Fibonacci Numbers

Dr. Jürgen Gerhard, Senior Director of Research

Many programming language tutorials have an example about computing Fibonacci numbers to illustrate recursion. Usually, however, these simple examples exhibit an abysmal runtime behaviour, namely, exponential in the index.

In this presentation, several more efficient ways of computing Fibonacci numbers, using Maple, are discussed. The best algorithm presented is based on doubling formulae for the Fibonacci numbers, which we also prove using Maple.

Definition

The well-known Fibonacci numbers are defined by

```
> F(0) := 0;
  F(1) := 1
```

$$F(0) := 0$$
$$F(1) := 1 \tag{1.1}$$

and the following recurrence

```
> rec := F(n+1) = F(n)+F(n-1):
  rec
```

$$F(n+1) = F(n) + F(n-1) \tag{1.2}$$

The first 12 Fibonacci numbers using the built-in fibonacci command:

```
> seq(combinat:-fibonacci(n),n=0..11)
```

$$0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89 \tag{1.3}$$

Using Maple to Prove the Doubling Formulae

We want to prove the following **doubling formulae** for $n \geq 1$

```
> claim1 := F(2*n-1) = F(n)^2+F(n-1)^2;
  claim2 := F(2*n) = F(n)^2+2*F(n-1)*F(n)
```

$$claim1 := F(2n-1) = F(n)^2 + F(n-1)^2$$
$$claim2 := F(2n) = F(n)^2 + 2F(n-1)F(n) \tag{2.1}$$

Base case $n = 1$

```
> eval(rec, n=1)
```

$$F(2) = 1 \tag{2.2}$$

```
> eval([claim1,claim2], n=1)
```

$$[1 = 1, F(2) = 1] \tag{2.3}$$

Induction step $n \to n+1$

```
> eval(claim1, n=n+1)
```

$$F(2\,n+1) = F(n+1)^2 + F(n)^2 \tag{2.4}$$

```
> eval((2.4), eval(rec,n=2*n))
```

$$F(2\,n) + F(2\,n-1) = F(n+1)^2 + F(n)^2 \tag{2.5}$$

Use the induction hypotheses

```
> eval((2.5), [claim1, claim2])
```

$$2\,F(n)^2 + 2\,F(n-1)\,F(n) + F(n-1)^2 = F(n+1)^2 + F(n)^2 \tag{2.6}$$

```
> eval((2.6), rec)
```

$$2\,F(n)^2 + 2\,F(n-1)\,F(n) + F(n-1)^2 = (F(n) + F(n-1))^2 + F(n)^2 \tag{2.7}$$

Simplify

```
> expand((lhs-rhs)((2.7)))
```

$$0 \tag{2.8}$$

This concludes the inductive step for the first claim.
Now for the second claim:

```
> eval(claim2, n=n+1)
```

$$F(2\,n+2) = F(n+1)^2 + 2\,F(n)\,F(n+1) \tag{2.9}$$

```
> eval((2.9), eval(rec,n=2*n+1))
```

$$F(2\,n+1) + F(2\,n) = F(n+1)^2 + 2\,F(n)\,F(n+1) \tag{2.10}$$

Use induction hypotheses and what we just proved:

```
> eval((2.10), [eval(claim1, n=n+1), claim2])
```

$$F(n+1)^2 + 2\,F(n)^2 + 2\,F(n-1)\,F(n) = F(n+1)^2 + 2\,F(n)\,F(n+1) \tag{2.11}$$

```
> eval((2.11), rec)
```

$$(F(n) + F(n-1))^2 + 2\,F(n)^2 + 2\,F(n-1)\,F(n) = (F(n) + F(n-1))^2 + 2\,F(n)\,(F(n) + F(n-1)) \tag{2.12}$$

```
> expand((lhs-rhs)((2.12)))
```

$$0 \tag{2.13}$$

Q.E.D.

