ネイピア数の数値計算

2020 年 10 月 3 日

1　はじめに

ネイピア数 e を複数の方法で数値計算しました. 今回は約 20 通りの方法を試しましたが、他にも数多くの方法があると思います. プログラミング言語には Python を用いました. ソースコードとその出力結果を添付しておきます.

2　方法 A

Lax and Terrell (2013) の 1.4 節を参考にする. ネイピア数 e は

$$e = \lim_{n\to\infty} \left(1 + \frac{1}{n}\right)^n$$

で定義される. 数列 e_n を

$$e_n = \left(1 + \frac{1}{n}\right)^n, \quad n \geq 1$$

とおくと, $\lim_{n\to\infty} e_n = e$ が成立する. したがって, 数列 e_n を計算する.

3　方法 B

Lax and Terrell (2013) の Problem 1.55 及び Perkins (2018) の 3.5 節を参考にする. ネイピア数 e の性質として,

$$e = \sum_{k=0}^{\infty} \frac{1}{k!}$$

が知られている. 数列 g_n を

$$g_n = \sum_{k=0}^{n} \frac{1}{k!}, \quad n \geq 0$$

とおくと, $\lim_{n\to\infty} g_n = e$ が成立する. したがって, 数列 g_n を計算する.

4 方法 C

Schinazi (2011) の Application 3.2 を参考にする. ネイピア数 e の性質として,

$$\frac{1}{e} = \sum_{k=0}^{\infty} \frac{(-1)^k}{k!}$$

が知られている. 数列 a_n 及び b_n を

$$a_n = \sum_{k=0}^{n} \frac{(-1)^k}{k!}, \quad n \geq 0$$
$$b_n = \frac{1}{a_n}, \quad n \geq 2$$

で定める. $\lim_{n\to\infty} b_n = e$ なので, 数列 b_n を計算する.

5 方法 D

Loya (2018) の 7.7.3 節を参考にする. 数列 a_n を $a_1 = 0,\ a_2 = 1$,

$$a_{n+2} = a_{n+1} + \frac{1}{n} a_n, \quad n \geq 1$$

を定める. このときネイピア数 e の性質として

$$\lim_{n\to\infty} \frac{n}{a_n} = e$$

が知られている. 数列 u_n を

$$u_n = \frac{n}{a_n}, \quad n \geq 2$$

と定める. $\lim_{n\to\infty} u_n = e$ なので, 数列 u_n を計算する.

6　方法 E

Stein (2008) の 5.4 節を参考にする. 数列 p_n 及び q_n を $p_0 = 1$, $p_1 = 3$, $q_0 = 1$, $q_1 = 1$,

$$p_n = 2(2n-1)p_{n-1} + p_{n-2}, \quad n \geq 2$$
$$q_n = 2(2n-1)q_{n-1} + q_{n-2}, \quad n \geq 2$$

と定める. このときネイピア数 e の性質として

$$\lim_{n\to\infty} \frac{p_n}{q_n} = e$$

が知られている. 数列 r_n を

$$r_n = \frac{p_n}{q_n}, \quad n \geq 0$$

と定める. $\lim_{n\to\infty} r_n = e$ なので, 数列 r_n を計算する.

7　方法 F2

Brothers (2004) の式 (2) を参考にする. ネイピア数 e の性質として

$$e = \sum_{k=0}^{\infty} \frac{2k+2}{(2k+1)!}$$

が知られている. 数列 a_n を

$$a_n = \sum_{k=0}^{n} \frac{2k+2}{(2k+1)!}, \quad n \geq 0$$

と定める. $\lim_{n\to\infty} a_n = e$ なので, 数列 a_n を計算する.

8　方法 F3

Brothers (2004) の式 (3) を参考にする. ネイピア数 e の性質として

$$e = \sum_{k=0}^{\infty} \frac{2k+1}{(2k)!}$$

が知られている. 数列 a_n を

$$a_n = \sum_{k=0}^{n} \frac{2k+1}{(2k)!}, \quad n \geq 0$$

と定める．$\lim_{n\to\infty} a_n = e$ なので, 数列 a_n を計算する．

9　方法 F4

Brothers (2004) の式 (4) を参考にする．ネイピア数 e の性質として

$$e = \frac{1}{2}\sum_{k=0}^{\infty}\frac{k+1}{k!}$$

が知られている．数列 a_n を

$$a_n = \frac{1}{2}\sum_{k=0}^{n}\frac{k+1}{k!}, \quad n \geq 0$$

と定める．$\lim_{n\to\infty} a_n = e$ なので, 数列 a_n を計算する．

10　方法 F9

Brothers (2004) の式 (9) を参考にする．ネイピア数 e の性質として,

$$\frac{1}{e} = \sum_{k=0}^{\infty}\frac{1-2k}{(2k)!}$$

が知られている．数列 a_n 及び b_n を

$$a_n = \sum_{k=0}^{n}\frac{1-2k}{(2k)!}, \quad n \geq 0,$$
$$b_n = \frac{1}{a_n}, \quad n \geq 0$$

と定める．$\lim_{n\to\infty} b_n = e$ なので, 数列 b_n を計算する．

11　方法 F10

Brothers (2004) の式 (10) を参考にする．ネイピア数 e の性質として

$$e = \sum_{k=0}^{\infty}\frac{3-4k^2}{(2k+1)!}$$

が知られている. 数列 a_n を

$$a_n = \sum_{k=0}^{n} \frac{3 - 4k^2}{(2k+1)!}, \quad n \geq 0$$

と定める. $\lim_{n\to\infty} a_n = e$ なので, 数列 a_n を計算する.

12　方法 F11

Brothers (2004) の式 (11) を参考にする. ネイピア数 e の性質として

$$e = \sum_{k=0}^{\infty} \frac{3k^2 + 1}{(3k)!}$$

が知られている. 数列 a_n を

$$a_n = \sum_{k=0}^{n} \frac{3k^2 + 1}{(3k)!}, \quad n \geq 0$$

と定める. $\lim_{n\to\infty} a_n = e$ なので, 数列 a_n を計算する.

13　方法 G

Coffey (2019) を参考にする. ネイピア数 e の性質として

$$e = 3 - \sum_{k=2}^{\infty} \frac{1}{k!(k-1)k}$$

が知られている. 数列 a_n を

$$a_n = 3 - \sum_{k=2}^{n} \frac{1}{k!(k-1)k}, \quad n \geq 2$$

と定める. $\lim_{n\to\infty} a_n = e$ なので, 数列 a_n を計算する.

14　方法 H2, H3, H4 及び H5

Mezo (2019) の 2.4 節を参考にする. ネイピア数 e の性質として

$$e = \frac{1}{B_n} \sum_{k=0}^{\infty} \frac{k^n}{k!}$$

が知られている．ここで B_n は n 番目のベル数を表し，$B_0 = 1, B_1 = 1, B_2 = 2, B_3 = 5, B_4 = 15, B_5 = 52, B_6 = 203, B_7 = 877, B_8 = 4140, \ldots$ である．この節では上式を用いた数値計算を説明する．

14.1　方法 H2

ネイピア数 e の性質として

$$e = \frac{1}{2}\sum_{k=0}^{\infty}\frac{k^2}{k!}$$

が成立する．数列 a_n を

$$a_n = \frac{1}{2}\sum_{k=0}^{n}\frac{k^2}{k!}, \quad n \geq 0$$

と定める．$\lim_{n\to\infty} a_n = e$ なので，数列 a_n を計算する．

14.2　方法 H3

ネイピア数 e の性質として

$$e = \frac{1}{5}\sum_{k=0}^{\infty}\frac{k^3}{k!}$$

が成立する．数列 a_n を

$$a_n = \frac{1}{5}\sum_{k=0}^{n}\frac{k^3}{k!}, \quad n \geq 0$$

と定める．$\lim_{n\to\infty} a_n = e$ なので，数列 a_n を計算する．

14.3　方法 H4

ネイピア数 e の性質として

$$e = \frac{1}{15}\sum_{k=0}^{\infty}\frac{k^4}{k!}$$

が成立する．数列 a_n を

$$a_n = \frac{1}{15}\sum_{k=0}^{n}\frac{k^4}{k!}, \quad n \geq 0$$

と定める．$\lim_{n\to\infty} a_n = e$ なので，数列 a_n を計算する．

14.4　方法 H5

ネイピア数 e の性質として

$$e = \frac{1}{52}\sum_{k=0}^{\infty}\frac{k^5}{k!}$$

が成立する. 数列 a_n を

$$a_n = \frac{1}{52}\sum_{k=0}^{n}\frac{k^5}{k!}, \quad n \geq 0$$

と定める. $\lim_{n\to\infty} a_n = e$ なので, 数列 a_n を計算する.

15　方法 I

Loya (2018) の式 8.15 及び Theorem 8.5 を参考にする. 数列 p_n 及び q_n を $p_{-1} = 1,\ p_0 = 2,\ q_{-1} = 0,\ q_0 = 1$,

$$\begin{aligned} p_n &= (n+1)p_{n-1} + (n+1)p_{n-2}, \quad n \geq 1 \\ q_n &= (n+1)q_{n-1} + (n+1)q_{n-2}, \quad n \geq 1 \end{aligned}$$

と定める. このときネイピア数 e の性質として

$$\lim_{n\to\infty}\frac{p_n}{q_n} = e$$

が知られている. 数列 r_n を

$$r_n = \frac{p_n}{q_n}, \quad n \geq 0$$

と定める. $\lim_{n\to\infty} r_n = e$ なので, 数列 r_n を計算する.

16　方法 J

Lu (2019) の 2.2 節を参考にする. 数列 p_n 及び q_n を $p_{-1} = 1,\ p_0 = 3,\ q_{-1} = 0,\ q_0 = 1$,

$$\begin{aligned} p_n &= (n+3)p_{n-1} - np_{n-2}, \quad n \geq 1 \\ q_n &= (n+3)q_{n-1} - nq_{n-2}, \quad n \geq 1 \end{aligned}$$

と定める．このときネイピア数 e の性質として

$$\lim_{n\to\infty}\frac{p_n}{q_n}=e$$

が知られている．数列 r_n を

$$r_n=\frac{p_n}{q_n},\quad n\geq 0$$

と定める．$\lim_{n\to\infty} r_n=e$ なので, 数列 r_n を計算する．

17　方法 K

Kadyrov and Mashurov (2019) の Theorem 2 及び Loya (2018) の Theorem 8.5 を参考にする．数列 p_n 及び q_n を $p_{-1}=1,\ p_0=3,\ q_{-1}=0,\ q_0=1$,

$$p_n=p_{n-1}+a_np_{n-2},\quad n\geq 1$$
$$q_n=q_{n-1}+a_nq_{n-2},\quad n\geq 1$$
$$a_n=\begin{cases}-1 & n\text{ が奇数のとき}\\ \frac{n}{2}+1 & n\text{ が偶数のとき}\end{cases}$$

と定める．このときネイピア数 e の性質として

$$\lim_{n\to\infty}\frac{p_n}{q_n}=e$$

が知られている．数列 r_n を

$$r_n=\frac{p_n}{q_n},\quad n\geq 0$$

と定める．$\lim_{n\to\infty} r_n=e$ なので, 数列 r_n を計算する．

18　方法 L

Ruiz (1997) を参考にする．ネイピア数 e の性質として

$$\lim_{n\to\infty}\left(\prod_{i=1}^{n}p_i\right)^{\frac{1}{p_n}}=e$$

が知られている．ここで p_n は n 番目の素数を表し, $p_1=2,\ p_2=3,\ p_3=5,\ p_4=7,\ p_5=11,\ldots$ である．数列 a_n を

$$a_n=\left(\prod_{i=1}^{n}p_i\right)^{\frac{1}{p_n}},\quad n\geq 1$$

と定める. $\lim_{n\to\infty} a_n = e$ なので, 数列 a_n を計算する.

参考文献

Brothers, H. J. (2004). Improving the convergence of newton's series approximation for e. *The College Mathematics Journal*, 35(1):34–39.

Coffey, J. (2019). A continued fraction for e. `http://www.mathstudio.co.uk/Q24-Continued%20fraction%20for%20e-B.pdf`. MathStudio (2020 年 9 月 29 日閲覧).

Kadyrov, S. and Mashurov, F. (2019). Generalized continued fraction expansions for π and e. *arXiv preprint arXiv:1912.03214*.

Lax, P. and Terrell, M. (2013). *Calculus With Applications*. Undergraduate Texts in Mathematics. Springer New York.

Loya, P. (2018). *Amazing and Aesthetic Aspects of Analysis*. Undergraduate Texts in Mathematics. Springer New York.

Lu, Z. (2019). Elementary proofs of generalized continued fraction formulae for e. *arXiv preprint arXiv:1907.05563*.

Mezo, I. (2019). *Combinatorics and Number Theory of Counting Sequences*. Discrete Mathematics and Its Applications. CRC Press.

Perkins, D. (2018). *Phi, Pi, e and i*. Spectrum. American Mathematical Society.

Ruiz, S. M. (1997). 81.27 a result on prime numbers. *The Mathematical Gazette*, 81(491):269–270.

Schinazi, R. (2011). *From Calculus to Analysis*. SpringerLink : Bücher. Birkhäuser Boston.

