17次方程式の判別式の計算について

木村 欣司(京都大学大学院情報学研究科)

講演内容

17次方程式

$$a_0x^{17} + a_1x^{16} + a_2x^{15} + a_3x^{14} + a_4x^{13} + a_5x^{12} + a_6x^{11} + a_7x^{10} + a_8x^9 + a_9x^8 + a_{10}x^7 + a_{11}x^6 + a_{12}x^5 + a_{13}x^4 + a_{14}x^3 + a_{15}x^2 + a_{16}x + a_{17}$$

の判別式の項数は, 21976689397

ディスク容量では, 427,470,114,659byte

判別式ってなに?

2次方程式ならば, $b^2 - 4ac$

終結式(resultant)のSylvester表現

$$f(x) = a_m x^m + a_{m-1} x^{m-1} + \cdots + a_0,$$

$$g(x) = b_n x^n + b_{n-1} x^{n-1} + \cdots + b_0.$$

$$\mathrm{Sylvester}(f(x), g(x)) =$$

$$\begin{pmatrix}
a_m & a_{m-1} & \cdots & a_0 & & \\
 & a_m & a_{m-1} & \cdots & a_0 & \\
 & & \ddots & \ddots & \ddots & \ddots \\
 & & & a_m & a_{m-1} & \cdots & a_0 \\
b_n & b_{n-1} & \cdots & \cdots & b_0 & & \\
 & b_n & b_{n-1} & \cdots & \cdots & b_0 & \\
 & & \ddots & \ddots & \ddots & \ddots & \\
 & & & b_n & b_{n-1} & \cdots & \cdots & b_0
\end{pmatrix}$$

$$f(x) = a_2x^2 + a_1x + a_0, g(x) = b_1x + b_0$$

$$\mathrm{res}_x(f(x), g(x)) =$$

$$\det(\mathrm{Sylvester}(f(x), g(x))) = \begin{vmatrix} a_2 & a_1 & a_0 \\ b_1 & b_0 & 0 \\ 0 & b_1 & b_0 \end{vmatrix}$$

判別式(discriminant)

$$\mathrm{discriminant}(f(x)) = (-1)^{\frac{1}{2}m(m-1)}\frac{1}{a_m}\mathrm{res}_x(f(x), f'(x))$$

判別式の特徴

$$f(x) = a_m x^m + a_{m-1} x^{m-1} + \cdots + a_0,$$
$$f'(x) = m a_m x^{m-1} + (m-1) a_{m-1} x^{m-2} + \cdots + a_1,$$
$$\mathrm{Sylvester}(f(x), g(x)) =$$
$$\begin{pmatrix}
a_m & a_{m-1} & \cdots & a_0 & & \\
 & a_m & a_{m-1} & \cdots & a_0 & \\
 & & \ddots & \ddots & \ddots & \ddots \\
 & & & a_m & a_{m-1} & \cdots & a_0 \\
m a_m & (m-1) a_{m-1} & \cdots & \cdots & a_1 & & \\
 & m a_m & (m-1) a_{m-1} & \cdots & \cdots & a_1 & \\
 & & \ddots & \ddots & \ddots & \ddots & \\
 & & m a_m & (m-1) a_{m-1} & \cdots & \cdots & a_1
\end{pmatrix}$$

行列式の定義を考えれば, $\mathrm{res}_x(f(x), f'(x))$が斉次多項式になることは自明，さらに，1列目での余因子展開を考えれば，a_mで除算を行った後の多項式が斉次多項式になることも自明, 以上より, $a_m = 1$と規格化しても一般性は失われない

平行移動に対する不変性を用いた級数展開法(Cayley法)

問題: $f_n(x) = x^n + a_1 x^{n-1} + \cdots + a_{n-1}x + a_n$ **の判別式を計算せよ**

一般係数の判別式を計算する場合，一般性を失うことなく，主係数は1に規格化できる

今，$f_{n-1}(x) = x^{n-1} + a_1 x^{n-2} + \cdots + a_{n-2}x + a_{n-1}$ の判別式が分かっていたとしたとき，なにができるであろうか?

仮定より,

$$\mathrm{disc.}(f_{n-1}(x))/(-1)^{\frac{1}{2}(n-1)(n-2)} =$$
$$\mathrm{res}_x(f_{n-1}(x), f'_{n-1}(x))$$

終結式のSylvester表現の行列式が計算できているといってよい

次からは, 具体例で考えてみる

3次の判別式を計算するとき，2次の判別式が計算できているとする

すなわち，

$$\begin{aligned}\mathrm{res}_x(f_2(x), f_2'(x)) &= \begin{vmatrix} 1 & a_1 & a_2 \\ 2 & a_1 & 0 \\ 0 & 1 & a_1 \end{vmatrix} \\ &= \frac{\mathrm{disc.}(f_2(x))}{(-1)^{\frac{1}{2}(3-1)(3-2)}}\end{aligned}$$

が計算できていると仮定する

$$\mathrm{res}_x(f_3(x), f_3'(x)) = \begin{vmatrix} 1 & a_1 & a_2 & a_3 & 0 \\ 0 & 1 & a_1 & a_2 & a_3 \\ 3 & 2a_1 & a_2 & 0 & 0 \\ 0 & 3 & 2a_1 & a_2 & 0 \\ 0 & 0 & 3 & 2a_1 & a_2 \end{vmatrix}$$

2次のSylvester表現と3次のSylvester表現になにか関係はないであろうか?

試しに, $a_3 = 0$にしてみよう

$$\mathrm{res}_x(\widehat{f}_3(x), \widehat{f}_3'(x)) = \begin{vmatrix} 1 & a_1 & a_2 & 0 & 0 \\ 0 & 1 & a_1 & a_2 & 0 \\ 3 & 2a_1 & a_2 & 0 & 0 \\ 0 & 3 & 2a_1 & a_2 & 0 \\ 0 & 0 & 3 & 2a_1 & a_2 \end{vmatrix}$$

$n = 3$のとき, $(-1)^{2(n+n-1)} = 1$により,

$$\mathrm{res}_x(\widehat{f}_3(x), \widehat{f}_3'(x)) = a_2 \begin{vmatrix} 1 & a_1 & a_2 & 0 \\ 0 & 1 & a_1 & a_2 \\ 3 & 2a_1 & a_2 & 0 \\ 0 & 3 & 2a_1 & a_2 \end{vmatrix}$$

3行目から1行目を引く

$$\mathrm{res}_x(\widehat{f}_3(x), \widehat{f}_3'(x)) = a_2 \begin{vmatrix} 1 & a_1 & a_2 & 0 \\ 0 & 1 & a_1 & a_2 \\ 2 & a_1 & 0 & 0 \\ 0 & 3 & 2a_1 & a_2 \end{vmatrix}$$

4行目から2行目を引く

$$\mathrm{res}_x(\widehat{f}_3(x), \widehat{f}_3'(x)) = a_2 \begin{vmatrix} 1 & a_1 & a_2 & 0 \\ 0 & 1 & a_1 & a_2 \\ 2 & a_1 & 0 & 0 \\ 0 & 2 & a_1 & 0 \end{vmatrix}$$

$n = 3$のとき,

$$(-1)^{n+n-1-1+n-1} = (-1)^{3n-3} = (-1)^{3(n-1)}$$

$$\mathrm{res}_x(\widehat{f}_3(x), \widehat{f}_3'(x)) = (-1)^{3(n-1)} a_2^2 \begin{vmatrix} 1 & a_1 & a_2 \\ 2 & a_1 & 0 \\ 0 & 2 & a_1 \end{vmatrix}$$

$$= (-1)^{3(n-1)} a_2^2 \frac{\mathrm{disc.}(f_2(x))}{(-1)^{\frac{1}{2}(n-1)(n-2)}}$$

$$= (-1)^{\frac{1}{2}(n-8)(n-1)} a_2^2 \mathrm{disc.}(f_2(x))$$

$$
\begin{aligned}
&\mathrm{disc.}(\widehat{f}_3(x)) \\
&= (-1)^{\frac{1}{2}n(n-1)}\det(\mathrm{Sylvester}(\widehat{f}_3(x), \widehat{f}_3'(x))) \\
&= (-1)^{\frac{1}{2}[n(n-1)+(n-8)(n-1)]}a_2^2\mathrm{disc.}(f_2(x)) \\
&= (-1)^{\frac{1}{2}[(2n-8)(n-1)]}a_2^2\mathrm{disc.}(f_2(x)) \\
&= (-1)^{(n-4)(n-1)}a_2^2\mathrm{disc.}(f_2(x)) \\
&= a_2^2\mathrm{disc.}(f_2(x))
\end{aligned}
$$

以上より, $\mathrm{disc.}(\widehat{f}_3(x)) = a_2^2\mathrm{disc.}(f_2(x))$ となる

この状況は, $n > 3$ でも成立する

ここまでのまとめ

$f_n(x) = x^n + a_1 x^{n-1} + \cdots + a_{n-1}x + a_n$

に対して,

$\widehat{f}_n(x) = x^n + a_1 x^{n-1} + \cdots + a_{n-1}x + 0$

とおくと,

$\mathrm{disc.}(\widehat{f}_n(x)) = a_{n-1}^2 \mathrm{disc.}(f_{n-1}(x))$ となる

ここから, a_n を変化させて, $\mathrm{disc.}(f_n(x))$ を求めることを企むわけである

判別式が満たす微分方程式

$$f_n(x) = x^n + a_1 x^{n-1} + \cdots + a_{n-1} x + a_n$$

の判別式Dは,

$$n\frac{\partial D}{\partial a_1} + (n-1)a_1\frac{\partial D}{\partial a_2} + \cdots + a_{n-1}\frac{\partial D}{\partial a_n} = 0$$

を満たす

証明

根を$x_1, \cdots, x_n$とすると, $D = \Pi_{i>j}(x_i - x_j)^2$となる

$D = (-1)^{\frac{1}{2}n(n-1)}\mathrm{res}_x(f(x), f'(x)) = \Pi_{i>j}(x_i - x_j)^2$となる

証明は, 後述する

$x_i \to x_i - h$と平行移動しても不変であるから

$$\frac{dD}{dh} = \frac{\partial D}{\partial a_1}\frac{da_1}{dh} + \frac{\partial D}{\partial a_2}\frac{da_2}{dh} + \cdots + \frac{\partial D}{\partial a_n}\frac{da_n}{dh} = 0$$

$\frac{da_i}{dh}$の計算

$x_i \longrightarrow x_i - h$と平行移動は,

$$f_n(x+h) = (x+h)^n + a_1(x+h)^{n-1} + \cdots +$$
$$a_{n-1}(x+h) + a_n$$

xの$n-1$次の係数は, $a_1 + hn$

xの$n-2$次の係数は, $a_2 + h(n-1)a_1 + h^2(\cdots)$

xの$n-3$次の係数は, $a_3 + h(n-2)a_2 + h^2(\cdots)$

$\cdots$

より, 証明終わり

判別式Dを，a_nで級数展開してみる

$$D = D_0 + D_1 a_n + D_2 a_n^2 + \cdots + D_{n-1} a_n^{n-1}$$

級数が，$n-1$次までであることはSylvester表現から自明

さらに,

$$\frac{\partial D}{\partial a_n} = \frac{-1}{a_{n-1}} \left(n \frac{\partial D}{\partial a_1} + (n-1) a_1 \frac{\partial D}{\partial a_2} + \cdots \right)$$

と変形し, 代入すると

$$D_1 = \frac{-1}{a_{n-1}} \left(n \frac{\partial D_0}{\partial a_1} + (n-1) a_1 \frac{\partial D_0}{\partial a_2} + \cdots \right)$$

$$D_2 = \frac{-1}{2a_{n-1}} \left(n \frac{\partial D_1}{\partial a_1} + (n-1) a_1 \frac{\partial D_1}{\partial a_2} + \cdots \right)$$

