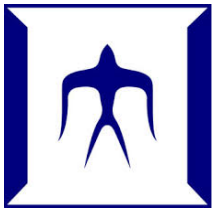
SuperCon2019 本選問題

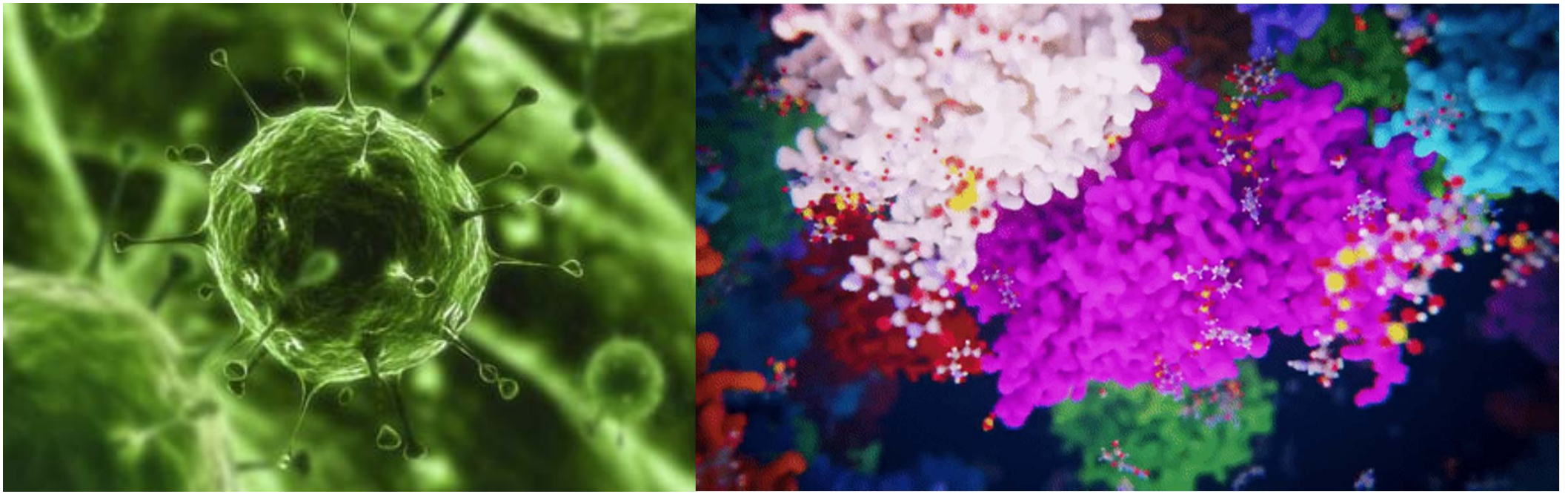
-多体問題の高速解法-

東京工業大学
学術国際情報センター
横田理央

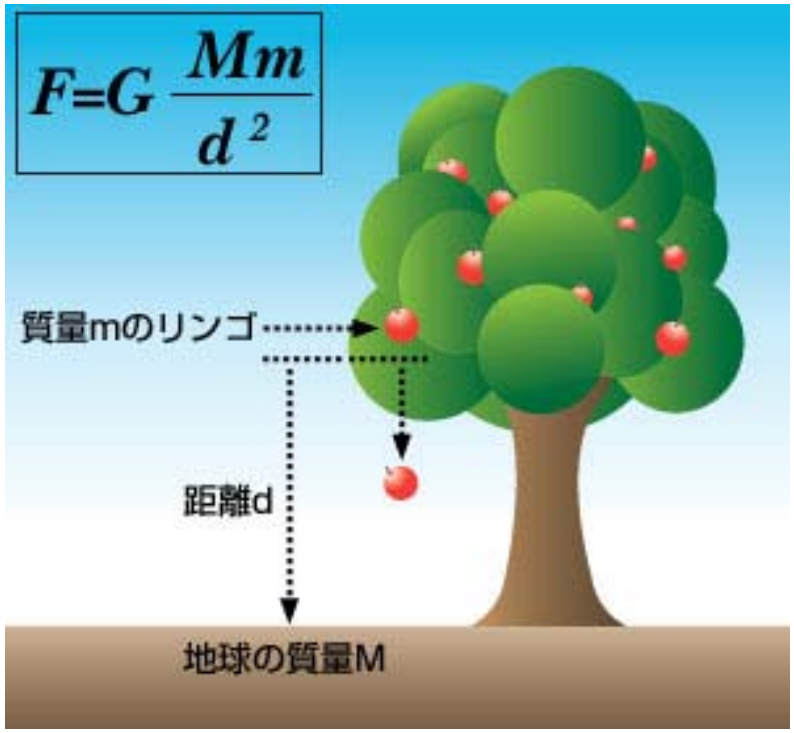


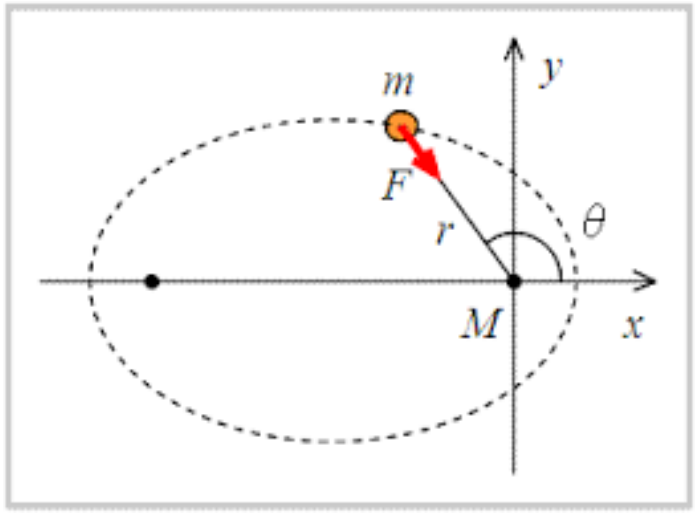
多体問題

多体問題



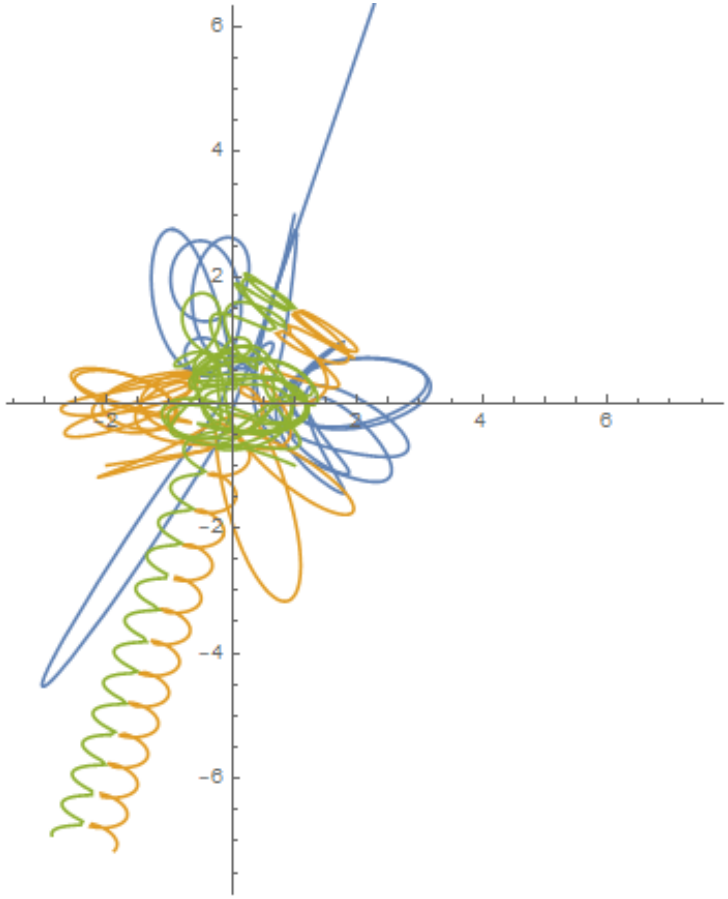
2体問題





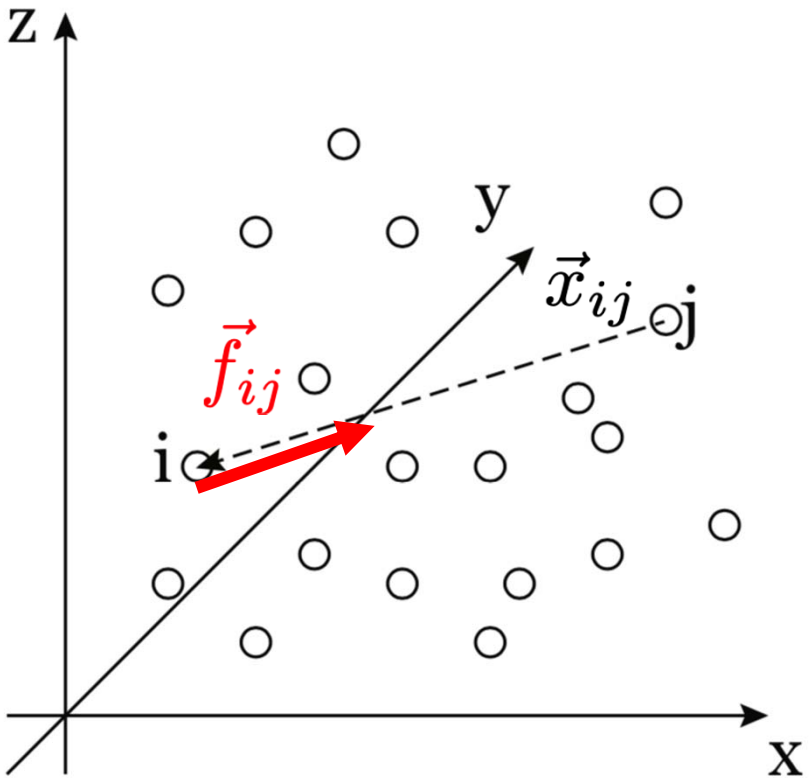
解析的に解ける

3体問題



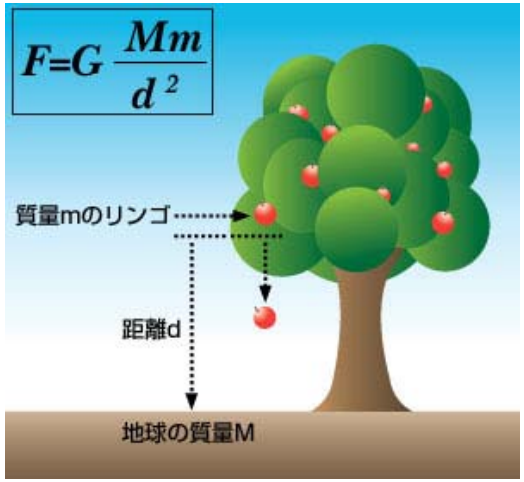
解析的に解けない

2体同士に働く力



万有引力の法則

$$f_{ij} = \frac{G m_i m_j}{|\vec{x}_{ij}|^2}$$



$$\vec{x}_i = \{x_i, y_i, z_i\}$$

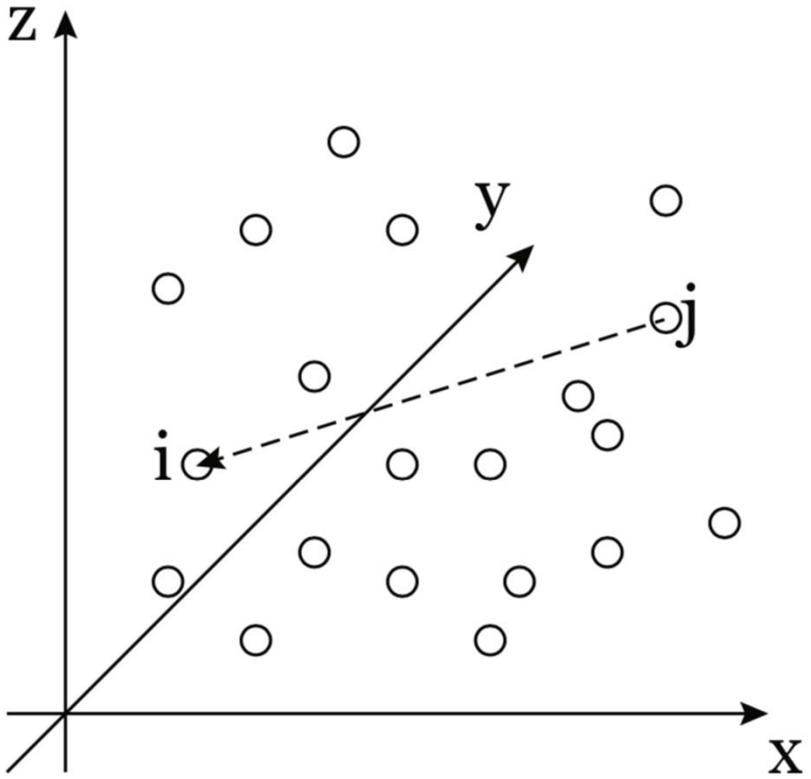
$$\vec{x}_j = \{x_j, y_j, z_j\}$$

$$\vec{x}_{ij} = \vec{x}_i - \vec{x}_j$$

万有引力の法則（ベクトル表記）

$$\vec{f}_{ij} = -\frac{G m_i m_j \vec{x}_{ij}}{|\vec{x}_{ij}|^3}$$

２体同士に働く力



全ての点jから受ける力

$$\vec{f}_i = \sum_{j=1}^{N} \vec{f}_{ij} = \sum_{j=1}^{N} -\frac{G m_i m_j \vec{x}_{ij}}{|\vec{x}_{ij}|^3}$$

ニュートンの運動の第２法則

