Arbres quarternaires, compression et Barnes-Hut

Algorithmique et structures de données, 2022-2023

P. Albuquerque (B410), P. Künzli et O. Malaspinas (A401), ISC, HEPIA

2023-05-05

En partie inspirés des supports de cours de P. Albuquerque

Le cours précédent (1/2)

Questions

- Qu'est-ce qu'un arbre quaternaire?

Le cours précédent (1/2)

Questions

- Qu'est-ce qu'un arbre quaternaire?
- Un arbre où chaque nœud a soit **4 enfants** soit **aucun**.
- A quoi peut servir un arbre quaternaire?

Le cours précédent (1/2)

Questions

- Qu'est-ce qu'un arbre quaternaire?
- Un arbre où chaque nœud a soit **4 enfants** soit **aucun**.
- A quoi peut servir un arbre quaternaire?
- Compression

Le cours précédent (2/2)

Questions

- Structure de données d'un arbre quaternaire?

Le cours précédent (2/2)

Questions

- Structure de données d'un arbre quaternaire?

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

Le cours précédent (2/2)

Questions

- Structure de données d'un arbre quaternaire?

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

- Dessin d'un nœud d'arbre quaternaire (avec correspondance `node`)?

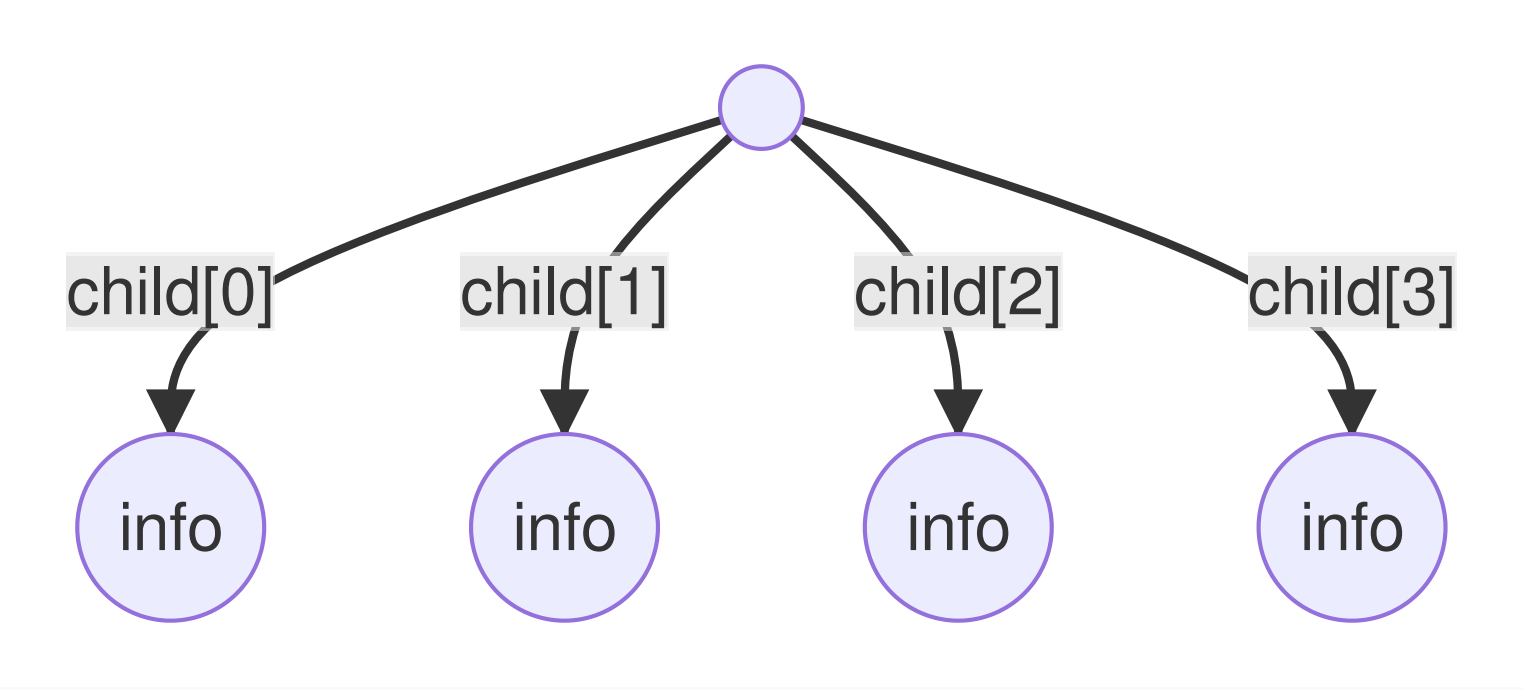
Le cours précédent (2/2)

Questions

- Structure de données d'un arbre quaternaire?

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

- Dessin d'un nœud d'arbre quaternaire (avec correspondance `node`)?



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

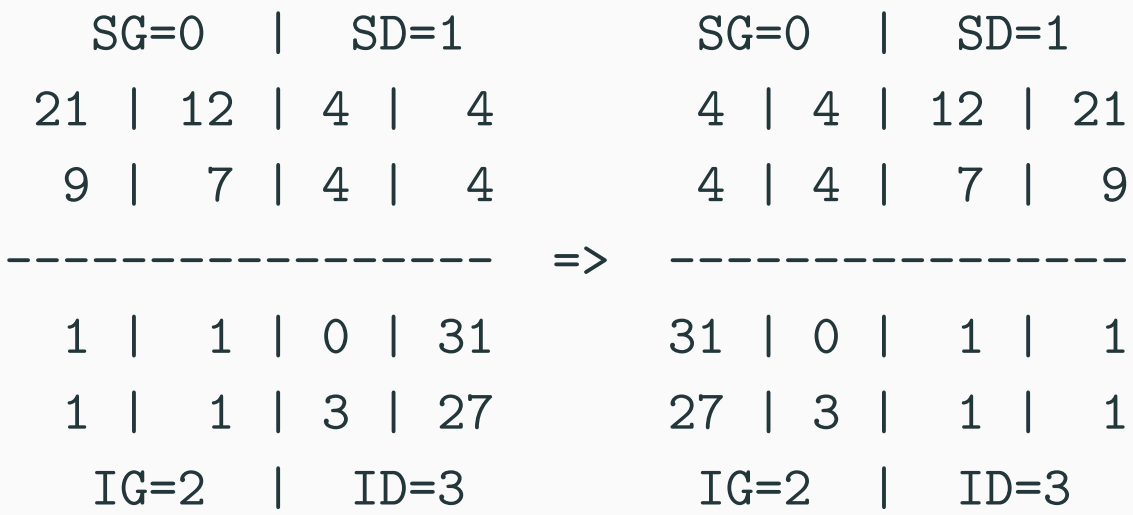
Comment faire sur une matrice (3min, matrix)?

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

La symétrie d'axe vertical

Comment faire sur un arbre?