$\cdots$

$$D_{n-1} = \frac{-1}{(n-1)a_{n-1}} \left(n \frac{\partial D_{n-2}}{\partial a_1} + (n-1) a_1 \frac{\partial D_{n-2}}{\partial a_2} + \cdots \right)$$

アルゴリズム

disc.$(f_2) = a_1^2 - 4a_2$とし，以下のアルゴリズムを繰り返し適用して，自分が必要なnまで計算する

$$f_n(x) = x^n + a_1 x^{n-1} + \cdots + a_{n-1}x + a_n$$

に対して，

$$\hat{f}_n(x) = x^n + a_1 x^{n-1} + \cdots + a_{n-1}x + 0$$

とおくと，

$D_0 = \text{disc.}(\hat{f}_n(x)) = a_{n-1}^2 \text{disc.}(f_{n-1}(x))$となる

このをD_0，seed solutionとして

$$D_1 = \frac{-1}{a_{n-1}} \left(n\frac{\partial D_0}{\partial a_1} + (n-1)a_1 \frac{\partial D_0}{\partial a_2} + \cdots \right)$$

$$D_2 = \frac{-1}{2a_{n-1}} \left(n\frac{\partial D_1}{\partial a_1} + (n-1)a_1 \frac{\partial D_1}{\partial a_2} + \cdots \right)$$

$\cdots$

$$D_{n-1} = \frac{-1}{(n-1)a_{n-1}} \left(n\frac{\partial D_{n-2}}{\partial a_1} + (n-1)a_1 \frac{\partial D_{n-2}}{\partial a_2} + \cdots \right)$$

により, $D = D_0 + D_1 a_n + D_2 a_n^2 + \cdots + D_{n-1} a_n^{n-1}$

の級数の係数を求める

実験データ

この方法を用いて，16次の判別式を計算した場合のタイミングデータを示す

Cayley method Singular-3-1-6	Kimura method Parallel
613m47.301s	372m10.574s

この方法を用いて，16次の判別式を計算した場合の使用メモリ量を示す

Cayley method Singular-3-1-6	Kimura method Parallel
765424MB	**622448MB**

CPU:E5-4650L，4 CPU，32コア，mem:1440Gbyte

非有用性について

判別式は，一般係数の場合にのみ求めれば十分なものであろうか?

$f_n(x) = x^n + (a_1 + 5a_2)x^{n-1} + \cdots + a_{n-1}x + a_n$ となったら，どうするのであろうか?

$f_n(x) = x^n + b_1x^{n-1} + \cdots + a_{n-1}x + a_n$

b_1 とおいて，すべての計算を行った後に，置換すればよいのでは?

$\Rightarrow$ そんな人には，ぜひ次の問題を紹介したい!

2008年に，神戸大学齋藤政彦先生のグループから，次の依頼がきた

次の多項式$E6(a)$の判別式を計算してください

```
E6(a)=a^27
+12*p2*a^25
+60*p2^2*a^23
-48*p1*a^22
+(168*p2^3+96*q2)*a^21
-336*p2*p1*a^20+(294*p2^4+528*q2*p2+480*p0)*a^19
+(-1008*p2^2*p1-1344*q1)*a^18
+(144*p1^2+336*p2^5+1152*q2*p2^2+2304*p0*p2)*a^17
+((-1680*p2^3-768*q2)*p1-5568*q1*p2)*a^16
+(608*p2*p1^2+252*p2^6+1200*q2*p2^3+4768*p0*p2^2+17280*q0-1248*q2^2)*a^15
+((-1680*p2^4-2688*q2*p2+2304*p0)*p1-8832*q1*p2^2)*a^14
+(976*p2^2*p1^2+3264*q1*p1+120*p2^7+480*q2*p2^4+5696*p0*p2^3+
(43776*q0-4800*q2^2)*p2+12288*q2*p0)*a^13
+(832*p1^3+(-1008*p2^5-3072*q2*p2^2+5888*p0*p2)*p1-6528*q1*p2^3
+10752*q2*q1)*a^12
+((704*p2^3+4224*q2)*p1^2+2688*q1*p2*p1+33*p2^8-144*q2*p2^5+4384*p0*p2^4
+(41472*q0-6720*q2^2)*p2^2+34560*q2*p0*p2-34560*p0^2)*a^11
+(2560*p2*p1^3+(-336*p2^6-768*q2*p2^3+3584*p0*p2^2+64512*q0+8448*q2^2)*p1
-2112*q1*p2^4+23040*q2*q1*p2-70656*p0*q1)*a^10
```

```
+((176*p2^4+8960*q2*p2-18944*p0)*p1^2-5504*q1*p2^2*p1+4*p2^9-192*q2*p2^6
+2176*p0*p2^5+(22528*q0-3840*q2^2)*p2^3+32768*q2*p0*p2^2-39936*p0^2*p2
+110592*q2*q0-40704*q1^2+5120*q2^3)*a^9
+(2688*p2^2*p1^3+4608*q1*p1^2+(-48*p2^7+768*q2*p2^4-1536*p0*p2^3
+(82944*q0+16128*q2^2)*p2-73728*q2*p0)*p1-192*q1*p2^5+13824*q2*q1*p2^2
-64512*p0*q1*p2)*a^8
+(-2560*p1^4+(-32*p2^5+5376*q2*p2^2-16384*p0*p2)*p1^2+(-6144*q1*p2^3
-15360*q2*q1)*p1-48*q2*p2^7+608*p0*p2^6+(9600*q0-480*q2^2)*p2^4
+10752*q2*p0*p2^3-20992*p0^2*p2^2+(156672*q2*q0-38400*q1^2+9984*q2^3)*p2
-165888*p0*q0-56832*q2^2*p0)*a^7
+((1024*p2^3-10240*q2)*p1^3+10240*q1*p2*p1^2+(384*q2*p2^5-1792*p0*p2^4
+(21504*q0+6912*q2^2)*p2^2-57344*q2*p0*p2+49152*p0^2)*p1+1536*q2*q1*p2^3
-19456*p0*q1*p2^2-110592*q1*q0-21504*q2^2*q1)*a^6
+(-1536*p2*p1^4+(-16*p2^6+768*q2*p2^3-4608*p0*p2^2+27648*q0-19200*q2^2)*p1^2
+(-1344*q1*p2^4+10752*q2*q1*p2-9216*p0*q1)*p1+64*p0*p2^7
+(2304*q0+192*q2^2)*p2^5-3072*p0^2*p2^3+(55296*q2*q0-12288*q1^2
+4608*q2^3)*p2^2+(-110592*p0*q0-46080*q2^2*p0)*p2+73728*q2*p0^2)*a^5
+((64*p2^4-4096*q2*p2+8192*p0)*p1^3-512*q1*p2^2*p1^2+(-256*p0*p2^5+
(3072*q0-768*q2^2)*p2^3-8192*q2*p0*p2^2+16384*p0^2*p2+73728*q2*q0
```

```
-39936*q1^2-18432*q2^3)*p1-1024*p0*q1*p2^3+(-36864*q1*q0-3072*q2^2*q1)*p2
+24576*q2*p0*q1)*a^4
+(256*p2^2*p1^4+15360*q1*p1^3+(128*q2*p2^4-1024*p0*p2^3+(-6144*q0
-2560*q2^2)*p2+8192*q2*p0)*p1^2+(-128*q1*p2^5+2048*q2*q1*p2^2
-14336*p0*q1*p2)*p1+256*q0*p2^6-256*q2*p0*p2^5+256*p0^2*p2^4+(9216*q2*q0
-2560*q1^2-256*q2^3)*p2^3+(-18432*p0*q0-7680*q2^2*p0)*p2^2
+24576*q2*p0^2*p2-110592*q0^2+55296*q2^2*q0-30720*q2*q1^2-16384*p0^3
-6912*q2^4)*a^3
+(-1024*p1^5+4096*p0*p2*p1^3+24576*q2*q1*p1^2-12288*q1^2*p2*p1)*a^2
+(-2048*q2*p1^4+2048*q1*p2*p1^3+((-3072*q0-256*q2^2)*p2^2+4096*q2*p0*p2
-4096*p0^2)*p1^2+(512*q2*q1*p2^3-1024*p0*q1*p2^2-36864*q1*q0
+9216*q2^2*q1)*p1-256*q1^2*p2^4-6144*q2*q1^2*p2+12288*p0*q1^2)*a
+(4096*q0-1024*q2^2)*p1^3+(2048*q2*q1*p2-4096*p0*q1)*p1^2
-1024*q1^2*p2^2*p1-4096*q1^3
```

以上より，Cayley法は，数式処理の研究内容としては相応しくない

Singularは分散表現を採用

Cayley法との相性が良いのは，分散表現

分散表現は，並列計算にまったく向いていない

分散表現とは?

多変数多項式を，係数と指数の組の結合されたリストで持つ

$$f = 5x^2y + 6x + 7y + 8$$

$$(5, [2, 1]) \to (6, [1, 0]) \to (7, [0, 1])$$

$$\to (8, [0, 0])$$

$E6(a)$は何か?

ガロア群がワイル群W($E6$)である27次6パラメータの代数方程式

cf. T.Shioda, Construction of elliptic curves with high rank via the invariants of the Weyl groups, J.Math.Soc.Japan 43(1991), 673-719

塩田先生は，次の３つの代数方程式を作られた

ガロア群がワイル群$W(E6)$:27次6パラメータ

ガロア群がワイル群$W(E7)$:56次7パラメータ

ガロア群がワイル群$W(E8)$:240次8パラメータ

多項式を係数(要素)にもつ終結式(行列式)の計算法

1. 小行列式展開, $O(2^n)$, 割り算なし
2. 久留島-ラプラス展開, ???, 割り算なし
3. fraction-free Gauss消去法, $O(n^3)$, 割り算あり
4. **補間法(木村版)**, ???, mod 演算あり
5. **Berkowitzの方法**, Fadeevの方法, $O(n^4)$, 割り算なし
6. Subresultant列を互除法で計算

補間法(木村版)

カット面付き多変数Newton補間法

中国剰余定理

未知の整数をXとする

$$X \bmod 2 = 1$$

$$X \bmod 3 = 2$$

この情報から，**中国剰余定理**により

$$X = \cdots, -13, -7, -1, 5, 11, 17, \cdots$$

が，Xの候補となる，解は一意に定まらない

もし，Xは1桁の整数であるという付加情報が手に入ったとすると，$X = -7, -1, 5$，まだ，3つも候補が残っている

もうすこし情報が手に入ったとする

$$X \bmod \underline{2} = 1$$
$$X \bmod \underline{3} = 2$$
$$X \bmod \underline{5} = 3$$

よって，答えは，中国剰余定理と付加情報により$X = -7$と一意的に定まる，**下線の部分には，素数を使う**

(付加情報) 行列式に付随した2つの上界公式

2つのノルムを定義する: qは, 多変数多項式

$$\|q\|_1 = \sum_{\alpha_1,\cdots,\alpha_s} |c(\alpha_1,\cdots,\alpha_s)|,$$
$$\|q\|_2 = \sqrt{\sum_{\alpha_1,\cdots,\alpha_s} c(\alpha_1,\cdots,\alpha_s)^2},$$

ここで,

$q = \Sigma_{\alpha_1,\cdots,\alpha_s} c(\alpha_1,\cdots,\alpha_s) x_1^{\alpha_1}\cdots x_s^{\alpha_s} \in \mathbb{Z}[x_1,\cdots,x_s]$

