression sans perte (3/5)

- Si un nœud a tous ses enfants égaux :
 - Stocker cette valeur dans ce nœud,
 - Supprimer ses enfants.
- Jusqu'à remonter à la racine.

Écrire le pseudo-code (5min, matrix)

Compression sans perte (3/5)

- Si un nœud a tous ses enfants égaux :
 - Stocker cette valeur dans ce nœud,
 - Supprimer ses enfants.
- Jusqu'à remonter à la racine.

Écrire le pseudo-code (5min, matrix)

```
rien compression_sans_perte(arbre)
  si !est_feuille(arbre)
    pour i de 0 à 3
      compression_sans_perte(arbre.enfant[i])
  si derniere_branche(arbre)
    valeur, toutes_égales = valeur_enfants(arbre)
    si toutes_égales
      arbre.info = valeur
      detruire_enfants(arbre)
```


Compression sans perte (4/5)

Écrire le code C (5min, matrix)

Compression sans perte (4/5)

Écrire le code C (5min, matrix)

```
void lossless_compression(node *qt) {
    if (!is_leaf(qt)) {
        for (int i=0; i<CHILDREN; i++) {
            lossless_compression(qt->child[i]);
        }
        if (is_last_branch(qt)) {
            int val = -1;
            if (last_value(qt, &val)) {
                qt->info = val;
                for (int i=0; i<CHILDREN; ++i) {
                    free(qt->child[i]);
                    qt->child[i] = NULL;
                }
            }
        }
    }
}
```

Compression sans perte (5/5)

```
bool is_last_branch(node *qt) {
    for (int i = 0; i < CHILDREN; ++i) {
        if (!is_leaf(qt)) {
            return false;
        }
    }
    return true;
}

bool last_value(node *qt, int *val) {
    int info = qt->child[0];
    for (int i = 1; i < CHILDREN; ++i) {
        if (info != qt->child[i]) {
            return false;
        }
    }
    *val = info;
    return true;
}
```

Compression avec perte (1/5)

Idée générale

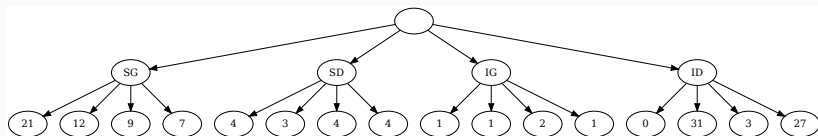
- Regrouper les pixels par valeur sous certaines conditions

| SG=0 SD=1 | | | | SG=0 SD=1 | | | |
|-------------|--|----|------|-------------|-------|----|------|
| 21 | | 12 | | 4 | | 3 | |
| 9 | | 7 | | 4 | | 4 | |
| ----- | | | | => | ----- | | |
| 1 | | 1 | | 0 | | 31 | |
| 2 | | 1 | | 3 | | 27 | |
| IG=2 | | | ID=3 | IG=2 | | | ID=3 |

- On enlève si l'écart à la moyenne est "petit" ?

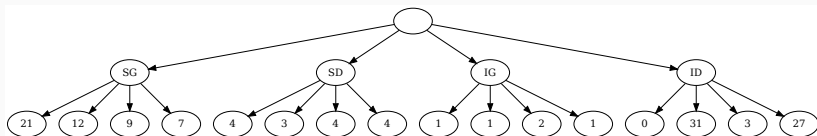
Compression avec perte (2/5)

Que devient l'arbre suivant si l'écart est petit ?

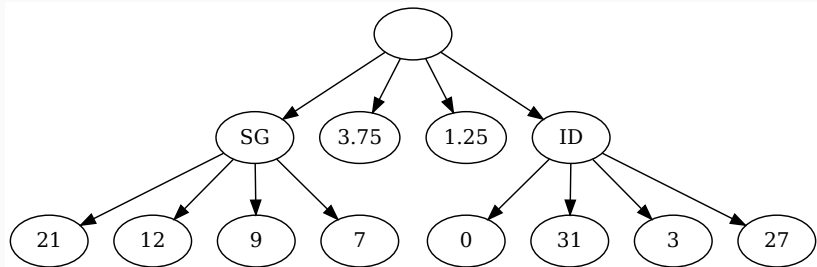


Compression avec perte (2/5)

Que devient l'arbre suivant si l'écart est petit ?



Arbre compressé



Compression avec perte (3/5)

Comment mesurer l'écart à la moyenne ?