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

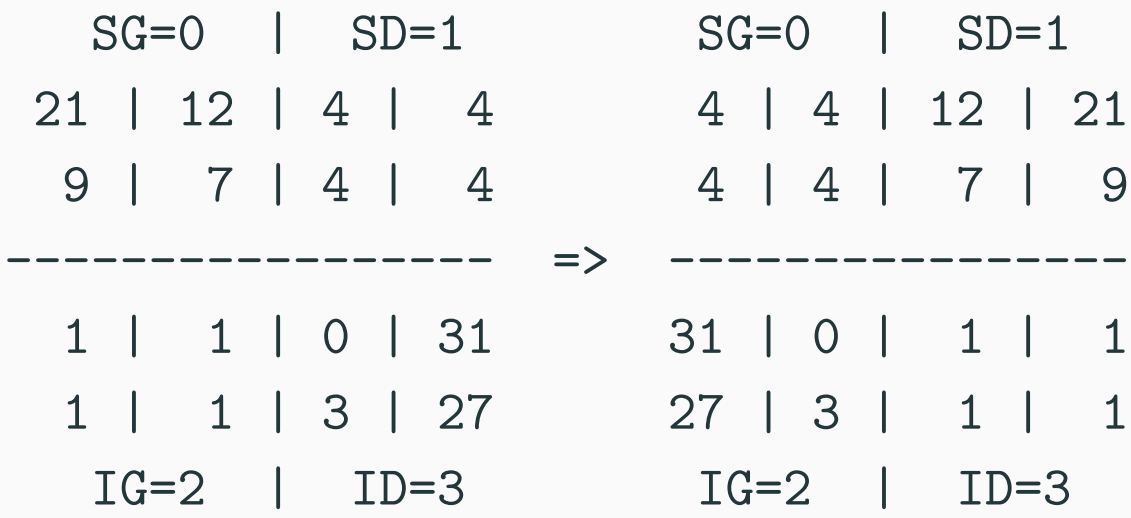


- Écrire le pseudo-code (3min, matrix)

La symétrie d'axe vertical

Comment faire sur un arbre?

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



- 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        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 vrai (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       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 vrai (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 < 4; i++) {
            rotate(qt->child[i]);
        }
    }
}
```

Compression sans perte (1/5)

Idée générale

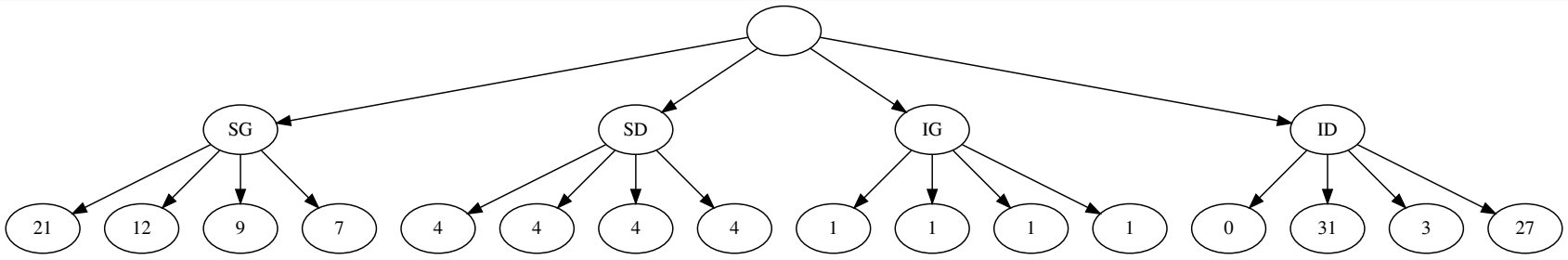
- Regrouper les pixels par valeur

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

- Comment faire?

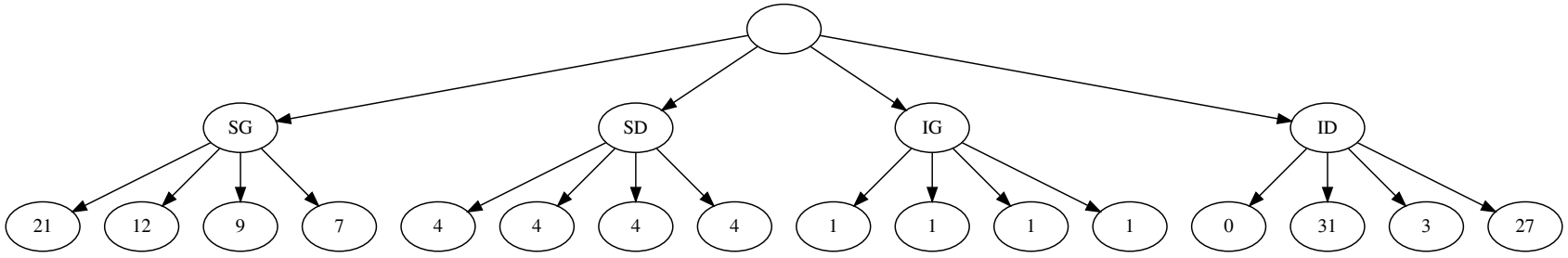
Compression sans perte (2/5)

Que devient l'arbre suivant?

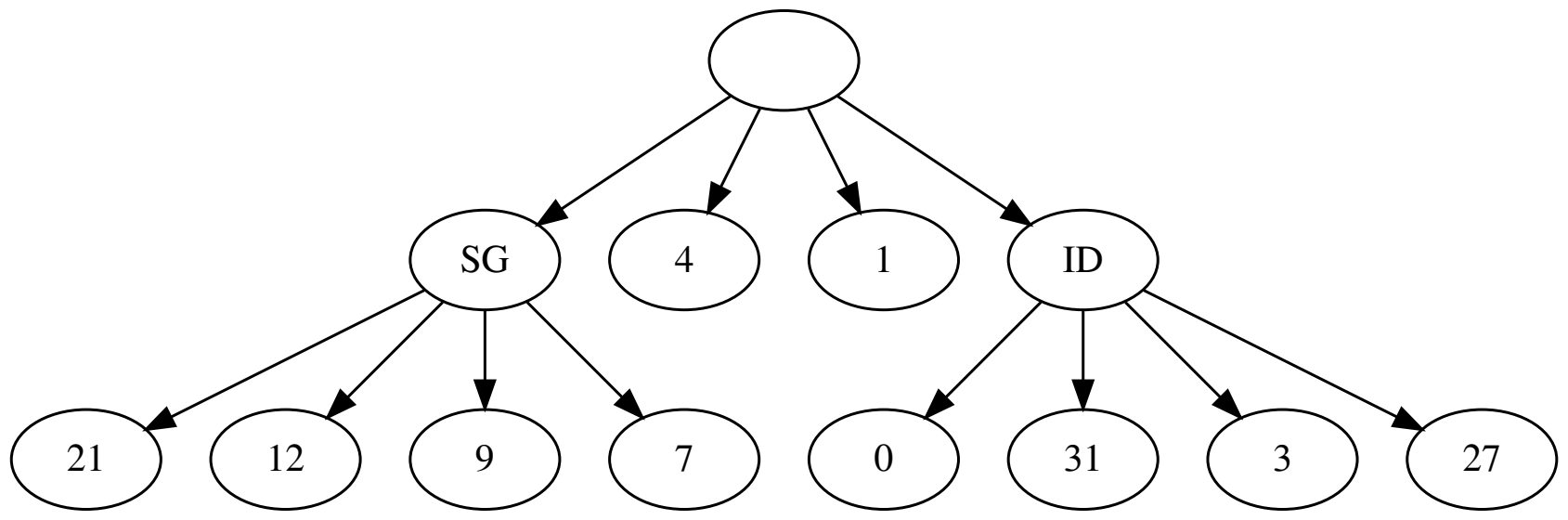


Compression sans perte (2/5)

Que devient l'arbre suivant?



Arbre compressé



Compression sans perte (3/5)

- Si un nœud a tous ses enfants égaux:
  - Donner la valeur au nœud,
  - Supprimer les enfants.
- Remonter jusqu'à la racine.

Écrire le pseudo-code (5min, matrix)

Compression sans perte (3/5)

- Si un nœud a tous ses enfants égaux:
  - Donner la valeur au nœud,
  - Supprimer les enfants.
- Remonter jusqu'à la racine.

Écrire le pseudo-code (5min, matrix)

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