Alternative: using the solve command

Collect the induction hypotheses, as well as all the required versions of the recursion formula from the definition of the Fibonacci numbers.

```
> eqns := [claim1, claim2, rec, eval(rec, n=2*n), eval(rec, n=2*
  n+1)]:
  Display(eqns)
```

$$\begin{aligned} F(2n-1) &= F(n)^2 + F(n-1)^2 \\ F(2n) &= F(n)^2 + 2F(n-1)F(n) \\ F(n+1) &= F(n) + F(n-1) \\ F(2n+1) &= F(2n) + F(2n-1) \\ F(2n+2) &= F(2n+1) + F(2n) \end{aligned} \tag{2.14}$$

Express everything in terms of $F(n+1)$ and $F(n)$. The first two equations provide the proof of the inductive step.

```
> sol := solve(eqns, [F(2*n+1), F(2*n+2), F(2*n), F(2*n-1), F(n-1)
  ])[]:
  Display(sol)
```

$$\begin{aligned} F(2n+1) &= F(n+1)^2 + F(n)^2 \\ F(2n+2) &= F(n+1)^2 + 2F(n)F(n+1) \\ F(2n) &= 2F(n)F(n+1) - F(n)^2 \\ F(2n-1) &= F(n+1)^2 - 2F(n)F(n+1) + 2F(n)^2 \\ F(n-1) &= F(n+1) - F(n) \end{aligned} \tag{2.15}$$

Computing large Fibonacci Numbers

The naive way

```
> fib := proc(n::nonnegint, $)
      if n <= 1 then
          return n;
      else
          return fib(n-1) + fib(n-2);
      end if;
  end proc
```

fib := **proc**(*n*::*nonnegint*, *$*) **(3.1.1)**

if *n* <= 1 **then return** *n* **else return** *fib*(*n* − 1) + *fib*(*n* − 2) **end if**

end proc

```
> seq(fib(n),n=0..11)
```

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89 **(3.1.2)**

```
> CodeTools:-Usage(fib(25))
memory used=16.67MiB, alloc change=0 bytes, cpu time=
141.00ms, real time=138.00ms, gc time=0ns
```

75025 **(3.1.3)**

```
> CodeTools:-Usage(fib(26))
memory used=26.98MiB, alloc change=0 bytes, cpu time=
234.00ms, real time=241.00ms, gc time=15.60ms
```

121393 **(3.1.4)**

```
> CodeTools:-Usage(fib(27))
memory used=43.65MiB, alloc change=0 bytes, cpu time=
374.00ms, real time=368.00ms, gc time=15.60ms
```

(3.1.5)

196418 **(3.1.5)**

```
> CodeTools:-Usage(fib(28))
memory used=70.62MiB, alloc change=0 bytes, cpu time=
577.00ms, real time=599.00ms, gc time=31.20ms
```

317811 **(3.1.6)**

```
> CodeTools:-Usage(fib(29))
memory used=114.27MiB, alloc change=0 bytes, cpu time=
936.00ms, real time=954.00ms, gc time=31.20ms
```

514229 **(3.1.7)**

The better way

Maple provides an option for procedures to remember previous results in a remember table. Subsequent invocations of the procedure with the same arguments simply retrieve the result from the remember table. Option remember allows for the coding of a function with a recursive definition in the most natural manner, without a loss of efficiency.

For example, the Fibonacci numbers computed with

$F := \mathbf{proc}(n)\ \mathbf{if}\ n < 2\ \mathbf{then}\ n\ \mathbf{else}\ F(n-1) + F(n-2)\ \mathbf{end\ if}\ \mathbf{end\ proc};$

takes exponential time to compute, whereas the following requires linear time.:

```
F := proc(n) option remember;
if n < 2 then n else F(n-1) + F(n-2) end if
end proc;
```

```
> fib := proc(n::nonnegint, $)
  option remember;
      if n <= 1 then
          return n;
      else
          return fib(n-1) + fib(n-2);
      end if;
  end proc
```

fib := **proc**(*n*::*nonnegint*, *$*) **(3.2.1)**
  option *remember*;
  if *n* <= 1 **then return** *n* **else return** *fib*(*n* − 1) + *fib*(*n* − 2) **end if**
end proc

```
> CodeTools:-Usage(fib(10000))
memory used=5.95MiB, alloc change=32.00MiB, cpu time=
78.00ms, real time=88.00ms, gc time=31.20ms
```