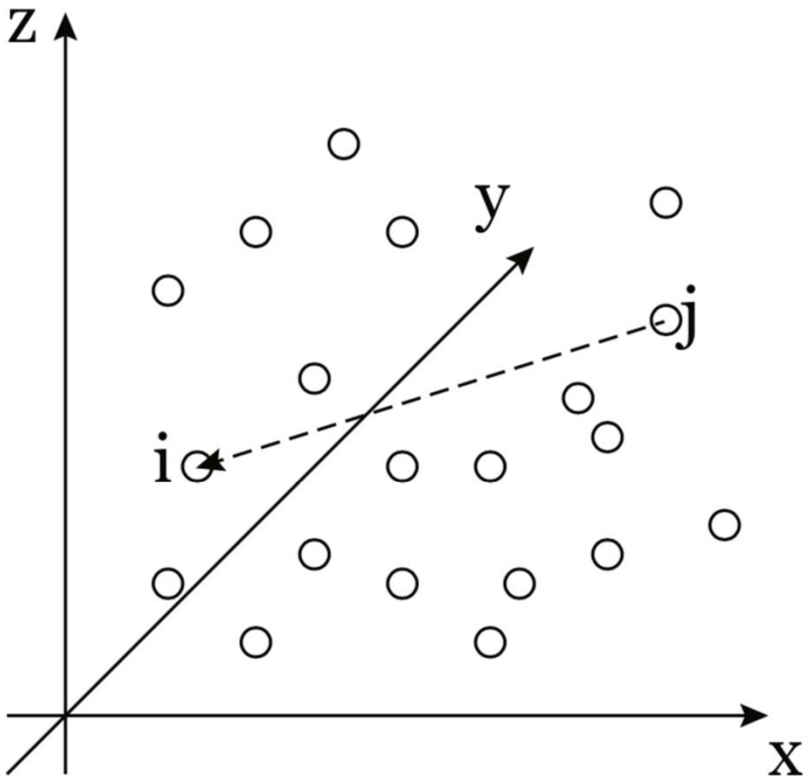
$$\vec{f}_i = m_i \vec{a}_i$$

全ての点jから受ける加速度

$$\vec{a}_i = -\sum_{j=1}^{N} \frac{G m_j \vec{x}_{ij}}{|\vec{x}_{ij}|^3}$$

$$\vec{F}_{ij} = -\frac{G\vec{x}_{ij}}{|\vec{x}_{ij}|^3}$$ とおくと $$\vec{a}_i = \sum_{j=1}^{N} \vec{F}_{ij} m_j$$

物体の運動



加速度は速度の時間微分

$$\vec{a}_i = \frac{d\vec{v}_i}{dt}$$

$$\vec{v}_i(t+h) \approx \vec{v}_i(t) + \vec{a}_i(t)h$$

速度は座標の時間微分

$$\vec{v}_i = \frac{d\vec{x}_i}{dt}$$

$$\vec{x}_i(t+h) \approx \vec{x}_i(t) + \vec{v}_i(t+h)h$$

多体問題の近似解法

テイラー展開

$$f(x+h) = f(x) + \frac{f'(x)}{1!}h + \frac{f''(x)}{2!}h^2 + \dots$$

p+1項目で打ち切ると

$$f(x+h) = \sum_{n=0}^{p} \frac{f^{(n)}(x)}{n!}h^n$$

以下の変換を行う

$$f = F_{ij}$$
$$x = x_{ic}$$
$$h = x_{cj}$$

ただし

$$x_{cj} \ll x_{ic}$$

簡単のため1次元で考えてみる

$$F_{ij} = F(x_{ij})$$
$$= F(x_{ic} + x_{cj})$$
$$= \sum_{n=0}^{p} \frac{F^{(n)}(x_{ic})}{n!}x_{cj}^n$$