```c
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 < 4; ++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 < 4; ++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 < 4; ++i) {
        if (info != qt->child[i]) {
            return false;
        }
    }
    *val = info;
    return true;
}
```

Compression avec perte (1/5)

Idée générale

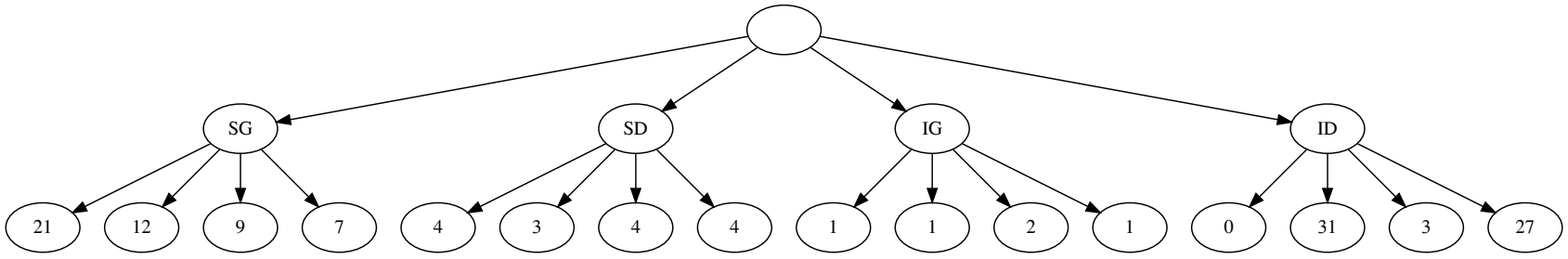
- Regrouper les pixels par valeur sous certaines conditions

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

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

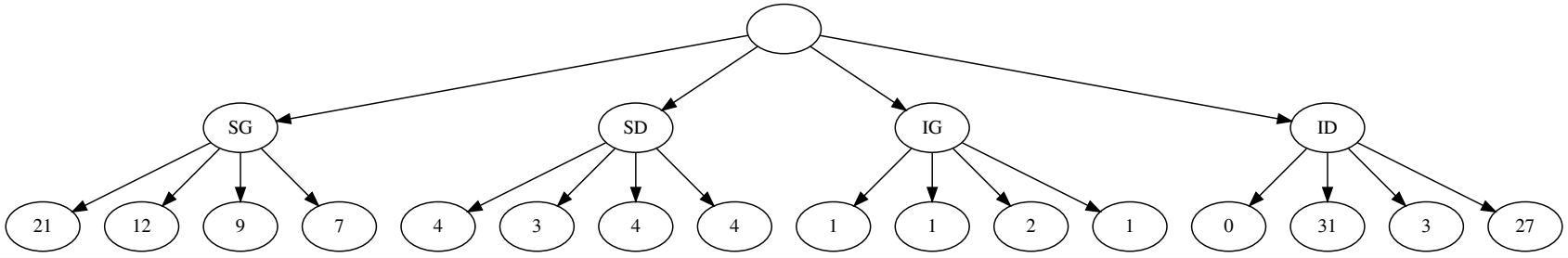
Compression avec perte (2/5)

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

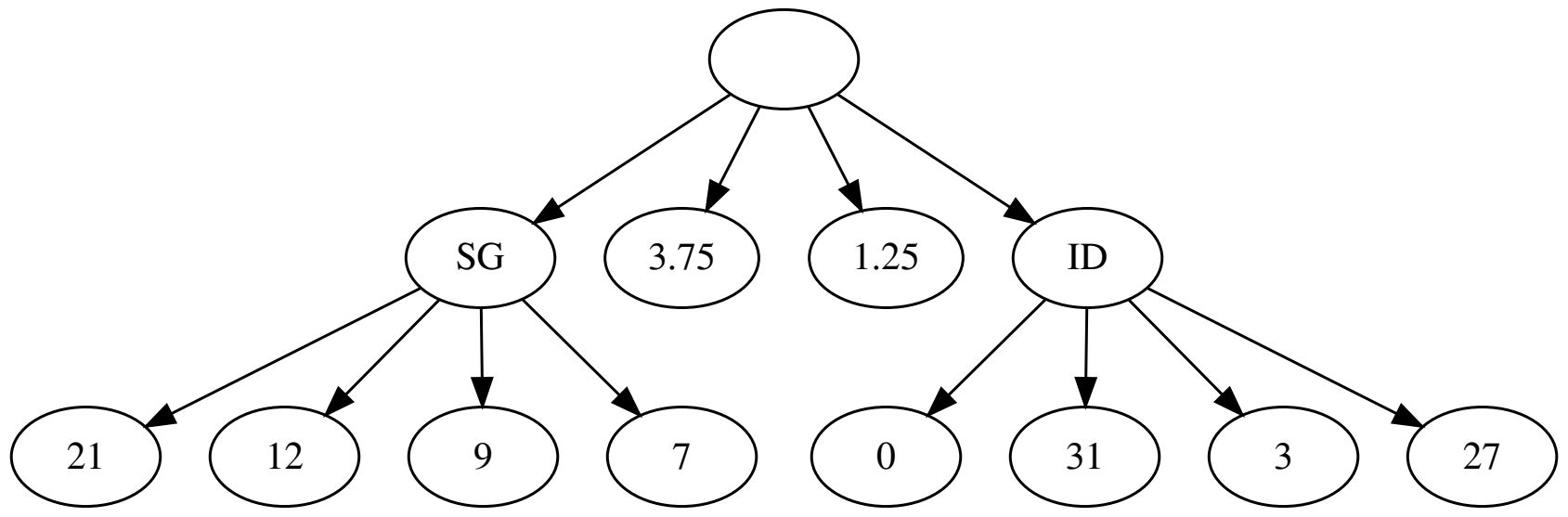


Compression avec perte (2/5)

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 $\sigma < \theta$, 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 $\sigma < \theta$, 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 $\theta = 0.5$?

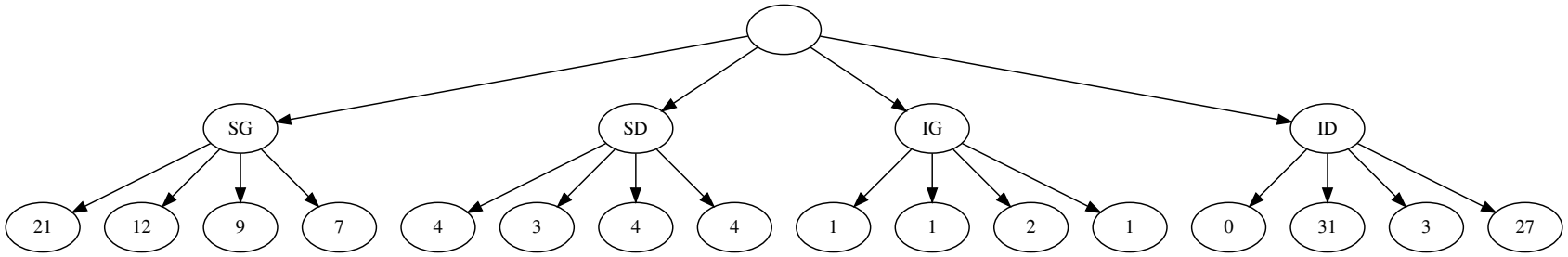


Figure 1: L'arbre original.

Compression avec perte (4/5)

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

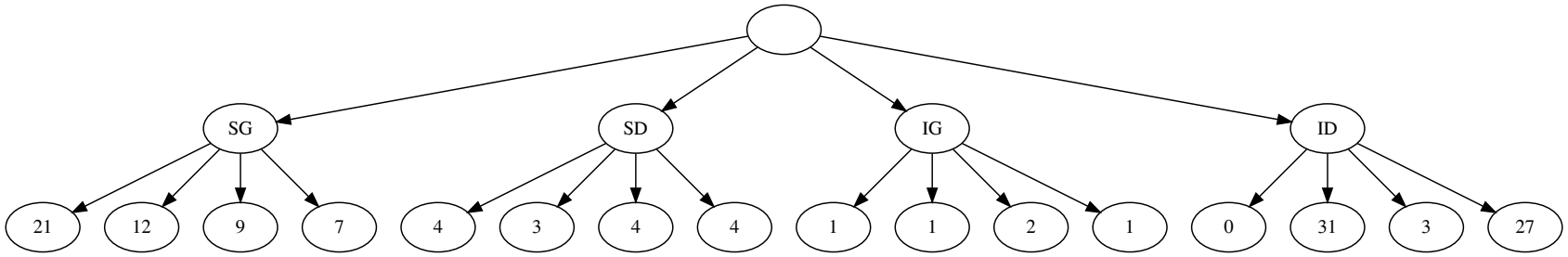


Figure 1: L'arbre original.

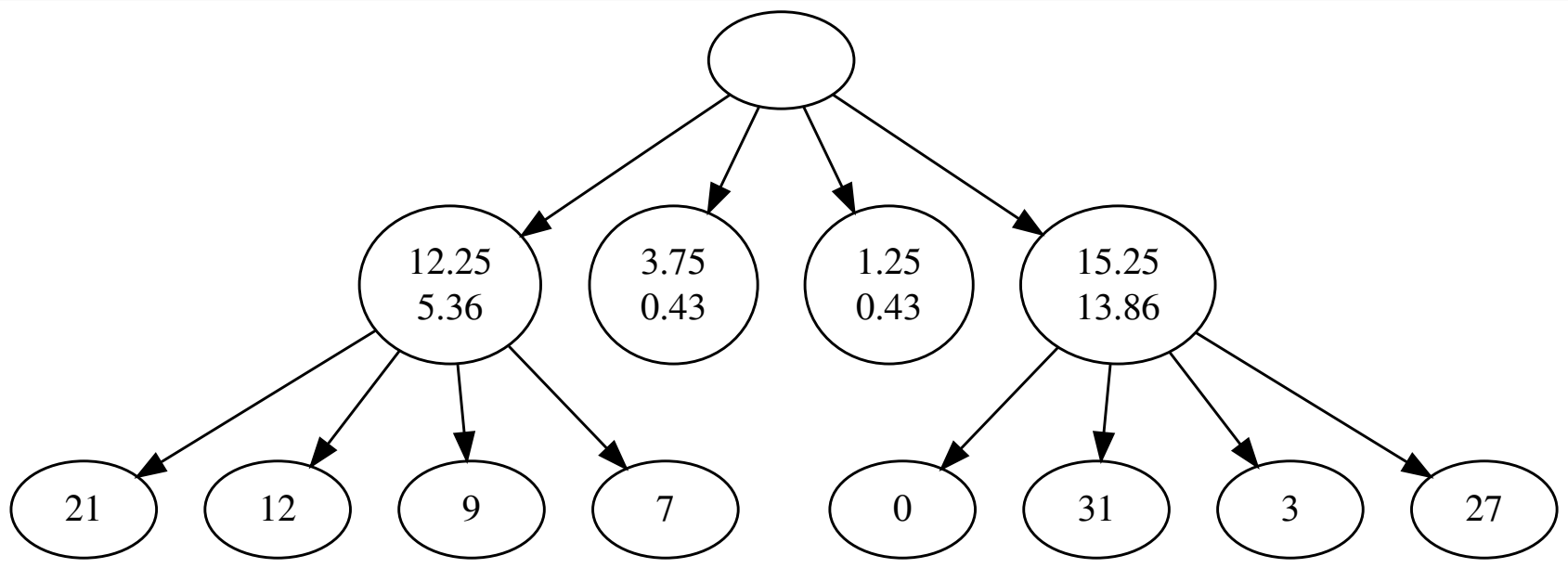


Figure 2: Arbre compressé.

Compression avec perte (5/5)

Modifications sur la structure de données?

Compression avec perte (5/5)

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_sans_pertes(arbre)
```