Stein, W. (2008). *Elementary Number Theory: Primes, Congruences, and Secrets: A Computational Approach*. Undergraduate Texts in Mathematics. Springer New York.

```
# method_A.py
# calculate the number e
# [0] init
from decimal import *
getcontext().prec = 200
from fractions import Fraction
N = 30

# [1] calculate a sequence e[n]
e = {}
for n in range(1, N+1):
    e[n] = (1 + Fraction(1, n)) ** n

# [2] print result
# [2.1] print e[n] as fraction
print("n, e[n]")
for n in range(1, N+1):
    print(f"{n:0>2d},", e[n].numerator, "/", e[n].denominator)

# [2.2] print e[n] as decimal
print("\nn, e[n]")
decimal_from_fraction = lambda x: Decimal(x.numerator) / Decimal(x.denominator)
for n in range(1, N+1):
    print(f"{n:0>2d},", str(decimal_from_fraction(e[n]))[:2 + 100])
```

method_A.py の出力(1/2)
n, e[n]
01, 2 / 1
02, 9 / 4
03, 64 / 27
04, 625 / 256
05, 7776 / 3125
06, 117649 / 46656
07, 2097152 / 823543
08, 43046721 / 16777216
09, 1000000000 / 387420489
10, 25937424601 / 10000000000
11, 743008370688 / 285311670611
12, 23298085122481 / 8916100448256
13, 793714773254144 / 302875106592253
14, 29192926025390625 / 11112006825558016
15, 1152921504606846976 / 437893890380859375
16, 48661191875666868481 / 18446744073709551616
17, 2185911559738696531968 / 827240261886336764177
18, 1041273502979112415328 41 / 39346408075296537575424
19, 52428800000000000000000000 / 19784196556603135891239 79
20, 27821842944695154863719640 1 / 10485760000000000000000000 0
21, 15519448971100888972574851072 / 584258701838598252138112 4421
22, 90784643477599617540674056 1329 / 34142787736421955739664672 3584
23, 55572324035428505185378394701824 / 20880467999847912034355032910567
24, 35527136788005009293556213378906 25 / 13337357768502841244490814728 43776
25, 23677383000796758887679516493846 9376 / 88817841970012523233890533447265625
26, 16423203268260658146231467800709255289 / 61561195802071573107966742884002037 76
27, 11837686826161919593775974376201644933 12 / 44342648824303776994824963061914989 2803
28, 88540901833145211536614766025207452637361 / 33145523113253374862572728253364605812736
29, 68630377364883000000000000000000000000000000 / 25676861531612111345618282147310161264834 69
30, 55061852034591083737453687190513918567886 2401 / 20589113209464900000000000000000000000000000000

method_A.py の出力(2/2)
n, e[n]
01, 2
02, 2.25
03, 2.3703703703703703703703703703703703703703703703703703703703703703703703703703703703703703703703703703
04, 2.44140625
05, 2.48832
06, 2.5216263717421124828532235939643347050754458161865569272976680384087791495198902606310013717421124828
07, 2.5464996970407131139479055738437458639075312400202539515240855668738608670099800496148956399362267665
08, 2.565784513950347900390625
09, 2.5811747917131971819900315081167532159095488622957161153136637541129116689540908612089434433603226390
10, 2.5937424601
11, 2.6041990118975308781817744553909617778905515982184429171387604917394978605227287310771856801785904846
12, 2.6130352902246781602995330443548766467020798169505175551001898611510967351032381769566141867112247643
13, 2.6206008878857322210793099478737464988024800389000434704638041612023425725699259164856674162703025705
14, 2.6271515563008693883842367121090552216121189649698484435230163792613635739114199962651881344002040583
15, 2.6328787177279190470443497891511876131053048152029976917405625390142956533204933309554518682154675613
16, 2.6379284973665998587631122129781857665875577367842197418212890625
17, 2.6424143751831096202574857116586844468956880036061964617429648061002606354850791747282053786629624053
18, 2.6464258210976854673490605270517860168008192400085816101126014886527599527247514197742462505974770193
19, 2.6500343266404449072632676129300099758062349272839852269853619567364108053338661681044565906613890509
20, 2.6532977051444201339454307651519775390625
21, 2.6562632139261049855383983572429240899618522554923002898545589081837890071708064459311723174496495726
22, 2.6589698585377882029213537001360108323939495206752861139639523406650777383601836769642600165546015552
23, 2.6614501186387814544955469044088252834580795308609836042586488219783821861062967580566599018296912076
24, 2.6637312580685940367405399559557309012795297472665259958165355907762898668205508246995634809225829896
25, 2.66583633148741999304064003395197815283391475482624
26, 2.6677849665337408840230069613300133011485855579403189190471504949809648014704376742236680902792007057
27, 2.6695939778125744088223082194170053019726603258113575086865728605041355110805404427464464678575165958
28, 2.6712778534408396403577865220737372674282453604895071747248539591475888012754682457649794996304766625
29, 2.6728491439808013627537148911336088635500295417439411501102166935832484094528295243552667764074635447
30, 2.6743187758702945960644354448555678214384083442150198515028446493514215890732841180153535292329385211

```
# method_B.py
# calculate the number e
# [0] init
from decimal import *
getcontext().prec = 200
from fractions import Fraction
from math import *
N = 30

# [1] calculate a sequence g[n]
g = {}
g[0] = 1
for n in range(1, N+1):
    g[n] = g[n-1] + Fraction(1, factorial(n))

# [2] print result
# [2.1] print g[n] as fraction
print("n, g[n]")
for n in range(0, N+1):
    print(f"{n:0>2d},", g[n].numerator, "/", g[n].denominator)

# [2.2] print g[n] as decimal
print("\nn, g[n]")
decimal_from_fraction = lambda x: Decimal(x.numerator) / Decimal(x.denominator)
for n in range(0, N+1):
    print(f"{n:0>2d},", str(decimal_from_fraction(g[n]))[:2 + 100])
```

method_B.py の出力(1/2)
n, g[n]
00, 1 / 1
01, 2 / 1
02, 5 / 2
03, 8 / 3
04, 65 / 24
05, 163 / 60
06, 1957 / 720
07, 685 / 252
08, 109601 / 40320
09, 98641 / 36288
10, 9864101 / 3628800
11, 13563139 / 4989600
12, 260412269 / 95800320
13, 8463398743 / 3113510400
14, 47395032961 / 17435658240
15, 888656868019 / 326918592000
16, 56874039553217 / 20922789888000
17, 7437374403113 / 2736057139200
18, 17403456103284421 / 6402373705728000
19, 82666416490601 / 30411275102208
20, 6613313319248080001 / 2432902008176640000
21, 69439789852104840011 / 25545471085854720000
22, 611070150698522592097 / 224800145555521536000
23, 1351405140967886501753 / 49715416805548032000
24, 337310723185584470837549 / 1240896803466478878720000
25, 8535190364007704221597 9 / 31399210614030336000000
26, 109625985035314953022203 4277 / 403291461126605635584000000
27, 739975398988375932899873137 / 27222173626045880401920000
28, 828772446866981044847857913441 / 3048883446117138605015040000000
29, 240344009591424503005878794 8979 / 884176199373970195454361600000
30, 554640022134056545398181834 37977 / 204040661393993122027929600000000

method_B.py の出力(2/2)
n, g[n]
00, 1
01, 2
02, 2.5
03, 2.6666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666
04, 2.708333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333
05, 2.716666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666
06, 2.718055555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555
07, 2.718253968253968253968253968253968253968253968253968253968253968253968253968253968253968253968253968253968253968253968
08, 2.718278769841269841269841269841269841269841269841269841269841269841269841269841269841269841269841269841269841269841269
09, 2.718281525573192239858906525573192239858906525573192239858906525573192239858906525573192239858906525573192239858906525
10, 2.7182818011463844797178130511463844797178130511463844797178130511463844797178130511463844797178130511
11, 2.7182818261984928651595318261984928651595318261984928651595318261984928651595318261984928651595318261
12, 2.7182818282861685639463417241195018972796750574528352306130083907861685639463417241195018972796750574
13, 2.7182818284467590023145578701134256689812245367800923356478912034467590023145578701134256689812245367
14, 2.7182818284582297479122875948272773669599066424463249860075256900653726050551447376844202241027637853
15, 2.7182818284589944642854695764748674801584854494907404960315013225066135119045171955224865277775330685
16, 2.7182818284590422590587934503278418622333966249310164654079997995341910685826029741373656717572061487
17, 2.7182818284590450705160477958486050611789796352510326989007350040652250425048433140558879743442457416
18, 2.7182818284590452267081174817108696833426231358243669340947758487613935966116344440513614355990812746
19, 2.7182818284590452349287527283351994002986043726966476833155148405875077310383076614195442493493357763
20, 2.7182818284590452353397844906664158861464034345402617207765517901788134377596413222879533900368485014
21, 2.7182818284590452353593574317298071473772510089137671511318392639688756142701810204245443014981586312
22, 2.7182818284590452353602471108690522047059258986580173979661705127775148041115691885216620702009454553
23, 2.7182818284590452353602857925707585115463030677773326260894023062039773775829338914824063210141100998
24, 2.7182818284590452353602874043083296076646521164906374272612036309300799848109074207724373314646586266
25, 2.7182818284590452353602874687778324515093860784391696193080756839191240891000263619440385718826805677
26, 2.7182818284590452353602874712574287147341835385141131651560323013417796315726847827583309272833737193
27, 2.7182818284590452353602874713492656133721389999983703335207714353203965035161165761218232367426586508
28, 2.7182818284590452353602874713525455026092088379085223752480835472482042489426677115990908192233473984
29, 2.7182818284590452353602874713526586022380733150778379628938529304181286539573763714431345289640608035
30, 2.7182818284590452353602874713526623722257021309834818158153785765237928007911999934379359859554179170

```
# method_C.py
# calculate the number e
# [0] init
from decimal import *
getcontext().prec = 200
from fractions import Fraction
from math import *
N = 30

# [1] calculate sequences a[n] and b[n]
# [1.1] calculate a[n]
a = {}
a[0] = 1
for n in range(1, N+1):
    a[n] = a[n-1] + Fraction((-1) ** n, factorial(n))

# [1.2] calculate b[n]
b = {}
for n in range(2, N+1):
    b[n] = 1 / a[n]

# [2] print result
# [2.1] print b[n] as fraction
print("n, b[n]")
for n in range(2, N+1):
    print(f"{n:0>2d},", b[n].numerator, "/", b[n].denominator)

# [2.2] print b[n] as decimal
print("\nn, b[n]")
decimal_from_fraction = lambda x: Decimal(x.numerator) / Decimal(x.denominator)
for n in range(2, N+1):
    print(f"{n:0>2d},", str(decimal_from_fraction(b[n]))[:2 + 100])
```

method_C.py の出力(1/2)
n, b[n]
02, 2 / 1
03, 3 / 1
04, 8 / 3
05, 30 / 11
06, 144 / 53
07, 280 / 103
08, 5760 / 2119
09, 45360 / 16687
10, 44800 / 16481
11, 3991680 / 1468457
12, 43545600 / 16019531
13, 172972800 / 63633137
14, 6706022400 / 2467007773
15, 93405312000 / 34361893981
16, 42268262400 / 15549624751
17, 22230464256000 / 8178130767479
18, 376610217984000 / 138547156531409
19, 250298560512000 / 92079694567171
20, 11640679464960000 / 4282366656425369
21, 196503623737344000 / 72289643288657479
22, 1784128139329536000 / 6563440628747948887
23, 10682651544993792000 / 39299278806015611311
24, 26976017466662584320000 / 9923922230666898717143
25, 215433472824041472000000 / 79253545592131482810517
26, 1613165844506422542336000 / 593450549393880543285151 3
27, 3807297010635787468800000 / 1400626296646396387124045 9
28, 1254684545727217532928000000 / 4615726495285737558884510 11
29, 3157772140621322126622720000 00 / 1161679450438521163480683669 47
30, 9146650338351415815045120000000 / 3364864615063302680426807870189

method_C.py の出力(2/2)
n, b[n]
02, 2
03, 3
04, 2.6666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666
05, 2.7272727272727272727272727272727272727272727272727272727272727272727272727272727272727272727272727272727272727272727272
06, 2.71698113207547169811320754716981132075471698113207547169811320754716981132075471698113207547169811320754716981132
07, 2.718446601941747572815533980582524271844660194174757281553398058252427184466019417475728155339805825242718446601941747572815533980582524
08, 2.7182633317602642756016989145823501651722510618216139688532326569136385087305332704105710240679565832
09, 2.718283693893449991010966620722718283693893449991010966620722718283693893449991010966620722718283693893449991010966620722718283693893
10, 2.7182816576664037376372792913051392512590255445664704811601237788969115951701959832534433590194769734
11, 2.718281842777827338492036198540372649658791507003609911628328238416242355070662607076679807444140345 9
12, 2.7182818273518744088075986743931517096224602330742391896491851103506088911092340968034582285835958618
13, 2.7182818285384861664135778815996451660083959085656896028872503959690058970375765067185042283865401763
14, 2.718281828453728183693079916372034876438145701780891794539149188161070297527595183624903803657370965 2
15, 2.718281828459378715874282005573131623824038944205367141284528020615104405819044308516924063086323390 6
16, 2.7182818284590255576129561867650242693628330632632898061888413219625225154090922666536314796501033583
17, 2.718281828459046331628328291213545319983841216038572064265639050671927082274318801876233351002038853 1
18, 2.7182818284590451775163631742914084928093139944691124479896463780625710417335970814398449907490949385
19, 2.718281828459045238259098180445180735491219128841911176155780842500949490317528386086047459445188603 4
20, 2.7182818284590452352219614301374920588826491041137852613693205622504719693380871936423528860317060144
21, 2.718281828459045235366586989675953424281542760219088100967057495303327495443394911411108195272747103 7
22, 2.718281828459045235360013100606023362217622820969826735708504657238133667655448688751341174433005772 1
23, 2.718281828459045235360298921869933364916053461500028805401887519915469480179029749298089875764257440 5
24, 2.718281828459045235360287012650603781470285516944552060881652600529825902027536270185856067390015162 4
25, 2.718281828459045235360287489019376964808116234724767566154719927378115574383946095963589517491144189 2
26, 2.718281828459045235360287470697501073141276591733217728605219954270280813714035282804537115176171770 7
27, 2.718281828459045235360287471376089069128937319251423273739645114757185233431298107571442430492670692 1
28, 2.718281828459045235360287471351853783557949436125773075693295942000394229065988494629829557761492474 9
29, 2.718281828459045235360287471352689483060397294164588599763852512687489138684851665484402615760071073
30, 2.7182818284590452353602874713526616264103156988966280822948339520525688703235316916282485595177415522

```
# method_D.py
# calculate the number e
# [0] init
from decimal import *
getcontext().prec = 200
from fractions import Fraction
N = 30

# [1] calculate sequences a[n] and u[n]
# [1.1] calculate a[n]
a = {}
a[1] = Fraction(0)
a[2] = Fraction(1)
for n in range(1, N-1):
    a[n+2] = a[n+1] + a[n] / n

# [1.2] calculate u[n]
u = {}
for n in range(2, N+1):
    u[n] = Fraction(n) / a[n]

# [2] print result
# [2.1] print u[n] as fraction
print("n, u[n]")
for n in range(2, N+1):
    print(f"{n:0>2d},", u[n].numerator, "/", u[n].denominator)

# [2.2] print u[n] as decimal
print("\nn, u[n]")
decimal_from_fraction = lambda x: Decimal(x.numerator) / Decimal(x.denominator)
for n in range(2, N+1):
    print(f"{n:0>2d},", str(decimal_from_fraction(u[n]))[:2 + 100])
```