33644764876431783266621612005107543310302148460680063906564769974680081\ **(3.2.2)**
44216666236815559551363373402558206533268083615937373479048386526826\
30408924630564318873545443695598274916066020998841839338646527313000\
88830269235673613135117579297437854413752130520504347701602264758318\
90652789085515436615958298727968298751063120057542878345321551510387\

```
08182989697916131278562650331954871402142875326981879620469360978799\
00350962302291026368131493195275630227837628441540360584402572114334\
96118002309120828704608892396232883546150577658327125254609359112820\
39252853934346209042452489294039017062338889910858410651831733604374\
70737908552631764325733993712871937587746897479926305837065742830161\
63740896917842637862421283525811282051637029808933209990570792006436\
74262023897831114700540749984592503606335609338838319233867830561364\
35351892133279732908133732642652633989763922723407882928177953580570\
99369104917547080893184105614632233821746563732124822638309210329770\
16480547262438423748624114530938122065649140327510866433945175121615\
26545361333111314042436854805106765843493523836959653428071768775328\
34823434555736671973139274627362910821067928078471803532913117677892\
46590899386354593278945237776744061922403376386740040213303432974969\
02028328145933418826817683893072003634795623117103101291953169794607\
63273758925353077255237594378843450406771555577905645044301664011946\
25809722167297586150269684431469520346149322911059706762432685159928\
34709891284706740862008587135016260312071903172086094081298321581077\
28207635318662461127824553720853236530577595643007251774431505153960\
09051686032203491632226408852488524331580515348496224348482993809050\
70483482449327453732624567755879089187190803662058009594743150052402\
53270974699531877072437682590741993963226598414749819360928522394503\
97071654431564213281576889080587831834049174345562705202235648464951\
96112460268313970975069382648706613264507665074611512677522748621598\
64253071129844118262266105716351506926002986170494542504749137811515\
41399415506712562711971332527636319396069028956502882686083622410820\
50562430701794976171121233066073310059947366875
```

```
> CodeTools:-Usage(fib(20000))
memory used=15.86MiB, alloc change=2.00MiB, cpu time=
31.00ms, real time=26.00ms, gc time=0ns
```

```
25311623237323612422401550035206072917663564858024852789519298419913127\(3.2.3)
817605413152301534234637588316374434882192110376890336735314627428855\
32972407155518761802693163044919315892277133164230203033197109868923\
57808434782585027792002936356518974833096860428609963644435145587721\
56043691404155819572984971754278513112487985892718229593329483578531\
41914880538028162426090036299355691663861393997707468501618825858431\
23291395263935580968408129704229524185589918557723068824425748555892\
37165219912238201311184749075137322987656049866305366913734924425822\
68133896650746385518023628358240986119921232383594789114376541491334\
50084560220094557042108916377919112654751677697044773348591098225900\
53774932978465651023851447920601310106288957894301592502061560528131\
20307277867749144342092182259070991044861732915613535546462089178845\
95660815728248895142963506709508242082451706676017264170911279999999\
41149913010424532046881958285409468463211897582215075436515584016297\
```