Compression avec perte (3/5)

Comment mesurer l'écart à la moyenne ?

- Avec l'écart-type

$$\mu = \frac{1}{4} \sum_{i=0}^3 p[i], \quad \sigma = \sqrt{\frac{1}{4} \sum_{i=0}^3 (\mu - p[i])^2} = \sqrt{\frac{1}{4} \left(\sum_{i=0}^3 p[i]^2 \right) - \mu^2}$$

Que devient l'algorithme ?

Compression avec perte (3/5)

Comment mesurer l'écart à la moyenne ?

- Avec l'écart-type

$$\mu = \frac{1}{4} \sum_{i=0}^3 p[i], \quad \sigma = \sqrt{\frac{1}{4} \sum_{i=0}^3 (\mu - p[i])^2} = \sqrt{\frac{1}{4} \left(\sum_{i=0}^3 p[i]^2 \right) - \mu^2}$$

Que devient l'algorithme ?

- Si $\sigma < \theta$, où θ est la **tolérance** :
 - Remplacer la valeur du pixel par la moyenne des enfants.
 - Remonter les valeurs dans l'arbre.

Quelle influence de la valeur de θ sur la compression ?

Compression avec perte (3/5)

Comment mesurer l'écart à la moyenne ?

- Avec l'écart-type

$$\mu = \frac{1}{4} \sum_{i=0}^3 p[i], \quad \sigma = \sqrt{\frac{1}{4} \sum_{i=0}^3 (\mu - p[i])^2} = \sqrt{\frac{1}{4} \left(\sum_{i=0}^3 p[i]^2 \right) - \mu^2}$$

Que devient l'algorithme ?

- Si $\sigma < \theta$, où θ est la **tolérance** :
 - Remplacer la valeur du pixel par la moyenne des enfants.
 - Remonter les valeurs dans l'arbre.

Quelle influence de la valeur de θ sur la compression ?

- Plus θ est grand, plus l'image sera compressée.

Compression avec perte (4/5)

Que devient l'arbre avec $\theta = 0.5$?

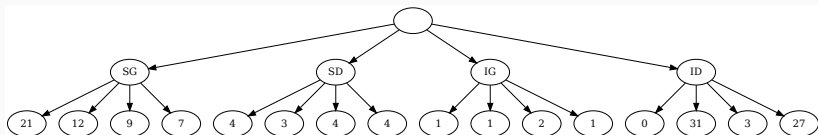


Figure 5 – L'arbre original.

Compression avec perte (4/5)

Que devient l'arbre avec $\theta = 0.5$?

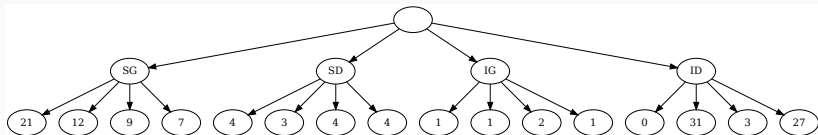
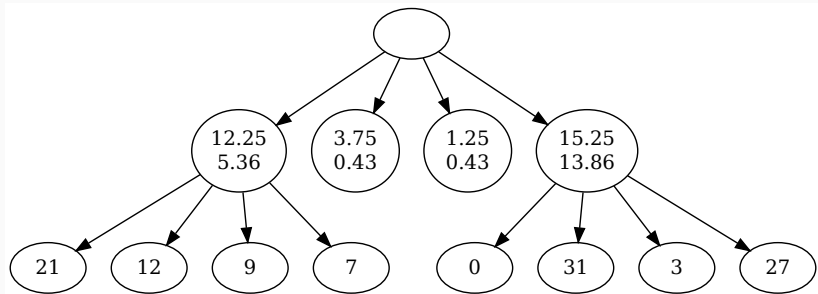


Figure 5 – L'arbre original.



Modifications sur la structure de données ?

Modifications sur la structure de données ?

- On stocke la moyenne, et la moyenne des carrés.

```
struct noeud  
    flottant moyenne, moyenne_carre  
    node enfants[4]
```

- Comment on calcule moyenne et moyenne_carre sur chaque nœud (pseudo-code) ?