method_D.py の出力(1/2)
n, u[n]
02, 2 / 1
03, 3 / 1
04, 8 / 3
05, 30 / 11
06, 144 / 53
07, 280 / 103
08, 5760 / 2119
09, 45360 / 16687
10, 44800 / 16481
11, 3991680 / 1468457
12, 43545600 / 16019531
13, 172972800 / 63633137
14, 6706022400 / 2467007773
15, 93405312000 / 34361893981
16, 42268262400 / 15549624751
17, 22230464256000 / 8178130767479
18, 376610217984000 / 138547156531409
19, 250298560512000 / 92079694567171
20, 11640679464960000 / 4282366656425369
21, 196503623737344000 / 72289643288657479
22, 1784128139329536000 / 6563440628747948887
23, 10682651544993792000 / 39299278806015611311
24, 26976017466662584320000 / 9923922230666898717143
25, 215433472824041472000000 / 79253545592131482810517
26, 1613165844506422542336000 / 593450549393880543285151 3
27, 3807297010635787468800000 / 1400626296646396387124045 9
28, 1254684545727217532928000000 / 4615726495285737558884510 11
29, 3157772140621322126622720000 00 / 1161679450438521163480683669 47
30, 9146650338351415815045120000000 / 3364864615063302680426807870189

method_D.py の出力(2/2)
n, u[n]
02, 2
03, 3
04, 2.6666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666
05, 2.7272727272727272727272727272727272727272727272727272727272727272727272727272727272727272727272727272727272727272727272
06, 2.7169811320754716981132075471698113207547169811320754716981132075471698113207547169811320754716981132075471698113207547169811320754716981132
07, 2.7184466019417475728155339805825242718446601941747572815533980582524271844660194174757281553398058252
08, 2.7182633317602642756016989145823501651722510618216139688532326569136385087305332704105710240679565832
09, 2.7182836938934499910109666207227182836938934499910109666207227182836938934499910109666207227182836938
10, 2.7182816576664037376372792913051392512590255445664704811601237788969115951701959832534433590194769734
11, 2.718281842777827338492036198540372649658791507003609911628328238416242355070662607076679807444140345 9
12, 2.7182818273518744088075986743931517096224602330742391896491851103506088911092340968034582285835958618
13, 2.7182818285384861664135778815996451660083959085656896028872503959690058970375765067185042283865401763
14, 2.7182818284537281836930799163720348764381457017808917945391491881610702975275951836249038036573709652
15, 2.7182818284593787158742820055731316238240389442053671412845280206151044058190443085169240630863233906
16, 2.7182818284590255576129561867650242693628330632632898061888413219625225154090922666536314796501033583
17, 2.7182818284590463316283282912135453199838412160385720642656390506719270822743188018762333510020388531
18, 2.7182818284590451775163631742914084928093139944691124479896463780625710417335970814398449907490949385
19, 2.7182818284590452382590981804451807354912191288419111761557808425009494903175283860860474594451886034
20, 2.7182818284590452352219614301374920588826491041137852613693205622504719693380871936423528860317060144
21, 2.7182818284590452353658698967595342428154276021908810096705749530332749544339491141110819527274710 37
22, 2.7182818284590452353600131006060233622176228209698267357085046572381336676554486887513411744330057721
23, 2.7182818284590452353602989218699333649160534615000288054018875199154694801790297492980898757642574405
24, 2.7182818284590452353602870126506037814702855169445520608816526005298259020275362701858560673900151624
25, 2.7182818284590452353602874890193769648081162347247675661547199273781155743839460959635895174911441892
26, 2.7182818284590452353602874706975010731412765917332177286052199542702808137140352828045371151761717707
27, 2.7182818284590452353602874713760890691289373192514232737396451147571852334312981075714424304926706921
28, 2.7182818284590452353602874713518537835579494361257730756932959420003942290659884946298295577614924749
29, 2.7182818284590452353602874713526894830603972941645885997638525126874891386848516654844402615760071073
30, 2.7182818284590452353602874713526616264103156988966280822948339520525688703235316916282485595177415522

```
# method_E.py
# calculate the constant e
# [0] init
from decimal import *
getcontext().prec = 200
N = 30

# [1] calculate sequences
# [1.1] calculate p[n] and q[n]
p = {}
q = {}
p[0] = 1
p[1] = 3
q[0] = 1
q[1] = 1
for n in range(2, N+1):
    p[n] = 2 * (2 * n - 1) * p[n-1] + p[n-2]
    q[n] = 2 * (2 * n - 1) * q[n-1] + q[n-2]

# [1.2] calculate r[n] = p[n] / q[n]
r = {}
for n in range(0, N+1):
    r[n] = Decimal(p[n]) / Decimal(q[n])

# [2] print result
# [2.1] print p[n] and q[n]
print("n, p[n], q[n]")
for n in range(0, N+1):
    print(f"{n:0>2d}, {p[n]}, {q[n]}")

# [2.2] print r[n]
print("\nn, p[n] / q[n]")
for n in range(0, N+1):
    print(f"{n:0>2d},", str(r[n])[:2 + 100])
```

method_E.py の出力(1/2)
n, p[n], q[n]
00, 1, 1
01, 3, 1
02, 19, 7
03, 193, 71
04, 2721, 1001
05, 49171, 18089
06, 1084483, 398959
07, 28245729, 10391023
08, 848456353, 312129649
09, 28875761731, 10622799089
10, 1098127402131, 403978495031
11, 46150226651233, 16977719590391
12, 2124008553358849, 781379079653017
13, 106246577894593683, 39085931702241241
14, 5739439214861417731, 2111421691000680031
15, 33299372103985682 2081, 12250154400974168 3039
16, 20651350143685984386753, 7597207150294985028449
17, 1363322103204314826347779, 501538173463478753560673
18, 954531985744457238287312 83, 351152693495938077342755 59
19, 706490001661218787815246 2721, 259903147004340525108995 2039
20, 551157654494325100219720823521, 202759569932735203392750534601
21, 452019925685512704058952599914 43, 166288837659543300834566337893 21
22, 388792251854990358000721208008 7619, 143028676344200512238066325641 6207
23, 349958228662059873471054982467877153, 128742437593546415344343149711247951
24, 328999614167521780098591755640605400 01, 121032194205568050474906367361137236 01
25, 322454617707037550483967026026040079 7251, 118624424565216044106942674328885 6160849
26, 328936610022595053671656225722124941859603, 121009016275940921794129018452199442130199
27, 348705052085721460647003995968055042379151 69, 128281419694953898706187453826764297219619 43
28, 383608450955295866217071561187432759111252 8193, 141121662566076882668985612111285946885794 3929
29, 437348504594245859633526280153270150891066129171, 160891523467297141632514216552248655879527569849
30, 516109596266305643954182717736977521327369157703 71, 189866109857667234814633674092864542532531111861 11

method_E.py の出力(2/2)
n, p[n] / q[n]
00, 1
01, 3
02, 2.71428571
03, 2.718309859154929577464788732394366197183098591549295774647887323943661971830985915492957746478873239436619718309859154929577464788732394
04, 2.71828182817182
05, 2.7182818287356957266847255237989938636740560561667311625849964066559787716291668970092321300237713527
06, 2.71828182845856341127785060620264237678558448361861745191861820387558621311964387317995082201429219 54
07, 2.718281828459045851404621084949961134721768973083785879407638689665108045665956085363298685798308 7901
08, 2.718281828459045234757560631479773329703773190735879115412070322098750702148132041118592998513896 3841
09, 2.718281828459045235360753230188480692633383946700754551002315393597669500270824523357414314361979 9284
10, 2.718281828459045235360287179900086259351744270348093473686536463570213977923065847067886518912090 8971
11, 2.718281828459045235360287471503357984170958205123068317904576741080414468916173090278597087001427 4124
12, 2.718281828459045235360287471352597036092056850785319415329167101183666688734502872353792745715608 4448
13, 2.718281828459045235360287471352662521984043872659327180034885464728257539941077452758403086210742 7749
14, 2.718281828459045235360287471352662497749516857440010498737420715557351491210657842345752677567245 7442
15, 2.718281828459045235360287471352662497757249242165525926092004950201074590726155680290438972064230 6685
16, 2.718281828459045235360287471352662497757247093175178600668246539353770455525591693797814588346125 7952
17, 2.718281828459045235360287471352662497757247093700073114263162890347693746837700504543354829834739 2473
18, 2.718281828459045235360287471352662497757247093699959553056722079310252820642154807529530582381532 6181
19, 2.718281828459045235360287471352662497757247093699959574970762255854256866169121965655585448024391 6763
20, 2.718281828459045235360287471352662497757247093699959574966967034629471038677376855098600141341388 9449
21, 2.718281828459045235360287471352662497757247093699959574966967627808155659121227052516061843292207 5967
22, 2.718281828459045235360287471352662497757247093699959574966967627724065770253438794111799644610224 8642
23, 2.718281828459045235360287471352662497757247093699959574966967627724076631636942259505018317709324 4166
24, 2.718281828459045235360287471352662497757247093699959574966967627724076630353408307266777661114104 4700
25, 2.718281828459045235360287471352662497757247093699959574966967627724076630353547608502885666400648 5056
26, 2.718281828459045235360287471352662497757247093699959574966967627724076630353547594570093895545490 1553
27, 2.718281828459045235360287471352662497757247093699959574966967627724076630353547594571382288993531 8628
28, 2.718281828459045235360287471352662497757247093699959574966967627724076630353547594571382178516358 5579
29, 2.718281828459045235360287471352662497757247093699959574966967627724076630353547594571382178525167 0820
30, 2.718281828459045235360287471352662497757247093699959574966967627724076630353547594571382178525166427 3

```
# method_F2.py
# calculate the number e
# [0] init
from decimal import *
getcontext().prec = 200
from fractions import Fraction
from math import *
N = 30

# [1] calculate sequence a[n]
a = {}
a[0] = 2
for n in range(1, N+1):
    a[n] = a[n-1] + Fraction(2 * n + 2, factorial(2 * n + 1))

# [2] print result
# [2.1] print a[n] as fraction
print("n, a[n]")
for n in range(0, N+1):
    print(f"{n:0>2d},", a[n].numerator, "/", a[n].denominator)

# [2.2] print a[n] as decimal
print("\nn, a[n]")
decimal_from_fraction = lambda x: Decimal(x.numerator) / Decimal(x.denominator)
for n in range(0, N+1):
    print(f"{n:0>2d},", str(decimal_from_fraction(a[n]))[:2 + 100])
```

method_F2.py の出力(1/2)
n, a[n]
00, 2 / 1
01, 8 / 3
02, 163 / 60
03, 685 / 252
04, 98641 / 36288
05, 13563139 / 4989600
06, 8463398743 / 3113510400
07, 888656868019 / 326918592000
08, 7437374403113 / 2736057139200
09, 82666416490601 / 30411275102208
10, 69439789852104840011 / 25545471085854720000
11, 1351405140967886501753 / 497154168055480320000
12, 85351903640077042215979 / 31399210614030336000000
13, 7399753989883759328999873137 / 272221736260458804019200000
14, 240344009591424503005878794897 9 / 884176199373970195454361600000
15, 55879982230006196948866819813761 83 / 205570966354448070443139072000000 0
16, 62115011826196362082108801814034202 7 / 228508358389786486724163010560000000
17, 17555255217378746833456000112691416538 11 / 64582174789913405810416570859520000000 0
18, 16266852138816341413678881495725194315660 9 / 59842404744462369766591215572090880000000
19, 27723596100184690671332917733164448672180871 89 / 10198941040598721679320140869951448678400000 00
20, 19768129393175170739559123948865085140163752130 27 / 72722883941660450235152308811827721011200000000 0
21, 15790933821802120041919363280537652049079267999752 99 / 58091599099397918882072242373571151449292800000000 0
22, 52446275686862783210194117527876021128167788169502188 73 / 19293906591206482993865026047686728236190924800000000 00
(途中省略)

method_F2.py の出力(2/2)
n, a[n]
00, 2
01, 2.6666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666
02, 2.7166666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666
03, 2.7182539682539682539682539682539682539682539682539682539682539682539682539682539682539682539682539682539682539682539682
04, 2.7182815255731922398589065255731922398589065255731922398589065255731922398589065255731922398589065255731922398589065255
05, 2.7182818261984928651595318261984928651595318261984928651595318261984928651595318261984928651595318261984928651595318261
06, 2.7182818284467590023145578701134256689812245367800923356478912034467590023145578701134256689812245367
07, 2.7182818284589944642854695764748674801584854494907404960315013225066135119045171955224865277775330685
08, 2.7182818284590450705160477958486050611789796352510326989007350040652250425048433140558879743442457416
09, 2.7182818284590452349287527283351994002986043726966476833155148405875077310383076614195442493493357763
10, 2.7182818284590452353593574317298071473772510089137671511318392639688756142701810204245443014981586312
11, 2.7182818284590452353602857925707585115463030677773326260894023062039773775829338914824063210141100998
12, 2.7182818284590452353602874687778324515093860784391696193080756839191240891000263619440385718826805677
13, 2.7182818284590452353602874713492656133721389999983703335207714353203965035161165761218232367426586508
14, 2.7182818284590452353602874713526586022380733150778379628938529304181286539573763714431345289640608035
15, 2.7182818284590452353602874713526624938382062863352767788128471457530077732697104328571231297293326626
16, 2.7182818284590452353602874713526624977537603973977398742123868534744029523078442917020590794341746146
17, 2.7182818284590452353602874713526624977572443308628473549695391873715351872669489090752280783019297879
18, 2.7182818284590452353602874713526624977572470917377154331366390328146468619609855702251317407551533176
19, 2.7182818284590452353602874713526624977572470936987033357420563918923108378645966481860308839204727462
20, 2.7182818284590452353602874713526624977572470936999588462894562017868422690686812772298473132499383470
21, 2.7182818284590452353602874713526624977572470936999595745822380083527252963187601113235815315024629540
22, 2.7182818284590452353602874713526624977572470936999595749667819473231342795516974807722536133899700262
23, 2.7182818284590452353602874713526624977572470936999595749669675454918043880582652822617514239913717108
24, 2.7182818284590452353602874713526624977572470936999595749669676276905398038872169580525188107059592080
25, 2.7182818284590452353602874713526624977572470936999595749669676277240639940175942559908805354440702555
26, 2.7182818284590452353602874713526624977572470936999595749669676277240766259414180835425766357925118235
27, 2.7182818284590452353602874713526624977572470936999595749669676277240766303521160142966787369374146339
28, 2.7182818284590452353602874713526624977572470936999595749669676277240766303535471618082239947351813966
29, 2.7182818284590452353602874713526624977572470936999595749669676277240766303535475944492017084913426755
30, 2.7182818284590452353602874713526624977572470936999595749669676277240766303535475945713498898130771109

```
# method_F3.py
# calculate the number e
# [0] init
from decimal import *
getcontext().prec = 200
from fractions import Fraction
from math import *
N = 30

# [1] calculate sequence a[n]
a = {}
a[0] = 1
for n in range(1, N+1):
    a[n] = a[n-1] + Fraction(2 * n + 1, factorial(2 * n))

# [2] print result
# [2.1] print a[n] as fraction
print("n, a[n]")
for n in range(0, N+1):
    print(f"{n:0>2d},", a[n].numerator, "/", a[n].denominator)

# [2.2] print a[n] as decimal
print("\nn, a[n]")
decimal_from_fraction = lambda x: Decimal(x.numerator) / Decimal(x.denominator)
for n in range(0, N+1):
    print(f"{n:0>2d},", str(decimal_from_fraction(a[n]))[:2 + 100])
```