$A = (a_{i,j}) \in \mathbb{Z}^{N\times N}$ についてのHadamardの上界は,

$$|\det(A)| \leq \min\left\{ \prod_{i=1}^{N} \sqrt{\sum_{j=1}^{N} |a_{i,j}|^2}, \prod_{j=1}^{N} \sqrt{\sum_{i=1}^{N} |a_{i,j}|^2} \right\}$$

$A = (a_{i,j}) \in \mathbb{Z}[x_1, \cdots, x_s]^{N\times N}$ についてのGoldsteinとGrahamの上界は,

$$\|\det(A)\|_2 \leq \min\left\{ \prod_{i=1}^{N} \sqrt{\sum_{j=1}^{N} \|a_{i,j}\|_1^2}, \prod_{j=1}^{N} \sqrt{\sum_{i=1}^{N} \|a_{i,j}\|_1^2} \right\}$$

$$B = \begin{vmatrix} a & 2b \\ 3c & 4d + f \end{vmatrix} = \underline{4}ad + \underline{1}af - \underline{6}bc$$

この例の場合，GoldsteinとGrahamの上界は，

$\min(\sqrt{1+4}\sqrt{9+25}, \sqrt{1+9}\sqrt{4+25}) < 13.1$

よって，行列式で定式化できるものは，すべて中国剰余定理が使える

行列式の計算結果の次数の上界公式:TYPE 1

A simple bound of degrees

$$A = \begin{pmatrix} x+y+z & xy \\ 2 & xyz \end{pmatrix} \begin{matrix} 1 \\ + \\ 1 \end{matrix}$$

$$1 \quad + \quad 1$$

上の考察からxに対する行列式の次数の最大値は, 2

変数とパラメータの解釈により

変数	パラメータ	(partial) total degree
x	y, z	2
y	x, z	2
z	x, y	2
x, y	z	3
y, z	x	3
z, x	y	3
x, y, z		4

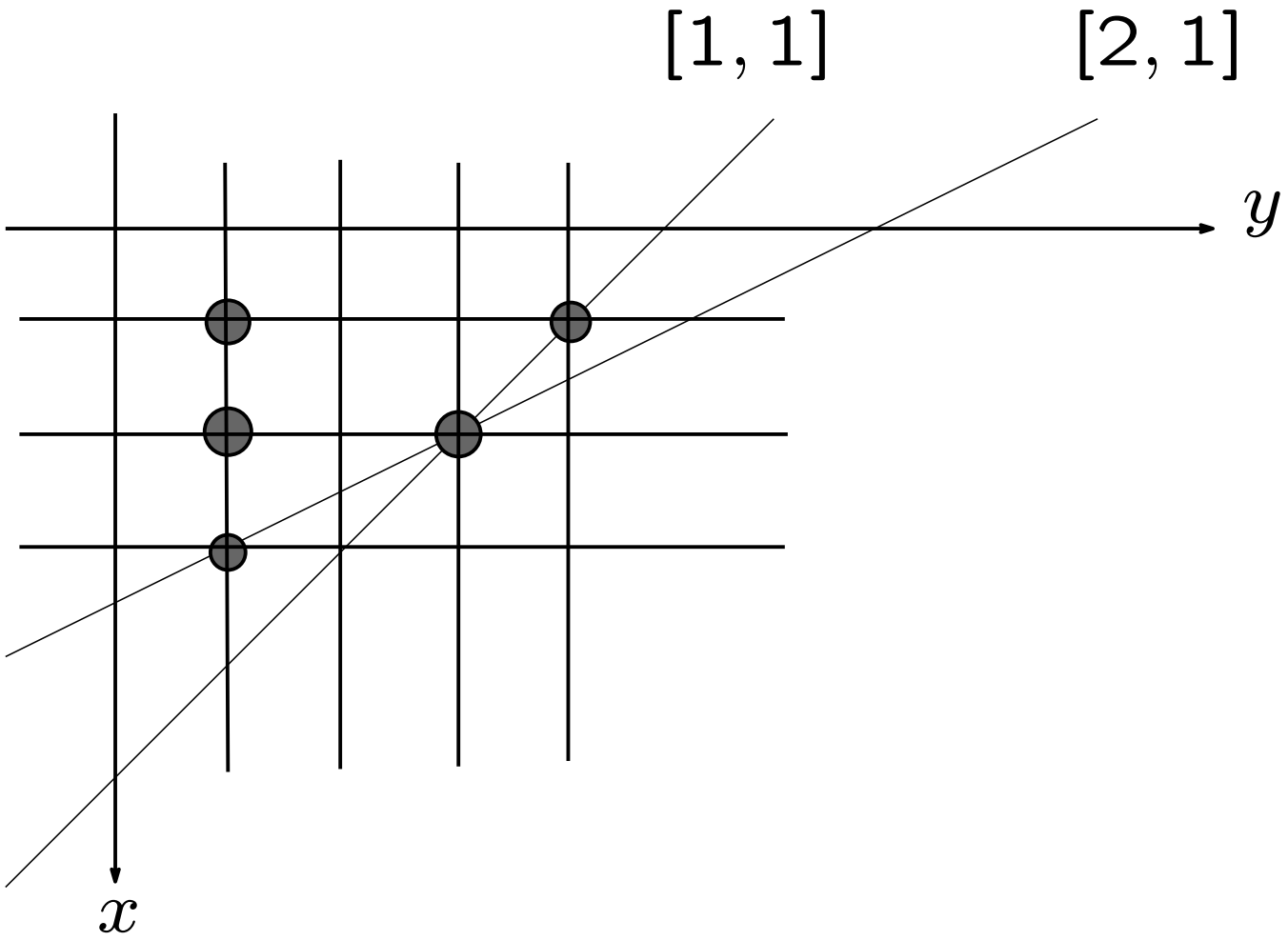
重み(weight)付き全次数

変数$[x, y, z]$に対する重み(weight)を，[1, 2, 3]とする

変数	パラメータ	(partial) total degree
x	y, z	2
y	x, z	4
z	x, y	6
x, y	z	5
y, z	x	8
z, x	y	7
x, y, z		9

weightとカット面の関係

weight($[x, y]$)=[2, 1]だとすると



たくさんのweightを使えば，原理的にはタイトな上界になる

優れた次数の上界公式

weightと，linear assignment problem:LAPの技術を両方使う

linear assignment problem:LAP

あなたは，働き手を持っています，Jim，SteveとAlanです，一人に，浴室の掃除をさせます，もう一人に，床の掃き掃除をさせます，三人目の人には，窓の洗浄をさせます，最大(最小)のコストになるには，どのように仕事を割り当てればよいでしょうか?
それぞれの仕事に対する3名のコストは，以下のテーブルで与えられています

	$Clean\ bathroom$	$Sweep\ floors$	$Wash\ windows$
Jim	\$1	\$2	\$3
$Steve$	\$3	\$3	\$3
$Alan$	\$3	\$3	\$2

最大コスト問題を最小コスト問題用の解法で解く方法

行列式の計算結果の次数に興味がある場合には，最大コスト問題となる，ところが，世の中に出回っているsolverはすべて，最小コスト問題用にできている

$$\begin{bmatrix} & Clean\ bathroom & Sweep\ floors & Wash\ windows \\ Jim & \$1 & \$2 & \$3 \\ Steve & \$3 & \$3 & \$3 \\ Alan & \$3 & \$3 & \$2 \end{bmatrix}$$

最大値を計算 $\rightarrow$ 3, そして, 最大値$-$各コストを計算

$$\begin{bmatrix} & Clean\ bathroom & Sweep\ floors & Wash\ windows \\ Jim & \$2 & \$1 & \$0 \\ Steve & \$0 & \$0 & \$0 \\ Alan & \$0 & \$0 & \$1 \end{bmatrix}$$

最小コストは, 0になる, 最後に, 最大値$\times$行列サイズ$-$最小コスト$=$ 9を行う

解法

1. 線形計画問題
2. 最大流問題

   Hungarian algorithm

   (Carpaneto and Toth version:LAPCT)
3. 最短路問題

   Dijkstra's shortest path methodを利用する

線形計画問題としての定式化

LAP

$$\max \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$$

$$s.t. \sum_{j=1}^{n} x_{ij} = 1 \quad (i = 1, 2, \cdots, n),$$

$$\sum_{i=1}^{n} x_{ij} = 1 \quad (j = 1, 2, \cdots, n),$$

$$x_{ij} \in \{0, 1\}$$

線形計画問題は，領域の端点を解として持つ為

LAP

$$\max \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$$

$$s.t. \sum_{j=1}^{n} x_{ij} = 1 \quad (i = 1, 2, \cdots, n),$$

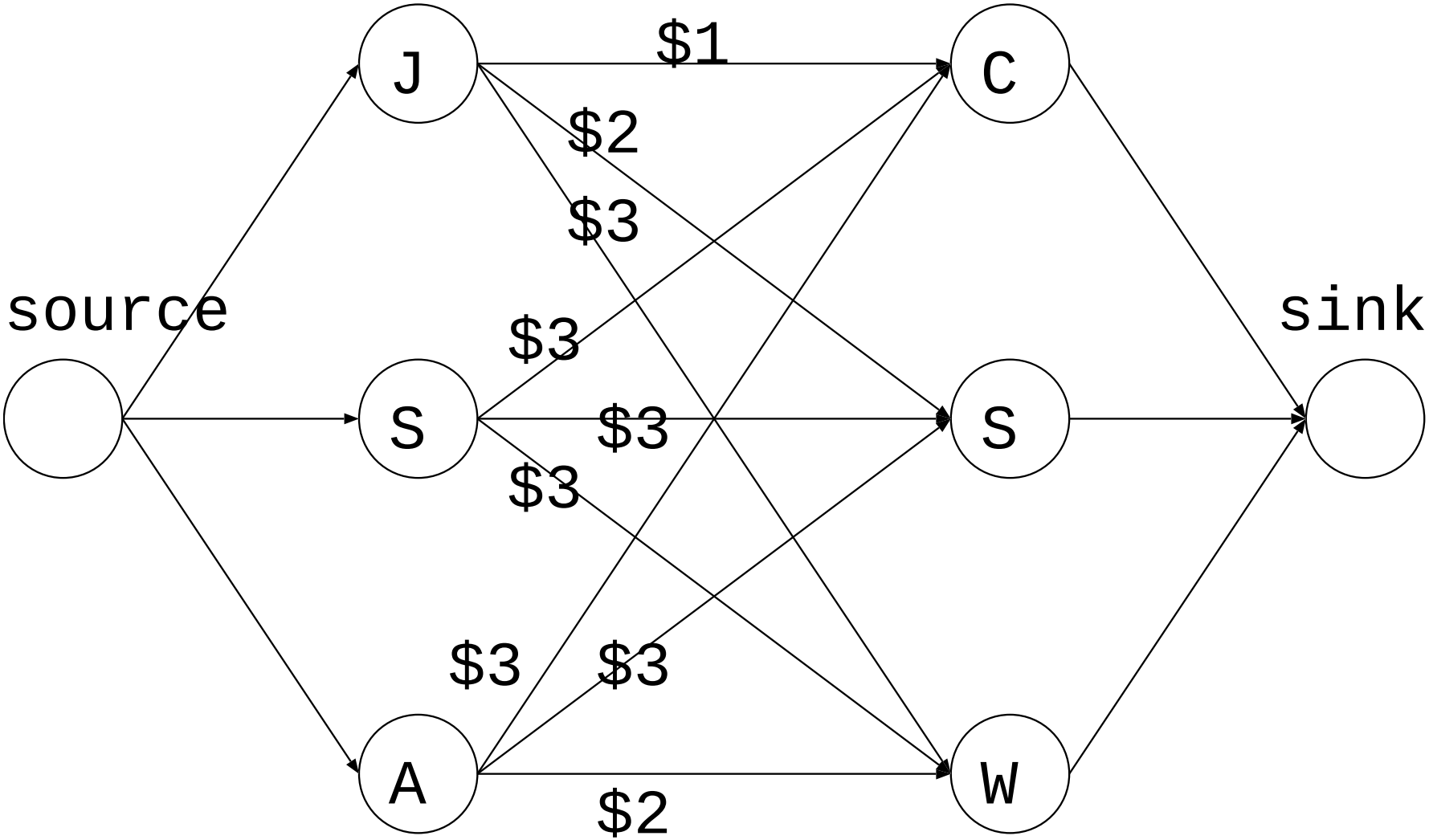
$$\sum_{i=1}^{n} x_{ij} = 1 \quad (j = 1, 2, \cdots, n),$$

$$x_{ij} \geq 0$$

Balinski. M. L. :Signature methods for the assignment problem. Operations Research 33, 527 - 536(1985).