Calcul de la moyenne

Pseudo-code (5min, matrix)

Calcul de la moyenne

Pseudo-code (5min, matrix)

```
rien moyenne(arbre) {  
    si !est_feuille(arbre)  
        pour enfant dans arbre.enfants  
            moyenne(enfant)  
    pour enfant dans arbre.enfants  
        arbre.moyenne += enfant.moyenne  
        arbre.moyenne_carre += enfant.moyenne_carre  
    arbre.moyenne /= 4  
    arbre.moyenne_carre /= 4  
}
```


La compression avec pertes

Pseudo-code (5min, matrix)

La compression avec pertes

Pseudo-code (5min, matrix)

```
rien compression_avec_pertes(arbre, theta)
  si !est_feuille(arbre)
    pour i de 0 à 3
      compression_avec_pertes(arbre.enfant[i])
    si derniere_branche(arbre)
      si racine(arbre.moyenne_carre - arbre.moyenne^2) < theta
        detruire_enfants(arbre)
```

Le code en entier

```
arbre = matrice_à_arbre(matrice)
moyenne(arbre)
compression_avec_pertes(arbre)
```

Slides très fortement inspirés du cours de J. Latt, Unige

Simulation du problème à N -corps

- Prédiction du mouvement d'un grand nombre de corps célestes.
- Modélisation :
 - On se limite aux étoiles ;
 - Chaque étoile est caractérisée par un point (coordonnées) et une masse ;
 - On simule en deux dimensions.
 - Interactions uniquement par les lois de la gravitation Newtonienne (oui-oui c'est de la **physique**!).

Les équations du mouvement

Mouvement de la i -ème étoile

- Algorithme de Verlet ($t_{n+1} = t_n + \delta t$)

$$\vec{x}_i(t_{n+1}) = 2\vec{x}_i(t_n) - \vec{x}_i(t_{n-1}) + \vec{a}_i(t_n)\delta t^2.$$

Force de gravitation

- $\vec{a}_i(t_n) = \vec{F}_i/m_i.$
- Sur l'étoile i , la force résultante est donnée par

$$\vec{F}_i = \sum_{j=1, j \neq i}^N \vec{F}_{ij}.$$

avec

$$\vec{F}_{ij} = \frac{Gm_i m_j (\vec{x}_j - \vec{x}_i)}{||\vec{x}_j - \vec{x}_i||^3}.$$

Algorithme du problème à n -corps

Pseudo-code : structure de données

```
struct étoile
    flottant m
    vec x, x_precedent, f
```

Pseudo-code : itération temporelle

```
rien iteration_temporelle(étoiles, dt)
    pour étoile_une dans étoiles
        étoile_une.f = 0
        pour étoile_deux dans étoiles
            si (étoile_un != étoile_deux)
                étoile_une.f +=
                    force(étoile_une, étoile_deux)
    pour étoile dans étoiles
        étoile.x, étoile.x_precedent =
            verlet(étoile.x, étoile.x_precedent,
                étoile.f / étoile.m, dt)
```

Algorithme du problème à n -corps

Complexité

- Complexité de chacune des parties ?

Algorithme du problème à n -corps

Complexité

- Complexité de chacune des parties ?
- $\mathcal{O}(N^2)$, $\mathcal{O}(N)$.

En temps CPU pour une itération

- Si le temps pour $N = 1$ est environ $1\mu s$, on a :

| N | N^2 | t [s] | t [réel] |
|-----------|-----------|---------|------------|
| 10 | 10^2 | $1e-4$ | |
| 10^4 | 10^8 | $1e+2$ | ~1min |
| 10^6 | 10^{12} | $1e+6$ | ~11j |
| 10^9 | 10^{18} | $1e+12$ | ~30K ans |
| 10^{11} | 10^{22} | $1e+16$ | ~300M ans |

- Typiquement, il y a des milliers-millions d'itérations.
- Il y a 10^{11} étoiles dans la galaxie.
- Houston we have a problem.

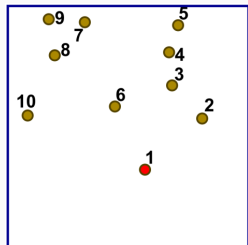
Comment faire mieux ? Des idées ?

Comment faire mieux ? Des idées ?

- Si un groupe d'étoiles est suffisamment loin, on le modélise comme un corps unique situé en son centre de masse.
- Exemple : Si on simule plusieurs galaxies, on considère chaque galaxie comme un corps unique !
- Un arbre quaternaire est une structure parfaite pour regrouper les étoiles.

