알고리즘 Assign 02

과제 제출일자 : 2019/09/23
학번 : 201502085, 이규봉
분반 : 05
과제 : 피보나치 수열의 성능 비교, powering a number 구현 및 연습문제 풀이

Test Environment

과제의 코드를 테스트한 환경은 아래와 같습니다. 따로 사용한 라이브러리는 없습니다.

- Pycharm 2019.02
- Python 3.7.4

피보나치 수열 성능 비교

1번 Recursion의 수행 시간이 심하게 오래 걸려 테스트 하기 위한 값을 40으로 낮춰 수행 시간을 기록하고 비교 했습니다. 아래 세 표는 세 방법의 수행 시간을 비교한 것입니다. 각 테스트는 main.py에서 반복문을 돌며 한 번 함수의 실행이 끝날 때 마다, 함수의 수행 시간을 기록하는 식으로 작성했습니다. main.py는 첫 번째 입력 값에서 방법으로 1, 2, 3 중 하나의 입력을 받고 두 번째 입력 값으로 피보나치 값을 몇 번째 까지 구할 것인지를 입력 받습니다.

함수의 수행 시간 측정에는 timeit를 import해 사용했습니다.

성능 비교 결과로 알 수 있는 것은, 아래와 같습니다.

1 - 첫 번째 피보나치 함수 (재귀)는 n이 커짐에 따라 기하급수적으로 증가하기 때문에, 이후에는 n을 구하는 시간 역시 피보나치 수열을 이루며 증가하여, 정상적으로 함수를 이용할 수 없을 정도로 오래 걸리게 된다는 것입니다.

2 - 두 개의 int 캐시 값을 이용하여 다음 피보나치 값을 구하는 두 번째 함수는 각 항목들을 단 한 번만 순회하기 때문에 O(n) 으로 작동하며, 첫 번째 함수에 비해 매우 빠릅니다.

3 - 세 번째 피보나치 함수는 행렬 A를 [[1, 1], [1, 0]] 로 정의하여 행렬 A의 n 제곱을 POW 함수를 이용하여 lgn 번만에 n 번째 피보나치 함수 값을 구할 수 있습니다. 그러나 POW를 DP 없이 정의하면 두 번째 피보나치 함수보다 훨씬 느린 결과를 내는 것을 알 수 있습니다.

이것은 세 번째 피보나치 함수가 n 번째 함수 값을 구하는 과정에서 이미 구한 POW 값을 반복해서 구하기 때문입니다. 아래 표를 보면 POW에 메모리제이션을 적용했을 때 훨씬 빠른 결과를 내는 것을 알 수 있습니다.

컴퓨터의 상태에 따라 실행할 때 마다 결과가 달라지긴 하지만, 아래의 표에서 이 수치는 두 번째 피보나치 함수보다 빠르게 작동합니다. 결과적으로, 함수의 실행 성능을 정리하면, 1 << 2 <= 3입니다.

1 – Recursion

Recursion		
f<0>=	0	0.0000634000
f<1>=	1	0.0001073000
f<2>=	1	0.0001228000
f<3>=	2	0.0001360000
f<4>=	3	0.0001521000
f<5>=	5	0.0001675000
f<6>=	8	0.0001849000
f<7>=	13	0.0002060000
f<8>=	21	0.0002324000
f<9>=	34	0.0002700000
f<10>=	55	0.0003287000
f<11>=	89	0.0004045000
f<12>=	144	0.0005189000
f<13>=	233	0.0006974000
f<14>=	377	0.0009823000
f<15>=	610	0.0014324000
f<16>=	987	0.0021459000
f<17>=	1597	0.0033130000
f<18>=	2584	0.0051776000
f<19>=	4181	0.0081698000
f<20>=	6765	0.0130124000
f<21>=	10946	0.0209129000
f<22>=	17711	0.0339745000
f<23>=	28657	0.0545698000
f<24>=	46368	0.0878534000
f<25>=	75025	0.1452187000
f<26>=	121393	0.2354954000
f<27>=	196418	0.3746956000
f<28>=	317811	0.6303978000
f<29>=	514229	1.0649098000
f<30>=	832040	1.7508482000
f<31>=	1346269	2.7597031000
f<32>=	2178309	4.4141557000
f<33>=	3524578	7.1122976000
f<34>=	5702887	11.2718737000
f<35>=	9227465	18.3502806000
f<36>=	4930352	29.4146165000
f<37>=	24157817	48.0767735000
f<38>=	39088169	78.8355435000
f<39>=	63245986	129.1767117000
f<40>=	102334155	208.7091947000