method_F3.py の出力(1/2)
n, a[n]
00, 1 / 1
01, 5 / 2
02, 65 / 24
03, 1957 / 720
04, 109601 / 40320
05, 9864101 / 3628800
06, 260412269 / 95800320
07, 47395032961 / 17435658240
08, 56874039553217 / 20922789888000
09, 17403456103284421 / 6402373705728000
10, 6613313319248080001 / 2432902008176640000
11, 611070150698522592097 / 224800145555521536000
12, 337310723185584470837549 / 124089680346647887872000
13, 109625985035314953022203427 7 / 40329146112660563558400000 0
14, 828772446866981044847857913441 / 304888344611713860501504000000
15, 554640022134056545398181834379 77 / 204040661393993122027929600000000
16, 286105509017631728378198117446460 57 / 105252334773477412066887204864000000
17, 160505190558891399620169143887464379777 / 590465598079208281695237219287040000 00
18, 777832846554627552005435081916173532765 49 / 286148712915308628821538037039104000000 00
19, 109363298225580633811964172517413998706827897 / 402325090358923932123082480076980224000000 00
20, 221788768801477525370663341865315589377446975120 1 / 815915283247897734345611269596115894272000000 00
21, 763840519752288597376564549384146889815927382313633 / 281001223550575979708628521248902313987276800000000
22, 152124869828561055393311592361558517424393116561830 91 / 559636121008094477482868591792550713751502848000000 0
(途中省略)

method_F3.py の出力(2/2)

n, a[n]

00, 1

01, 2.5

02, 2.7083333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333

03, 2.7180555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555

04, 2.7182787698412698412698412698412698412698412698412698412698412698412698412698412698412698412698412698

05, 2.7182818011463844797178130511463844797178130511463844797178130511463844797178130511463844797178130511

06, 2.7182818282861685639463417241195018972796750574528352306130083907861685639463417241195018972796750574

07, 2.7182818284582297479122875948272773669599066424463249860075256900653726050551447376844202241027637853

08, 2.7182818284590422590587934503278418622333966249310164654079997995341910685826029741373656717572061487

09, 2.7182818284590452267081174817108696833426231358243669340947758487613935966116344440513614355990812746

10, 2.7182818284590452353397844906664158861464034345402617207765517901788134377596413222879533900368485014

11, 2.7182818284590452353602471108690522047059258986580173979661705127775148041115691885216620702009454553

12, 2.7182818284590452353602874043083296076646521164906374272612036309300799848109074207724373314646586266

13, 2.7182818284590452353602874712574287147341835385141131651560323013417796315726847827583309272833737193

14, 2.7182818284590452353602874713525455026092088379085223752480835472482042489426677115990908192233473984

15, 2.7182818284590452353602874713526623722257021309834818158153785765237928007911999934379359859554179170

16, 2.7182818284590452353602874713526624976385970411900203714065180385414207411596638840889727279722674984

17, 2.7182818284590452353602874713526624977571475549332610360596182892077259585180848919259733838889365886

18, 2.7182818284590452353602874713526624977572470190831136416059258789871965547321951317738184864800684878

19, 2.7182818284590452353602874713526624977572470936496786381769209579153692384670063712370084053413397605

20, 2.7182818284590452353602874713526624977572470936999289531811847777417343778495364051097564458849510708

21, 2.7182818284590452353602874713526624977572470936999595580301293309307734093358037741850875719967237584

22, 2.7182818284590452353602874713526624977572470936999595749584222964759514755683727553494563942185024812

23, 2.7182818284590452353602874713526624977572470936999595749669636788632904274643784530640535529371758424

24, 2.7182818284590452353602874713526624977572470936999595749669676260465650955706379245367034629716674581

25, 2.7182818284590452353602874713526624977572470936999595749669676277234192980535485387228351176606450430

26, 2.7182818284590452353602874713526624977572470936999595749669676277240763920169028274768044857860592019

27, 2.7182818284590452353602874713526624977572470936999595749669676277240766302733535512474983422741127980

28, 2.7182818284590452353602874713526624977572470936999595749669676277240766303535224868511283868421164524

29, 2.7182818284590452353602874713526624977572470936999595749669676277240766303535475872385187465954066542

30, 2.7182818284590452353602874713526624977572470936999595749669676277240766303535475945693797578562749426

```
# method_F4.py
# calculate the number e
# [0] init
from decimal import *
getcontext().prec = 200
from fractions import Fraction
from math import *
N = 30

# [1] calculate sequence a[n]
a = {}
a[0] = Fraction(1, 2)
for n in range(1, N+1):
    a[n] = a[n-1] + Fraction(n + 1, 2 * factorial(n))

# [2] print result
# [2.1] print a[n] as fraction
print("n, a[n]")
for n in range(0, N+1):
    print(f"{n:0>2d},", a[n].numerator, "/", a[n].denominator)

# [2.2] print a[n] as decimal
print("\nn, a[n]")
decimal_from_fraction = lambda x: Decimal(x.numerator) / Decimal(x.denominator)
for n in range(0, N+1):
    print(f"{n:0>2d},", str(decimal_from_fraction(a[n]))[:2 + 100])
```

method_F4.py の出力(1/2)
n, a[n]
00, 1 / 2
01, 3 / 2
02, 9 / 4
03, 31 / 12
04, 43 / 16
05, 217 / 80
06, 3913 / 1440
07, 9133 / 3360
08, 73067 / 26880
09, 1972819 / 725760
10, 6576067 / 2419200
11, 24112247 / 8870400
12, 372017527 / 136857600
13, 1612075951 / 593049600
14, 157983443203 / 58118860800
15, 7109254944151 / 2615348736000
16, 37916026368811 / 13948526592000
17, 644572448269793 / 237124952064000
18, 34806912206568841 / 12804747411456000
19, 2422459091299663 / 891172896768000
20, 7775794614048301 / 2860554977280000
21, 277759159408419360043 / 102181884343418880000
22, 2036900502328408640323 / 749333818518405120000
23, 4684871155355339872743 7 / 17234677825923317760000
24, 3373107231855844708375489 / 124089680346647887872000 0
25, 2810922693213203923646241 7 / 10340806695553990656000000
26, 1044057000336332885925746 93 / 38408710583486251008000000
27, 8456861702724296375998550137 / 311110555726238633164800000 0
28, 5525149645779873632319052756 27 / 20325889640780924033433600000 0
29, 5340977990920544511241750997731 / 19648359986088226565652480000 00
30, 1442064057548547018035272769387401 / 530505719624382117272616960000000

method_F4.py の出力(2/2)
n, a[n]
00, 0.5
01, 1.5
02, 2.25
03, 2.58333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333
04, 2.6875
05, 2.7125
06, 2.71736111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
07, 2.7181547619047619047619047619047619047619047619047619047619047619047619047619047619047619047619047619
08, 2.7182663690476190476190476190476190476190476190476190476190476190476190476190476190476190476190476190
09, 2.71828014770723104056437389770723104056437389770723104056437389770723104056437389770723104056437389770
10, 2.7182816633597883597883597883597883597883597883597883597883597883597883597883597883597883597883597883
11, 2.7182818136724386724386724386724386724386724386724386724386724386724386724386724386724386724386724386
12, 2.7182818272423307145529367751589973812196034418256640478862701084923307145529367751589973812196034418
13, 2.7182818283664637831304497971164637831304497971164637831304497971164637831304497971164637831304497971
14, 2.7182818284524943751134227324703515179705655896132086608277084467560658036848513038989229465419941610
15, 2.7182818284586121060988785856510724235591960459685327410195135062859930584798309666034533759401484269
16, 2.7182818284590183616721315134013546711959410372108784807197505610204022902435600848299260997673696086
17, 2.7182818284590436647874206230882234617061881300910245821543674017997080555437231440966268230507259452
18, 2.7182818284590451486120826387797373722608013855376998164977554264133093195582388790536247049716635081
19, 2.7182818284590452308184351050230345418206137542605073087051453446744506638249710527354528424742085255
20, 2.7182818284590452351342686095008076432225039036184547020460333153831605843989744918537488196930921389
21, 2.7182818284590452353495709611981115167618272217270144359541955270738445260149111713562488457675035663
22, 2.7182818284590452353598022712994296760415884537858922745490048883731952091908751044731031858495520432
23, 2.7182818284590452353602664517199053581261144832176750120277864094907460908472515400020341956075277775
24, 2.7182818284590452353602865984395440596054775921339850266753029685670286811969206561274218262393843632
25, 2.7182818284590452353602874365430810295870190974649035232846396574246020369554668913582379516736695972
26, 2.7182818284590452353602874700176305831217848084766413922320539926304518603363555723511847495830271435
27, 2.7182818284590452353602874713033471640531612692562417493384018683310880675444006794400770820130161851
28, 2.7182818284590452353602874713509055579906739189534463543844274912843003762293921438604570279830030246
29, 2.7182818284590452353602874713526020524236410764931801690709682388331664514500220415211126740937041009
30, 2.7182818284590452353602874713526604872318877230306598893546157534709607273742881824405352574597393602

```
# method_F9.py
# calculate the number e
# [0] init
from decimal import *
getcontext().prec = 200
from fractions import Fraction
from math import *
N = 30

# [1] calculate sequences a[n] and b[n]
# [1.1] calculate a[n]
a = {}
a[0] = Fraction(1)
for n in range(1, N+1):
    a[n] = a[n-1] + Fraction(1 - 2 * n, factorial(2 * n))

# [1.2] calculate b[n]
b = {}
for n in range(0, N+1):
    b[n] = 1 / a[n]

# [2] print result
# [2.1] print b[n] as fraction
print("n, b[n]")
for n in range(0, N+1):
    print(f"{n:0>2d},", b[n].numerator, "/", b[n].denominator)

# [2.2] print b[n] as decimal
print("\nn, b[n]")
decimal_from_fraction = lambda x: Decimal(x.numerator) / Decimal(x.denominator)
for n in range(0, N+1):
    print(f"{n:0>2d},", str(decimal_from_fraction(b[n]))[:2 + 100])
```

method_F9.py の出力(1/2)
n, b[n]
00, 1 / 1
01, 2 / 1
02, 8 / 3
03, 144 / 53
04, 5760 / 2119
05, 44800 / 16481
06, 43545600 / 16019531
07, 6706022400 / 2467007773
08, 42268262400 / 15549624751
09, 376610217984000 / 138547156531409
10, 11640679464960000 / 4282366656425369
11, 1784128139329536000 / 656344062874794888 7
12, 26976017466662584320000 / 9923922230666898717143
13, 161316584450642254233600 00 / 59345054939388054328515 13
14, 125468454572721753292800 0000 / 46157264952857375588845 1011
15, 914665033835141581504512 0000000 / 33648646150633026804268078701 89
16, 848809151399011387636187136 0000000 / 31225943627787448874360777035353 91
17, 208541922045351515750242713 6000000 00 / 76718285742862403528149787867745 97
18, 106283807654257490705142699 47166720000000 / 39099627765421309682928595686372469102 43
19, 116824350562117737717222967 187865600000000 / 42977276800008547006855398564359821410109
20, 697363489955468149013342965466765721600000 0000 / 25654569097818435326625549249685189393741065 73
21, 311531289967379134931960666573062432358400 0000000 / 11460595686061799199548904455723001880076690636 11
22, 618202691811267155358982746747585090772008960 00000000 / 22742406079421034331584846001936724930824184898296683
(途中省略)