最大流問題に帰着

Lawler algorithm(1976)



最短路問題に帰着

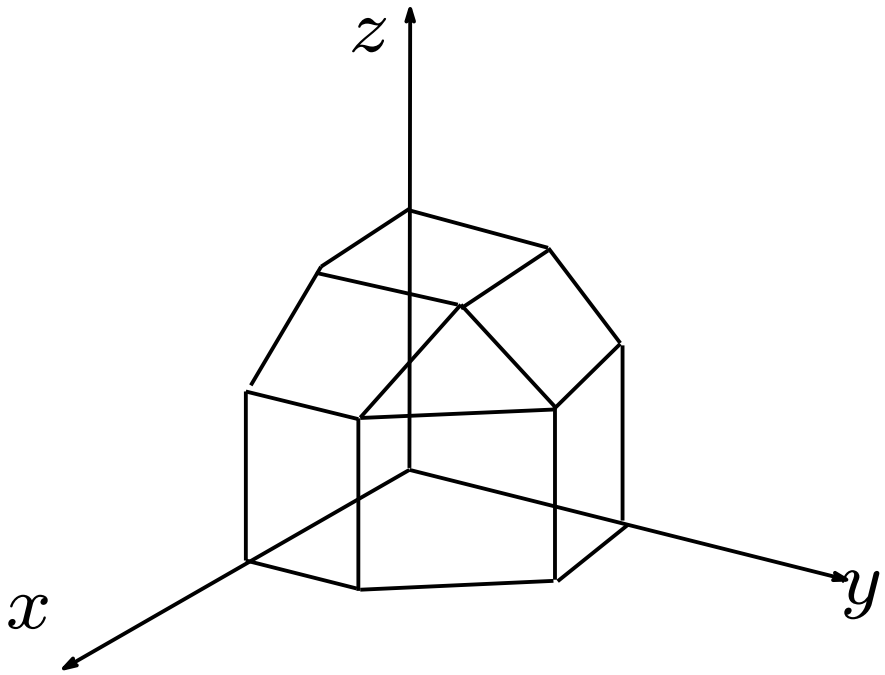
Tomizawa, N. : On some techniques useful for the solution of transportation problems. Networks 1, 173 194 (1971).

Jonker-Volgenant algorithm(LAPJV), 1987

カット面付き多変数Newton補間

下の図は，weightが1種類の場合，最近は，複数のweightでカットを入れる

下の凸包の内点格子の数分の評価点で，終結式の値をサンプリングし，補間する



カット面付き多変数Newton補間の動作原理

"Remarks on Newton Type Multivariate Interpolation for Subsets of Grids"

H. Werner, Mtinster

Computing 25, 181–191, (1980)

$$B = \begin{vmatrix} \alpha+\beta & 2 \\ 3 & \alpha\beta \end{vmatrix}$$

行列式または終結式の解が,

$$\begin{aligned} &(c_0 + c_1\beta + c_2\beta(\beta-1)) \\ &+\alpha(c_3 + c_4\beta + c_5\beta(\beta-1)) \\ &+\alpha(\alpha-1)(c_6 + c_7\beta + c_8\beta(\beta-1)), \\ &c_8 = 0, \end{aligned}$$

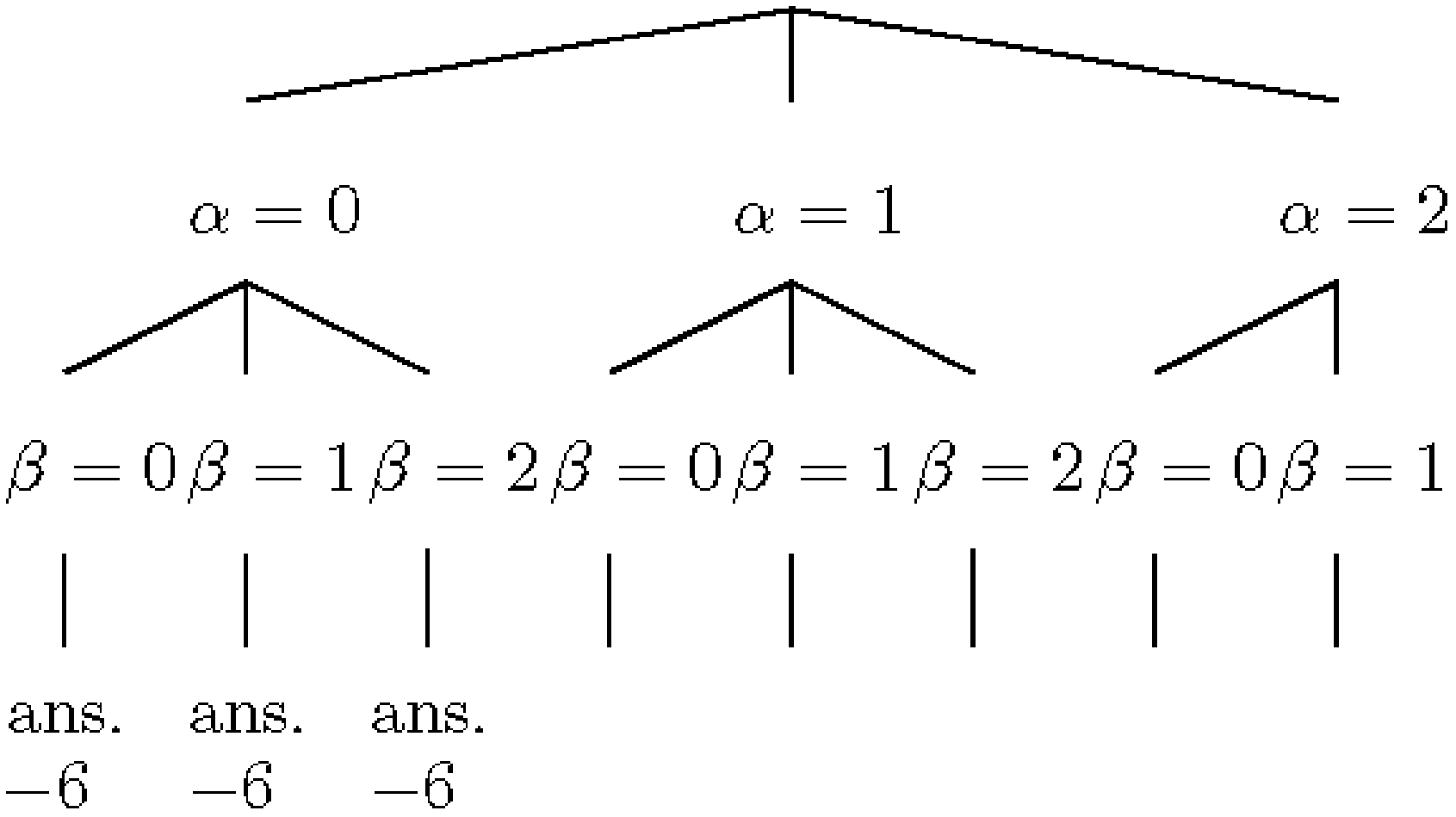
と仮定できるとする. $c_0, \cdots, c_7$ は, 未知数である.

行列式または終結式の解が,

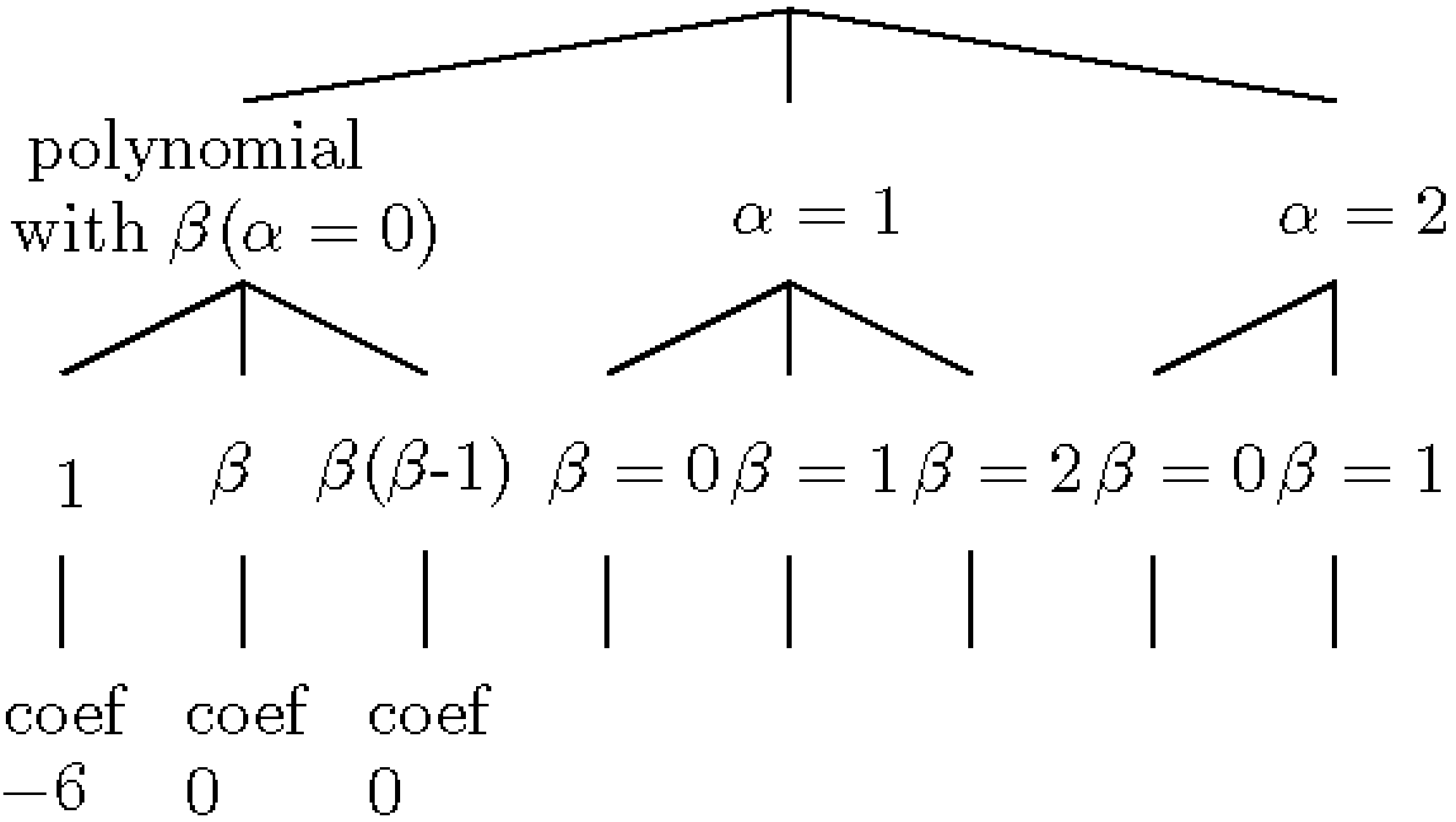
$$
\begin{aligned}
&(c_0 + c_1\beta + c_2\beta(\beta - 1)) \\
&+\alpha(c_3 + c_4\beta + c_5\beta(\beta - 1)) \\
&+\alpha(\alpha - 1)(c_6 + c_7\beta + c_8\beta(\beta - 1)), c_8 = 0,
\end{aligned}
$$