2 – Array

Array		
f<0>=	0	0.0000648000
f<1>=	1	0.0001129000
f<2>=	1	0.0001273000
f<3>=	2	0.0001402000
f<4>=	3	0.0001528000
f<5>=	5	0.0001649000
f<6>=	8	0.0001781000
f<7>=	13	0.0001908000
f<8>=	21	0.0002033000
f<9>=	34	0.0002164000
f<10>=	55	0.0002374000
f<11>=	89	0.0002504000
f<12>=	144	0.0002634000
f<13>=	233	0.0002762000
f<14>=	377	0.0002890000
f<15>=	610	0.0003022000
f<16>=	987	0.0003159000
f<17>=	1597	0.0003292000
f<18>=	2584	0.0003425000
f<19>=	4181	0.0003562000
f<20>=	6765	0.0003779000
f<21>=	10946	0.0003913000
f<22>=	17711	0.0004049000
f<23>=	28657	0.0004184000
f<24>=	46368	0.0004327000
f<25>=	75025	0.0004465000
f<26>=	121393	0.0004609000
f<27>=	196418	0.0004753000
f<28>=	317811	0.0004904000
f<29>=	514229	0.0005055000
f<30>=	832040	0.0005285000
f<31>=	1346269	0.0005428000
f<32>=	2178309	0.0005579000
f<33>=	3524578	0.0005723000
f<34>=	5702887	0.0005868000
f<35>=	9227465	0.0006015000
f<36>=	14930352	0.0006162000
f<37>=	24157817	0.0006310000
f<38>=	39088169	0.0006458000
f<39>=	63245986	0.0006607000
f<40>=	102334155	0.0006837000

3 – Recursive Squaring (Not DP)