$$
87457218390794925728626160861240137963948471310113812040467173219045\\
13278814332010251840275416961241144634886653593858709103314761566658\\
89459832092710304159637019707297988417848767011085425271875588008671\\
42249143400511528833434383777879228238357673634141441024899408156483\\
02023638205041900745045666125159651346656832893561887275494637328300\\
75811851574961558669278847363279870595320099844676879457196432535973\\
35712830539029047134948025875181289031477972350810422952516174064398\\
44239786596382330744631003665005719772345084647100781025813048232354\\
36518145074482824812996511614161933313389889630935320139507075992100\\
56107753402820725757425770627820130830264263467811259109184308266572\\
16971178387264317667411587435542988645609932555476084966868501858046\\
59790217122426535133253371422250684486113457341827911625517128815447\\
32595854791211324236720199067223068130881919594101615600196195470024\\
15765537507376815522568454211593868583994334500459039751670842528768\\
48848085910156941603293424067793097271128806817514906531652407763118\\
30816237703346320351465753121041314919121359545528038763103066559458\\
91836015753400271729972224890816311447288736218055286487685113689486\\
39522975539046995395707688938978847084621586473529546678958226255042\\
38999871814130305503606077200388777303842236691382039774855079317816\\
72201933460174300241344961411459918962277418425157189978986272699182\\
36920453493946658273870473264523119133765447653295022886429174942653\\
01465652190946961318498367143146593496548942551598106754608734234835\\
07242075835444361072940876379750251478462545269384424356449282310278\\
68701394819091132912397475713787593612758364812687556725146456646878\\
91216927421920970816667866815218494157859020195314403051938192227325\\
26666526717175263186066767545561703793509563420954556127802021999226\\
15392785572481747913435560866995432578680971243966868110016581395696\\
31092251980368583746079535838461801721546812288044225234368454723366\\
85023132393283526713181306042474604521341218333052843987264385737877\\
98499612760939462427922917659263046333084007208056631996856315539698\\
23402295345221150567562915363786725269505692534522008402007161122057\\
57008412683026389952728421609942196326845753641801609918848850918582\\
59996299627148614456696661412745040519981575543804847463997422326563\\
89704380373297039748847164490618331014469124364914954239469152497202\\
39351906336728273061165257128829591084342116524656211447020153366574\\
59532134026915214509960877430595844287585350290234547564574848753110\\
28110154593154722581176344171021745297966817802528646015832465885290\\
41057924724681089961354766372120575081921769109004228269695234389853\\
32067597093454021924077101784215936539638808624420121459718286059401\\
82361421321432600427047175280272562581095378771389884614425690983511\\
63712350195270131802040301676015670642685738206979488689826309041646\\
85161783088076506964317303709708574052747204405282785965604677674192\\
$$

```
569851918643651835755242670293612851920696732320545562286110332140065
912751551110134916256237884844001366366654055079721985816714803952465
293015580969682022616988370960903778630177970204880448266288174628665
854321356787305635653577619877987998113667928954840972022833505708585
756190202341139891582348762729796894762141691281636751612509656370515
742204606398576839712130931255
```

> CodeTools:-Usage(fib(50000))

```
memory used=95.50MiB, alloc change=153.12MiB, cpu time=
734.00ms, real time=497.00ms, gc time=468.00ms
```

107777348930729747802790388551194808296251067694115797849023092100327445 **(3.2.4)**
735364652304984884440204760295[...10250 digits...]
214921063358870308182304266593049366953768037220397037490781969011125
665240202976183053642523735531255

> CodeTools:-Usage(fib(100000))

```
memory used=316.78MiB, alloc change=0.60GiB, cpu time=
1.48s, real time=782.00ms, gc time=998.41ms
```

259740693472217241661550340212759154148804853865176965847247707039525345 **(3.2.5)**
543511273686265556772836716745[...20699 digits...]
925913043557232163566089560351438388393901895316627435560997001569978
802892363623498953746534287468755

> CodeTools:-Usage(fib(200000))

```
Error, (in fib) too many levels of recursion
```

For an iterative version of the procedure, only two intermediate variables for storing the previous two results are required, and no stack memory for recursive calls is needed, either.

```
> fib := proc(n::nonnegint, $)
  local a, b, i;
      if n = 0 then
          return n;
      end if;
      a, b := 0, 1;
      for i from 2 to n do
          a, b := b, a+b;
      end do;
      return b;
  end proc:
```

> seq(fib(n), n=0..11)

