Algoritmo 1

F1	
N	OP
0	0
100	197
200	397
300	597
400	797
500	997
600	1197
700	1397
800	1597
900	1797
1000	1997

Expoente
1,005911839

Fórmula
0
102,7599011
206,3637038
310,2884401
414,4221412
518,7115037
623,1250815
727,6420701
832,2476677
936,9307974
1041,682855

Foi um código de execução rápida, seu gráfico é de complexidade n, linear. Observa-se que, à medida que o tamanho da entrada dobra, o tempo de execução também cresce de forma proporcional. Esse comportamento é típico de algoritmos que percorrem todos os elementos de forma sequencial, sem repetições aninhadas. Por isso, o Algoritmo 1 se mostrou bastante eficiente, mantendo tempos de execução baixos mesmo para entradas maiores.

Algoritmo 2

F2	
N	OP
0	0
100	247500
200	1990
300	6727500
400	15960000
500	31187500
600	53910000
700	85627500
800	127840000
900	182047500
1000	249750000

Expoente
3,003930294

Fórmula
0
1018264,463
8168338,193
27612108,93
65524970,45
128090746,8
221499475,1
351946127,3
525629773,3
748752988,5
1027521412

Foi um código de execução rápida, seu gráfico é de complexidade n ao cubo. O tempo de execução aumentou de forma muito acentuada à medida que o valor de n cresceu. Embora o tempo inicial tenha sido relativamente rápido para entradas pequenas, o crescimento se mostrou insustentável em entradas maiores, tornando esse algoritmo pouco escalável.

Algoritmo 3

F3	
N	OP
0	1
100	4951
200	19901
300	44851
400	79801
500	124751
600	179701
700	244651
800	319601
900	404551
1000	499501

Área de Plotagem

Expoente
2,003843436

Fórmula
0
10178,57243
40822,89991
91994,77587
163727,1993
256043,2472
368960,7313
502494,1746
656655,8439
831456,3606
1026905,091

Foi um código de execução rápida, seu gráfico é de complexidade n ao quadrado. O gráfico evidencia que o tempo de execução cresce mais rapidamente do que no Algoritmo 1 (linear), mas ainda menos abruptamente que no Algoritmo 2 (cúbico). Apesar de mais custoso que a versão linear, o Algoritmo 3 ainda pode ser considerado viável para entradas de tamanho moderado.

Algoritmo 4

F4	
N	OP
0	0
100	405553791
200	-2080247576
300	-203690645
400	-470844127
500	-1329816081
600	-806682940
700	1466048028
800	-1636671090
900	1022618534
1000	1706379110

Expoente
0,62402706

Fórmula
0
17,70329558
27,28380795
35,13910491
42,04901696
48,33150486
54,1553738
59,62359609
64,80473072
69,74727886
74,48711966

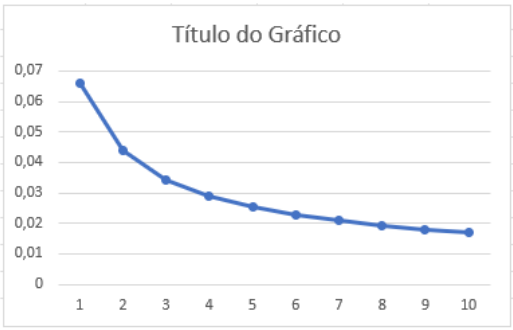
Foi um código de execução mais lenta comparado aos outros, que demorou em torno de uma hora para finalizar, seu gráfico é de complexidade log n. Esse comportamento cresce de forma muito mais lenta do que os algoritmos lineares, quadráticos ou cúbicos, o que o torna bastante eficiente em cenários de entrada muito grande. Assim, apesar do tempo inicial elevado, tende a superar os demais em escalabilidade.

Algoritmo 5

F5	
N	OP
0	0
100	1899035032
200	255065140
300	-1987830736
400	2051388164
500	-1109181112
600	-303447386
700	-1220062488
800	215126802
900	519000666
1000	-487529366

Expoente
-0,5905322

Fórmula
#DIV/0!
0,065907617
0,043769109
0,03444928
0,029066972
0,025478351
0,022877694
0,020887093
0,019303315
0,018006308
0,016920119



Foi um código de execução lenta comparado aos outros, foi o que mais demorou para ser realizado, da operação 900 para 1000 demorou mais de uma hora. Com as fórmulas dadas no material, não era possível calcular o expoente por não haver log de número negativo, a formula do expoente adaptada foi:

=(LOG(ABS(OP final))-LOG(ABS(OP inicial em 100)))/(LOG(ABS(1000))-LOG(ABS(100))).

O gráfico sugere um crescimento superior ao polinomial, possivelmente de ordem exponencial, o que explica sua inviabilidade prática para entradas maiores. Dessa forma, embora seja possível executá-lo em instâncias pequenas, o algoritmo se torna impraticável em escalas maiores.