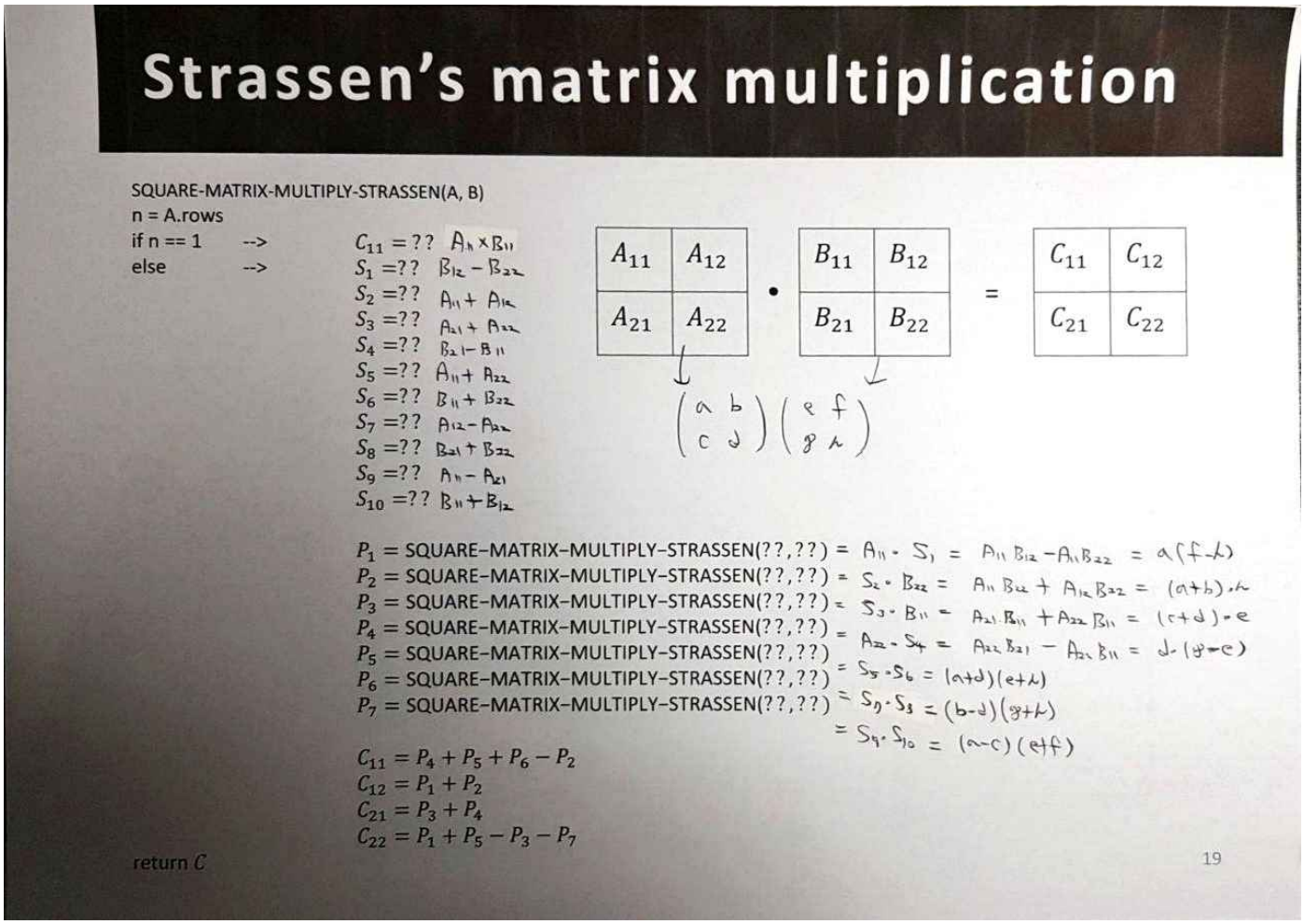
Recursive Squaring		
f<0>=	0	0.0000407000
f<1>=	1	0.0000653000
f<2>=	1	0.0000838000
f<3>=	2	0.0000998000
f<4>=	3	0.0001240000
f<5>=	5	0.0001471000
f<6>=	8	0.0001744000
f<7>=	13	0.0002046000
f<8>=	21	0.0002399000
f<9>=	34	0.0002788000
f<10>=	55	0.0003260000
f<11>=	89	0.0003724000
f<12>=	144	0.0004227000
f<13>=	233	0.0004769000
f<14>=	377	0.0005377000
f<15>=	610	0.0006002000
f<16>=	987	0.0006689000
f<17>=	1597	0.0007407000
f<18>=	2584	0.0008160000
f<19>=	4181	0.0008956000
f<20>=	6765	0.0009829000
f<21>=	10946	0.0010697000
f<22>=	17711	0.0011602000
f<23>=	28657	0.0012546000
f<24>=	46368	0.0013532000
f<25>=	75025	0.0014569000
f<26>=	121393	0.0015756000
f<27>=	196418	0.0016866000
f<28>=	317811	0.0018013000
f<29>=	514229	0.0019193000
f<30>=	832040	0.0020452000
f<31>=	1346269	0.0021725000
f<32>=	2178309	0.0023200000
f<33>=	3524578	0.0024667000
f<34>=	5702887	0.0026122000
f<35>=	9227465	0.0027587000
f<36>=	14930352	0.0029080000
f<37>=	24157817	0.0030624000
f<38>=	39088169	0.0032210000
f<39>=	63245986	0.0033838000
f<40>=	102334155	0.0035540000

4 – Recursive Squaring (Apply DP)