と仮定できるとする．$c_0, \cdots, c_7$は，未知数である．ここでは説明のため$\mathbb{Q}$上で表記するが，実際のプログラムでは，$\mathbb{Z}/p\mathbb{Z}$を利用して計算する．

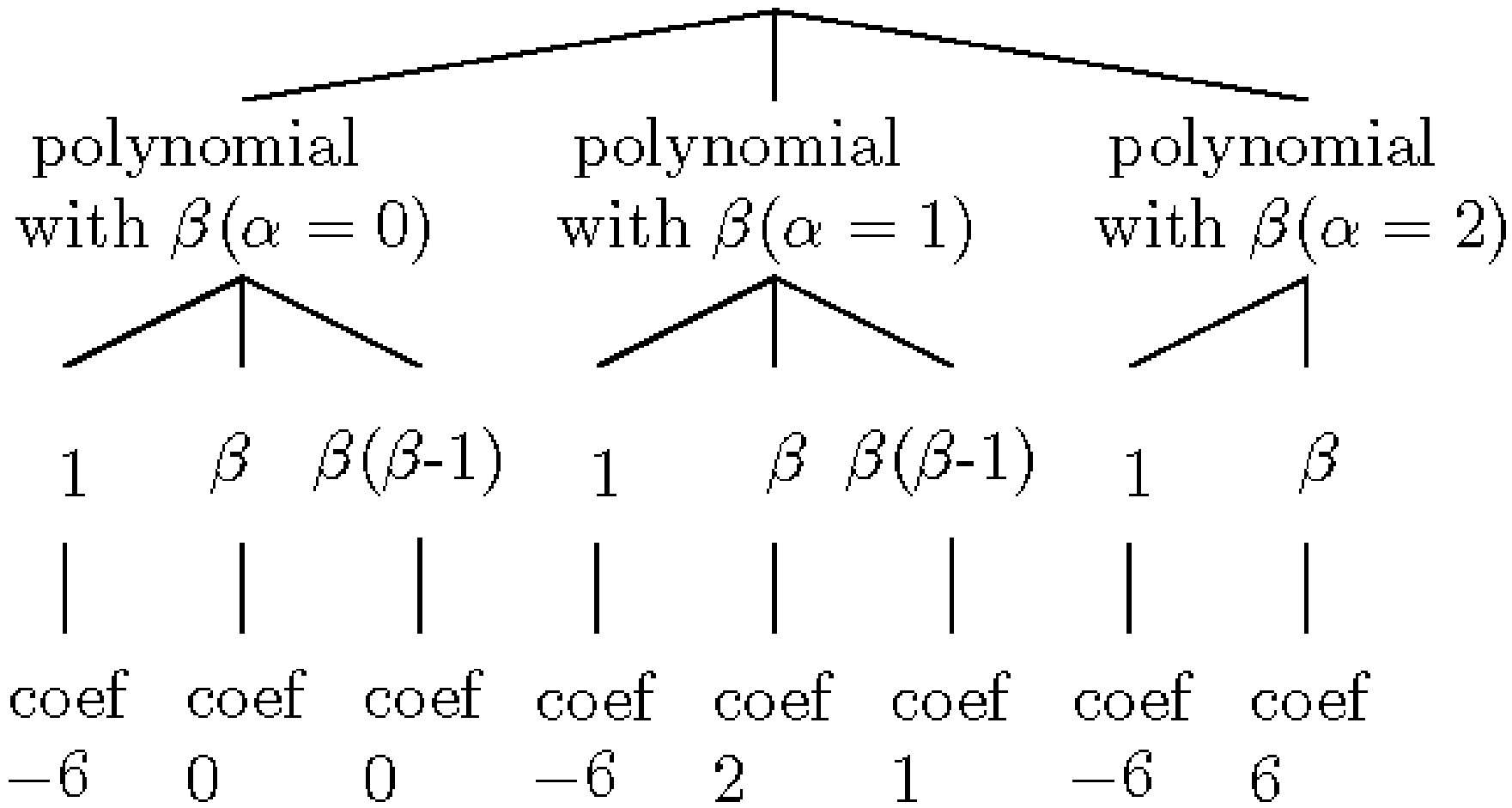
はじめに，$\alpha = 0$，$\beta = 0, 1, 2$における値を計算したとする．



次に, $\alpha = 0$においてNewton補間をする.

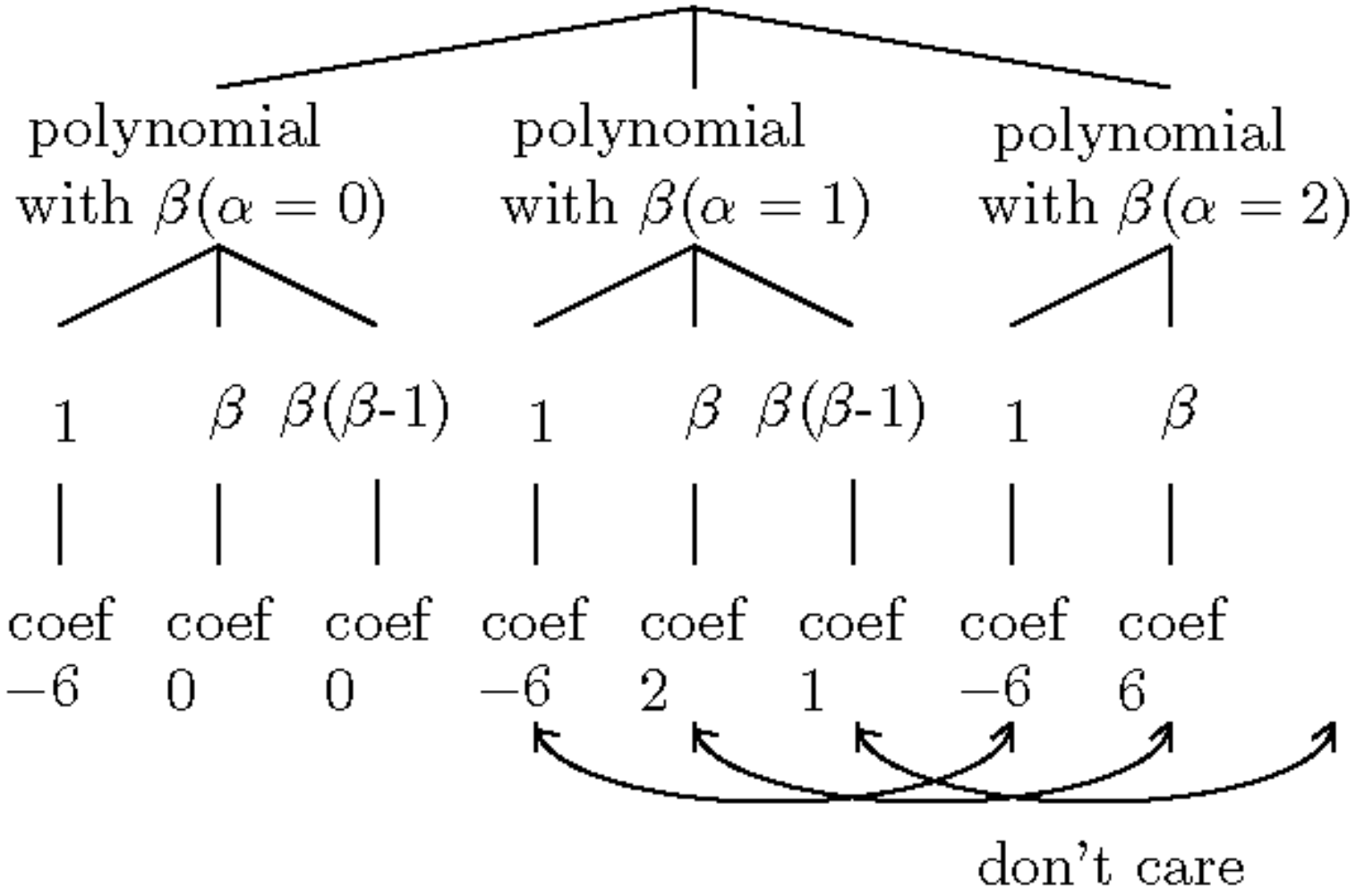


$\alpha = 1, \alpha = 2$においても同様の計算をおこなう.

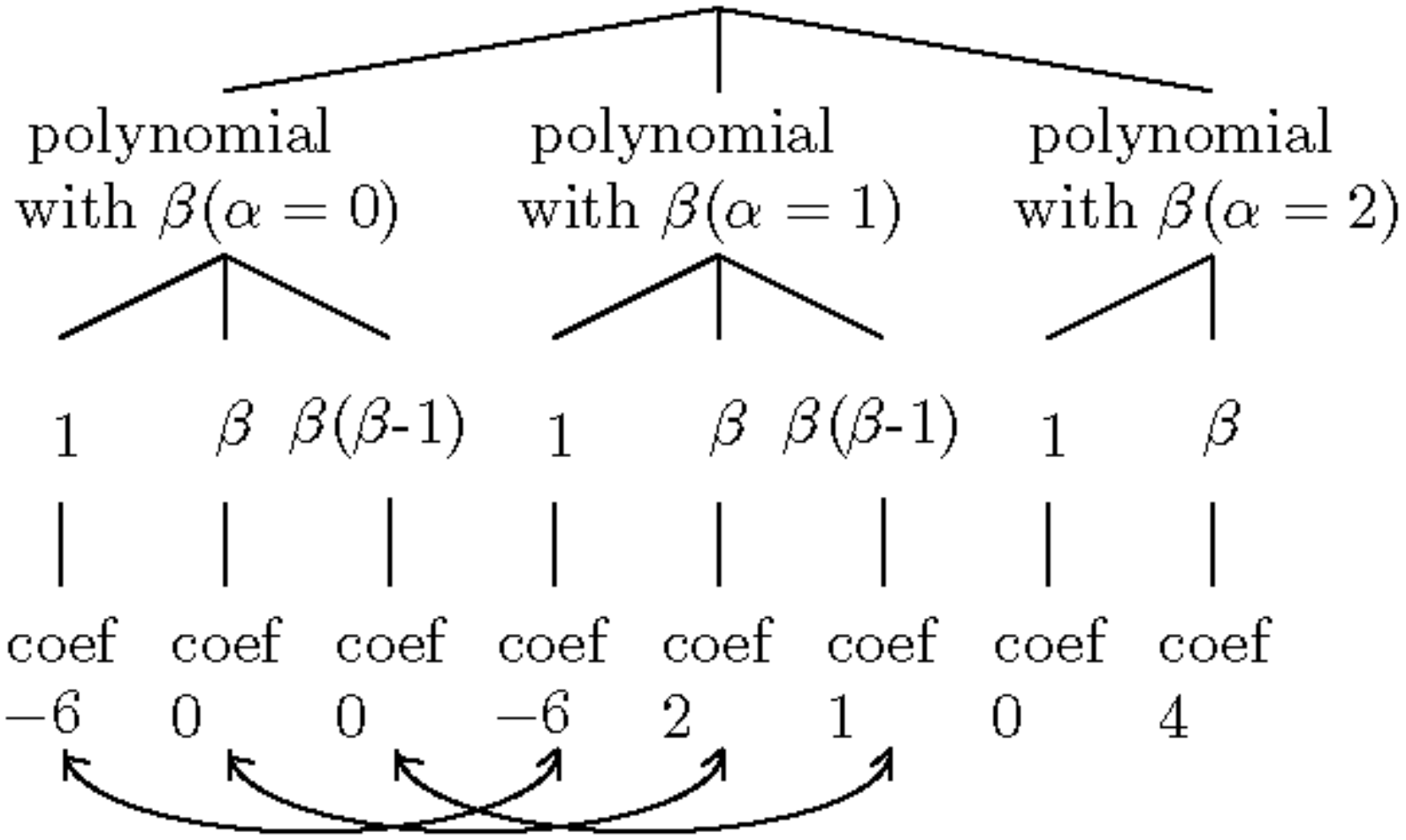


$\beta = 2$の多項式は，あえて次数1で打ち切っている．Newton補間は，逐次補間であるため，このような途中で打ち切ったものを入力としても正しい計算ができる．