Le cas à 10 corps

Illustration : le cas à 10 corps



Problématique

- On veut calculer la force sur 1.

Le cas à 10 corps

Illustration : le cas à 10 corps

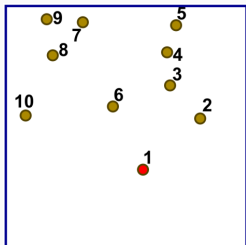
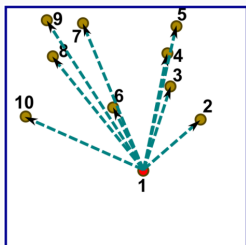


Illustration : le cas à 10 corps



Problématique

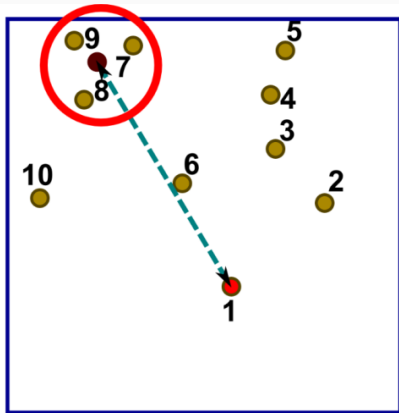
- On veut calculer la force sur 1.

Résultat

- Calcul et somme des forces venant des 9 autres corps.

Le cas à 10 corps

Réduction d'un groupe à un seul corps

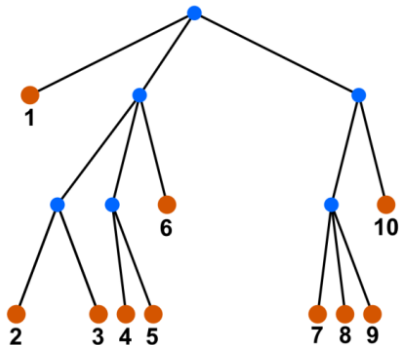
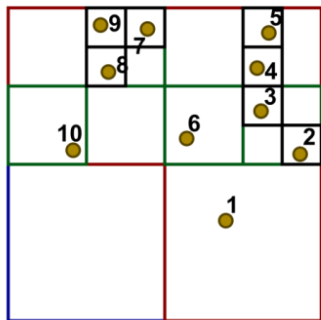


Idée

- On accélère le calcul en traitant un groupe comme un seul corps.
- Fonctionne uniquement si le groupe est assez loin.
- Autrement l'approximation est trop grossière.

Solution : l'arbre quaternaire

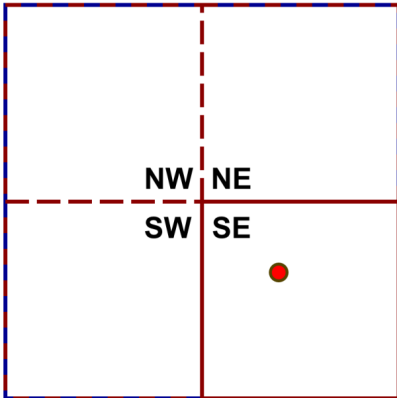
Corps célestes - arbre



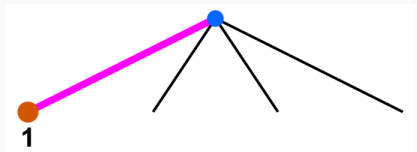
- On omet les nœuds vides pour alléger la représentation.
- La numérotation est :
 - 0 : ID
 - 1 : SD
 - 2 : IG
 - 3 : SG

Exemple d'insertion

Insertion corps 1



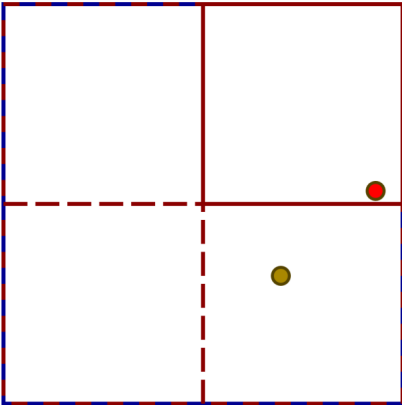
Arbre, niveau 1



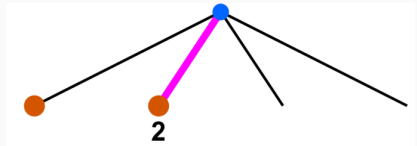
- Quadrant ID.
- La feuille est vide, on insère.