method_F9.py の出力(2/2)
n, b[n]
00, 1
01, 2
02, 2.6666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666
03, 2.71698113207547169811320754716981132075471698113207547169811320754716981132075471698113207547169811320754716981132
04, 2.7182633317602642756016989145823501651722510618216139688532326569136385087305332704105710240679565832
05, 2.7182816576664037376372792913051392512590255445664704811601237788969115951701959832534433590194769734
06, 2.7182818273518744088075986743931517096224602330742391896491851103506088911092340968034582285835958618
07, 2.718281828453728183693079916372034876438145701780891794539149188161070297527595183624903803657370965 2
08, 2.7182818284590255576129561867650242693628330632632898061888413219625225154090922666536314796501033583
09, 2.7182818284590451775163631742914084928093139944691124479896463780625710417335970814398449907490949385
10, 2.71828182845904523522196143013749205888264910411378526136932056225047196933808719364235288603170601 44
11, 2.718281828459045235360131006060233622176228209698267357085046572381336676554486887513411744330057721
12, 2.7182818284590452353602870126506037814702855169445520608816526005298259020275362701858560673900151624
13, 2.7182818284590452353602874706975010731412765917332177286052199542702808137140352828045371151761717707
14, 2.7182818284590452353602874713518537835579494361257730756932959420003942290659884946298295577614924749
15, 2.7182818284590452353602874713526616264103156988966280822948339520525688703235316916282485595177415522
16, 2.7182818284590452353602874713526624969306307487487518484657407820637679143741348090568209953499567613
17, 2.7182818284590452353602874713526624977565513512628430474848782648246896203415507583140936657486813722
18, 2.7182818284590452353602874713526624977572465706252286461372512761738149753131152147512197252444102318
19, 2.7182818284590452353602874713526624977572470933465537480911252408440022876100863159800753432235475289
20, 2.7182818284590452353602874713526624977572470936997438326194724529822929547129896748322570703886736372
21, 2.7182818284590452353602874713526624977572470936999594553816997711826215814938827782259397210157866070
22, 2.7182818284590452353602874713526624977572470936999595749065125357714244244133843819417898776957351176
23, 2.7182818284590452353602874713526624977572470936999595749669396403035724541620797821458929364200424727
24, 2.7182818284590452353602874713526624977572470936999595749669676158149299301017782522385800211694518743
25, 2.7182818284590452353602874713526624977572470936999595749669676277194028481766462090343641433331750272
26, 2.7182818284590452353602874713526624977572470936999595749669676277240749333093286532874237132398326894
27, 2.7182818284590452353602874713526624977572470936999595749669676277240766297817806076971016578912149180
28, 2.7182818284590452353602874713526624977572470936999595749669676277240766303533683610510132744493858689
29, 2.7182818284590452353602874713526624977572470936999595749669676277240766303535475421649173219780874899
30, 2.7182818284590452353602874713526624977572470936999595749669676277240766303535475945570558904130374026

```
# method_F10.py
# calculate the number e
# [0] init
from decimal import *
getcontext().prec = 200
from fractions import Fraction
from math import *
N = 30

# [1] calculate sequence a[n]
a = {}
a[0] = 3
for n in range(1, N+1):
    a[n] = a[n-1] + Fraction(3 - 4 * n ** 2, factorial(2 * n + 1))

# [2] print result
# [2.1] print a[n] as fraction
print("n, a[n]")
for n in range(0, N+1):
    print(f"{n:0>2d},", a[n].numerator, "/", a[n].denominator)

# [2.2] print a[n] as decimal
print("\nn, a[n]")
decimal_from_fraction = lambda x: Decimal(x.numerator) / Decimal(x.denominator)
for n in range(0, N+1):
    print(f"{n:0>2d},", str(decimal_from_fraction(a[n]))[:2 + 100])
```

method_F10.py の出力(1/2)
~/kyoaca$ python3 method_F10.py
n, a[n]
00, 3 / 1
01, 17 / 6
02, 109 / 40
03, 4567 / 1680
04, 986411 / 362880
05, 36168371 / 13305600
06, 5642265829 / 2075673600
07, 323147952007 / 118879488000
08, 1085138801801 / 399200256000
09, 110221888654134667 / 40548366802944000
10, 138879579704209680023 / 51090942171709440000
11, 23424355776776699363719 / 8617338912961658880000
12, 1405461346066019618231209 / 5170403347776995328000000
13, 295990159595350373159949254 81 / 1088886945041835216076800000 0
14, 80114669863808167668626264965 97 / 2947253997913233984847872000000
15, 74506642973341595931822426418 34911 / 27409462180593076059085209600000 00
16, 10262480214762877213565802038840433 49 / 37753554864399506502253019136000000 0
17, 31209342608673327703921777978118073845 53 / 11481275518206827699629612597248000000 00
18, 66691194152009955884957981176769958869 911 / 24534319235697584752791407453798400000000
19, 50406538363972164856968941333026270313056130 71 / 18543529164724948507854801581729906688000000 00
20, 30311131736201928467323990054926463881584419933081 / 11150842204387935702723354017813583888384000000 00
21, 54741903915580682811987126039197193770141462399143699 / 20138421021124611879118377356171332502421504000000 00
22, 32516690925854925590320352867283133099464028665091357012 7 / 11962222086548019456196316149565771506438373376000000 0000
(途中省略)

method_F10.py の出力(2/2)

n, a[n]

00, 3

01, 2.8333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333

02, 2.725

03, 2.7184523809523809523809523809523809523809523809523809523809523809523809523809523809523809523809523809

04, 2.7182842813051146384479717813051146384479717813051146384479717813051146384479717813051146384479717813

05, 2.7182818512506012506012506012506012506012506012506012506012506012506012506012506012506012506012506012

06, 2.7182818286073494406827740161073494406827740161073494406827740161073494406827740161073494406827740161

07, 2.7182818284597591806586515581224575933570642565351560060554769549478544187538896533605528314523023517

08, 2.7182818284590478819733021413693682601245626455710489323934702085962590164270836539744102769312853346

09, 2.7182818284590452431493879749595291172545856095689284325362538324136218654649808787877270630995902781

10, 2.7182818284590452353789303727931984086080985832872725814871267377589377907807207185611352129594687609

11, 2.7182818284590452353603244742724648183866802368966478542126340996304399510542985944431505718272747443

12, 2.7182818284590452353602875332473352953541200403877018113549477369081681933891453031156398123007025088

13, 2.7182818284590452353602874714411025120100944614826275018855105692990133754595483694853155462019435824

14, 2.7182818284590452353602874713527717018669377922471535505396223135880530589720850312871782387047742086

15, 2.7182818284590452353602874713526626154507104416870717418103157149822227457482208722763102735032474082

16, 2.7182818284590452353602874713526624978689237536054593770182556684073851634560246993151454308960817308

17, 2.71828182845904523536028747135266249775734110679243367387946008553534441601581292622244827727149229871

18, 2.7182818284590452353602874713526624977572471643923172246673521866420971691897760086764449950302381473

19, 2.7182818284590452353602874713526624977572470937477280333071918258692524372621869251350533624996057319

20, 2.7182818284590452353602874713526624977572470936999887393977276258319501602878261493499381806149256232

21, 2.7182818284590452353602874713526624977572470936999595911343466857746771833017164484620754910082021496

22, 2.7182818284590452353602874713526624977572470936999595749751415981703170835350222061950508325614375713

23, 2.7182818284590452353602874713526624977572470936999595749669714121203183486521521114594492950455675793

24, 2.7182818284590452353602874713526624977572470936999595749669676293345145122037959915683341584402509580

25, 2.7182818284590452353602874713526624977572470936999595749669676277247086899816399732589259532274954679

26, 2.7182818284590452353602874713526624977572470936999595749669676277240768598659333396083487857989644451

27, 2.7182818284590452353602874713526624977572470936999595749669676277240766304308784773458591316007164698

28, 2.7182818284590452353602874713526624977572470936999595749669676277240766303535718367653196026282463408

29, 2.7182818284590452353602874713526624977572470936999595749669676277240766303535476016598846703872786969

30, 2.7182818284590452353602874713526624977572470936999595749669676277240766303535475945733200217698792792

```
# method_F11.py
# calculate the number e
# [0] init
from decimal import *
getcontext().prec = 200
from fractions import Fraction
from math import *
N = 30

# [1] calculate sequence a[n]
a = {}
a[0] = 1
for n in range(1, N+1):
    a[n] = a[n-1] + Fraction((3 * n) ** 2 + 1, factorial(3 * n))

# [2] print result
# [2.1] print a[n] as fraction
print("n, a[n]")
for n in range(0, N+1):
    print(f"{n:0>2d},", a[n].numerator, "/", a[n].denominator)

# [2.2] print a[n] as decimal
print("\nn, a[n]")
decimal_from_fraction = lambda x: Decimal(x.numerator) / Decimal(x.denominator)
for n in range(0, N+1):
    print(f"{n:0>2d},", str(decimal_from_fraction(a[n]))[:2 + 100])
```

method_F11.py の出力(1/2)
~/kyoaca$ python3 method_F11.py
n, a[n]
00, 1 / 1
01, 8 / 3
02, 1957 / 720
03, 98641 / 36288
04, 260412269 / 95800320
05, 888656868019 / 326918592000
06, 17403456103284421 / 6402373705728000
07, 69439789852104840011 / 25545471085854720000
08, 337310723185584470837549 / 124089680346647887872000
09, 739975398988375932899873137 / 272221736260458804019200000
10, 554640022134056545398181834379 77 / 204040661393993122027929600000000
11, 621150118261963620821088018140342027 / 228508358389786486724163010560000000
12, 777832846554627552005435081916173532765 49 / 286148712915308628821538037039104000000000
13, 277235961001846906713329177331644486721808718 9 / 101989410405987216793201408699514486784000000 0
14, 763840519752288597376564549384146889815927382313633 / 2810012235505759797086285212489023139872768000000 00
15, 52446275686862783210194117527876021128167788169502188 73 / 19293906591206482993865026047686728236190924800000000 00
(途中省略)

method_F11.py の出力(2/2)
n, a[n]
00, 1
01, 2.6666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666
02, 2.718055555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555
03, 2.718281525573192239858906525573192239858906525573192239858906525573192239858906525573192239858906525573192239858906525
04, 2.7182818282861685639463417241195018972796750574528352306130083907861685639463417241195018972796750574
05, 2.718281828458994464285469576474867480158485449490740496031501322506613511904517195522486527777533068 5
06, 2.7182818284590452670811748171086968334262313582436693409477584876139359661163444405136143559908127 46
07, 2.718281828459045235359357431729807147377251008913767151131839263968875614270181020424544301498158631 2
08, 2.7182818284590452353602874043083296076646521164906374272612036309300799848109074207724373314646586266
09, 2.718281828459045235360287471349265613372138999998370333520771435320396503516116576121823236742658650 8
10, 2.7182818284590452353602874713526623722257021309834818158153785765237928007911999934379359859554179170
11, 2.718281828459045235360287471352662497753760397397739874212386853474402952307844291702059079434174614 6
12, 2.7182818284590452353602874713526624977572470190831136416059258789871965547321951317738184864800684878
13, 2.718281828459045235360287471352662497757247093698703335742056391892310837864596648186030883920472746 2
14, 2.7182818284590452353602874713526624977572470936999595580301293309307734093358037741850875719967237584
15, 2.718281828459045235360287471352662497757247093699959574966781947323134279551697480772253613389970026 2
16, 2.718281828459045235360287471352662497757247093699959574966967626046565095570637924536703462971667458 1
17, 2.7182818284590452353602874713526624977572470936999595749669676277240639940175942559908805354440702555
18, 2.718281828459045235360287471352662497757247093699959574966967627724076630273353551247498342274112798 0
19, 2.7182818284590452353602874713526624977572470936999595749669676277240766303535471618082239947351813966
20, 2.7182818284590452353602874713526624977572470936999595749669676277240766303535475945693797578562749426
21, 2.7182818284590452353602874713526624977572470936999595749669676277240766303535475945713821705210226661
22, 2.7182818284590452353602874713526624977572470936999595749669676277240766303535475945713821785251385993
23, 2.7182818284590452353602874713526624977572470936999595749669676277240766303535475945713821785251664273
24, 2.7182818284590452353602874713526624977572470936999595749669676277240766303535475945713821785251664274
25, 2.7182818284590452353602874713526624977572470936999595749669676277240766303535475945713821785251664274
26, 2.7182818284590452353602874713526624977572470936999595749669676277240766303535475945713821785251664274
27, 2.7182818284590452353602874713526624977572470936999595749669676277240766303535475945713821785251664274
28, 2.7182818284590452353602874713526624977572470936999595749669676277240766303535475945713821785251664274
29, 2.7182818284590452353602874713526624977572470936999595749669676277240766303535475945713821785251664274
30, 2.7182818284590452353602874713526624977572470936999595749669676277240766303535475945713821785251664274

```
# method_G.py
# calculate the number e
# [0] init
from decimal import *
getcontext().prec = 200
from fractions import Fraction
from math import *
N = 30

# [1] calculate sequence a[n]
a = {}
a[2] = Fraction(11, 4)
for n in range(3, N+1):
    a[n] = a[n-1] - Fraction(1, factorial(n) * (n - 1) * n)

# [2] print result
# [2.1] print a[n] as fraction
print("n, a[n]")
for n in range(2, N+1):
    print(f"{n:0>2d},", a[n].numerator, "/", a[n].denominator)

# [2.2] print a[n] as decimal
print("\nn, a[n]")
decimal_from_fraction = lambda x: Decimal(x.numerator) / Decimal(x.denominator)
for n in range(2, N+1):
    print(f"{n:0>2d},", str(decimal_from_fraction(a[n]))[:2 + 100])
```

method_G.py の出力(1/2)
n, a[n]
02, 11 / 4
03, 49 / 18
04, 87 / 32
05, 1631 / 600
06, 11743 / 4320
07, 31967 / 11760
08, 876809 / 322560
09, 8877691 / 3265920
10, 4697191 / 1728000
11, 1193556233 / 439084800
12, 2232105163 / 821145600
13, 2222710781 / 817689600
14, 3317652307271 / 1220496076800
15, 53319412081141 / 19615115520000
16, 303328210950491 / 111588212736000
17, 2348085347268533 / 863812325376000
18, 313262209859119579 / 115242726703104000
19, 42739099682215483 / 15722836107264000
20, 10174328183458584617 / 3742926166425600000
21, 2916471173788403280463 / 1072909785605898240000
22, 276616117600154259797 / 101761382761758720000
23, 1616280548597592256096589 / 594596384994354462720000
24, 525679049120391383123453 / 193386514825944760320000
25, 20668549214802970026810601 / 760353433496616960000000
26, 4071822301311698255110413029 / 1497939712755963789312000000
27, 799173430907446007531862987961 / 293999475161295508340736000000
28, 773520950409182308524667385878 3 / 284562454970932936468070400000 0
29, 6969976278151310587170485052039 11 / 256411097818451356681764864000000
30, 1272409462542835604148770090635943 / 468093282021513632887603200000000

method_G.py の出力(2/2)

n, a[n]