ここからは，多項式を入力としてNewton補間をする． まず，$\alpha = 2$の多項式から$\alpha = 1$の多項式の減算をおこない，その結果を再度$\alpha = 2$に格納する．

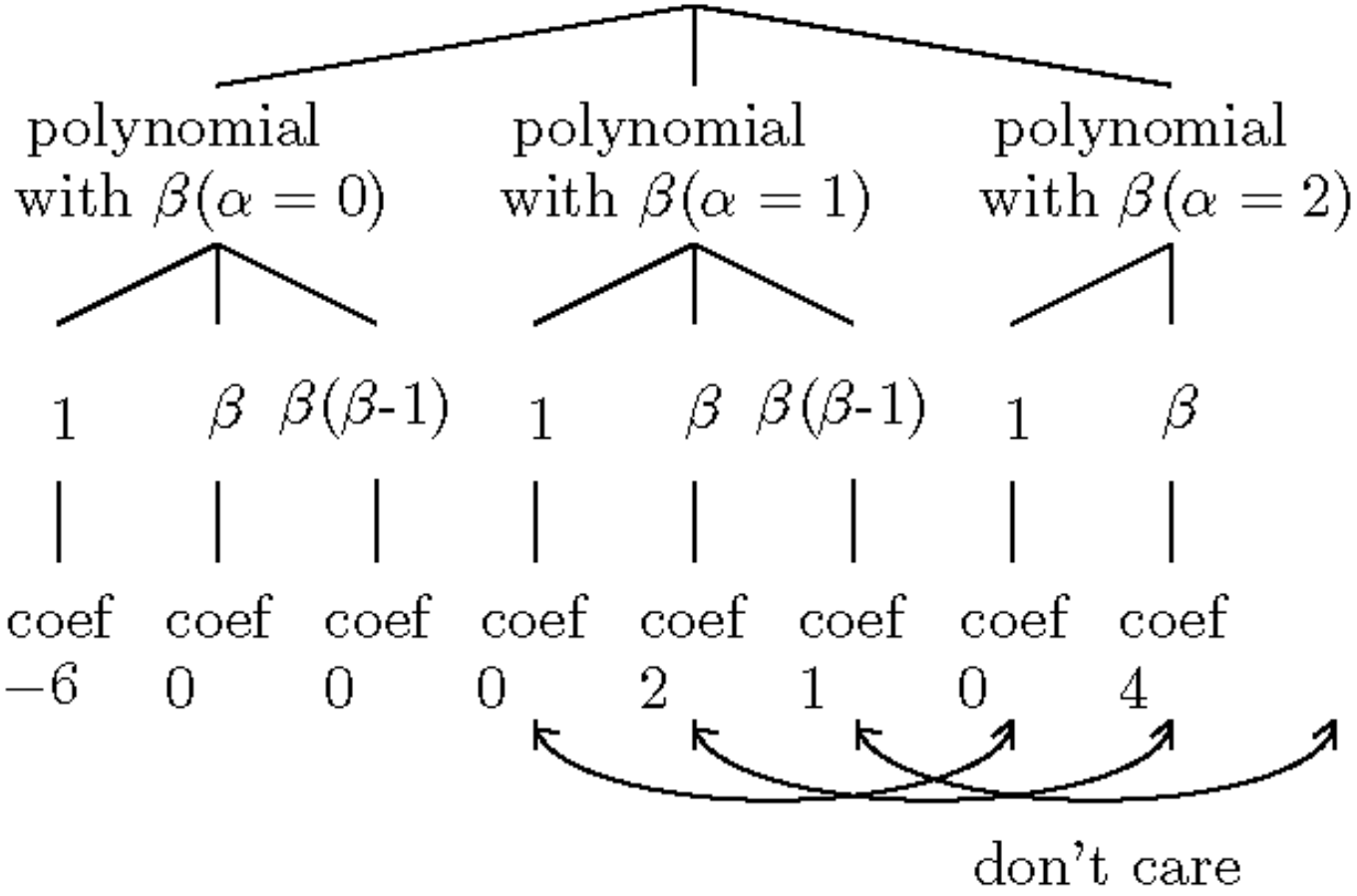


つぎに, $\alpha = 1$の多項式から$\alpha = 0$の多項式の減算をおこない, その結果を$\alpha = 1$に格納する.

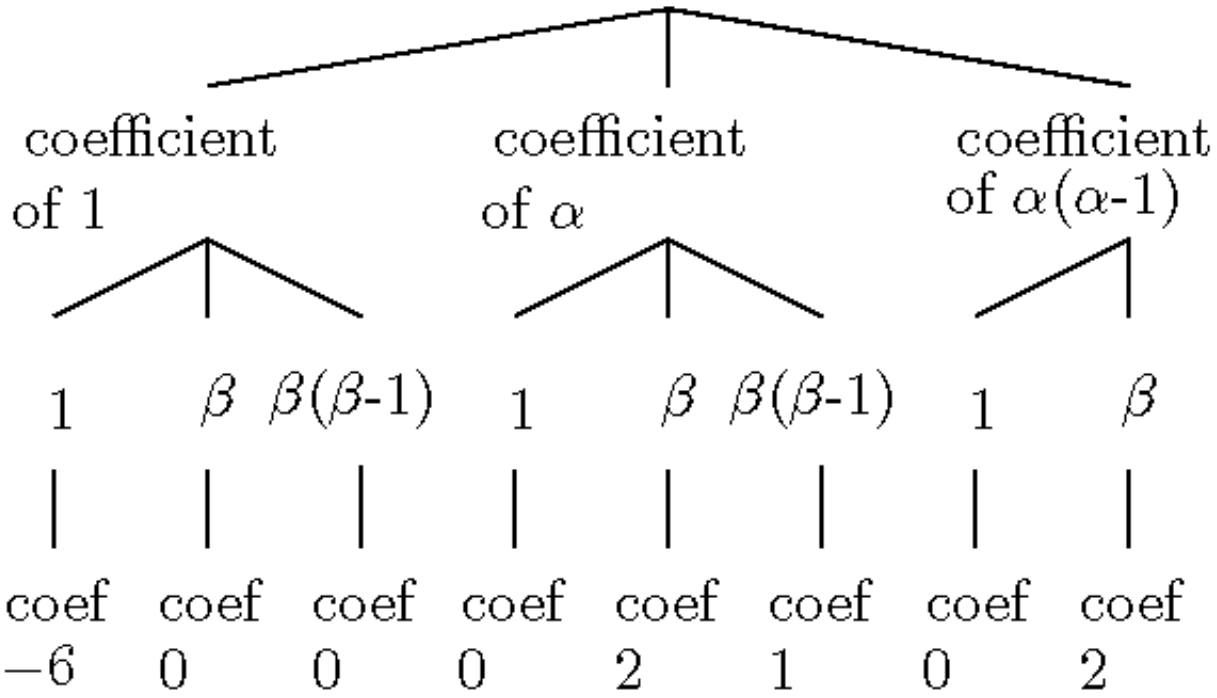


最後に$\alpha = 2$の多項式から$\alpha = 1$の多項式の減算をおこなうと,

多項式を入力とするNewton補間が完了する.



以上より, つぎの結果を得る.



$$(-6 + 0\beta + 0\beta(\beta - 1))$$

$$+\alpha(0 + 2\beta + 1\beta(\beta - 1))$$

$$+\alpha(\alpha - 1)(0 + 2\beta + 0\beta(\beta - 1))$$

最後に展開する

問題の再確認

```
E6(a)=a^27
+12*p2*a^25
+60*p2^2*a^23
-48*p1*a^22
+(168*p2^3+96*q2)*a^21
-336*p2*p1*a^20+(294*p2^4+528*q2*p2+480*p0)*a^19
+(-1008*p2^2*p1-1344*q1)*a^18
+(144*p1^2+336*p2^5+1152*q2*p2^2+2304*p0*p2)*a^17
+((-1680*p2^3-768*q2)*p1-5568*q1*p2)*a^16
+(608*p2*p1^2+252*p2^6+1200*q2*p2^3+4768*p0*p2^2+17280*q0-1248*q2^2)*a^15
+((-1680*p2^4-2688*q2*p2+2304*p0)*p1-8832*q1*p2^2)*a^14
+(976*p2^2*p1^2+3264*q1*p1+120*p2^7+480*q2*p2^4+5696*p0*p2^3+
(43776*q0-4800*q2^2)*p2+12288*q2*p0)*a^13
+(832*p1^3+(-1008*p2^5-3072*q2*p2^2+5888*p0*p2)*p1-6528*q1*p2^3
+10752*q2*q1)*a^12
+((704*p2^3+4224*q2)*p1^2+2688*q1*p2*p1+33*p2^8-144*q2*p2^5+4384*p0*p2^4
+(41472*q0-6720*q2^2)*p2^2+34560*q2*p0*p2-34560*p0^2)*a^11
```

```
+(2560*p2*p1^3+(-336*p2^6-768*q2*p2^3+3584*p0*p2^2+64512*q0+8448*q2^2)*p1
-2112*q1*p2^4+23040*q2*q1*p2-70656*p0*q1)*a^10
+((176*p2^4+8960*q2*p2-18944*p0)*p1^2-5504*q1*p2^2*p1+4*p2^9-192*q2*p2^6
+2176*p0*p2^5+(22528*q0-3840*q2^2)*p2^3+32768*q2*p0*p2^2-39936*p0^2*p2
+110592*q2*q0-40704*q1^2+5120*q2^3)*a^9
+(2688*p2^2*p1^3+4608*q1*p1^2+(-48*p2^7+768*q2*p2^4-1536*p0*p2^3
+(82944*q0+16128*q2^2)*p2-73728*q2*p0)*p1-192*q1*p2^5+13824*q2*q1*p2^2
-64512*p0*q1*p2)*a^8
+(-2560*p1^4+(-32*p2^5+5376*q2*p2^2-16384*p0*p2)*p1^2+(-6144*q1*p2^3
-15360*q2*q1)*p1-48*q2*p2^7+608*p0*p2^6+(9600*q0-480*q2^2)*p2^4
+10752*q2*p0*p2^3-20992*p0^2*p2^2+(156672*q2*q0-38400*q1^2+9984*q2^3)*p2
-165888*p0*q0-56832*q2^2*p0)*a^7
+((1024*p2^3-10240*q2)*p1^3+10240*q1*p2*p1^2+(384*q2*p2^5-1792*p0*p2^4
+(21504*q0+6912*q2^2)*p2^2-57344*q2*p0*p2+49152*p0^2)*p1+1536*q2*q1*p2^3
-19456*p0*q1*p2^2-110592*q1*q0-21504*q2^2*q1)*a^6
+(-1536*p2*p1^4+(-16*p2^6+768*q2*p2^3-4608*p0*p2^2+27648*q0-19200*q2^2)*p1^2
+(-1344*q1*p2^4+10752*q2*q1*p2-9216*p0*q1)*p1+64*p0*p2^7
+(2304*q0+192*q2^2)*p2^5-3072*p0^2*p2^3+(55296*q2*q0-12288*q1^2
+4608*q2^3)*p2^2+(-110592*p0*q0-46080*q2^2*p0)*p2+73728*q2*p0^2)*a^5
```

```
+((64*p2^4-4096*q2*p2+8192*p0)*p1^3-512*q1*p2^2*p1^2+(-256*p0*p2^5+
(3072*q0-768*q2^2)*p2^3-8192*q2*p0*p2^2+16384*p0^2*p2+73728*q2*q0
-39936*q1^2-18432*q2^3)*p1-1024*p0*q1*p2^3+(-36864*q1*q0-3072*q2^2*q1)*p2
+24576*q2*p0*q1)*a^4
+(256*p2^2*p1^4+15360*q1*p1^3+(128*q2*p2^4-1024*p0*p2^3+(-6144*q0
-2560*q2^2)*p2+8192*q2*p0)*p1^2+(-128*q1*p2^5+2048*q2*q1*p2^2
-14336*p0*q1*p2)*p1+256*q0*p2^6-256*q2*p0*p2^5+256*p0^2*p2^4+(9216*q2*q0
-2560*q1^2-256*q2^3)*p2^3+(-18432*p0*q0-7680*q2^2*p0)*p2^2
+24576*q2*p0^2*p2-110592*q0^2+55296*q2^2*q0-30720*q2*q1^2-16384*p0^3
-6912*q2^4)*a^3
+(-1024*p1^5+4096*p0*p2*p1^3+24576*q2*q1*p1^2-12288*q1^2*p2*p1)*a^2
+(-2048*q2*p1^4+2048*q1*p2*p1^3+((-3072*q0-256*q2^2)*p2^2+4096*q2*p0*p2
-4096*p0^2)*p1^2+(512*q2*q1*p2^3-1024*p0*q1*p2^2-36864*q1*q0
+9216*q2^2*q1)*p1-256*q1^2*p2^4-6144*q2*q1^2*p2+12288*p0*q1^2)*a
+(4096*q0-1024*q2^2)*p1^3+(2048*q2*q1*p2-4096*p0*q1)*p1^2
-1024*q1^2*p2^2*p1-4096*q1^3
```