多体問題の近似解法

加速度の式（1次元）

$$a_i = \sum_{j=1}^{N} F_{ij} m_j$$

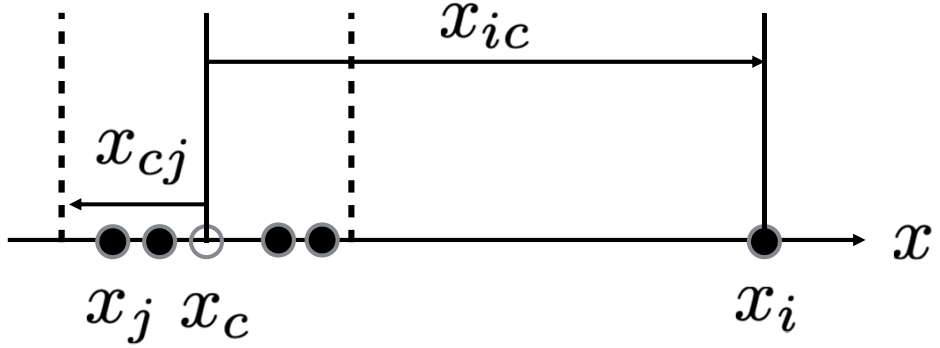
N x N回

$$\begin{aligned} F_{ij} &= F(x_{ij}) \\ &= F(x_{ic} + x_{cj}) \\ &= \sum_{n=0}^{p} \frac{F^{(n)}(x_{ic})}{n!} x_{cj}^n \end{aligned}$$

テイラー展開を用いると

$$a_i = \sum_{j=1}^{N} \sum_{n=0}^{p} \frac{F^{(n)}(x_{ic})}{n!} x_{cj}^n m_j$$

$x_{cj} \ll x_{ic}$



iと関係ない部分は1回だけ計算

$$M_n = \sum_{j=1}^{N} \frac{x_{cj}^n m_j}{n!}$$

N x p回

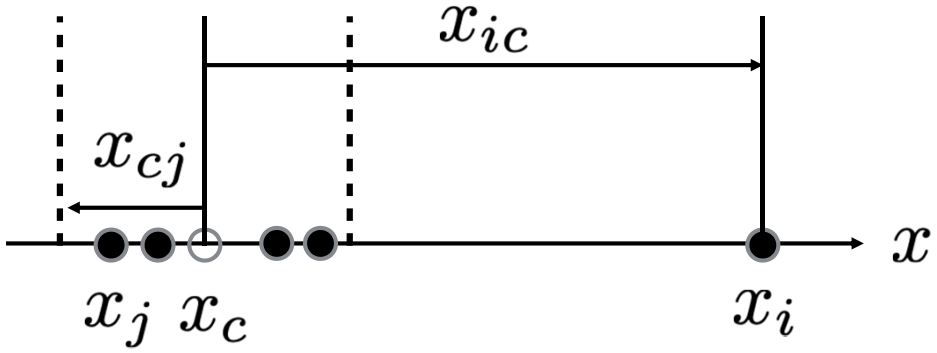
$$a_i = \sum_{n=0}^{p} F_{ic}^{(n)} M_n$$

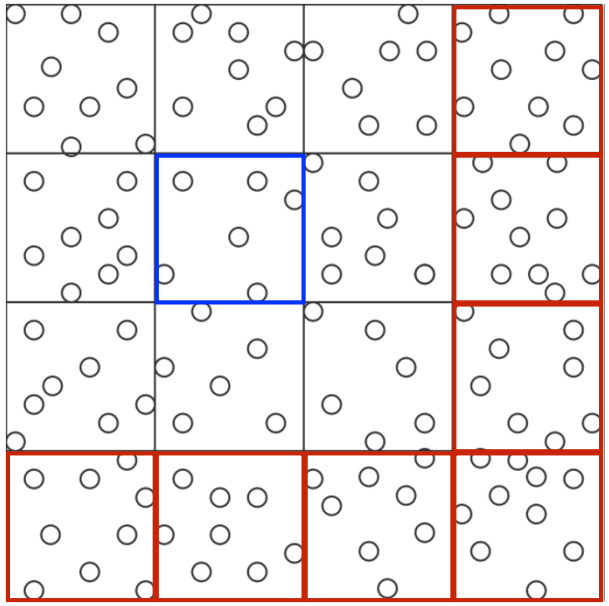
N x p回

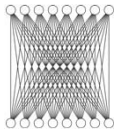
多体問題の近似解法

この近似はjとcの距離がiとcの距離に比べて近いときにしか成り立たない

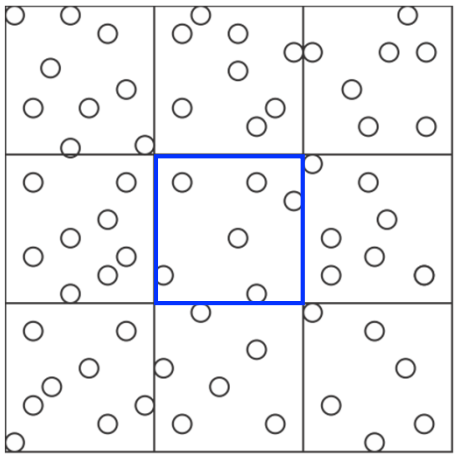
$$x_{cj} \ll x_{ic}$$





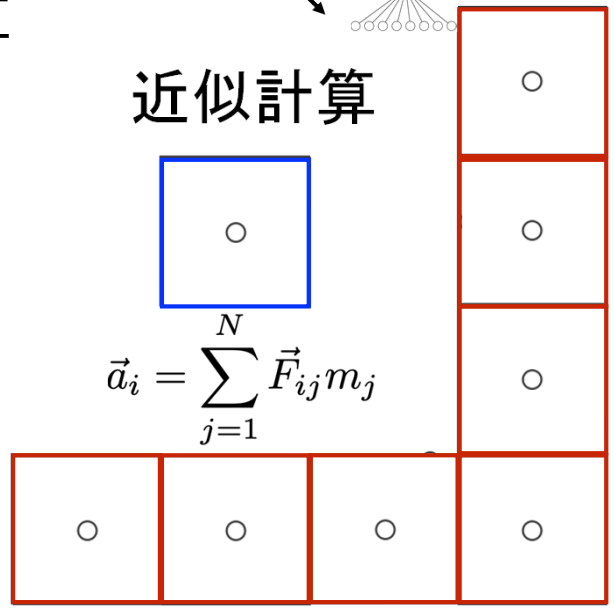


直接計算



$$\vec{a}_i = -\sum_{j=1}^{N} \frac{G m_j \vec{x}_{ij}}{|\vec{x}_{ij}|^3}$$

近似計算



$$\vec{a}_i = \sum_{j=1}^{N} \vec{F}_{ij} m_j$$

領域を分割して場合分け

多体問題の近似解法

粒子が増えてくると箱をもっと
細かく分割することで近傍で

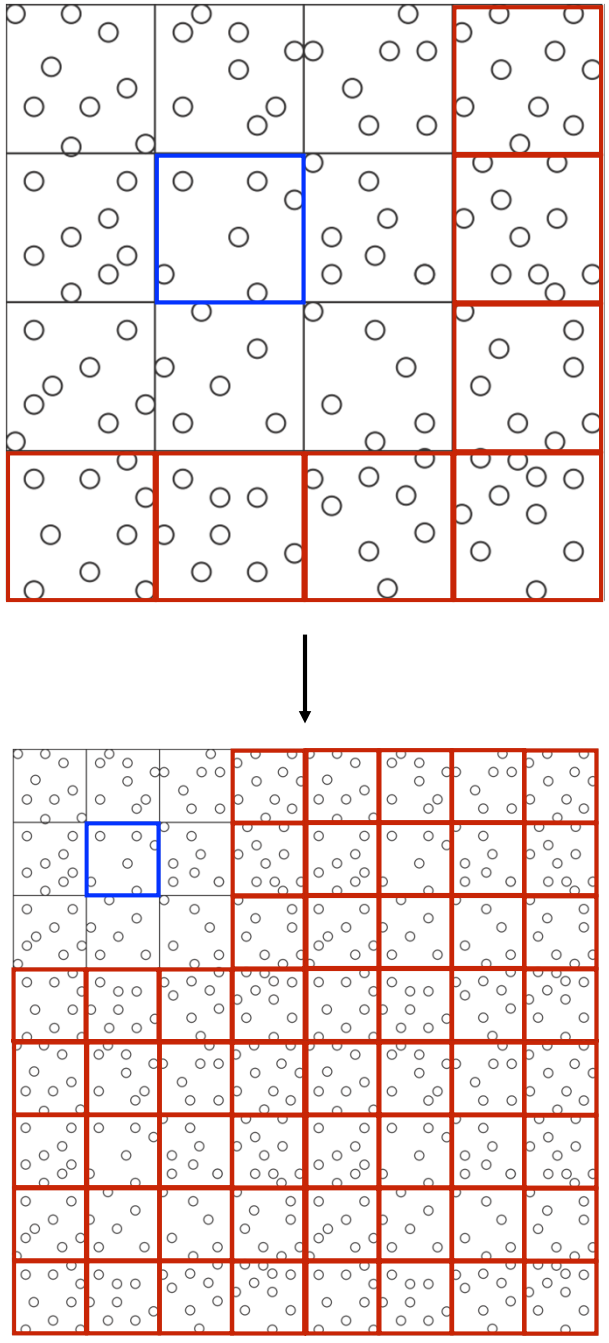
$$\vec{a}_i = -\sum_{j=1}^{N} \frac{G m_j \vec{x}_{ij}}{|\vec{x}_{ij}|^3}$$