Exemple d'insertion

Insertion corps 2



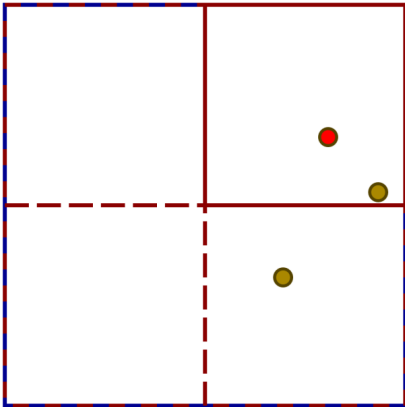
Arbre, niveau 1



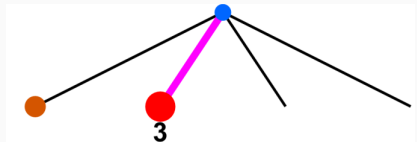
- Quadrant SD.
- La feuille est vide, on insère.

Exemple d'insertion

Insertion corps 3 (1/N)



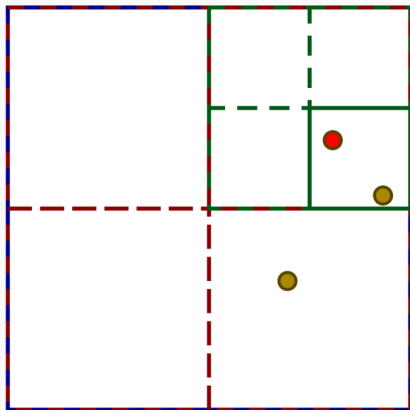
Arbre, niveau 1



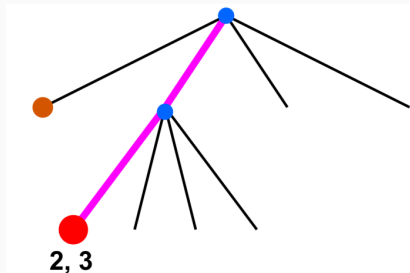
- Quadrant SD.
- La feuille est prise par 2.

Exemple d'insertion

Insertion corps 3 (2/N)



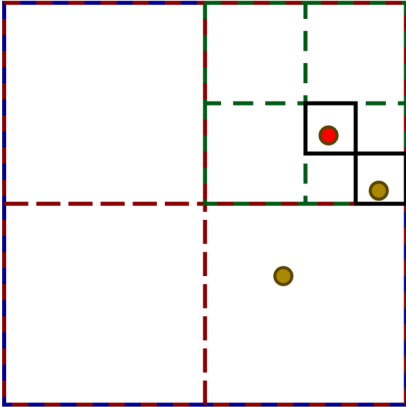
Arbre, niveau 2



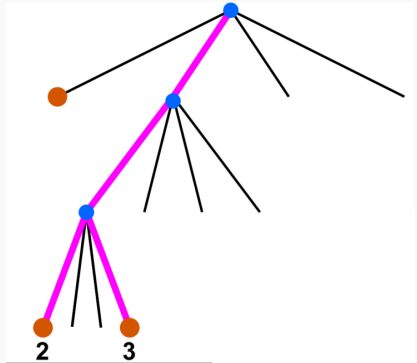
- On crée un nouveau nœud.
- Deux corps dans le nœud ID.
- On crée un nouveau nœud.

Exemple d'insertion

Insertion corps 3 (3/N)



Arbre, niveau 3



- 2 va dans ID.
- 3 va dans SG.
- C'est des feuilles vides, tout va bien.

Exemple d'insertion

Que fait-on avec les nœuds intérieurs ?

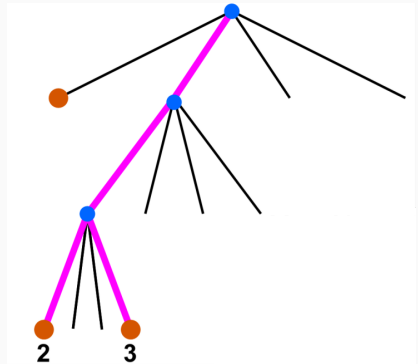
- On les utilise pour :
 - stocker la masse totale ;
 - stocker le centre de masse.

$$m = m_2 + m_3, \quad (1)$$

$$\vec{x} = \frac{m_2 \vec{x}_2 + m_3 \vec{x}_3}{m}. \quad (2)$$

Chaque feuille contient une étoile

Arbre



- Insertion du corps c dans le nœud n en partant de la racine.
- Si le nœud n
 - ne contient pas de corps, on y dépose c ;
 - est interne, on met à jour masse et centre de masse, c est inséré récursivement dans le bon quadrant ;
 - est externe, on subdivise n , on met à jour la masse et centre de masse, on insère récursivement les deux nœuds dans les quadrants appropriés.