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89 **(3.2.6)**

> CodeTools:-Usage(fib(100000))

```
memory used=421.79MiB, alloc change=-4.00MiB, cpu time=
483.00ms, real time=420.00ms, gc time=109.20ms
```

259740693472217241661550340212759154148804853865176965847247707039525345 **(3.2.7)**
543511273686265556772836716745[...20699 digits...]
925913043557232163566089560351438388393901895316627435560997001569978
802892363623498953746534287468755

> CodeTools:-Usage(fib(200000))

```
memory used=1.63GiB, alloc change=0 bytes, cpu time=1.44s,
real time=1.31s, gc time=249.60ms
```

(3.2.8)

150856835579889389926368789481731344418484165616586897402143008140078755 (3.2.8)
839363948123516190995732093475[...41598 digits...]
486192612635235963032921420195770467711978919314265479877731245093503
108942980265224625940817585312 5

```
> CodeTools:-Usage(fib(500000))
memory used=10.14GiB, alloc change=0 bytes, cpu time=7.16s,
real time=7.04s, gc time=670.80ms
```

295556140826951512599006844757840509167027654471016272968257332870870535 (3.2.9)
183700218928163438697493576475[...104294 digits...]
658183135098783658399739851044270021891035772777060604335742720797401
022820167547380851846978045312 5

Memory usage is still high because the numbers in the intermediate results are getting big.

The best way: use the duplication formula

```
> claim1;
  claim2
```

$$F(2\,n-1) = F(n)^2 + F(n-1)^2$$

$$F(2\,n) = F(n)^2 + 2\,F(n-1)\,F(n) \qquad (3.3.1)$$

```
> restart:
> fib := proc(n::nonnegint, $)
  option remember;
      if n <= 1 then
          return n;
      elif type(n, 'odd') then
          return fib((1/2)*n+1/2)^2+fib((1/2)*n-1/2)^2;
      else # type(n,'even')
          return fib((1/2)*n)*(fib((1/2)*n)+2*fib((1/2)*n-1));
      end if;
  end proc:
> seq(fib(n),n=0..11)
```

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89 (3.3.2)

```
> CodeTools:-Usage(fib(500000));
memory used=384.27KiB, alloc change=0 bytes, cpu time=0ns,
real time=1000.00us, gc time=0ns
```

295556140826951512599006844757840509167027654471016272968257332870870531 8 (3.3.3)
370021892816343869749357647[...104294 digits...]
658183135098783658399739851044270021891035772777060604335742720797401
022820167547380851846978045312 5

```
> CodeTools:-Usage(fib(1000000));
memory used=0.63MiB, alloc change=0 bytes, cpu time=0ns, real
time=3.00ms, gc time=0ns
```

195328212870775773163201494759625633244354299659187339695340519457162525 7 (3.3.4)
887015694766641987634150146[...208788 digits...]
832396965438555237837814138667507928683720580204334722541903368468430 1
719893411568996526838242546875

```
> CodeTools:-Usage(fib(2000000));
```

```
memory used=1.77MiB, alloc change=0 bytes, cpu time=0ns, real
time=5.00ms, gc time=0ns
```

85312949175076415430516606545038251619554336100240130700596358588383980061 (3.3.5)
887605974653683126461696733[...417775 digits...]
247646358470776374996874962715191981012227644442159375938945392769179949310896082512918877780345312 5

```
> CodeTools:-Usage(fib(5000000)); length(%)*digits;
memory used=8.17MiB, alloc change=2.91MiB, cpu time=31.00ms,
real time=30.00ms, gc time=0ns
```

[Length of output exceeds limit of 1000000]

1044938 *digits* (3.3.6)

```
> CodeTools:-Usage(fib(10000000)): length(%)*digits;
memory used=10.88MiB, alloc change=2.60MiB, cpu time=31.00ms,
real time=34.00ms, gc time=0ns
```

2089877 *digits* (3.3.7)

```
> CodeTools:-Usage(fib(20000000)): length(%)*digits;
memory used=22.90MiB, alloc change=6.46MiB, cpu time=78.00ms,
real time=83.00ms, gc time=0ns
```