の直接計算を行う
部分を一定にすることができる

ただし，このままだと

$$\vec{a}_i = \sum_{j=1}^{N} \vec{F}_{ij} m_j$$

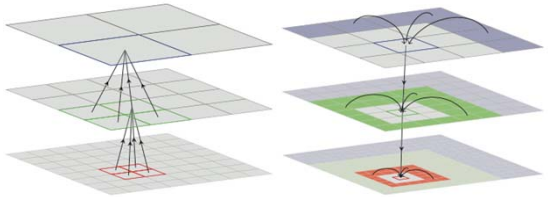
の近似計算を行う部分が粒子数
に比例して増えてしまう



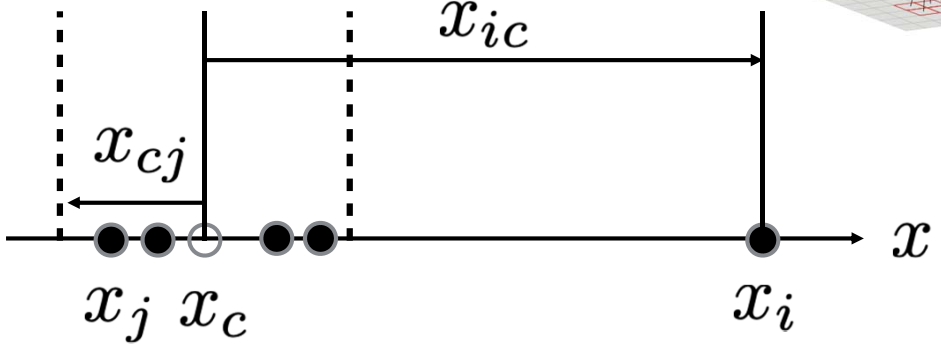
多体問題の近似解法

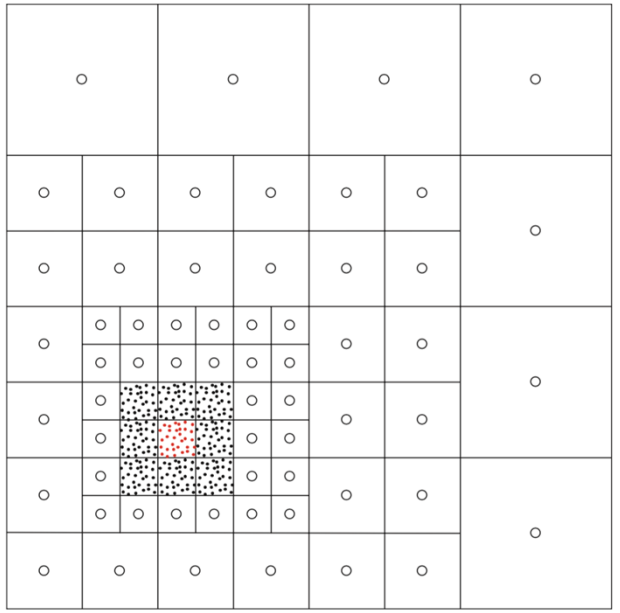
赤い点がi, 黒い点がj, 白丸がc

木構造を使うともっと速くできる

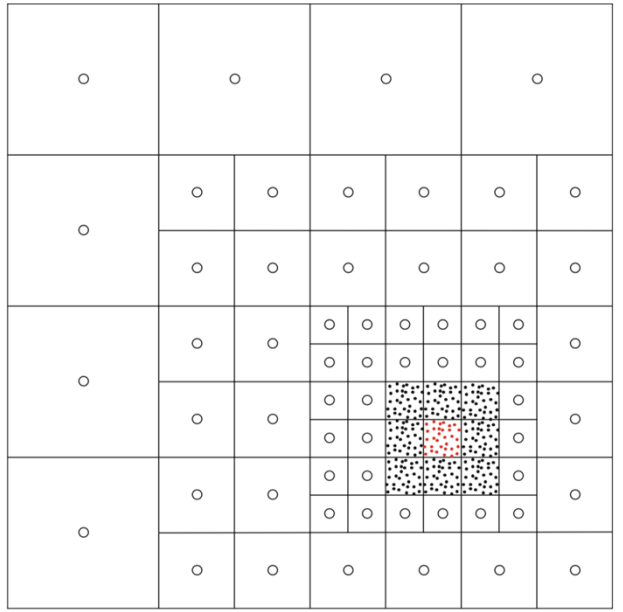


$x_{cj} \ll x_{ic}$





x_c は箱の中心に置く



木構造を使って空間を分割する

木の全ての節点で $M_n = \sum_{j=1}^{N} \frac{x_{cj}^n m_j}{n!}$ を計算しておく

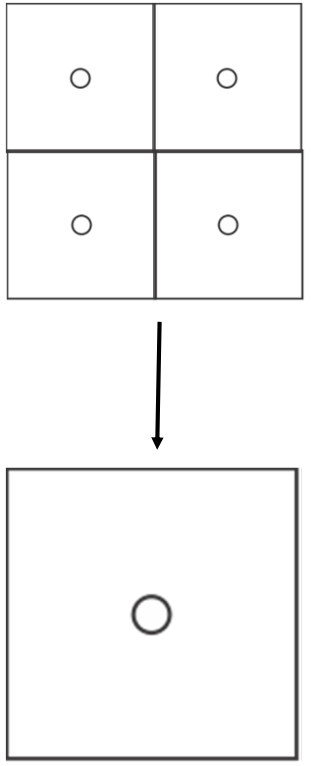
iが変わるたびに異なる M_nを使って

$a_i = \sum_{n=0}^{p} F_{ic}^{(n)} M_n$を計算

多体問題の近似解法

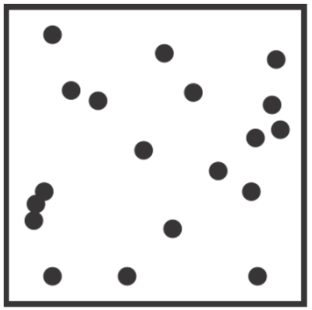
小さい箱のMnから大きい箱のMnを計算する方法

小さい箱の中心をx_c，大きな箱の中心をx_dとする

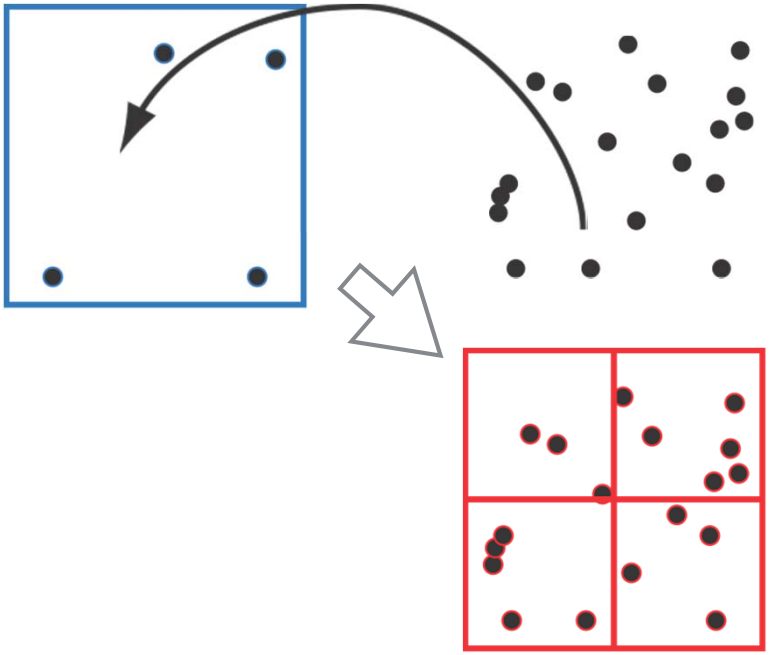


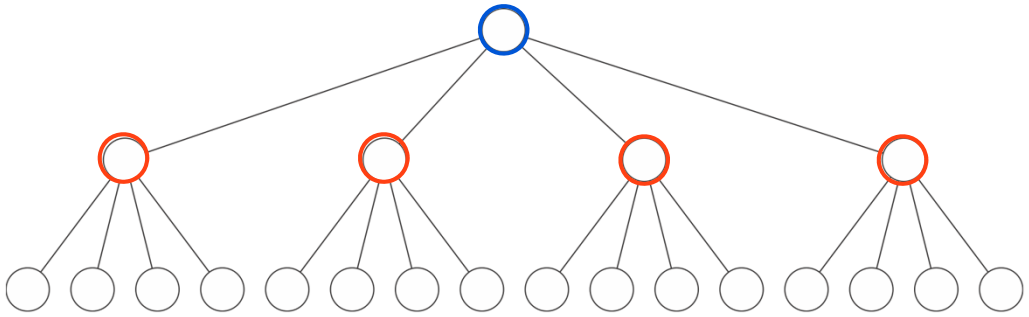
$$
\begin{aligned}
M_n(x_d) &= \sum_{j=1}^{N} \frac{1}{n!} x_{dj}^n m_j \\
&= \sum_{j=1}^{N} \frac{1}{n!} (x_{dc} + x_{cj})^n m_j \\
&= \sum_{j=1}^{N} \frac{1}{n!} \sum_{k=0}^{n} \frac{n!}{(n-k)!k!} x_{dc}^{n-k} x_{cj}^k m_j \qquad \text{二項定理} \\
&= \sum_{k=0}^{n} \frac{1}{(n-k)!} x_{dc}^{n-k} \sum_{j=1}^{N} \frac{1}{k!} x_{cj}^k m_j \qquad \text{jに関係ない部分を前に} \\
&= \sum_{k=0}^{n} \frac{1}{(n-k)!} x_{dc}^{n-k} \sum_{j=1}^{N} M_k(x_{cj})
\end{aligned}
$$