La dynamique des corps célestes

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 temps pour $N = 1$ on calcule en $1\mu s$:

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.

Question

Comment faire mieux, des idées?

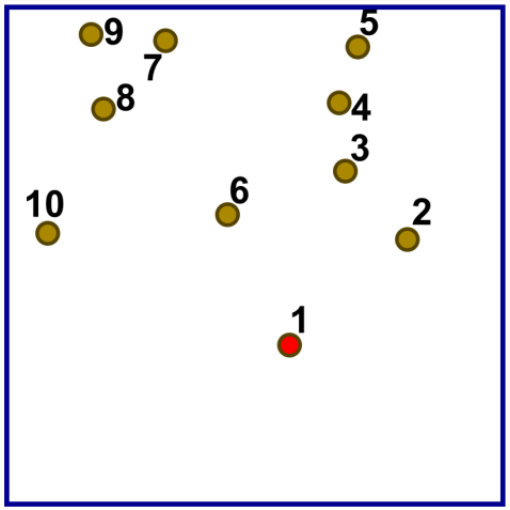
Question

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

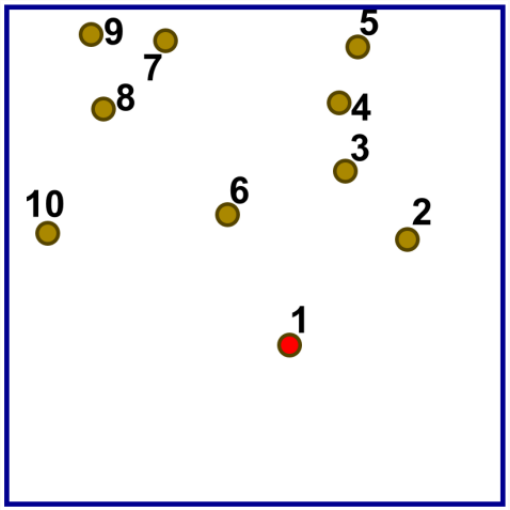


Problématique

- On veut calculer la force sur 1.