Remarque

- Il faut stocker les coordonnées des quadrants.
- Un nœud a un comportement différent s'il est interne ou externe.

Algorithme d'insertion

Structure de données

```
struct node
    etoile e // externe: pour stocker
    etoile sup_etoile // interne: pour stocker m, x
    quadrant q // coordonnées du quadrant
    node enfants[4]
```

Remarque :

- On fait une simplification “moche” : sup_etoile pourrait juste avoir une masse et une position.

Algorithme d'insertion

Algorithme d'insertion, pseudo-code (15min, matrix)

Algorithme d'insertion

Algorithme d'insertion, pseudo-code (15min, matrix)

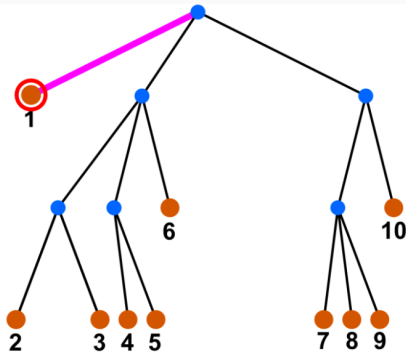
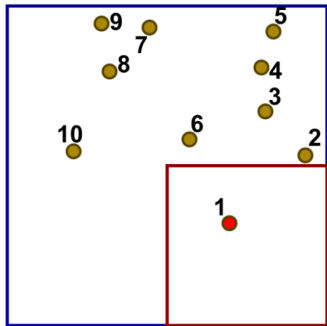
```
rien insertion_etoile(arbre, e)
  si (!est_vide(arbre) && dans_le_quadrant(arbre.q, e.x)) {
    si (est_feuille(arbre))
      si (!contient_etoile(arbre))
        arbre.e = e
    sinon
      // on crée enfants et arbre.sup_etoile est initialisée
      subdivision_arbre(arbre, e)
      pour enfant dans arbre.enfants
        insertion_etoile(enfant, arbre.e)
      pour enfant dans arbre.enfants
        insertion_etoile(enfant, e)
      destruction(arbre.e)
  }
  sinon
    maj_masse_cdm(arbre.sup_etoile, e)
    pour enfant dans arbre.enfants
      insertion_etoile(enfant, e)
```

Utilisation de l'arbre

- L'arbre est rempli : comment on calcule la force sur le corps 1 ?
- Parcours de l'arbre :
 - Si la distance entre 1 et le centre de masse est suffisante, on utilise la masse totale et centre de masse pour calculer la force.
 - Sinon on continue le parcours.

Calcul de la force

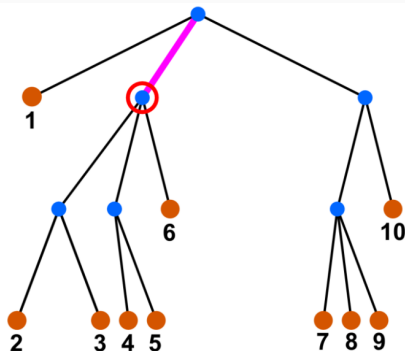
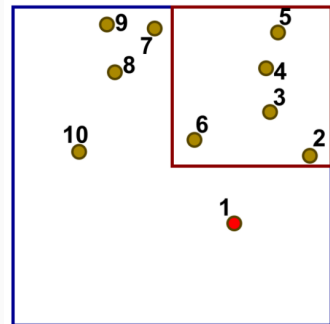
Calcul de la force sur 1



- Le quadrant ID ne contient que 1, rien à faire.