木構造の構築



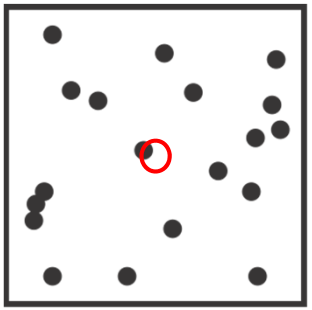
1. 全ての粒子の座標の最小値，最大値を計算し全体の領域を木構造の出発点とする
2. 領域に順番に粒子を入れていき一定数以上になった箱は分割する
3. 2.を繰り返すことで粒子の密度が高いところは深くなるような木構造ができあがる





木構造の構築

4. 木構造の下から順番に $M_n = \sum_{j=1}^{N} \frac{x_{cj}^n m_j}{n!}$ を計算していく
5. この際，小さな箱の M_n から大きな箱の M_n を計算する方法がある
6. 木構造の全ての節点について M_n が計算できたら完成

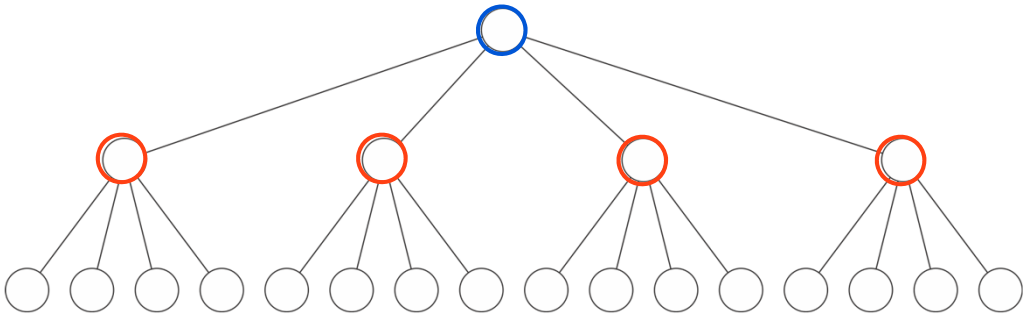


箱の中心にcを置く

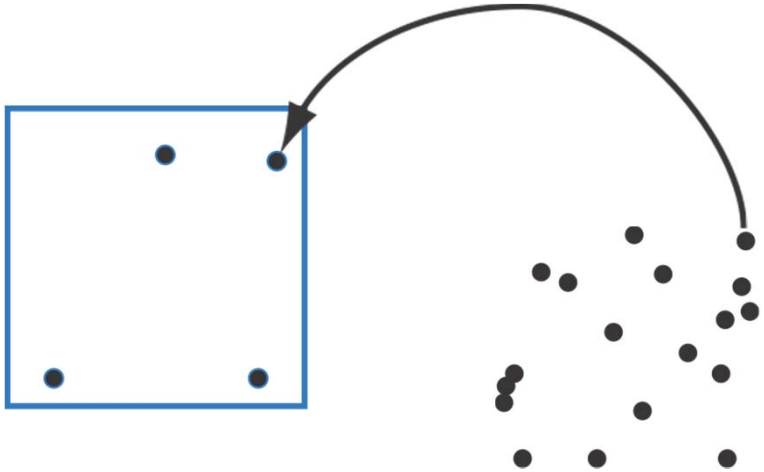
ヒント：このような構造体を使うと便利

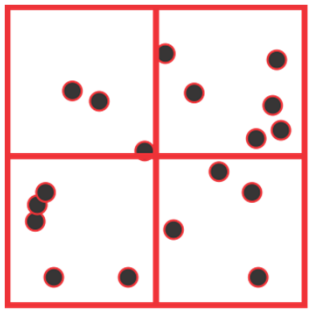
```
struct Node {
  Node *parent;
  Node *child[8];
  double M[P+1];
  その箱に含まれる全ての粒子の情報{x,y,z,m}
}
```

木構造の構築



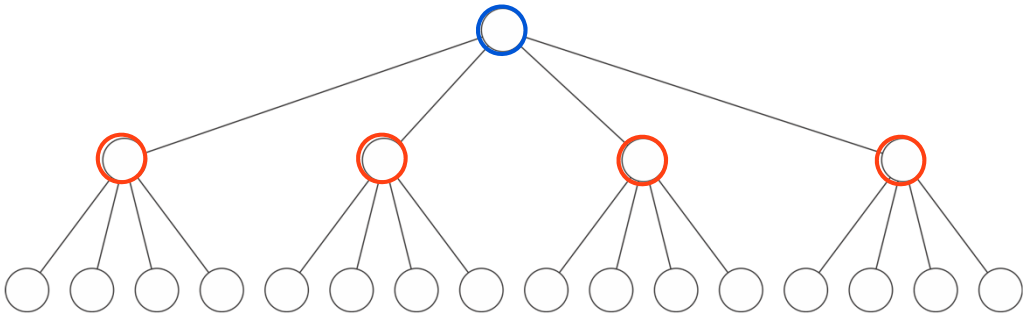
```
Node  {
  Node *parent;
  Node *child[8];
  double M[P+1];
}
```

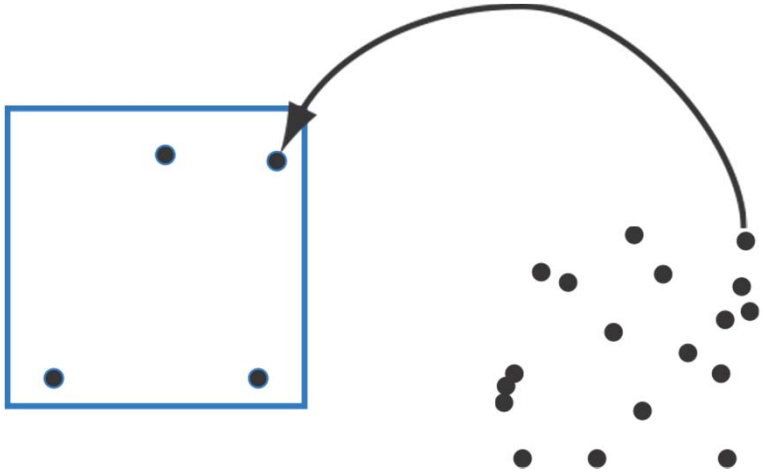


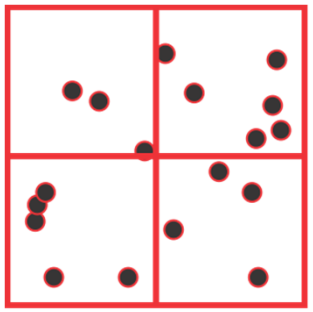


```
Node node[N];
node[0].child[0]=&node[1];
node[0].child[1]=&node[2];
                :
                :
node[1].parent=&node[0];
node[2].parent=&node[0];
                :
                :
```