Le cas à 10 corps

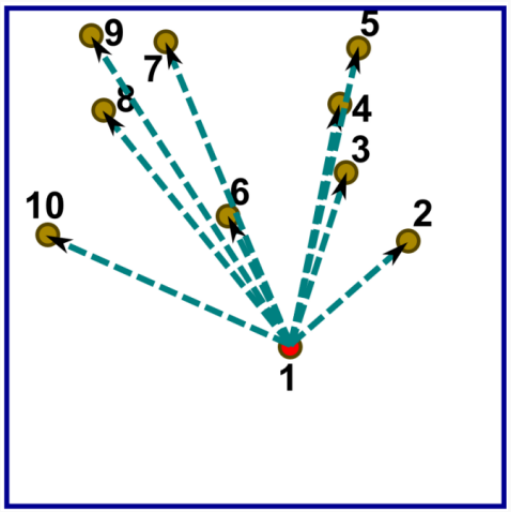
Illustration: le cas à 10 corps



Problématique

- On veut calculer la force sur 1.

Illustration: le cas à 10 corps

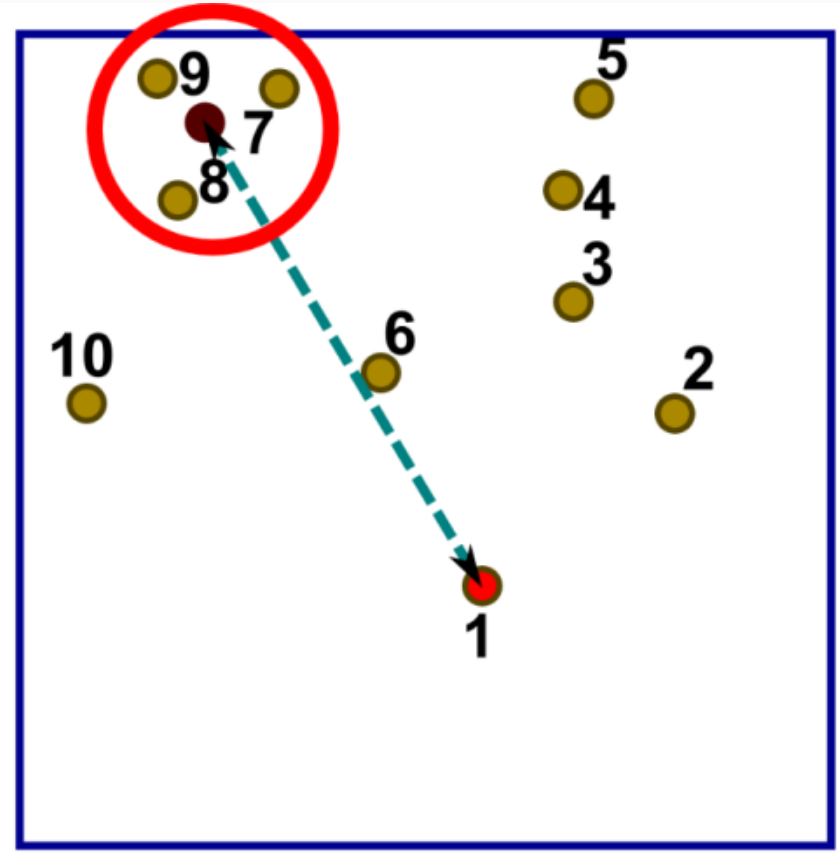


Résultat

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

Le cas à 10 corps

Réduction d'un groupe à un seul corps

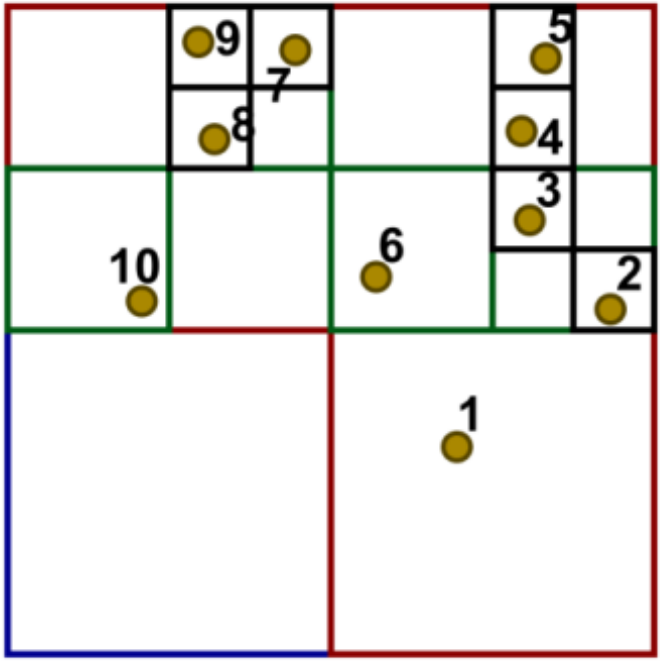


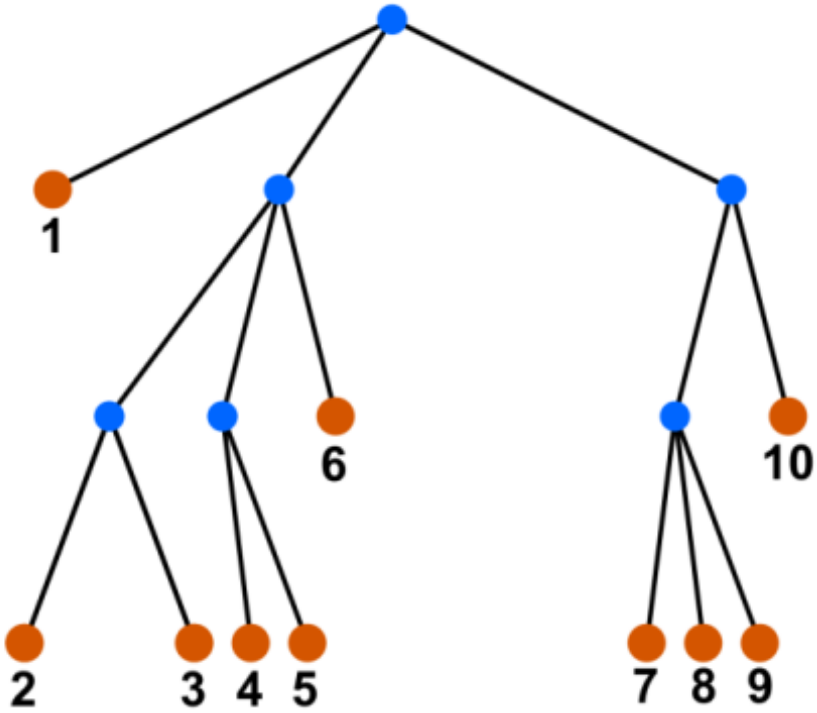
Idée