$E6(a)$の判別式を計算せよ

$E6_k(a) = E6(a) \bmod a^{k+1}$ として，性能評価(1)

TRIP 1.2.26とは，直接的な多項式の操作をする計算において，世界最速の数式処理ソフト(Cayley法の実装には不向き)

k	TRIP minor Serial	TRIP minor Parallel	Kimura Serial	Kimura Parallel
7	1m59.969s	21.477s	6m28.043s	19.864s
8	7m38.860s	48.304s	16m45.179s	45.613s
9	30m28.221s	2m27.779s	41m32.633s	1m53.171s
10	94m33.848s	7m04.401s	82m55.905s	3m40.043s
11	297m54.106s	21m15.628s	165m27.601s	8m38.818s
12	785m36.664s	60m45.546s	284m07.984s	13m55.036s

CPU:E5-4650L, 4 CPU, 32コア, mem:1440Gbyte

Intel C++ Compiler:13.1.2

$E6_k(a) = E6(a) \bmod a^{k+1}$ として，性能評価(2)

sdmpとは，Maple 14の多項式演算高速化パッケージ(Cayley法の実装には不向き)

k	sdmp minor Serial	sdmp minor Parallel	Kimura Serial	Kimura Parallel
7	42.780s	43.126s	6m28.043s	19.864s
8	2m37.466s	2m55.536s	16m45.179s	45.613s
9	11m07.446s	11m24.410s	41m32.633s	1m53.171s
10	36m01.061s	37m16.865s	82m55.905s	3m40.043s
11	118m39.614s	120m03.297s	165m27.601s	8m38.818s
12	347m48.068s	346m14.324s	284m07.984s	13m55.036s

CPU:E5-4650L, 4 CPU, 32コア, mem:1440Gbyte

Intel C++ Compiler:13.1.2

$E6_k(a) = E6(a) \bmod a^{k+1}$ として，性能評価(3)

TRIP vs. sdmp

k	TRIP minor Serial	TRIP minor Parallel	sdmp minor Serial	sdmp minor Parallel
7	1m59.969s	21.477s	42.780s	43.126s
8	7m38.860s	48.304s	2m37.466s	2m55.536s
9	30m28.221s	2m27.779s	11m07.446s	11m24.410s
10	94m33.848s	7m04.401s	36m01.061s	37m16.865s
11	297m54.106s	21m15.628s	118m39.614s	120m03.297s
12	785m36.664s	60m45.546s	347m48.068s	346m14.324s

CPU:E5-4650L, 4 CPU, 32コア, mem:1440Gbyte

Intel C++ Compiler:13.1.2

$E6(a)$の判別式計算

CPU:Intel Core i7 980X(6Core)

Mem:24G

計算結果の項数:27329463項(txt形式:2.5G)

Serial: 10913m45.857s

Parallel: 1773m28.272s

Speed Up: 6.15 **“superlinear”**

どんなweightを使っているのか?

	q_0	q_1	p_0	q_2	p_1	p_2
weight_1	12	9	8	6	5	2
weight_2	2	5	6	8	9	12
weight_3	1	1	1	1	1	1

上記の3種類を使う

再び，一般係数の判別式計算へ

不変式論の立場では,

平行移動$x \leftarrow x + \alpha$することで,

3次式$x^3 + c'x + d'$の判別式を計算すれば十分であるということになっている,

$\alpha = -b/3$であるから, 計算結果に対して

$$c' = -1/3b^2 + c$$

$$d' = 2/27b^3 - 1/3cb + d$$

置換を行わなければならないため一般公式設計の立場からは, 不変式論の立場は使えない

斉重多項式

3次式$f(x) = x^3 + bx^2 + cx + d$の判別式

$$-27d^2 + 18bcd - 4c^3 - 4b^3d + b^2c^2$$

の計算結果を，じっと眺める，すると，

weight$([b, c, d]) = [1, 2, 3]$において，

weighted homogeniousになっている，

すなわち，斉重多項式になっている

どうして，斉重多項式になるのか?(1)

3次方程式$f(x) = x^3 + bx^2 + cx + d$の解$\xi_1, \xi_2, \xi_3$とすると，
解と係数の関係より，

$$\begin{aligned} -b &= \xi_1 + \xi_2 + \xi_3 \\ c &= \xi_1\xi_2 + \xi_1\xi_3 + \xi_2\xi_3 \\ -d &= \xi_1\xi_2\xi_3 \end{aligned}$$

判別式の定義式は，

$$\{(\xi_3 - \xi_1)(\xi_1 - \xi_2)(\xi_2 - \xi_3)\}^2$$

ξ_1, ξ_2, ξ_3について，6次の斉次式

終結式と根の関係の理論

$$
\begin{aligned}
&\mathrm{discriminant}(f(x)) \\
&= (-1)^{\frac{1}{2}3(3-1)}\mathrm{res}_x(f(x), f'(x)) \\
&= \{(\xi_3 - \xi_1)(\xi_1 - \xi_2)(\xi_2 - \xi_3)\}^2
\end{aligned}
$$

2行目を定義としたので，上記を証明する必要がある

$$f(x) = a_m x^m + a_{m-1}x^{m-1} + \cdots + a_0 = a_m(x-\xi_1)\cdots(x-\xi_m),$$
$$g(x) = b_n x^n + b_{n-1}x^{n-1} + \cdots + b_0 = b_n(x-\eta_1)\cdots(x-\eta_n),$$

とおく時,

$$\begin{aligned}
\mathrm{res}_x(f(x),g(x)) &= a_m^n b_n^m \prod_{i=1}^{m}\prod_{j=1}^{n}(\xi_i-\eta_j) \\
&= a_m^n \prod_{i=1}^{m} g(\xi_i) \\
&= (-1)^{mn} b_n^m \prod_{j=1}^{n} f(\eta_j)
\end{aligned}$$

cf. グレブナー基底1, コックス、リトル、オシー

$f(x) = (x-\xi_1)(x-\xi_2)(x-\xi_3)$ と改めて定義すれば,

$$f'(x) = (x-\xi_2)(x-\xi_3) + (x-\xi_1)(x-\xi_3) + (x-\xi_1)(x-\xi_2)$$

よって,

$$\begin{aligned}
\mathrm{res}_x(f(x), f'(x)) &= \prod_{i=1}^{3} f'(x)|_{x=\xi_i} \\
&= \{(\xi_1-\xi_2)(\xi_1-\xi_3)\}\{(\xi_2-\xi_1)(\xi_2-\xi_3)\} \times \\
&\quad \{(\xi_3-\xi_1)(\xi_3-\xi_2)\} \\
&= -\{(\xi_3-\xi_1)(\xi_1-\xi_2)(\xi_2-\xi_3)\}^2
\end{aligned}$$

以上より,

$$\begin{aligned}
&\mathrm{discriminant}(f(x)) \\
&= (-1)^{\frac{1}{2}3(3-1)}\mathrm{res}_x(f(x), f'(x)) \\
&= \{(\xi_3-\xi_1)(\xi_1-\xi_2)(\xi_2-\xi_3)\}^2
\end{aligned}$$

どうして，斉重多項式になるのか?(2)

判別式がb, c, dの多項式で表現できるならば(容易に証明可能),

$b^k c^l d^m$の次数は，ξ_1, ξ_2, ξ_3について6次の項を生成するため,

$k + 2l + 3m = 6$とならなければならない

判別式の公式設計問題

3次式$x^3 + x^2 + cx + d$の判別式が計算できることは，3次式

$ax^3 + bx^2 + cx + d$の判別式が計算できることに等しい

解くべき問題の再確認

17次方程式の判別式を計算したのであれば,

$$f(x) = x^{17} + x^{16} + a_2x^{15} + a_3x^{14} + a_4x^{13} + a_5x^{12} + a_6x^{11} + a_7x^{10} + a_8x^9 + a_9x^8 + a_{10}x^7 + a_{11}x^6 + a_{12}x^5 + a_{13}x^4 + a_{14}x^3 + a_{15}x^2 + a_{16}x + a_{17}$$

について, $\mathrm{res}_x(f(x), f'(x))$ を計算すればよい

より高速に計算するための方法(1)

$f = a_2x^2 + a_1x + a_0, g = b_1x + b_0$

関孝和，Bezoutの表現法

$$\mathrm{res}_x(f,g) = \begin{vmatrix} b_0a_2 - b_1a_1 & -b_1a_0 \\ b_1 & b_0 \end{vmatrix}$$