木構造の構築







分割する前に粒子をソート
第何象限にいるかでソート

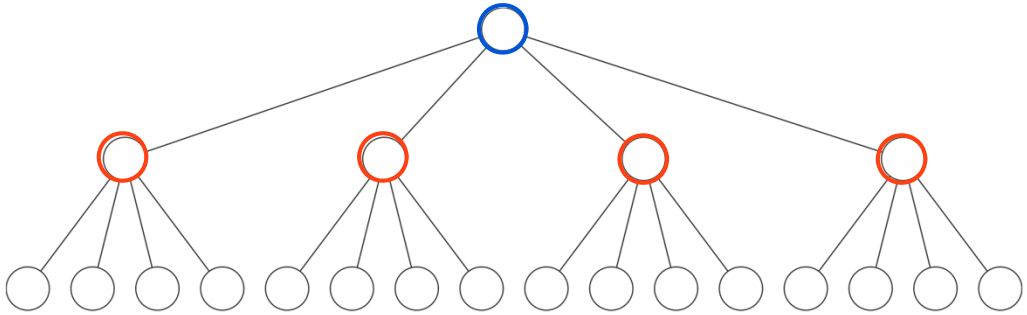
↓

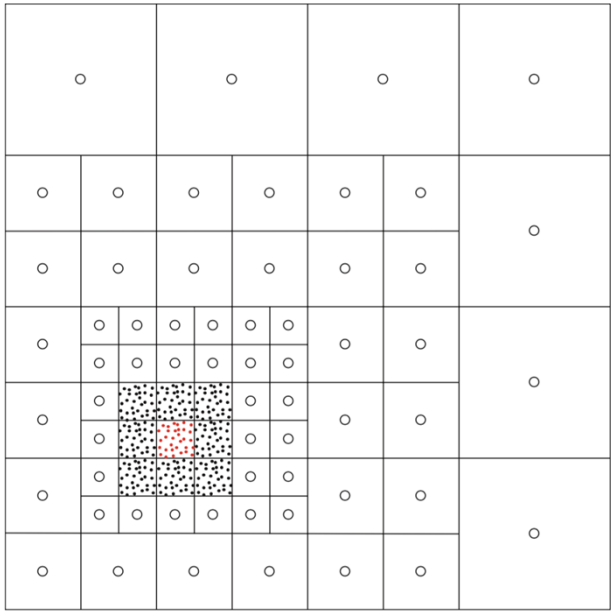
分割した後に粒子が連続になる
粒子番号の始点と終点を持っておく

```
Node  {
  Node *parent;
  Node *child[8];
  double M[P+1];
  int begin;
  int end;
}
```

相互作用の計算

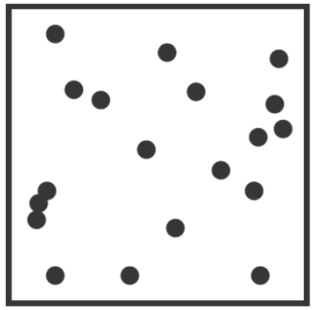
1. 木構造の末端の節点を一つづつ見ていき，それぞれに対して相互作用リストを作る
2. 木構造を辿っていき，末端の節点との距離が箱の大きさに対して十分大きければ近似計算を行い，そうでなければ木構造をさらに辿る
3. 末端まで辿っても距離が十分大きくならなければ粒子同士の直接計算を行う

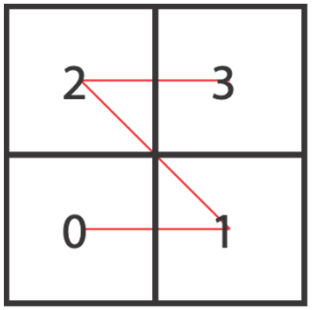


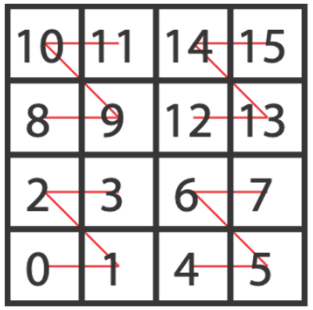


Z階数曲線を利用した木構造

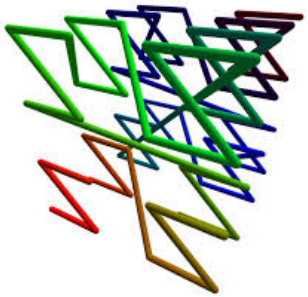
2次元







3次元



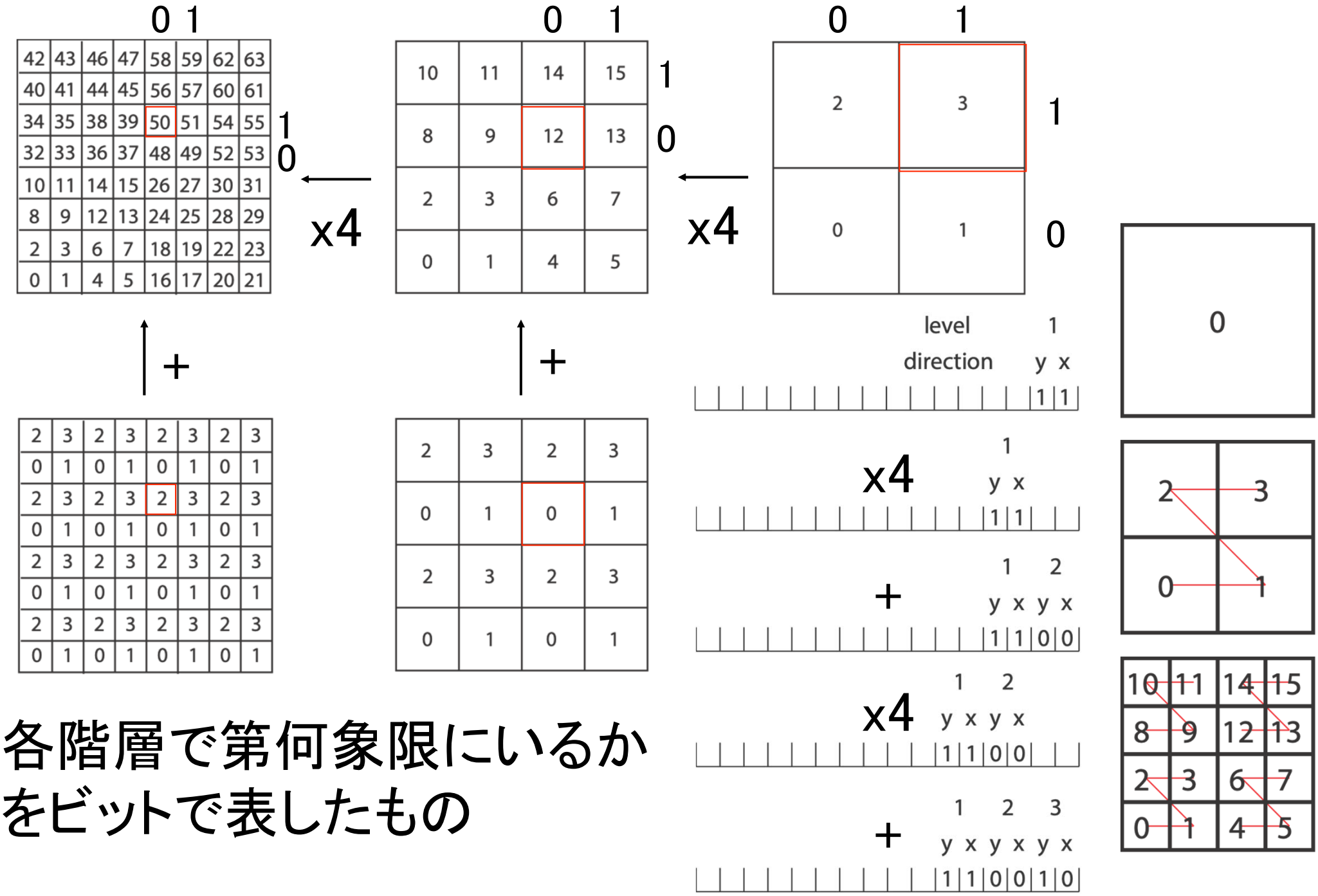
int parent = i/4;
int child[4] = i*4+c;
c=0,1,2,3

int parent = i/8;
int child[8] = i*8+c;
c=0,1,2,3,4,5,6,7

- Z階数曲線は3次元空間に一筆書きで番号を振る
- 木構造を節点同士のリンクで持つ必要がない

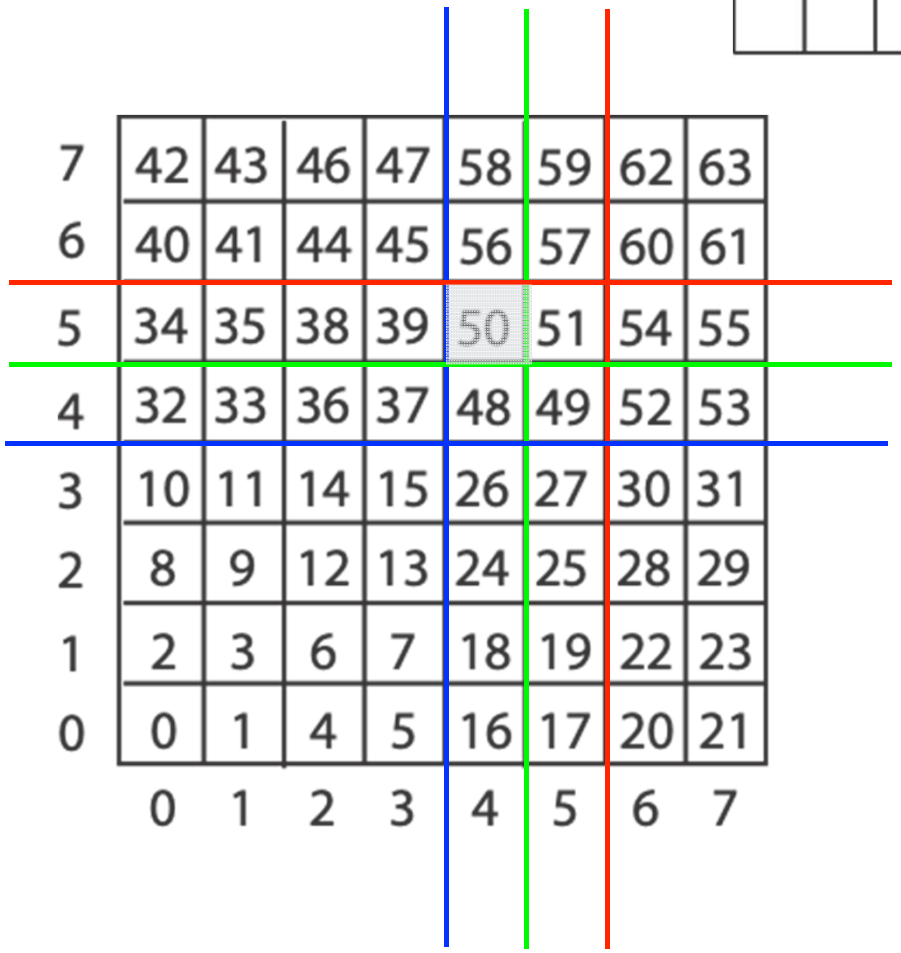
（四則演算で親や子を参照できる）
（階層ごとのオフセットを足した番号を使うことで
全階層の箱に固有の番号を振ることができる）