- 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

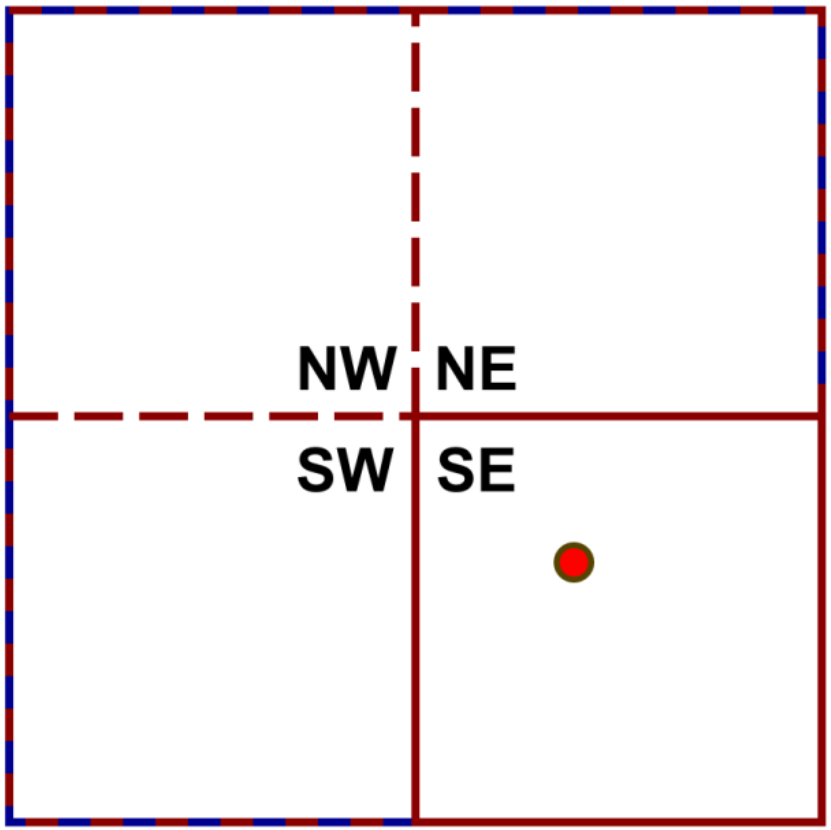




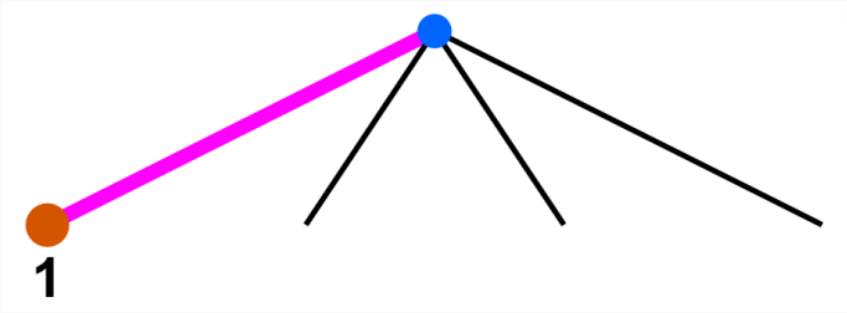
- On omet les nœuds vides pour éviter la surcharge.
- La numérotation est:
  - 0: ID
  - 1: SD
  - 2: IG
  - 3: SG

Exemple d'insertion

Insertion corps 1



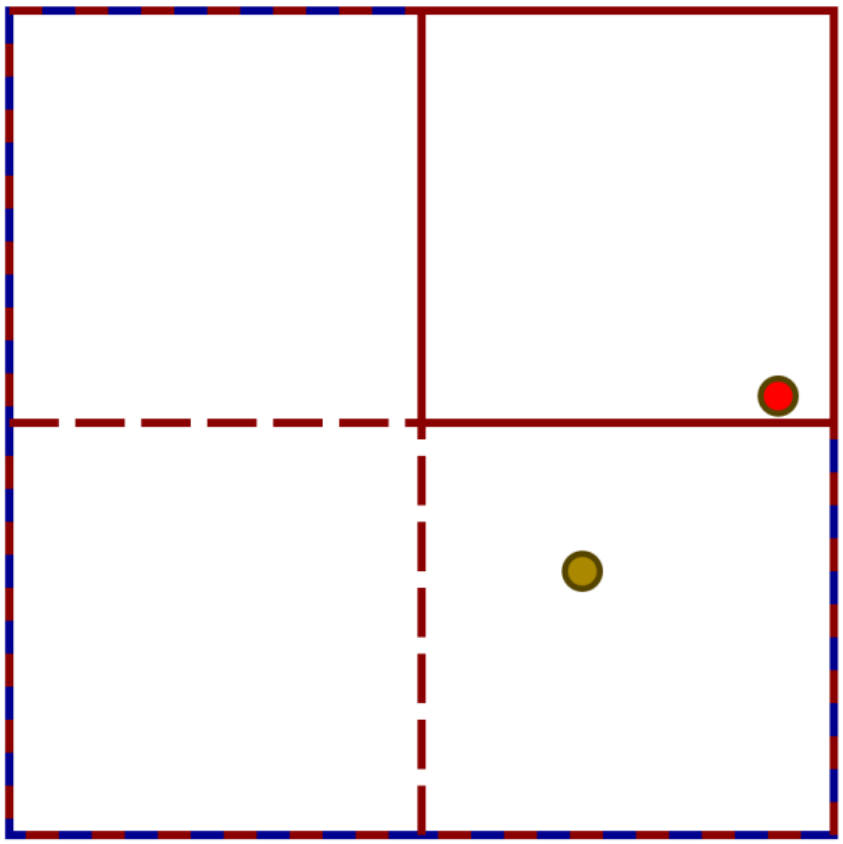
Arbre, niveau 1



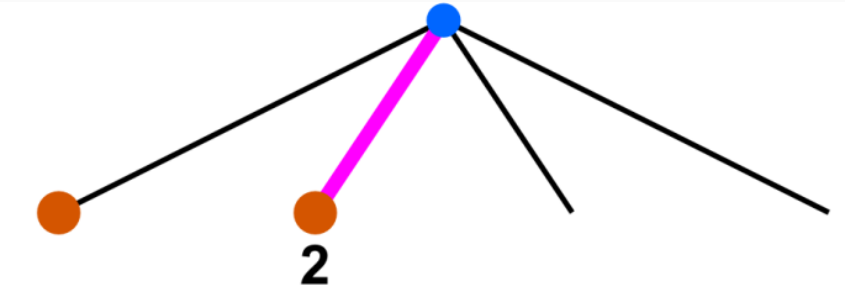
- Quadrant ID.
- La feuille est vide, on insère.