4179753 *digits* (3.3.8)

```
> CodeTools:-Usage(fib(50000000)): length(%)*digits;
memory used=113.53MiB, alloc change=31.53MiB, cpu time=
453.00ms, real time=460.00ms, gc time=0ns
```

10449382 *digits* (3.3.9)

```
> CodeTools:-Usage(fib(100000000)): length(%)*digits;
memory used=118.49MiB, alloc change=31.44MiB, cpu time=
546.00ms, real time=556.00ms, gc time=0ns
```

20898764 *digits* (3.3.10)

```
> evalf(10^8*log[10]((1+sqrt(5))/2))
```

$2.089876400\ 10^{7}$ (3.3.11)

Fibonacci polynomials

```
> F(0) := 0;
  F(1) := 1
```

$$F(0) := 0$$
$$F(1) := 1 \quad (4.1)$$

Recurrence formula

```
> F(n+1) = x*F(n)+F(n-1)
```

$$F(n+1) = x\,F(n) + F(n-1) \quad (4.2)$$

First try

```
> fp := proc(n::integer, x, $)
  option remember;
      if n <= 1 then
          return n;
      else
          return x*fp(n-1,x)+fp(n-2,x);
      end if;
```

```
  end proc:
> for n from 0 to 9 do
      print(fp(n,x));
  end do
```

$$0$$

$$1$$

$$x$$

$$x^2+1$$

$$x\left(x^2+1\right)+x$$

$$x\left(x\left(x^2+1\right)+x\right)+x^2+1$$

$$x\left(x\left(x\left(x^2+1\right)+x\right)+x^2+1\right)+x\left(x^2+1\right)+x$$

$$x\left(x\left(x\left(x\left(x^2+1\right)+x\right)+x^2+1\right)+x\left(x^2+1\right)+x\right)+x\left(x\left(x^2+1\right)+x\right)+x^2+1$$

$$x\left(x\left(x\left(x\left(x^2+1\right)+x\right)+x^2+1\right)+x\left(x^2+1\right)+x\right)+x\left(x\left(x^2+1\right)+x\right)+x^2+1\right)$$
$$+x\left(x\left(x\left(x^2+1\right)+x\right)+x^2+1\right)+x\left(x^2+1\right)+x$$

$$x\left(x\left(x\left(x\left(x\left(x^2+1\right)+x\right)+x^2+1\right)+x\left(x^2+1\right)+x\right)+x\left(x\left(x^2+1\right)+x\right)+x^2\right. \quad \textbf{(4.3)}$$
$$\left.+1\right)+x\left(x\left(x\left(x^2+1\right)+x\right)+x^2+1\right)+x\left(x^2+1\right)+x\right)+x\left(x\left(x\left(x^2+1\right)\right.\right.$$
$$\left.\left.+x\right)+x^2+1\right)+x\left(x^2+1\right)+x\right)+x\left(x\left(x^2+1\right)+x\right)+x^2+1$$

Second try: use the **expand** command

```
> fp := proc (n::integer, x, $)
  option remember;
      if n <= 1 then
          return n;
      else
          return expand(x*fp(n-1,x)+fp(n-2,x));
      end if;
  end proc:
> for n from 0 to 9 do
      print(fp(n,x));
  end do
```

$$0$$

$$1$$

$$x$$

$$x^2+1$$

$$x^3+2x$$

$$x^4+3x^2+1$$

$$x^5+4x^3+3x$$

$$x^6+5x^4+6x^2+1$$

$$x^7+6x^5+10x^3+4x$$

$$x^8+7x^6+15x^4+10x^2+1 \quad \textbf{(4.4)}$$

What is the factored form of the first 20 Fibonacci polynomials?

```
> for n from 0 to 19 do
      print(factor(fp(n,x)));
  end do
```