02, 2.75
03, 2.7222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222
04, 2.71875
05, 2.7183333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333
06, 2.7182870370370370370370370370370370370370370370370370370370370370370370370370370370370370370370370370
07, 2.7182823129251700680272108843537414965986394557823129251700680272108843537414965986394557823129251700
08, 2.7182818700396825396825396825396825396825396825396825396825396825396825396825396825396825396825396825
09, 2.7182818317656280619243582206545169508132471095434058397021359984322947285910248873211836174799137762
10, 2.7182818287037037037037037037037037037037037037037037037037037037037037037037037037037037037037037037
11, 2.7182818284759572638360517148395936274724153512032299911087789875668663547451426239305027183815062602
12, 2.7182818284601415388452425489462526499563536600573637610674647711684748721785758822795859832896869933
13, 2.7182818284591121129582668044206505744967283428821890360351898813437274975736514198052659591121129582
14, 2.7182818284590490868835540037354096311012410785653416038903567248238450052509009425109198351828737316
15, 2.7182818284590454453770150419180401543717240366270348633664330313360295723611420260450242813558510207
16, 2.7182818284590452462321261924436527611130785733960337134940309543484146658749833353007956182559357162
17, 2.7182818284590452358958862867615911317051904005639748302826606043317564527355633340510951686140716001
18, 2.7182818284590452353854546864809954956850477747451077249388892290222918496175672846066655167799054709
19, 2.7182818284590452353614177413154272801383928588478203543271326822625663696923430939126065027046123290
20, 2.7182818284590452353603360787829767104387933876324424226496036376583787230957080053313738470712241377
21, 2.7182818284590452353602894765423495883882437505506007430535196198636404798183019584310486306153638754
22, 2.7182818284590452353602875508299269800390474845554833182768219331779074945589050143442583324147084927
23, 2.7182818284590452353602874743838761770611020751303463316599776015703453155599497481328734623538129104
24, 2.7182818284590452353602874714640617366695833268536917939766953527936675934454063178261886235667648153
25, 2.7182818284590452353602874713566125652631754369171109069899505660386858532715911195909026214994014453
26, 2.7182818284590452353602874713527978017812911331323802246117229404734202293600947220204190947987849943
27, 2.7182818284590452353602874713526697998835957264593541383057658843071564692142886476491554450040099446
28, 2.7182818284590452353602874713526626415105327606910278053097732655313402398507588235804218043119434251
29, 2.7182818284590452353602874713526625022252755384285040176402587712170915644751249459205153465413267830
30, 2.7182818284590452353602874713526624978919564248470032775794294313939816056856607808377627011884631541

```
# method_H2.py
# calculate the number e
# [0] init
from decimal import *
getcontext().prec = 200
from fractions import Fraction
from math import *
N = 30

# [1] calculate sequence a[n]
a = {}
a[0] = 0
for n in range(1, N+1):
    a[n] = a[n-1] + Fraction(n**2, 2 * factorial(n))

# [2] print result
# [2.1] print a[n] as fraction
print("n, a[n]")
for n in range(0, N+1):
    print(f"{n:0>2d},", a[n].numerator, "/", a[n].denominator)

# [2.2] print a[n] as decimal
print("\nn, a[n]")
decimal_from_fraction = lambda x: Decimal(x.numerator) / Decimal(x.denominator)
for n in range(0, N+1):
    print(f"{n:0>2d},", str(decimal_from_fraction(a[n]))[:2 + 100])
```

method_H2.py の出力(1/2)
n, a[n]
00, 0 / 1
01, 1 / 2
02, 3 / 2
03, 9 / 4
04, 31 / 12
05, 43 / 16
06, 217 / 80
07, 3913 / 1440
08, 9133 / 3360
09, 73067 / 26880
10, 1972819 / 725760
11, 6576067 / 2419200
12, 24112247 / 8870400
13, 372017527 / 136857600
14, 1612075951 / 593049600
15, 157983443203 / 58118860800
16, 7109254944151 / 2615348736000
17, 37916026368811 / 13948526592000
18, 644572448269793 / 237124952064000
19, 34806912206568841 / 12804747411456000
20, 2422459091299663 / 891172896768000
21, 7775794614048301 / 2860554977280000
22, 277759159408419360043 / 102181884343418880000
23, 2036900502328408640323 / 749333818518405120000
24, 46848711553553398727437 / 17234677825923317760000
25, 337310723185584470837548 9 / 124089680346647887872000 0
26, 28109226932132039236462417 / 10340806695553990656000000
27, 10440570003363328859257469 3 / 38408710583486251008000000
28, 84568617027242963759985501 37 / 31111055572623863316480000 00
29, 55251496457798736323190527 5627 / 20325889640780924033433600 0000
30, 53409779909205445112417509977 31 / 19648359986088226565652480000 00

method_H2.py の出力(2/2)

n, a[n]
00, 0
01, 0.5
02, 1.5
03, 2.25
04, 2.5833333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333
05, 2.6875
06, 2.7125
07, 2.7173611111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
08, 2.7181547619047619047619047619047619047619047619047619047619047619047619047619047619047619047619047619
09, 2.7182663690476190476190476190476190476190476190476190476190476190476190476190476190476190476190476190
10, 2.7182801477072310405643738977072310405643738977072310405643738977072310405643738977072310405643738977
11, 2.7182816633597883597883597883597883597883597883597883597883597883597883597883597883597883597883597883
12, 2.7182818136724386724386724386724386724386724386724386724386724386724386724386724386724386724386724386
13, 2.7182818272423307145529367751589973812196034418256640478862701084923307145529367751589973812196034418
14, 2.7182818283664637831304497971164637831304497971164637831304497971164637831304497971164637831304497971
15, 2.7182818284524943751134227324703515179705655896132086608277084467560658036848513038989229465419941610
16, 2.7182818284586121060988785856510724235591960459685327410195135062859930584798309666034533759401484269
17, 2.7182818284590183616721315134013546711959410372108784807197505610204022902435600848299260997673696086
18, 2.7182818284590436647874206230882234617061881300910245821543674017997080555437231440966268230507259452
19, 2.7182818284590451486120826387797373722608013855376998164977554264133093195582388790536247049716635081
20, 2.7182818284590452308184351050230345418206137542605073087051453446744506638249710527354528424742085255
21, 2.7182818284590452351342686095008076432225039036184547020460333153831605843989744918537488196930921389
22, 2.718281828459045235349570961198111516761827221727014435954195527073844526014911171356248845767503566 3
23, 2.7182818284590452353598022712994296760415884537858922745490048883731952091908751044731031858495520432
24, 2.7182818284590452353602664517199053581261144832176750120277864094907460908472515400020341956075277775
25, 2.7182818284590452353602865984395440596054775921339850266753029685670286811969206561274218262393843632
26, 2.7182818284590452353602874365430810295870190974649035232846396574246020369554668913582379516736695972
27, 2.7182818284590452353602874700176305831217848084766413922320539926304518603363555723511847495830271435
28, 2.7182818284590452353602874713033471640531612692562417493384018683310880675444006794400770820130161851
29, 2.7182818284590452353602874713509055579906739189534463543844274912843003762293921438604570279830030246
30, 2.7182818284590452353602874713526020524236410764931801690709682388331664514500220415211126740937041009

```
# method_H3.py
# calculate the number e
# [0] init
from decimal import *
getcontext().prec = 200
from fractions import Fraction
from math import *
N = 30

# [1] calculate sequence a[n]
a = {}
a[0] = 0
for n in range(1, N+1):
    a[n] = a[n-1] + Fraction(n**3, 5 * factorial(n))

# [2] print result
# [2.1] print a[n] as fraction
print("n, a[n]")
for n in range(0, N+1):
    print(f"{n:0>2d},", a[n].numerator, "/", a[n].denominator)

# [2.2] print a[n] as decimal
print("\nn, a[n]")
decimal_from_fraction = lambda x: Decimal(x.numerator) / Decimal(x.denominator)
for n in range(0, N+1):
    print(f"{n:0>2d},", str(decimal_from_fraction(a[n]))[:2 + 100])
```

method_H3.py の出力(1/2)
n, a[n]
00, 0 / 1
01, 1 / 5
02, 1 / 1
03, 19 / 10
04, 73 / 30
05, 317 / 120
06, 1621 / 600
07, 391 / 144
08, 68489 / 25200
09, 547993 / 201600
10, 4932037 / 1814400
11, 49320491 / 18144000
12, 108505109 / 39916800
13, 6510306709 / 2395008000
14, 84633987413 / 31135104000
15, 1184875824007 / 435891456000
16, 17773137360361 / 6538371840000
17, 56874039553213 / 20922789888000
18, 4834293362023429 / 1778437140480000
19, 87017280516422083 / 32011868528640000
20, 1653328329812019977 / 608225502044160000
21, 3006051508749127271 / 1105864549171200000
22, 664495596670859713 / 244454268764160000
23, 15276753767463064802399 / 5620003638888038400000
24, 2066854921480297002 6809 / 760353433496616960000 0
25, 8432768079639611770938697 / 3102242008666197196800000
26, 2108192019909902942734 68101 / 7755605021665492992000 0000
27, 1096259850353149530222034271 / 4032914611266056355840000 00
28, 1479950797976751865799746273 69 / 5444434725209176080384 0000 0
29, 4143862234334905224239289567173 / 1524441723058569302507520000000
30, 5224869773726619630562582497779 / 1922122172552109120552960000000

method_H3.py の出力(2/2)
n, a[n]
00, 0
01, 0.2
02, 1
03, 1.9
04, 2.4333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333
05, 2.6416666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666
06, 2.7016666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666
07, 2.7152777777777777777777777777777777777777777777777777777777777777777777777777777777777777777777777777
08, 2.7178174603174603174603174603174603174603174603174603174603174603174603174603174603174603174603174603
09, 2.7182192460317460317460317460317460317460317460317460317460317460317460317460317460317460317460317460
10, 2.7182743606701940035273368606701940035273368606701940035273368606701940035273368606701940035273368606
11, 2.7182810295414462081128747795414462081128747795414462081128747795414462081128747795414462081128747795
12, 2.7182817510421677088343755010421677088343755010421677088343755010421677088343755010421677088343755010
13, 2.7182818216056063278285500507722729944952167174389396611618833841056063278285500507722729944952167174
14, 2.7182818279007515118626229737340848451959563070674181785292896404007515118626229737340848451959563070
15, 2.7182818284169350637604605858574112542366510620478874447128415382383636351890320144288398256652224906
16, 2.7182818284560885420673780462140250500038859827219615579403939192298980658769018557378345738134097922
17, 2.7182818284588510799654979549159443339337519231699125879020058914238808418702598596778490958385138279
18, 2.7182818284590332623955795446613996256075309919069645182312471450348823520314338863980943034786794512
19, 2.7182818284590445394630108639169053458225917333016962992409961320982519585417534720712782060778049296
20, 2.7182818284590451971138305938632827023010906830841562369001154781873827126756108615259033060981650683
21, 2.7182818284590452333668320314765767540769679376909143409635744321405460454972397501195895147367874210
22, 2.7182818284590452352614927264128508412230131370462399993554018950185647317174825297415897441916079823
23, 2.7182818284590452353556207793449779065968164719879161144276480189725910169363507144166496729464539700
24, 2.7182818284590452353600769113815444546082663545330303942239506217010794808375644954943873666230210193
25, 2.7182818284590452353602783785779314694018974436961305406991162124639053843342556567482636729415868762
26, 2.7182818284590452353602870948547159572099290991376829054362177765826682842231365031487513774581533095
27, 2.7182818284590452353602874563798511353853987780644518900682925968058463767367342578725767948792148098
28, 2.7182818284590452353602874707798768418168151387959758896593888046529718974668394572681709180950920753
29, 2.7182818284590452353602874713315542114919618752835493081932860309102346782127404445445782913469394140
30, 2.7182818284590452353602874713519121446875677657603550844317750014966275808602992164724460446753523299

```
# method_H4.py
# calculate the number e
# [0] init
from decimal import *
getcontext().prec = 200
from fractions import Fraction
from math import *
N = 30

# [1] calculate sequence a[n]
a = {}
a[0] = 0
for n in range(1, N+1):
    a[n] = a[n-1] + Fraction(n**4, 15 * factorial(n))

# [2] print result
# [2.1] print a[n] as fraction
print("n, a[n]")
for n in range(0, N+1):
    print(f"{n:0>2d},", a[n].numerator, "/", a[n].denominator)

# [2.2] print a[n] as decimal
print("\nn, a[n]")
decimal_from_fraction = lambda x: Decimal(x.numerator) / Decimal(x.denominator)
for n in range(0, N+1):
    print(f"{n:0>2d},", str(decimal_from_fraction(a[n]))[:2 + 100])
```

method_H4.py の出力(1/2)
n, a[n]
00, 0 / 1
01, 1 / 15
02, 3 / 5
03, 3 / 2
04, 199 / 90
05, 307 / 120
06, 1607 / 600
07, 29269 / 10800
08, 13693 / 5040
09, 547963 / 201600
10, 14796001 / 5443200
11, 16440149 / 6048000
12, 180841831 / 66528000
13, 3906183989 / 1437004800
14, 84633987343 / 31135104000
15, 1184875823927 / 435891456000
16, 53319412080811 / 19615115520000
17, 284370197765963 / 104613949440000
18, 966858672404663 / 355687428096000
19, 261051841549265869 / 96035605585920000
20, 183703147756891093 / 67580611338240000
21, 111335241064782491 / 40957946265600000
22, 14567788080861155243 / 5359189738291200000
23, 3055350753492612960443 / 1124000727777607680000
24, 351365336651650490455553 / 129260083694424883200000
25, 25298304238918835312815441 / 9306726025998591590400000
26, 2108192019909902942734678 67 / 775560502166549299200000 00
27, 32242936775092633241824535 9 / 11861513562547224576000000 0
28, 88797047878605111947984776259 / 326666083512550564823040000 00
29, 13812874114449684080797631889 61 / 508147241019523100835840000000
30, 21082807858896886228585859201 51 / 775593157345587890749440000000