Exemple d'insertion

Insertion corps 2



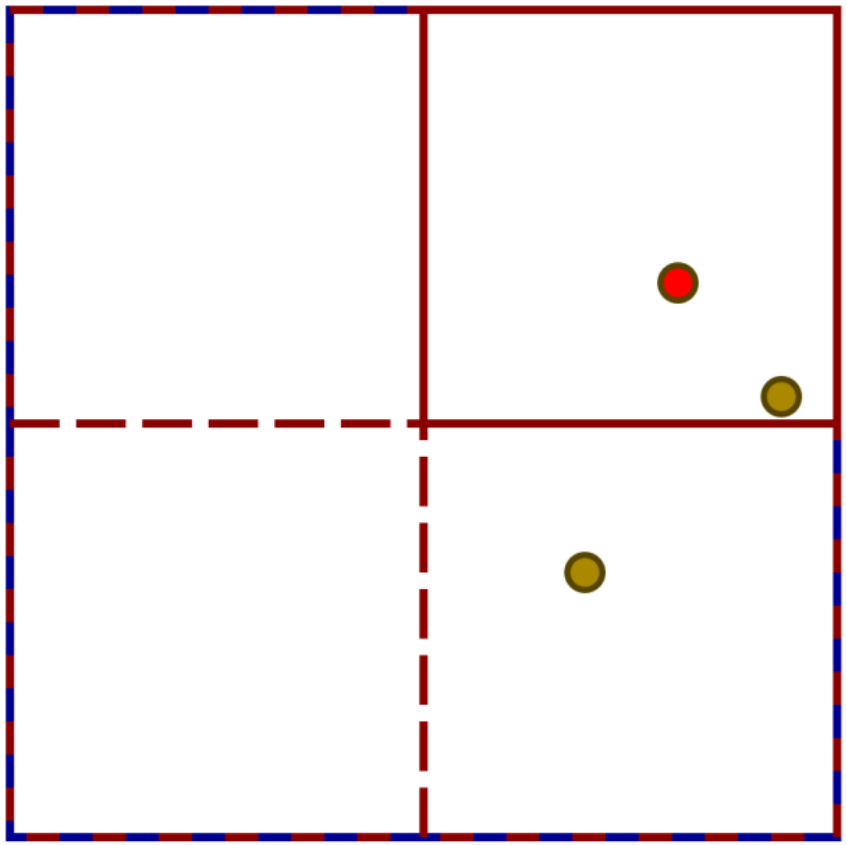
Arbre, niveau 1



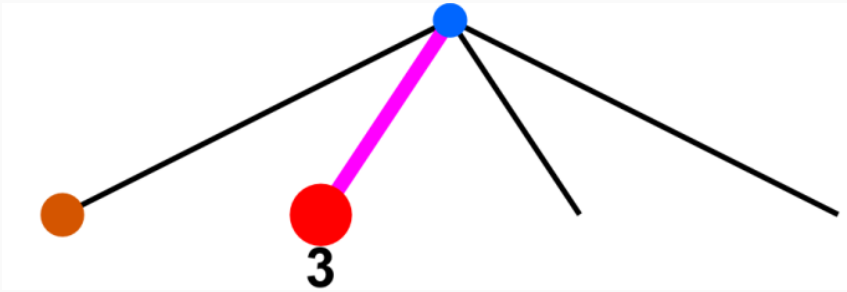
- Quadrant SD.
- La feuille est vide, on insère.

Exemple d'insertion

Insertion corps 3 (1/N)



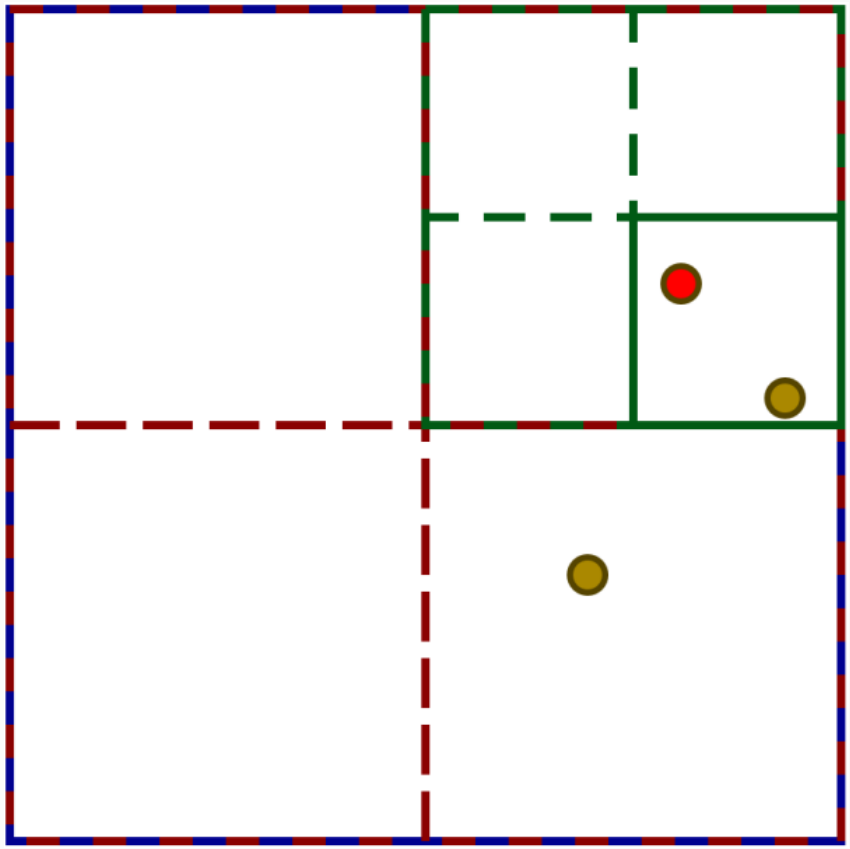
Arbre, niveau 1



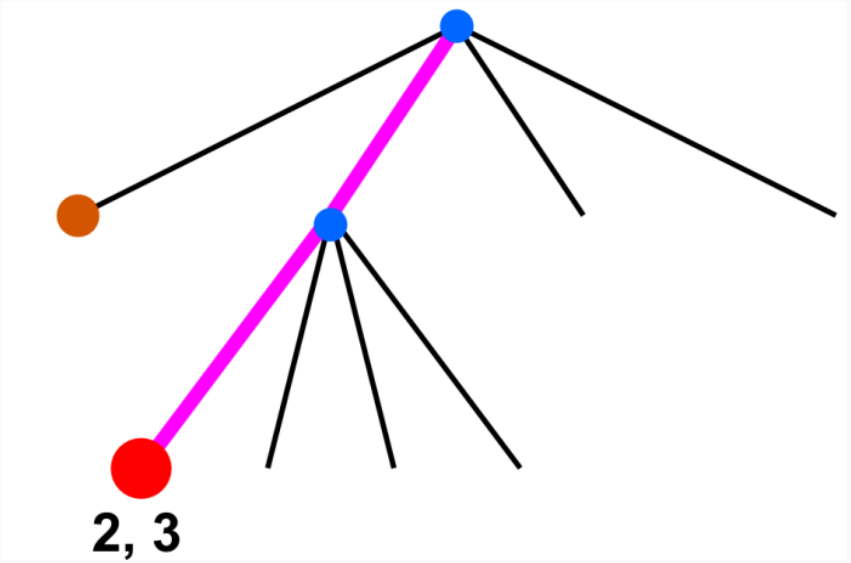
- Quadrant SD.
- La feuille est prise par 2.