Calcul de la force

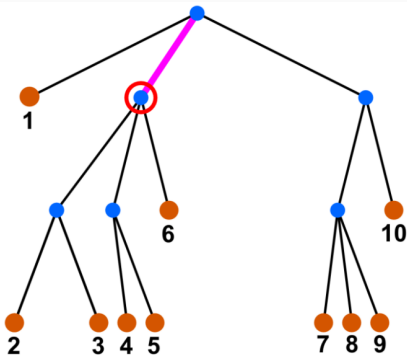
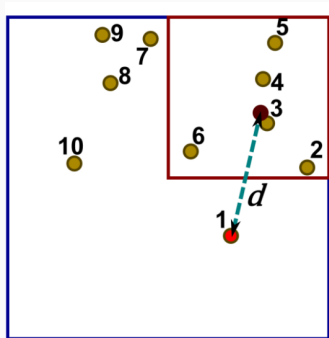
Calcul de la force sur 1



- Le quadrant SG contient 5 corps.

Calcul de la force

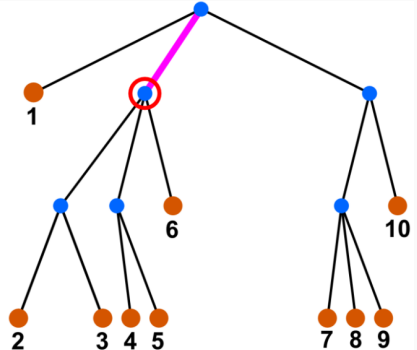
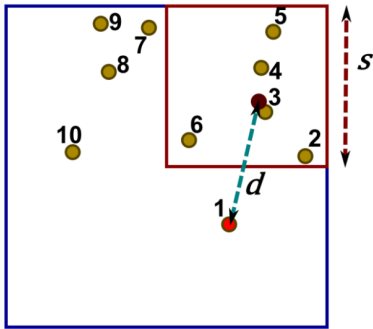
Calcul de la force sur 1



- La distance entre 1 et le centre de masse de SG est d .

Calcul de la force

Calcul de la force sur 1



- La distance entre 1 et le centre de masse de SG est d .
- Est-ce que d est assez grand ?
- On va comparer avec la distance d avec la taille du quadrant s .

Critère θ

- On compare $d = \|\vec{x}_1 - \vec{x}_{cm}\|$ avec s la taille du quadrant.
- Le domaine est assez éloigné si

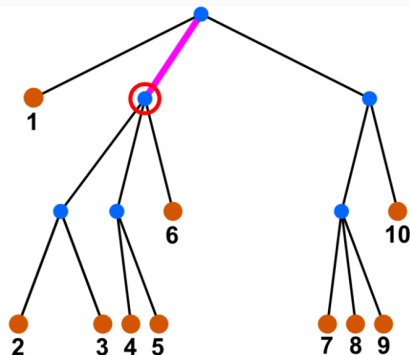
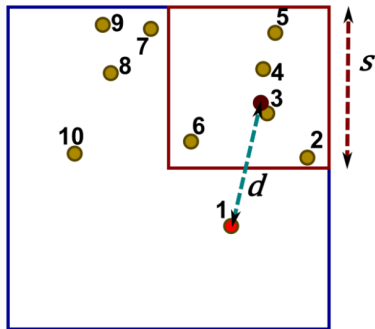
$$\frac{s}{d} < \theta,$$

- θ est la valeur de seuil.
- Une valeur typique est $\theta = 0.5$, donc la condition devient

$$d > 2s.$$

Calcul de la force

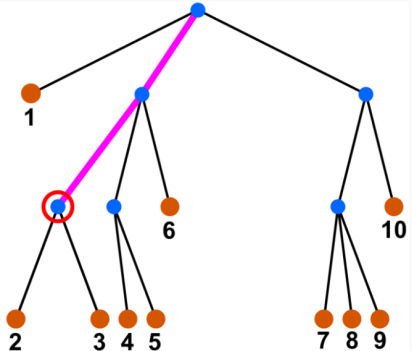
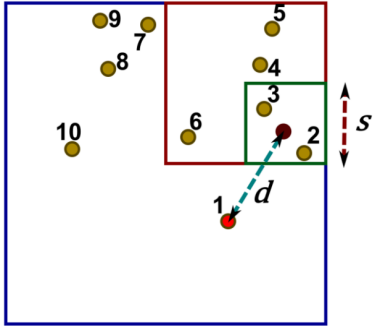
Calcul de la force sur 1



- Ici $d < 2s$, domaine rejeté.
- On descend dans l'arbre.