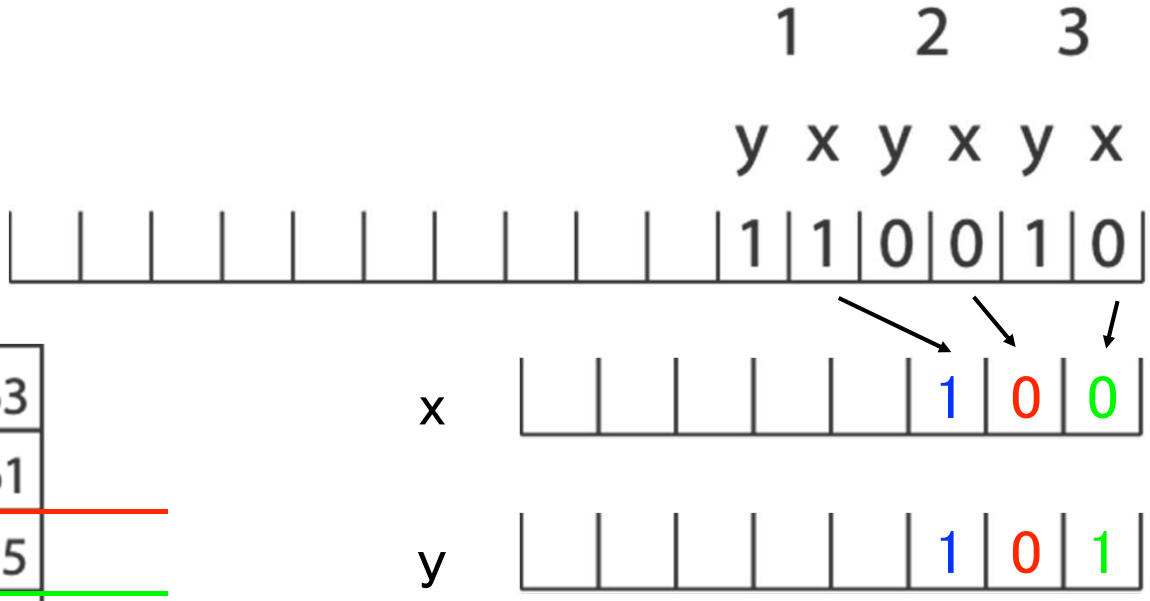
2次元のZ階数曲線



各階層で第何象限にいるか
をビットで表したもの

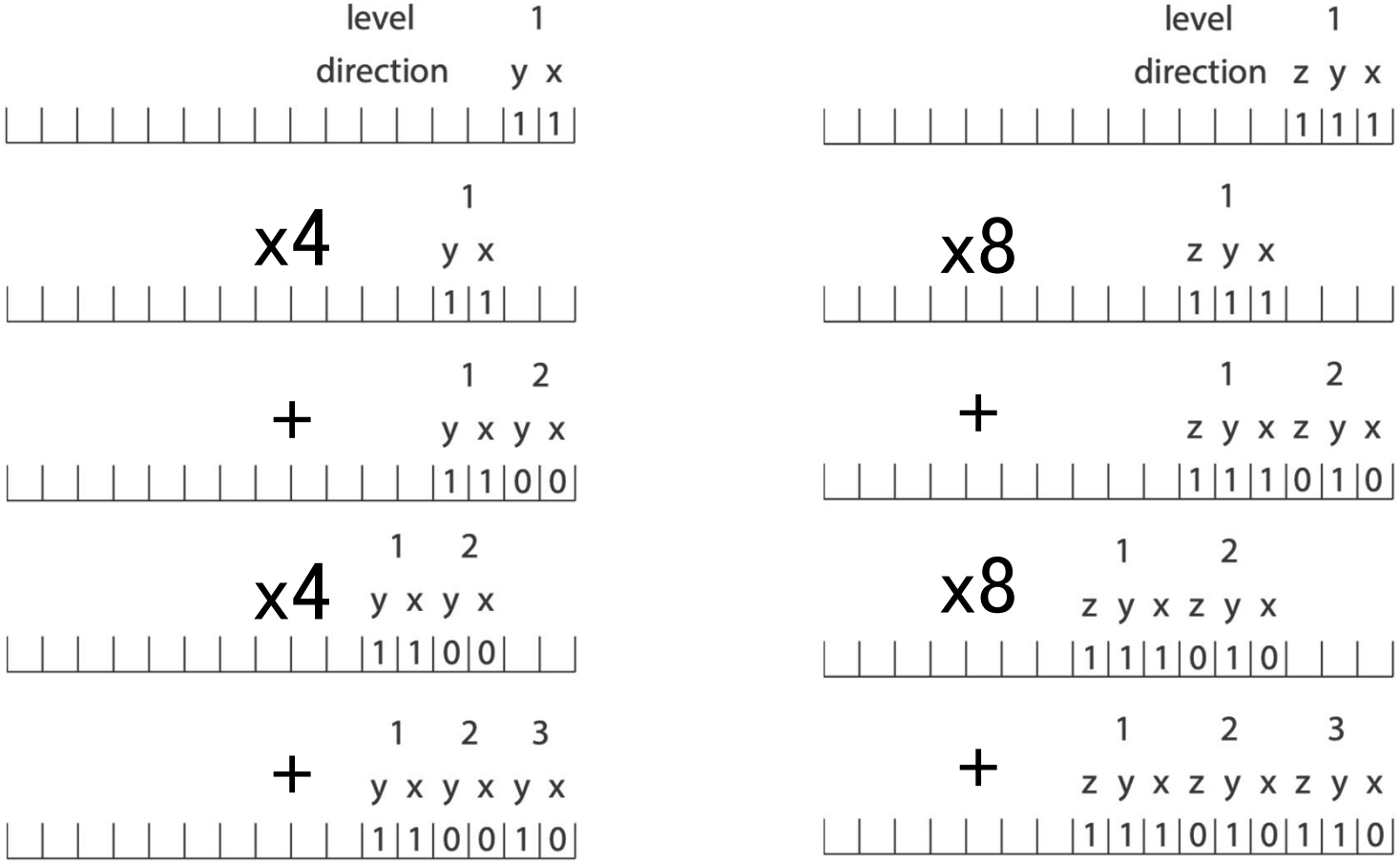
曲線の番号から2次元の番地への変換





50 ⟶ (4,5)