Exemple d'insertion

Insertion corps 3 (2/N)



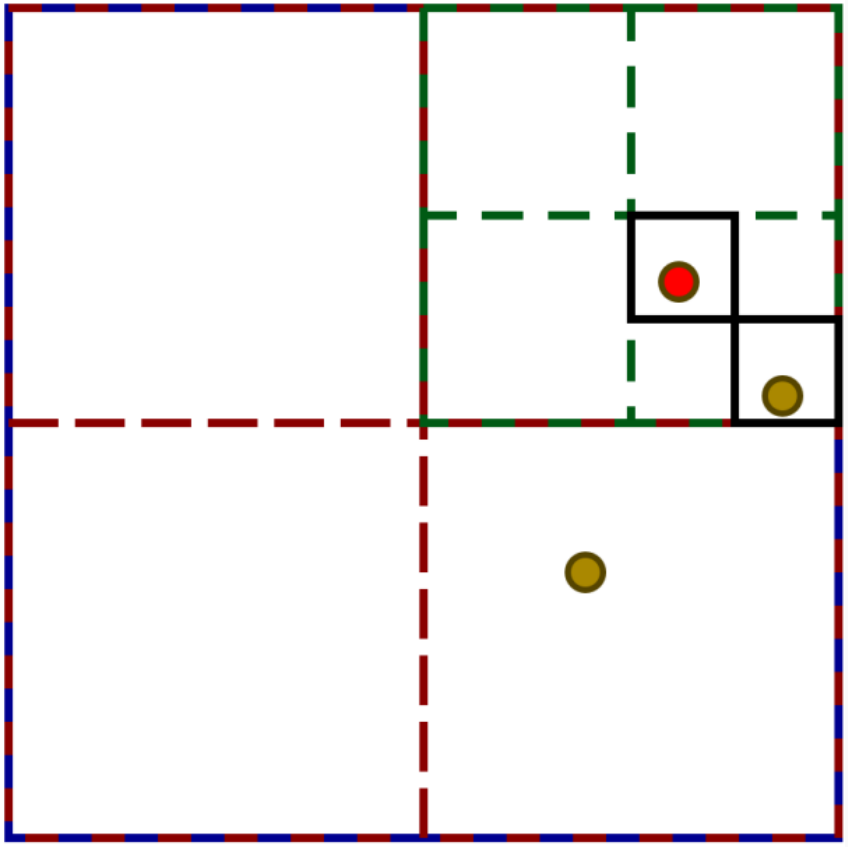
Arbre, niveau 2



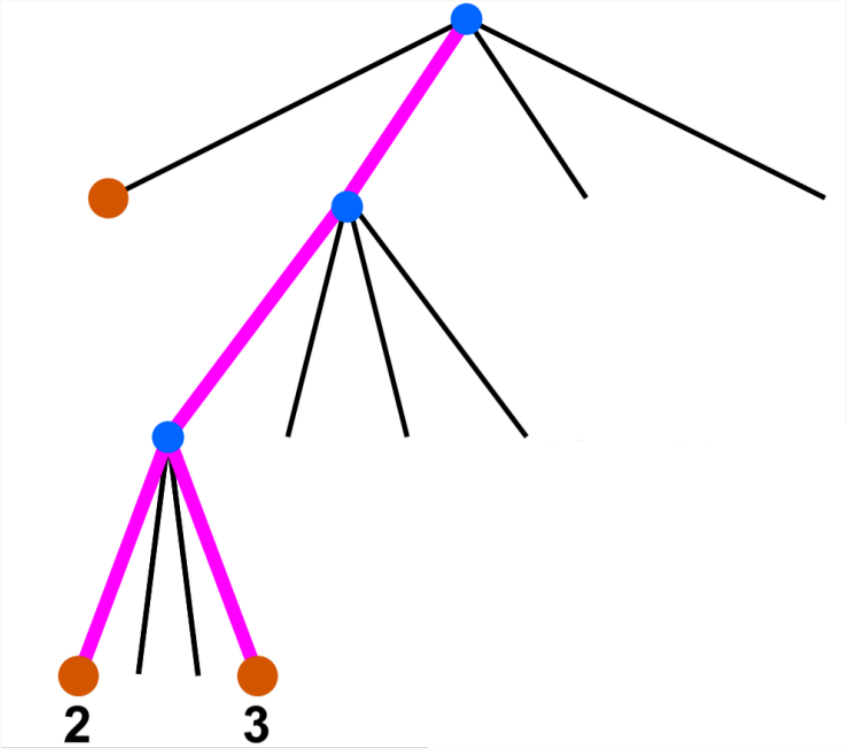
- On crée un nouveau nœud.
- Deux corps dans le nœud ID.
- On crée un nouveau nœud.

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

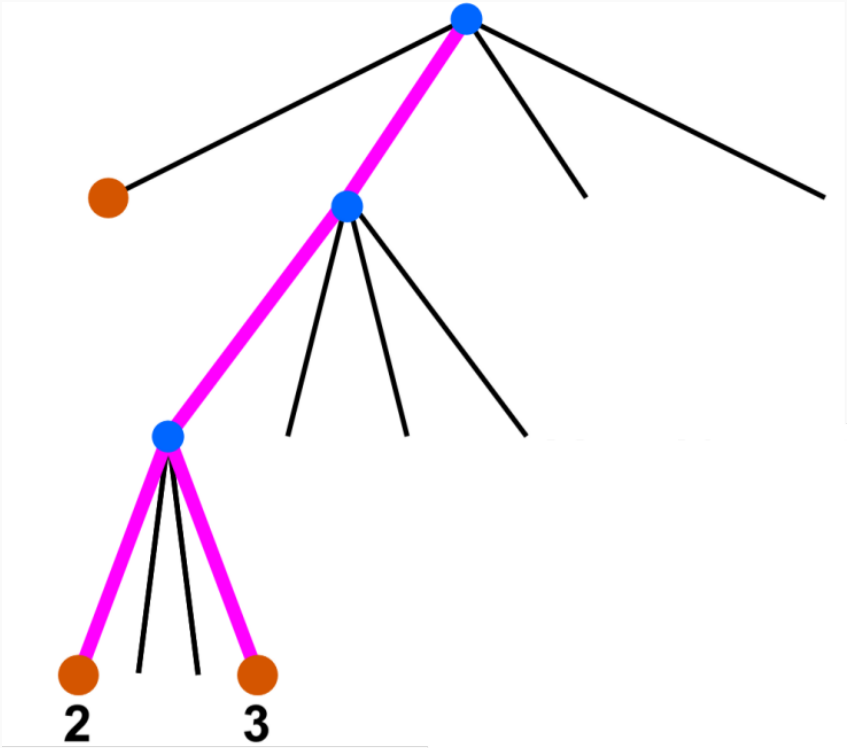
Que fait-on avec les nœuds intérieurs?

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

$$m = m_2 + m_3, \tag{1}$$

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

Arbre



Chaque feuille contient une étoile

Résumé

- 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.