Calcul de la force

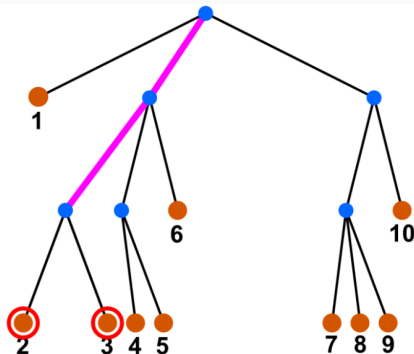
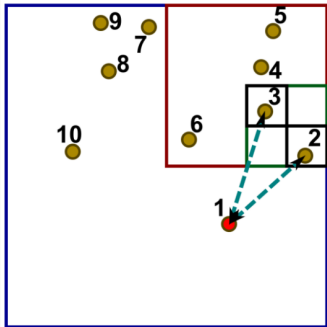
Calcul de la force sur 1



- s est plus petit, mais....
- Cela ne suffit pas $d < 2s$, domaine rejeté.

Calcul de la force

Calcul de la force sur 1



- Les nœuds sont des feuilles, on calcule la force.
- On ajoute la force qu'ils exercent sur 1.

Algorithme pour le calcul de la force

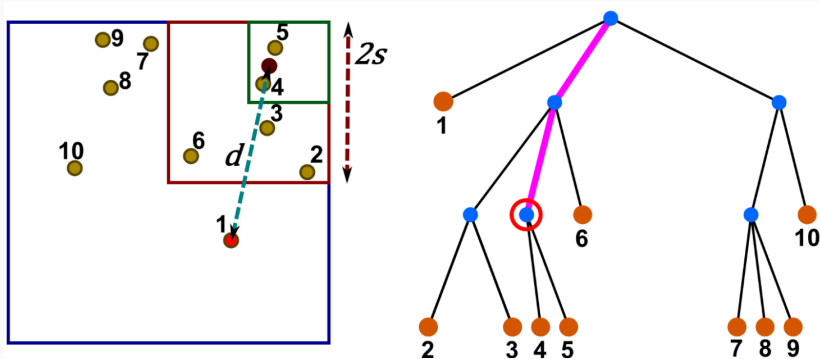
Pour calculer la force sur un corps c , on parcourt l'arbre en commençant par la racine :

- Si le nœud n est une feuille et n'est pas c , on ajoute la force dûe à n sur c ;
- Sinon, si $s/d < \theta$, on traite n comme une feuille et on ajoute la force dûe à n sur c ;
- Sinon on continue sur les enfants récursivement.

Continuons notre exemple précédent !

Calcul de la force

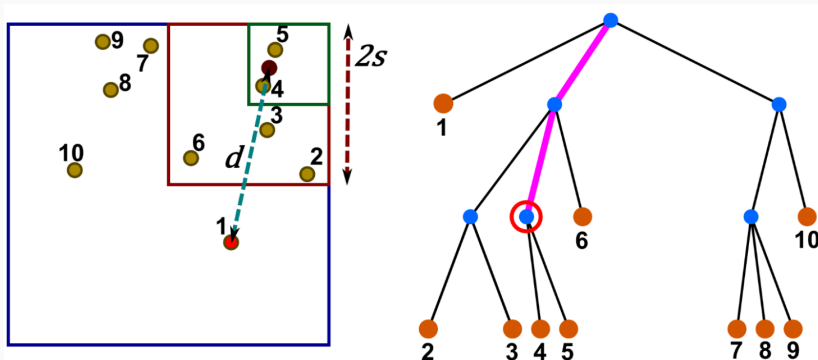
Calcul de la force sur 1



- Il y a deux corps dans le quadrant vert.
- Quel est le critère pour remplacer les étoiles par leur centre de masse ?

Calcul de la force

Calcul de la force sur 1



- Il y a deux corps dans le quadrant vert.
- Quel est le critère pour remplacer les étoiles par leur centre de masse ?
- Et oui ! $d > 2s$, donc on peut remplacer les étoiles par leur centre de masse !

Algorithme du calcul de force

Écrire le pseudo-code-code du calcul de la force

```
rien maj_force_sur_etoile(arbre, e, theta)
    si est_vide(arbre)
        retourne

    si est_feuille(arbre) && contient_etoile(arbre)
        && dans_le_quadrant(arbre.q, e.x)
        maj_force(e, arbre.e)
    sinon si noeud_assez_loin(arbre, e, theta)
        maj_force(e, arbre.sup_etoile)
    sinon
        pour enfant dans enfants
            maj_force_sur_etoile(enfant, e, theta)
```