補間法を利用することを念頭に，

$a_2 = 3, a_1 = 2, a_0 = 1, b_0 = 4$を代入する

$$\mathrm{res}_x(f,g) = \begin{vmatrix} 12 - 2b_1 & -b_1 \\ b_1 & 4 \end{vmatrix}$$

mパラメータの問題においては，$m-1$パラメータに値を代入すると，行列係数多項式の行列式計算問題と思える

$$\begin{aligned}\mathrm{res}_x(f,g) &= \begin{vmatrix} 12-2b_1 & -b_1 \\ b_1 & 4 \end{vmatrix} \\ &= \left| \begin{pmatrix} 12 & 0 \\ 0 & 4 \end{pmatrix} + \begin{pmatrix} -2 & -1 \\ 1 & 0 \end{pmatrix} b_1 \right| \\ &= |A_0 + A_1 b_1|\end{aligned}$$

$$(A_0 + A_1 b_1)\, v = 0$$

とすると，行列多項式の固有多項式計算問題とみなせる

行列多項式の固有多項式計算問題の解法

A_iを，$n \times n$の行列とする

$$A = A_0 + A_1 s + \cdots + A_\alpha s^\alpha$$

Aの行列式のsの最大次数は，$n\alpha$,

方法1:$n\alpha + 1$個のサンプリング点で，$\det(A)$を評価すれば，

$\det(A)$を決められる，その計算量は，$O(n^4\alpha)$

方法2:

$$(A_0 + A_1 s + \cdots + A_\alpha s^\alpha)\, v = 0$$

を書き換える

$$|A_0 + A_1 s_0 + \cdots + A_\alpha s_0^\alpha| \neq 0$$

となるs_0を見つける(s_0が存在しないものは，そもそも方法2では扱えない),

$s = t + s_0$と変数変換する

$B_0 = A_0 + A_1 s_0 + \cdots + A_\alpha s_0^\alpha, \cdots$であり

$$(B_0 + B_1 t + \cdots + B_\alpha t^\alpha)\, v = 0$$

となる，B_0は逆行列が存在するため

$$\left(I + B_0^{-1} B_1 t + \cdots + B_0^{-1} B_\alpha t^\alpha\right) v = 0$$

さらに，$t = 1/u$として

$$\left(I u^\alpha + B_0^{-1} B_1 u^{\alpha-1} + \cdots + B_0^{-1} B_\alpha\right) v = 0$$

ブロックコンパニオン行列の理論を用いることで

$$\begin{pmatrix} 0 & I & & \\ & \ddots & & \\ & & & I \\ -B_0^{-1}B_\alpha & \cdots & \cdots & -B_0^{-1}B_1 \end{pmatrix} \begin{pmatrix} v \\ uv \\ \vdots \\ u^{\alpha-1}v \end{pmatrix} = u \begin{pmatrix} v \\ uv \\ \vdots \\ u^{\alpha-1}v \end{pmatrix}$$

右辺の固有多項式を計算することで，Aの行列式を計算できる，

右辺の固有多項式の計算量は$O(n^3\alpha^3)$

計算量の比較

$A = A_0 + A_1 s + \cdots + A_\alpha s^\alpha$, $\det(A)$を計算する

Aが一般の場合,

方法1:$O(n^4\alpha)$ 方法2:$O(n^3\alpha^3)$

高速化が期待できる場合もある

AがSylverster行列，関孝和，Bezout行列の場合，**行列式をEuclidの互除法を用いて計算できるため**

方法1:$O(n^3\alpha)$ 方法2:$O(n^3\alpha^3)$

終結式の計算では，方法２は役に立たない?

$\alpha = 1$とすると，計算量は同じ

実は，剰余算の回数を数えると，
方法1:$O(n^3)$ 方法2:$O(n^2)$であることがわかる

よって，$\alpha = 1$のときは，どちらの算法が高速であるかは，やってみるまで分からない

一般係数の判別式計算では，$\alpha = 1$に設定できる

方法2の剰余算の回数は，なぜ$O(n^2)$なのか?

$\alpha = 1$のとき,

方法1の演算量:$O(n^3)$

方法1の剰余算の回数:$O(n^3)$

方法2の演算量:$O(n^3)$

方法2の剰余算の回数:$O(n^2)$

有限体$\mathbb{Z}/p\mathbb{Z}$の実装法

unsigned long longは，0から$2^{64}-1$までの数を表現できる

有限体$\mathbb{Z}/p\mathbb{Z}$において，長さnのベクトルx, yの内積を計算することを考える，

与えられたnに対して，$n(p-1)^2 \leq 2^{64}-1$となるpを採用すれば，

$s_0 = 0, s_i = (s_{i-1} + x_i \times y_i) \bmod p, i = 1, \ldots, n$

と計算するのではなく，

$s_n = (\Sigma_{i=1}^{n}(x_i \times y_i)) \bmod p$と計算すればよい

四則演算以外に，内積も基本演算に加えよう

タイミングデータ(1)

k	Cayley method Singular-3-1-6	Kimura Serial	Kimura Parallel
12	17.197s	2m41.004s	8.912s
13	1m55.036s	18m42.406s	56.499s
14	15m26.978s	177m38.849s	7m46.184s
15	95m09.931s	1265m33.327s	52m55.039s
16	613m47.301s	???	372m10.574s

CPU:E5-4650L, 4 CPU, 32コア, mem:1440Gbyte

Intel C++ Compiler:13.1.2

???は, あまりにも時間を必要とするため測定していない

タイミングデータ(2)

k	TRIP minor Serial	TRIP minor Parallel	Kimura Serial	Kimura Parallel
12	2m59.687s	2m54.455s	2m41.004s	8.912s
13	25m35.621s	20m26.959s	18m42.406s	56.499s
14	237m14.666s	141m35.767s	177m38.849s	7m46.184s
15	-	-	1265m33.327s	52m55.039s
16	-	-	???	372m10.574s

CPU:E5-4650L, 4 CPU, 32コア, mem:1440Gbyte

Intel C++ Compiler:13.1.2

???は, あまりにも時間を必要とするため測定していない

-は, out of memoryを意味する

TRIP minor Serial 14次:483060 MB

タイミングデータ(3)

k	sdmp minor Serial	sdmp minor Parallel	Kimura Serial	Kimura Parallel
12	2m55.216s	3m00.169s	2m41.004s	8.912s
13	31m41.283s	31m47.976s	18m42.406s	56.499s
14	316m12.479s	323m23.322s	177m38.849s	7m46.184s
15	-	-	1265m33.327s	52m55.039s
16	-	-	???	372m10.574s

CPU:E5-4650L, 4 CPU, 32コア, mem:1440Gbyte

Intel C++ Compiler:13.1.2

???は, あまりにも時間を必要とするため測定していない

-は, out of memoryを意味する

sdmp Serial 14次:421558 MB

タイミングデータ(4)

TRIP vs. sdmp

k	TRIP minor Serial	TRIP minor Parallel	sdmp minor Serial	sdmp minor Parallel
12	2m59.687s	2m54.455s	2m55.216s	3m00.169s
13	25m35.621s	20m26.959s	31m41.283s	31m47.976s
14	237m14.666s	141m35.767s	316m12.479s	323m23.322s

どんなweightを使っているのか?(2)

17次の判別式ならば

	a_{15}	a_{14}	a_{13}	a_{12}	a_{11}	a_{10}	a_9	a_8	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0
weight_1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
weight_2	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	17

上記の2種類を使う

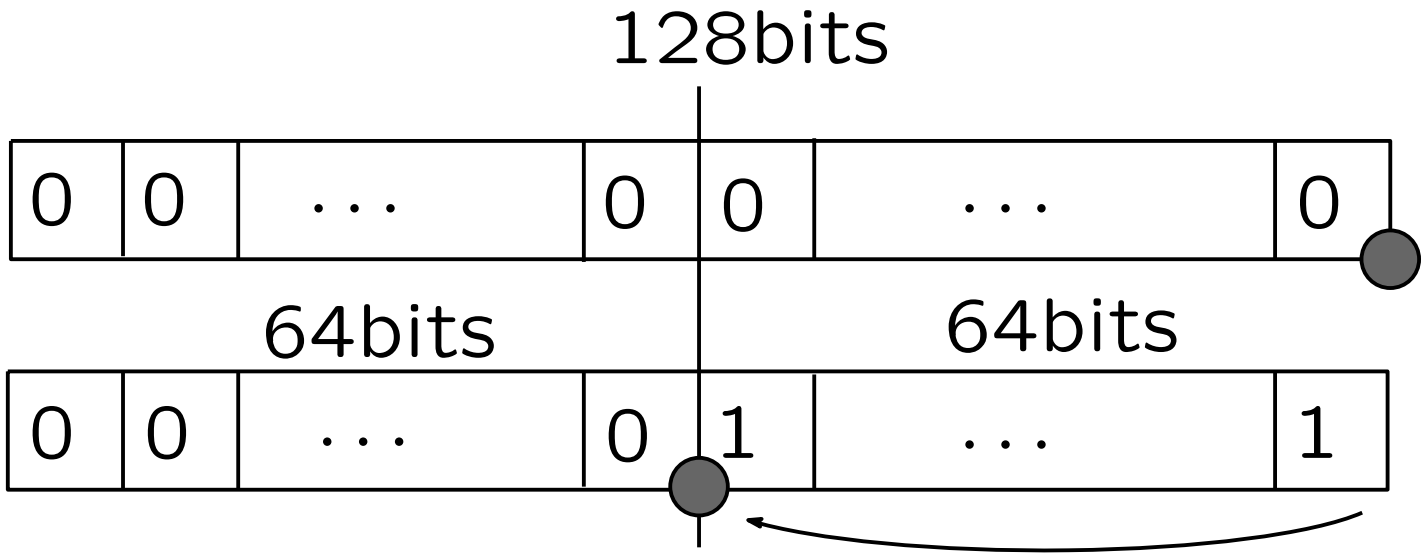
整数の剰余計算の高速化

$0 \leq a_0, \cdots, a_{99} < 2^{64}, 0 < b < 2^{64}$の条件を満たす整数が与えられているとき，$a_i \bmod b$を考える．$c_i = a_i \bmod b$が，$0 \leq c_i < b$の範囲にある必要があるときは，やはりCPUの命令をそのまま呼び出す以外に方法はない．**もし，c_iが，$0 \leq c_i < 2b$の範囲にあればよいという緩い条件の場合には，次のようにして演算を高速化することが可能である．**

$$M = \frac{2^{64} - 1}{b} \quad \text{(切り捨て演算)}$$

を用意する，Mを，擬似逆元という

擬似逆元の意味を考える，原点を移動する



$0.1111111111111111111111111111111111111111111111111111111111111111_{(2)} \approx 1.0$

より，

$$M = \frac{2^{64}-1}{b} \approx 1.0/b \times 2^{64}$$

bが一定ならば，Mも一定． 以下，証明は省略するが，

$$a_i \times M\text{の結果の上位64bits}$$

は，a_i/bの除算をした時の真の商に等しいか，または，真の商より1小さい(ただし，負の数にはならない)，以上より，

$$c_i = a_i - b \times (a_i \times M\text{の結果の上位64bits})$$

SIMD演算の活用

```
int
calbound0 (int PRODH, short min,
           short * restrict tdegree)
{
  int min0;
  min0 = __sec_reduce_min (tdegree[PRODH:PRODH]);
  min = min0 < min ? min0 : min;
  return min;
}
```

17 次方程式の判別式の計算

子問題0:871m13.504s

子問題1:852m10.139s

子問題2:851m32.733s

子問題3:877m35.759s

子問題4:869m11.776s

子問題5:856m06.433s

子問題6:842m22.586s

子問題7:873m54.573s

子問題8:848m30.900s

子問題9:777m07.483s

子問題10:863m36.559s

子問題11:850m15.090s

子問題12:849m59.693s

子問題13:869m23.785s

子問題14:851m27.902s

子問題15:818m04.673s

子問題16:880m35.452s

子問題17:791m25.893s

計算結果の統合:612m15.268s

項数のカウント:20m34.528s

検算１回:109m47.591s

まとめ

・17次方程式の判別式を，1通りの方法で計算した

1通りの計算であるために，報道機関での発表予定はなし