$$0$$
$$1$$
$$x$$
$$x^2+1$$
$$x\left(x^2+2\right)$$
$$x^4+3x^2+1$$
$$x\left(x^2+3\right)\left(x^2+1\right)$$
$$x^6+5x^4+6x^2+1$$
$$x\left(x^2+2\right)\left(x^4+4x^2+2\right)$$
$$\left(x^2+1\right)\left(x^6+6x^4+9x^2+1\right)$$
$$x\left(x^4+3x^2+1\right)\left(x^4+5x^2+5\right)$$
$$x^{10}+9x^8+28x^6+35x^4+15x^2+1$$
$$x\left(x^2+3\right)\left(x^2+2\right)\left(x^2+1\right)\left(x^4+4x^2+1\right)$$
$$x^{12}+11x^{10}+45x^8+84x^6+70x^4+21x^2+1$$
$$x\left(x^6+5x^4+6x^2+1\right)\left(x^6+7x^4+14x^2+7\right)$$
$$\left(x^2+1\right)\left(x^4+3x^2+1\right)\left(x^8+9x^6+26x^4+24x^2+1\right)$$
$$x\left(x^2+2\right)\left(x^4+4x^2+2\right)\left(x^8+8x^6+20x^4+16x^2+2\right)$$
$$x^{16}+15x^{14}+91x^{12}+286x^{10}+495x^8+462x^6+210x^4+36x^2+1$$
$$x\left(x^2+3\right)\left(x^2+1\right)\left(x^6+6x^4+9x^2+1\right)\left(x^6+6x^4+9x^2+3\right)$$
$$x^{18}+17x^{16}+120x^{14}+455x^{12}+1001x^{10}+1287x^8+924x^6+330x^4+45x^2+1 \quad \textbf{(4.5)}$$

Observed properties:
- except for x there are no terms of odd degree in the factorization
- all coefficients are positive
- 0 is the only zero
- $fp(n,x)$ is of degree $n-1$
- $fp(n,x)$ is irreducible whenever n is prime, and it is a factor of $fp(k\cdot n,x)$ for all integers k

$fp(n,x)$ evaluated at $x=1$ gives the nth Fibonacci number:

```
> seq(fp(n,1),n=0..11)
```

$$0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89 \quad \textbf{(4.6)}$$

Both Fibonacci numbers and polynomials are already available in Maple's **combinat** package:

```
> with(combinat)
```

[Chi, *bell*, binomial, *cartprod*, *character*, *choose*, *composition*, *conjpart*, *decodepart*, *encodepart*, *eulerian1*, *eulerian2*, *fibonacci*, *firstcomb*, *firstpart*, *firstperm*, *graycode*, *inttovec*, *lastcomb*, *lastpart*, *lastperm*, *multinomial*, *nextcomb*, *nextpart*, *nextperm*, *numbcomb*, *numbcomp*, *numbpart*, *numbperm*, *partition*, *permute*, *powerset*, *prevcomb*, *prevpart*, *prevperm*, *randcomb*, *randpart*, *randperm*, *rankcomb*, *rankperm*, *setpartition*, stirling1, stirling2, *subsets*, *unrankcomb*, *unrankperm*, *vectoint*] **(4.7)**

```
> seq(fibonacci(n), n=0..11)
```