method_H4.py の出力(2/2)
n, a[n]
00, 0
01, 0.06666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666
02, 0.6
03, 1.5
04, 2.21111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
05, 2.55833333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333
06, 2.67833333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333
07, 2.710092592592592592592592592592592592592592592592592592592592592592592592592592592592592592592592592592592592592592592592592
08, 2.716865079365079365079365079365079365079365079365079365079365079365079365079365079365079365079365079365079365079365079365079
09, 2.718070436507936507936507936507936507936507936507936507936507936507936507936507936507936507936507936507936507936507936507936
10, 2.718254151969429747207524985302763080540858318636096413874191651969429747207524985302763080540858318636
11, 2.718278604497354497354497354497354497354497354497354497354497354497354497354497354497354497354497354497354497354497354497354
12, 2.718281490500240500240500240500240500240500240500240500240500240500240500240500240500240500240500240500240500240500240500240
13, 2.718281796275141182548589955997363404770812178219585626993034400441807849215256622664030071437478844
14, 2.7182818256524853747075969298191520413742635964858187080409302631524853747075969298191520413742635964
15, 2.718281828233403134196784990435784086577737371388165038958689752340545991339642133292926943720594514
16, 2.7182818284422216851670114456710576640029902816498936428389691176287296216749479536075656005109267895
17, 2.7182818284578760665896909283152669396055639441882828126214369600612986856373099759343145586531829918
18, 2.7182818284589691611701804667879986896482383566105943945968844817273077466043541362557858044941767319
19, 2.718281828459040582597245488739534917676956385443895674325294733128648587836378178852617187621971428 4
20, 2.718281828459044966936043688382050627533616050660295258719423707056186948728760775216784521091039019 8
21, 2.7182818284590452207070537516751089899647568329076019871636363847283302784801629953725879815613954883
22, 2.7182818284590452346012321812077856290357549615133234820370377791671339774286100459339229975634129382
23, 2.7182818284590452353228805870207597969015805293995070309242580628146688307732661284427157846838988439
24, 2.7182818284590452353585296433132921809931795897604212692946788846425765419829763770646173340964352383
25, 2.7182818284590452353602085366165173042734386661195891565877254743327924044554027208469198867511507125
26, 2.7182818284590452353602840776819828652763796799463763176426056966954042034923700563178133258947264672
27, 2.7182818284590452353602873314081994688556067902872971793312790787040070361147498488322420826842799702
28, 2.7182818284590452353602874658084393955488261571148545088481770186105118962623983765244538993658011146
29, 2.7182818284590452353602874711413206357419112764947308880091835390973854434727745868630585074669920558
30, 2.71828182845904523536028747134489996769797018126278865039407324496131446994836230614173604075112121 44

```
# method_H5.py
# calculate the number e
# [0] init
from decimal import *
getcontext().prec = 200
from fractions import Fraction
from math import *
N = 30

# [1] calculate sequence a[n]
a = {}
a[0] = 0
for n in range(1, N+1):
    a[n] = a[n-1] + Fraction(n**5, 52 * factorial(n))

# [2] print result
# [2.1] print a[n] as fraction
print("n, a[n]")
for n in range(0, N+1):
    print(f"{n:0>2d},", a[n].numerator, "/", a[n].denominator)

# [2.2] print a[n] as decimal
print("\nn, a[n]")
decimal_from_fraction = lambda x: Decimal(x.numerator) / Decimal(x.denominator)
for n in range(0, N+1):
    print(f"{n:0>2d},", str(decimal_from_fraction(a[n]))[:2 + 100])
```

method_H5.py の出力(1/2)
n, a[n]
00, 0 / 1
01, 1 / 52
02, 17 / 52
03, 115 / 104
04, 601 / 312
05, 233 / 96
06, 16441 / 6240
07, 101047 / 37440
08, 10945 / 4032
09, 5697961 / 2096640
10, 51291649 / 18869760
11, 512931131 / 188697600
12, 5642263177 / 2075673600
13, 13541437337 / 4981616640
14, 880193465321 / 323805081600
15, 12322708565119 / 4533271142400
16, 184840628542321 / 67999067136000
17, 2957450056760657 / 1087985074176000
18, 773486937923633 / 284549942476800
19, 4839463729255513 / 1780339212288000
20, 17194614630044997689 / 6325545221259264000
21, 26453253276992319097 / 9731608032706560000
22, 722173814461890334773 7 / 2656728992928890880000
23, 3177564783632317478601 1 / 1168960756888711987200 0
24, 3654199501177165100723041 / 1344304870422018785280000
25, 8770078802825196241774360 9 / 3226331689012845084672000 0
26, 9532694350896952436713248 7 / 3506882270666135961600000 0
27, 5700551221836377557154575866 7 / 2097115597858349305036800000 0
28, 3078297659791643880863472197 33 / 1132442422843508624719872000 00
29, 4309616723708301433208861146990 1 / 1585419391980912074607820800000 0
30, 1032883347004468938537660936724 9 / 3799765484912929765588992000000

method_H5.py の出力(2/2)

n, a[n]

00, 0

01, 0.0192307692307692307692307692307692307692307692307692307692307692307692307692307692307692307692307692

02, 0.3269230769230769230769230769230769230769230769230769230769230769230769230769230769230769230769230769

03, 1.1057692307692307692307692307692307692307692307692307692307692307692307692307692307692307692307692307

04, 1.9262820512820512820512820512820512820512820512820512820512820512820512820512820512820512820512820512

05, 2.4270833333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333

06, 2.6347756410256410256410256410256410256410256410256410256410256410256410256410256410256410256410256410

07, 2.6989049145299145299145299145299145299145299145299145299145299145299145299145299145299145299145299145

08, 2.7145337301587301587301587301587301587301587301587301587301587301587301587301587301587301587301587301

09, 2.7176630227411477411477411477411477411477411477411477411477411477411477411477411477411477411477411477

10, 2.7181929711877628544295210961877628544295210961877628544295210961877628544295210961877628544295210961

11, 2.7182705609398317731651064984398317731651064984398317731651064984398317731651064984398317731651064984

12, 2.7182805509498217831551164884498217831551164884498217831551164884498217831551164884498217831551164884

13, 2.7182816976056993418104529215640326751437862548973660084771195882306993418104529215640326751437862548

14, 2.7182818162449739639910580081520252460423400594340765280936221107161278101449041619981790921961862132

15, 2.7182818274124065771654294242816831339419862008384596907185429773952362474950997539520128042716565305

16, 2.7182818283761845047203207561367919528277688635848994009352169710330068490426650784811142971501131859

17, 2.7182818284529511828507682191035874389557743241096924450607804291157974511657865341219024572707926391

18, 2.7182818284586268662494638996350792180235068501486179668563733300739214215715927511756954645221062899

19, 2.7182818284590183106093395007156143139501345081772884422909295156389625706317245231006366989725195305

20, 2.7182818284590436048716368063455126400462479613488245061032120575286069603954702713554482382171402498

21, 2.7182818284590451421383323820630777201579661615007787264864234703502444386972337203761807391433380880

22, 2.7182818284590452303129262617896794681085312084217035977983938581349602204854553874000375714638334432

23, 2.7182818284590452351007858772796042356798739184358059894539899707964895359451928578910666398593649326

24, 2.7182818284590452353475870362279053563140212593959814858645956603743121520124176560426927511769245865

25, 2.7182818284590452353596944398569326876620434446784422115356047206402919294577999429342976982061227176

26, 2.7182818284590452353602609978479243951841010483793459194472063883598804222350549589659984917829408778

27, 2.7182818284590452353602863393694190961376968116115180152916817674653447916978206506649147706246556606

28, 2.7182818284590452353602874249098185040444686206033272152358574359409609698134433743327794438215572115

29, 2.7182818284590452353602874695214211864289306769542161562942773669369223743617828262037987615911352778

30, 2.7182818284590452353602874712831654052794404297547160230865921292209235650159073199615851073191760733

```
# method_I.py
# calculate the constant e
# [0] init
from decimal import *
getcontext().prec = 200
N = 30

# [1] calculate sequences
# [1.1] calculate p[n] and q[n]
p = {}
q = {}
p[-1] = 1
p[0] = 2
q[-1] = 0
q[0] = 1
for n in range(1, N+1):
    p[n] = (n + 1) * p[n-1] + (n + 1) * p[n-2]
    q[n] = (n + 1) * q[n-1] + (n + 1) * q[n-2]

# [1.2] calculate r[n] = p[n] / q[n]
r = {}
for n in range(0, N+1):
    r[n] = Decimal(p[n]) / Decimal(q[n])

# [2] print result
# [2.1] print p[n] and q[n]
print("n, p[n], q[n]")
for n in range(0, N+1):
    print(f"{n:0>2d}, {p[n]}, {q[n]}")

# [2.2] print r[n]
print("\nn, p[n] / q[n]")
for n in range(0, N+1):
    print(f"{n:0>2d},", str(r[n])[:2 + 100])
```

method_I.py の出力(1/2)
n, p[n], q[n]
00, 2, 1
01, 6, 2
02, 24, 9
03, 120, 44
04, 720, 265
05, 5040, 1854
06, 40320, 14833
07, 362880, 133496
08, 3628800, 1334961
09, 39916800, 14684570
10, 479001600, 176214841
11, 6227020800, 2290792932
12, 87178291200, 32071101049
13, 1307674368000, 481066515734
14, 20922789888000, 7697064251745
15, 355687428096000, 130850092279664
16, 6402373705728000, 2355301661033953
17, 121645100408832000, 44750731559645106
18, 2432902008176640000, 895014631192902121
19, 51090942171709440000, 18795307255050944540
20, 1124000727777607680000, 413496759611120779881
21, 25852016738884976640000, 9510425471055777937262
22, 620448401733239439360000, 228250211305338670494289
23, 15511210043330985984000000, 5706255282633466762357224
24, 403291461126605635584000000, 148362637348470135821287825
25, 10888869450418352160768000000, 4005791208408693667174771274
26, 304888344611713860501504000000, 112162153835443422680893595673
27, 8841761993739701954543616000000, 3252702461227859257745914274516
28, 265252859812191058636308480000000, 975810738368357777323774282354 81
29, 8222838654177922817725562880000000, 302501328894190910970370027529 9910
30, 263130836933693530167218012160000000, 968004252461410915105184088095 97121

method_I.py の出力(2/2)
n, p[n] / q[n]
00, 2
01, 3
02, 2.6666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666
03, 2.72727272727272727272727272727272727272727272727272727272727272727272727272727272727272727272727272727272727272727272
04, 2.71698113207547169811320754716981132075471698113207547169811320754716981132075471698113207547169811320754716981132
05, 2.71844660194174757281553398058252427184466019417475728155339805825242718446601941747572815533980582524271844660194174757281553398058252
06, 2.71826333176026427560169891458235016517225106182161396885323265691363850873053327041057102406795658326
07, 2.71828369389344999101096662072271828369389344999101096662072271828369389344999101096662072271828369389344999101096662072271828369389
08, 2.71828165766640373763727929130513925125902554456647048116012377889691159517019598325344335901947697349
09, 2.71828184277782733849203619854037264965879150700360991162832823841624235507066260707667980744414034593
10, 2.718281827351874408807598674393151709622460233074239189649185110350608891109234096803458228583595861837
11, 2.718281828538486166413577881599645166008395908565689602887250395969005897037576506718504228386540176312
12, 2.718281828453728183693079916372034876438145701780891794539149188161070297527595183624903803657370965212
13, 2.718281828459378715874282005573131623824038944205367141284528020615104405819044308516924063086323390612
14, 2.718281828459025557612956186765024269362833063263289806188841321962522515409092266653631479650103358312
15, 2.718281828459046331628328291213545319983841216038572064265639050671927082274318801876233351002038853112
16, 2.718281828459045177516363174291408492809313994469112447989646378062571041733597081439844990749094938512
17, 2.718281828459045238259098180445180735491219128841911176155780842500949490317528386086047459445188603412
18, 2.718281828459045235221961430137492058882649104113785261369320562250471969338087193642352886031706014412
19, 2.718281828459045235365865989675953424281542760219088100967057495303327495443394911411108195272747103712
20, 2.718281828459045235360013100606023362217622820969826735708504657238133667655448688751341174433005772112
21, 2.718281828459045235360298921869933364916053461500028805401887519915469480179029749298089875764257440512
22, 2.718281828459045235360287012650603781470285516944552060881652600529825902027536270185856067390015162412
23, 2.718281828459045235360287489019376964808116234724767566154719927378115574383946095963589517491144189212
24, 2.718281828459045235360287470697501073141276591733217728605219954270280813714035282804537115176171770712
25, 2.718281828459045235360287471376089069128937319251423273739645114757185233431298107571442430492670692112
26, 2.718281828459045235360287471351853783557949436125773075693295942000394229065988494629829557761492474912
27, 2.718281828459045235360287471352689483060397294164588599763852512687489138684851665484402615760071073112
28, 2.718281828459045235360287471352661626410315698896628082294833952052568870323531691628248559517741552212
29, 2.718281828459045235360287471352662525011931234227852615116415195935106198828881934566706530732796752912
30, 2.718281828459045235360287471352662496930630748748751848465740782063767914374134809056820995349956761312

```
# method_J.py
# calculate the constant e
# [0] init
from decimal import *
getcontext().prec = 200
N = 30

# [1] calculate sequences
# [1.1] calculate p[n] and q[n]
p = {}
q = {}
p[-1] = 1
p[0] = 3
q[-1] = 0
q[0] = 1
for n in range(1, N+1):
    p[n] = (n + 3) * p[n-1] - n * p[n-2]
    q[n] = (n + 3) * q[n-1] - n * q[n-2]

# [1.2] calculate r[n] = p[n] / q[n]
r = {}
for n in range(0, N+1):
    r[n] = Decimal(p[n]) / Decimal(q[n])

# [2] print result
# [2.1] print p[n] and q[n]
print("n, p[n], q[n]")
for n in range(0, N+1):
    print(f"{n:0>2d}, {p[n]}, {q[n]}")

# [2.2] print r[n]
print("\nn, p[n] / q[n]")
for n in range(0, N+1):
    print(f"{n:0>2d},", str(r[n])[:2 + 100])
```