Recursive Squaring		
f<0>=	0	0.0000395000
f<1>=	1	0.0000656000
f<2>=	1	0.0000732000
f<3>=	2	0.0000797000
f<4>=	3	0.0000860000
f<5>=	5	0.0000922000
f<6>=	8	0.0000986000
f<7>=	13	0.0001051000
f<8>=	21	0.0001117000
f<9>=	34	0.0001181000
f<10>=	55	0.0001289000
f<11>=	89	0.0001357000
f<12>=	144	0.0001436000
f<13>=	233	0.0001506000
f<14>=	377	0.0001574000
f<15>=	610	0.0001643000
f<16>=	987	0.0001712000
f<17>=	1597	0.0001782000
f<18>=	2584	0.0001849000
f<19>=	4181	0.0001917000
f<20>=	6765	0.0002029000
f<21>=	10946	0.0002100000
f<22>=	17711	0.0002172000
f<23>=	28657	0.0002244000
f<24>=	46368	0.0002316000
f<25>=	75025	0.0002388000
f<26>=	121393	0.0002537000
f<27>=	196418	0.0002627000
f<28>=	317811	0.0002700000
f<29>=	514229	0.0002774000
f<30>=	832040	0.0002890000
f<31>=	1346269	0.0002966000
f<32>=	2178309	0.0003040000
f<33>=	3524578	0.0003114000
f<34>=	5702887	0.0003189000
f<35>=	9227465	0.0003268000
f<36>=	14930352	0.0003344000
f<37>=	24157817	0.0003421000
f<38>=	39088169	0.0003497000
f<39>=	63245986	0.0003574000
f<40>=	102334155	0.0003696000

스트라센 행렬 곱셈 문제 풀이

PPT 19 page에 ??로 표시된 부분의 코드는 아래와 같습니다.
(PPT를 종이에 인쇄 후 빈칸을 채우고 사진을 찍었습니다.)



Strassen's matrix multiplication

SQUARE-MATRIX-MULTIPLY-STRASSEN(A, B)
n = A.rows
if n == 1 --> $C_{11} = ??$ $A_{11} \times B_{11}$
else --> $S_1 = ??$ $B_{12} - B_{22}$
$S_2 = ??$ $A_{11} + A_{12}$
$S_3 = ??$ $A_{21} + A_{22}$
$S_4 = ??$ $B_{21} - B_{11}$
$S_5 = ??$ $A_{11} + A_{22}$
$S_6 = ??$ $B_{11} + B_{22}$
$S_7 = ??$ $A_{12} - A_{22}$
$S_8 = ??$ $B_{21} + B_{22}$
$S_9 = ??$ $A_{11} - A_{21}$
$S_{10} = ??$ $B_{11} + B_{12}$

A_{11}	A_{12}
A_{21}	A_{22}

•

B_{11}	B_{12}
B_{21}	B_{22}

=

C_{11}	C_{12}
C_{21}	C_{22}

$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} e & f \\ g & h \end{pmatrix}$

P_1 = SQUARE-MATRIX-MULTIPLY-STRASSEN(??,??) = $A_{11} \cdot S_1 = A_{11}B_{12} - A_{11}B_{22} = a(f-h)$
P_2 = SQUARE-MATRIX-MULTIPLY-STRASSEN(??,??) = $S_2 \cdot B_{22} = A_{11}B_{12} + A_{12}B_{22} = (a+b)\cdot h$
P_3 = SQUARE-MATRIX-MULTIPLY-STRASSEN(??,??) = $S_3 \cdot B_{11} = A_{21}B_{11} + A_{22}B_{11} = (c+d)\cdot e$
P_4 = SQUARE-MATRIX-MULTIPLY-STRASSEN(??,??) = $A_{22} \cdot S_4 = A_{22}B_{21} - A_{22}B_{11} = d\cdot(g-e)$
P_5 = SQUARE-MATRIX-MULTIPLY-STRASSEN(??,??) = $S_5 \cdot S_6 = (a+d)(e+h)$
P_6 = SQUARE-MATRIX-MULTIPLY-STRASSEN(??,??) = $S_7 \cdot S_8 = (b-d)(g+h)$
P_7 = SQUARE-MATRIX-MULTIPLY-STRASSEN(??,??) = $S_9 \cdot S_{10} = (a-c)(e+f)$

$C_{11} = P_4 + P_5 + P_6 - P_2$
$C_{12} = P_1 + P_2$
$C_{21} = P_3 + P_4$
$C_{22} = P_1 + P_5 - P_3 - P_7$

return C

19