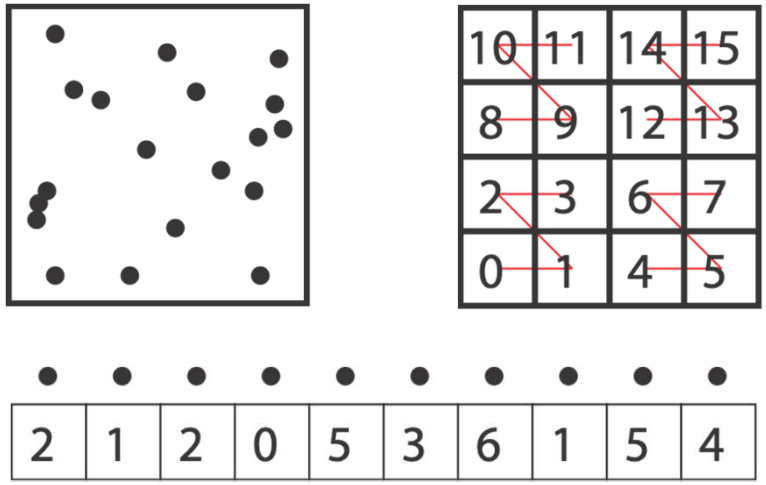
3次元では



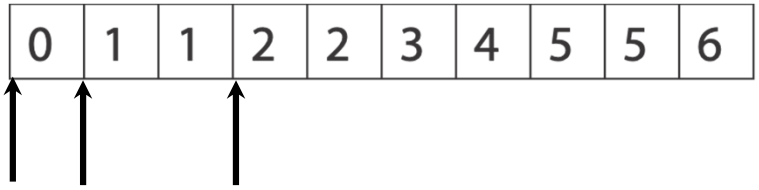
(ix,iy,iz) ——変換——→ i

(ix,iy,iz) ←——逆変換—— i

Z階数曲線を利用した木構造



sort

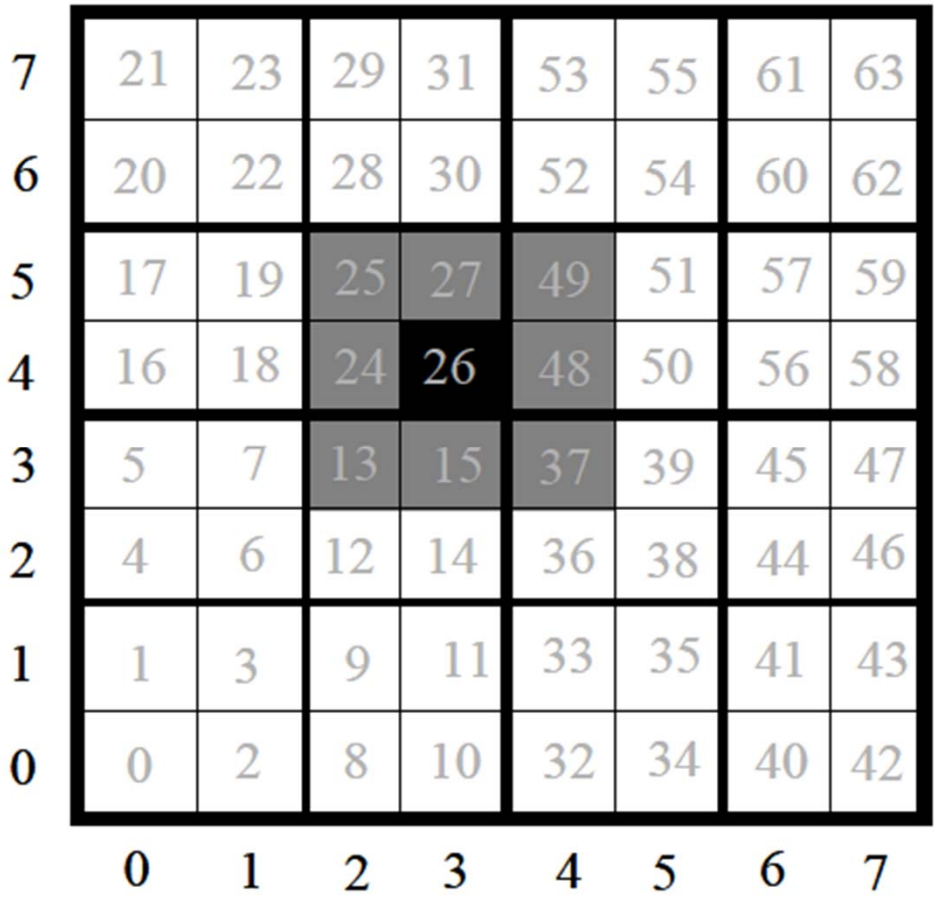



```
offset[0]=0;
offset[1]=1;
offset[2]=3;

int parent = i/8;
int child[8] = i*8+c;
for c=0:7
```

1. 全ての粒子の座標の最小値，最大値を計算する
2. 領域を8分割してZ階数曲線に従って粒子に番号を振る
3. 粒子の情報{x,y,z,m}を番号をキーとしてソート
4. 同じ番号の粒子が閾値以上の場合その領域をまた8分割（異なる階層で同じ番号にならないようにする工夫が必要）
5. 2,3,4を繰り返すことで木構造が構築できる

Z階数曲線を利用した近傍探索



26の近傍を探す

2次元の番地に変換 26→(3,4)

番地が近い
ものを求める

(2,5),(3,5),(4,5)
(2,4),(3,4),(4,4)
(2,3),(3,3),(4,3)

↓

曲線の番号に
逆変換

25,27,49
24,26,48
13,15,37

3次元の場合も同様の手順で求められる

Z階数曲線を利用した相互作用リスト

parent = i/4;

neighbor(parent);

child[c] = i*4+c;

nonneighbor(child[c]);

親

↓

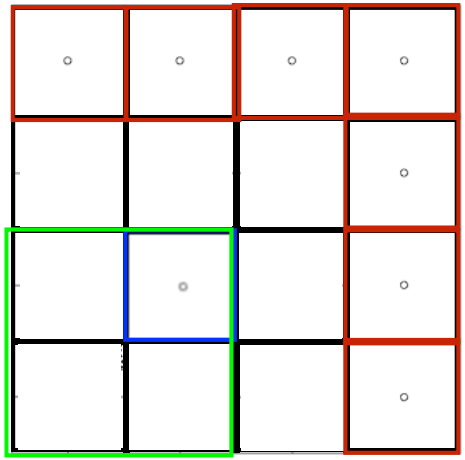
その近傍

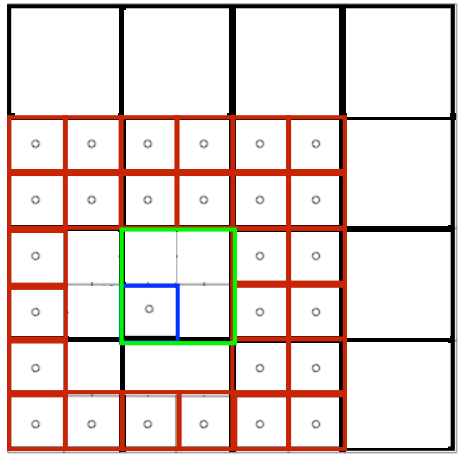
↓

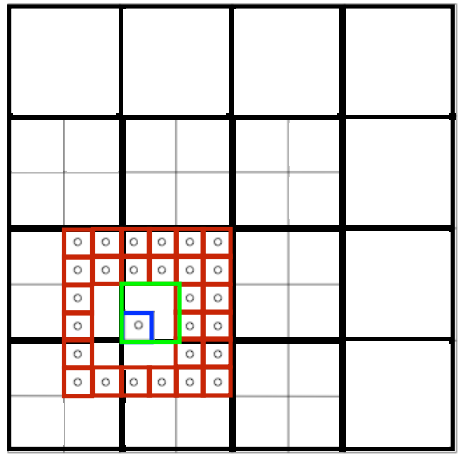
それらの子

↓

近傍でないもの







２次元の番地に変換 26→(3,4)

番地が近い
ものを求める

(2,5),(3,5),(4,5)
(2,4),(3,4),(4,4)
(2,3),(3,3),(4,3)

↓

曲線の番号に
逆変換

25,27,49
24,26,48
13,15,37

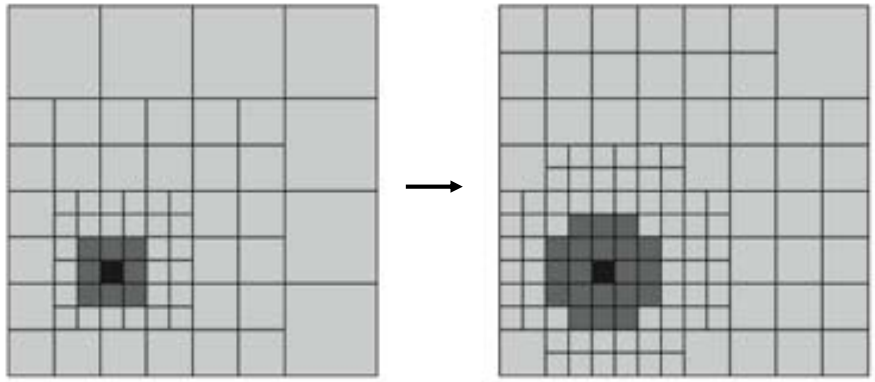
各階層でこれらの
赤い箱を見つけて

$$\vec{a}_i = \sum_{j-1}^{N} \vec{F}_{ij} m_j$$

を計算

精度と速度のトレードオフ

1. 近傍の定義を変える
- 近似計算する箱同士の距離が大きくなり精度が向上
- 問題点：箱の数が増える



2. 級数展開の次数Pを増やす
- 近似計算自体の精度が向上
- 問題点：近似計算自体が重くなる

$$M_n = \sum_{j=1}^{N} \frac{x_{cj}^n m_j}{n!}$$

$$a_i = \sum_{n=0}^{p} F_{ic}^{(n)} M_n$$

その他の調節すべきパラメータ

木構造を構築する際の木の深さ
・これは末端の箱の最大粒子数を変えることで制御

末端の箱の最大粒子数が大きいと…
・直接計算の部分が増える
・直接計算の部分はGPU利用効率が高い

末端の箱の最大粒子数が小さいと…
・近似計算の部分が増える
・全体の演算量が減るが，GPU利用効率が下がる