method_J.py の出力(1/2)
n, p[n], q[n]
00, 3, 1
01, 11, 4
02, 49, 18
03, 261, 96
04, 1631, 600
05, 11743, 4320
06, 95901, 35280
07, 876809, 322560
08, 8877691, 3265920
09, 98641011, 36288000
10, 1193556233, 439084800
11, 15624736141, 5748019200
12, 220048367319, 80951270400
13, 3317652307271, 1220496076800
14, 53319412081141, 19615115520000
15, 909984632851473, 334764638208000
16, 16436597430879731, 6046686277632000
17, 313262209859119579, 115242726703104000
18, 6282647653285676001, 2311256907767808000
19, 132266266384961600021, 48658040163532800000
20, 2916471173788403280463, 1072909785605898240000
21, 67217716576837485130671, 24728016011107368960000
22, 1616280548597592256096589, 594596384994354462720000
23, 404772867822701365005058 81, 148907616415977465446400 00
24, 10540960099549514713673406 51, 38778025108327464960000 0000
25, 28502756109181887785772891203, 10485577989291746525184000000
26, 799173430907446007531862987961, 293999475161295508340736000000
27, 23205628512275469255740021576349, 8536873649127988094042112000000
28, 696997627815131058717048505203911, 256411097818451356681764864000000
29, 216309608632282052705290915408 11031, 795758579436573175908925440000 0000
30, 692911779652076842165948565690646693, 254907998279515607349492449280000000

method_J.py の出力(2/2)
n, p[n] / q[n]
00, 3
01, 2.75
02, 2.722222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222
03, 2.71875
04, 2.718333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333
05, 2.71828703703703703703703703703703703703703703703703703703703703703703703703703703703703703703703703703703703703703
06, 2.71828231292517006802721088435374149659863945578231292517006802721088435374149659863945578231292517 00
07, 2.7182818700396825396825396825396825396825396825396825396825396825396825396825396825396825396825396825396825396825
08, 2.7182818317656280619243582206545169508132471095434058397021359984322947285910248873211836174799137762
09, 2.718281828703703703703703703703703703703703703703703703703703703703703703703703703703703703703703703703703703703703
10, 2.7182818284759572638360517148395936274724153512032299911087789875668663547451426239305027183815062602
11, 2.7182818284601415388452425489462526499563536600573637610674647711684748721785758822795859832896869933
12, 2.718281828459112112958266804420650574496728342882189036035189881343727497573651419805265959112112958 2
13, 2.7182818284590490868835540037354096311012410785653416038903567248238450052509009425109198351828737316
14, 2.7182818284590454453770150419180401543717240366270348633664330313360295723611420260450242813558510207
15, 2.7182818284590452462321261924436527611130785733960337134940309543484146658749833353007956182559357162
16, 2.7182818284590452358958862867615911317051904005639748302826606043317564527355633340510951686140716001
17, 2.7182818284590452353854546864809954956850477747451077249388892290222918496175672846066655167799054709
18, 2.7182818284590452353614177413154272801383928588478203543271326822625663696923430939126065027046123290
19, 2.7182818284590452353603360787829767104387933876324424226496036376583787230957080053313738470712241377
20, 2.7182818284590452353602894765423495883882437505506007430535196198636404798183019584310486306153638754
21, 2.7182818284590452353602875508299269800390474845554833182768219331779074945589050143442583324147084927
22, 2.7182818284590452353602874743838761770611020751303463316599776015703453155599497481328734623538129104
23, 2.7182818284590452353602874714640617366695833268536917939766953527936675934454063178261886235667648153
24, 2.7182818284590452353602874713566125652631754369171109069899505660386858532715911195909026214994014453
25, 2.7182818284590452353602874713527978017812911331323802246117229404734202293600947220204190947987849943
26, 2.7182818284590452353602874713526669799883595726459354138305765884307156469214288647649155445004099446
27, 2.7182818284590452353602874713526626415105327606910278053097732655313402398507588235804218043119434251
28, 2.7182818284590452353602874713526625022527553842850401764025877121709156447512494592051534654132678 30
29, 2.7182818284590452353602874713526624978919564248470032775794294313939816056856607808377627011884631541
30, 2.7182818284590452353602874713526624977611902913466250034256687125023372885109527050964517472704266866

```
# method_K.py
# calculate the constant e
# [0] init
from decimal import *
getcontext().prec = 200
N = 30

# [1] calculate sequences
# [1.1] calculate p[n] and q[n]
def a(n):
    if n % 2 == 0:
        return n // 2 + 1
    else:
        return -1

p = {}
q = {}
p[-1] = 1
p[0] = 3
q[-1] = 0
q[0] = 1
for n in range(1, N+1):
    p[n] = p[n-1] + a(n) * p[n-2]
    q[n] = q[n-1] + a(n) * q[n-2]

# [1.2] calculate r[n] = p[n] / q[n]
r = {}
for n in range(0, N+1):
    r[n] = Decimal(p[n]) / Decimal(q[n])

# [2] print result
# [2.1] print p[n] and q[n]
print("n, p[n], q[n]")
for n in range(0, N+1):
    print(f"{n:0>2d}, {p[n]}, {q[n]}")

# [2.2] print r[n]
print("\nn, p[n] / q[n]")
for n in range(0, N+1):
    print(f"{n:0>2d},", str(r[n])[:2 + 100])
```

method_K.py の出力(1/2)
n, p[n], q[n]
00, 3, 1
01, 2, 1
02, 8, 3
03, 6, 2
04, 30, 11
05, 24, 9
06, 144, 53
07, 120, 44
08, 840, 309
09, 720, 265
10, 5760, 2119
11, 5040, 1854
12, 45360, 16687
13, 40320, 14833
14, 403200, 148329
15, 362880, 133496
16, 3991680, 1468457
17, 3628800, 1334961
18, 43545600, 16019531
19, 39916800, 14684570
20, 518918400, 190899411
21, 479001600, 176214841
22, 6706022400, 2467007773
23, 6227020800, 2290792932
24, 93405312000, 34361893981
25, 87178291200, 32071101049
26, 1394852659200, 513137616783
27, 1307674368000, 481066515734
28, 22230464256000, 8178130767479
29, 20922789888000, 7697064251745
30, 376610217984000, 138547156531409

method_K.py の出力(2/2)
n, p[n] / q[n]
00, 3
01, 2
02, 2.66666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666
03, 3
04, 2.72727272727272727272727272727272727272727272727272727272727272727272727272727272727272727272727272727272727272727272
05, 2.66666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666
06, 2.71698113207547169811320754716981132075471698113207547169811320754716981132075471698113207547169811320754716981132
07, 2.72727272727272727272727272727272727272727272727272727272727272727272727272727272727272727272727272727272727272727272
08, 2.71844660194174572815533980582524271844660194174572815533980582524271844660194174572815533980582524
09, 2.71698113207547169811320754716981132075471698113207547169811320754716981132075471698113207547169811320754716981132
10, 2.71826333176026427560169891458235016517225106182161396885323265691363850873053327041057102406795658326
11, 2.71844660194174572815533980582524271844660194174572815533980582524271844660194174572815533980582524
12, 2.718283693893449991010966620722718283693893449991010966620722718283693893449991010966620722718283693838
13, 2.71826333176026427560169891458235016517225106182161396885323265691363850873053327041057102406795658326
14, 2.71828165766640373763729291305139251259025544566470481160123778896911595170195983253443359019476973 4
15, 2.718283693893449991010966620722718283693893449991010966620722718283693893449991010966620722718283693838
16, 2.71828184277782733849203619854037264965879150700360991162832823841624235507066260707667980744414034 59
17, 2.71828165766640373763729291305139251259025544566470481160123778896911595170195983253443359019476973 4
18, 2.71828182735187440880759867439315170962246023307423918964918511035060889110923409680345822858359586 18
19, 2.71828184277782733849203619854037264965879150700360991162832823841624235507066260707667980744414034 59
20, 2.71828182853848616641357788159964516600839590856568960288725039596900589703757650671850422838654017 63
21, 2.71828182735187440880759867439315170962246023307423918964918511035060889110923409680345822858359586 18
22, 2.71828182845372818369307991637203487643814570178089179453914918816107029752759518362490380365737096 52
23, 2.71828182853848616641357788159964516600839590856568960288725039596900589703757650671850422838654017 63
24, 2.71828182845937871587428200557313162382403894420536714128452802061510440581904430851692406308632339 06
25, 2.71828182845372818369307991637203487643814570178089179453914918816107029752759518362490380365737096 52
26, 2.71828182845902555761295618676502426936283306326328980618884132196252251540909226665363147965010335 83
27, 2.71828182845937871587428200557313162382403894420536714128452802061510440581904430851692406308632339 06
28, 2.71828182845904633162832829121354531998384121603857206426563905067192708227431880187623335100203885 31
29, 2.71828182845902555761295618676502426936283306326328980618884132196252251540909226665363147965010335 83
30, 2.71828182845904517751636317429140849280931399446911244798964637806257104173359708143984499074909493 85

```
# method L
# calculate the number e
# [0] init
from decimal import *
getcontext().prec = 200
from sympy import prime
N = 30

# [1] define sequence a(n)
def a(n):
    q = 1
    for i in range(1, n + 1):
        q *= prime(i)
    a_n = Decimal(q) ** (1 / Decimal(prime(n)))
    return a_n

# [2] print a[n] as decimal
print("n, a[n]")
for n in range(1, N+1):
    print(f"{n:0>2d},", str(a(n))[:2 + 100])
```

method_L.py の出力(1/1)
n, a[n]
01, 1.41421356237309504880168872420969807856967187537694807317667973799073247846210703885038753432764157 27
02, 1.81712059283213965889121175632726050242821046314121967148133429793130973945930186564714170412641707 21
03, 1.97435048583481984267283617240845318268226724809535471679407794339424719810702124614532324978243046 58
04, 2.14657275836544392815497869531416578609150798968799185898683469122233020542976239267667000988954231 93
05, 2.02200817568580954718641626450962959658746589780549719047369827287963365071624676886639961710021066 61
06, 2.21017841524702456910988033529808074012892794078395754717726735210963325845710164611479280606633491 28
07, 2.16653124213280393999080511818331084326554994702663768690120486368487308123087573232525990736801344 39
08, 2.33197611343692868250780395695094539004429199218702245485350250169372723839005052226577644059398087 57
09, 2.30662842024558362798669411247147531261401722704886468497758365288807207254258816339987515398421669 32
10, 2.17924638423188508032495357642134859353750074481854631191679709317694041477968734611238406506187043 97
11, 2.31520025025470522154565452984181587318888052048262331275108092504454378259402773067568565154834692 56
12, 2.22766520678654010052898527209296976372146541095367575353657283860690107162758549319189124624227269 77
13, 2.25553481866597940917147502546336284467025943508010163469834441801407849834053780882889385947460607 29
14, 2.37031990315158708159421423828038623321066936511657045368060335638618428687296988550371933901357108 04
15, 2.39047963630243097890113580105639428033482840470618847233531874180292100259671057619089383955644537 13
16, 2.33437805274503614665518827692817896498683830188154445456418484059397081770671861171724130485387387 9
17, 2.29479816640970029809537460572050570180345452897362641728271001505970359585234106061661692771275048 46
18, 2.38882594944092663207694123320380372448445805753582560511558331252114918376515785536880086274964322 16
19, 2.35272997957579504430720251560248962283332729282998678976903267894188857589895515678346624746526806 14
20, 2.38074296167119137433214069921389637771653625801960603101216154464089614277518834824679548281095853 10
21, 2.46556575349135315757260385248336871828263804109339777311297228229514871434415855511987978409820562 30
22, 2.43316392205124238369204496034819900603517394776438209690473581075453852567099470321889062531585089 71
23, 2.45856742299527342514059963109933517465759904012051276731093025061858623296897883422329241930449280 98
24, 2.43358915098132959414096649463871570918742283667289810244767268167604293730944474475645039415559199 34
25, 2.37068580494883101855628780485629618770023104127321800114326369522792122575414243607871513281527531 61
26, 2.39812750573274513439221098589558875442685426653768837684463836139539187581102276494240166607044921 83
27, 2.46625636278001173463985766209410310504332844809663971319144273914600237491169268274703357882736523 88
28, 2.49085695684058107751876675412733964221522373752801936336764769227816537130454150646467007364835302 05
29, 2.55722133606582385997494608191797588143888594042523706657963120975279432776424259069180204223178016 03
30, 2.57930605872999733569435897789726177757567961474651216720657111905719024865007629275833531853707510 83

出力まとめ
各手法の数列の第 30 項をまとめておく.
2.67431877587029459606443544485556782143840834421501985150284464935142158907328411801535352923293852 11, 方法 A
2.7182818284590452353602874713526623722257021309834818158153785765237928007911999934379359859554179170, 方法 B
2.7182818284590452353602874713526616264103156988966280822948339520525688703235316916282485595177415522, 方法 C
2.7182818284590452353602874713526616264103156988966280822948339520525688703235316916282485595177415522, 方法 D
2.7182818284590452353602874713526624977572470936999595749669676277240766303535475945713821785251664273, 方法 E
2.7182818284590452353602874713526624977572470936999595749669676277240766303535475945713498898130771109, 方法 F2
2.7182818284590452353602874713526624977572470936999595749669676277240766303535475945693797578562749426, 方法 F3
2.7182818284590452353602874713526604872318877230306598893546157534709607273742881824405352574597393602, 方法 F4
2.7182818284590452353602874713526624977572470936999595749669676277240766303535475945570558904130374026, 方法 F9
2.7182818284590452353602874713526624977572470936999595749669676277240766303535475945733200217698792792, 方法 F10
2.7182818284590452353602874713526624977572470936999595749669676277240766303535475945713821785251664274, 方法 F11
2.7182818284590452353602874713526624978919564248470032775794294313939816056856607808377627011884631541, 方法 G
2.7182818284590452353602874713526020524236410764931801690709682388331664514500220415211126740937041009, 方法 H2
2.7182818284590452353602874713519121446875677657603550844317750014966275808602992164724460446753523299, 方法 H3
2.7182818284590452353602874713448999676979701812627886503940732449613144699483623061417360407511212144, 方法 H4
2.7182818284590452353602874712831654052794404297547160230865921292209235650159073199615851073191760733, 方法 H5
2.7182818284590452353602874713526624969306307487487518484657407820637679143741348090568209953499567613, 方法 I
2.7182818284590452353602874713526624977611902913466250034256687125023372885109527050964517472704266866, 方法 J
2.7182818284590451775163631742914084928093139944691124479896463780625710417335970814398449907490949385, 方法 K
2.5793060587299973356943589778972617775756796147465121672065711905719024865007629275833531853707510 83, 方法 L