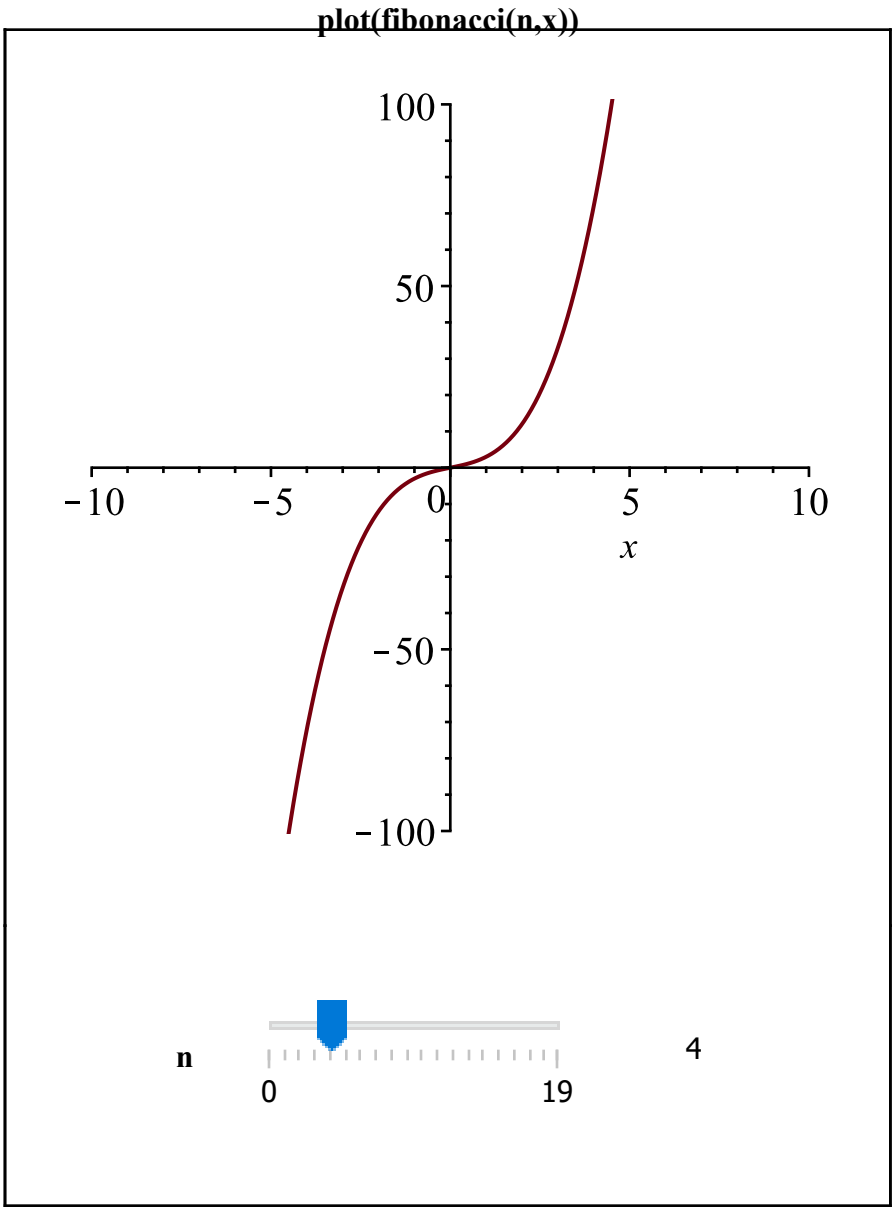
$$0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89 \quad \textbf{(4.8)}$$

```
> for n from 0 to 9 do
      print(fibonacci(n,x));
  end do:
  unassign('n'):
```

$$0$$
$$1$$
$$x$$
$$x^2+1$$
$$x^3+2\,x$$
$$x^4+3\,x^2+1$$
$$x^5+4\,x^3+3\,x$$
$$x^6+5\,x^4+6\,x^2+1$$
$$x^7+6\,x^5+10\,x^3+4\,x$$
$$x^8+7\,x^6+15\,x^4+10\,x^2+1 \qquad \textbf{(4.9)}$$

Explore and compare Fibonacci polynomials for various parameter values

```
> Explore(plot(fibonacci(n,x),x=-10..10,view=[-10..10,-100..100]),
          n=0..19,title="plot(fibonacci(n,x))")
```




```
> Explore(factor(fibonacci(n,x)), n=0..19, width=500, size=[800,
  50],
          echoexpression=false, tableborders=false,
          title="factor(fibonacci(n,x))", aligntowardcontrols=
  false,
          initialvalues=[i=10], fillcolor=Cyan)
```

factor(fibonacci(n,x))

$$x\left(x^6+5x^4+6x^2+1\right)\left(x^6+7x^4+14x^2+7\right)$$

n 14

0 19