2.3 Complexity of algorithms

2.3.1 Constant or O(1) algorithms

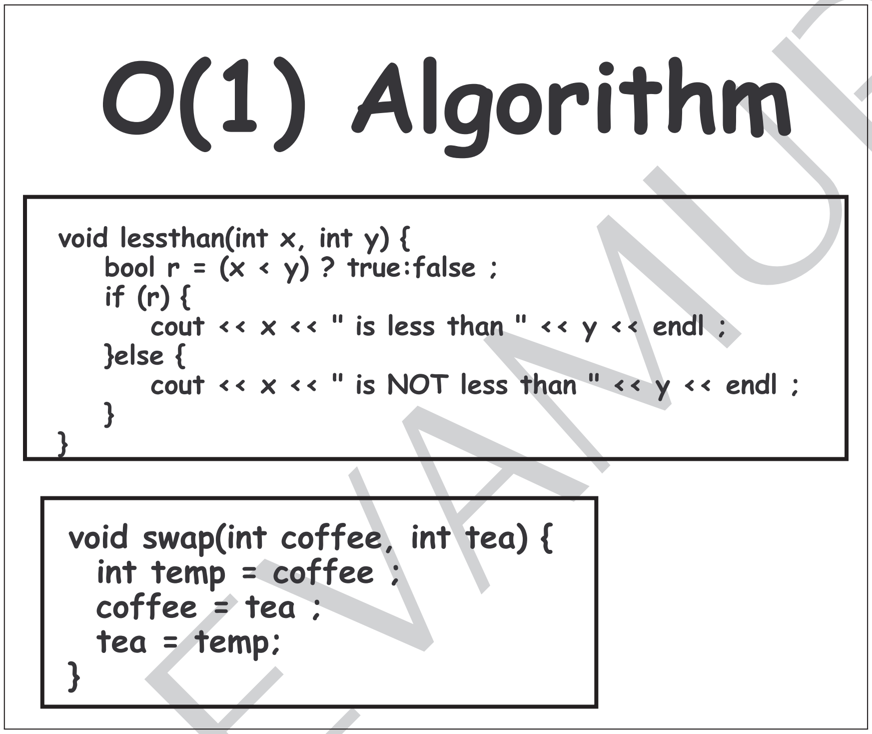


Figure 2.9: Constant algorithms

2.3.2 Logarithmic or O(log n) algorithms

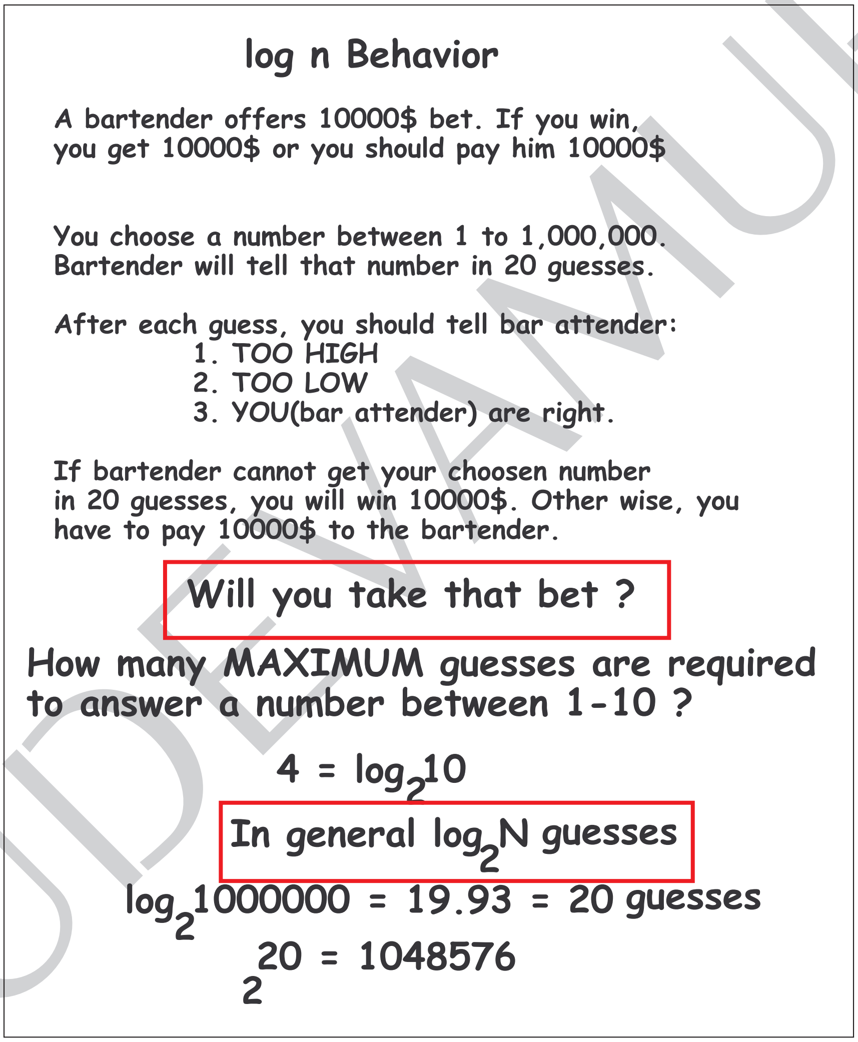


Figure 2.10: log n algorithm

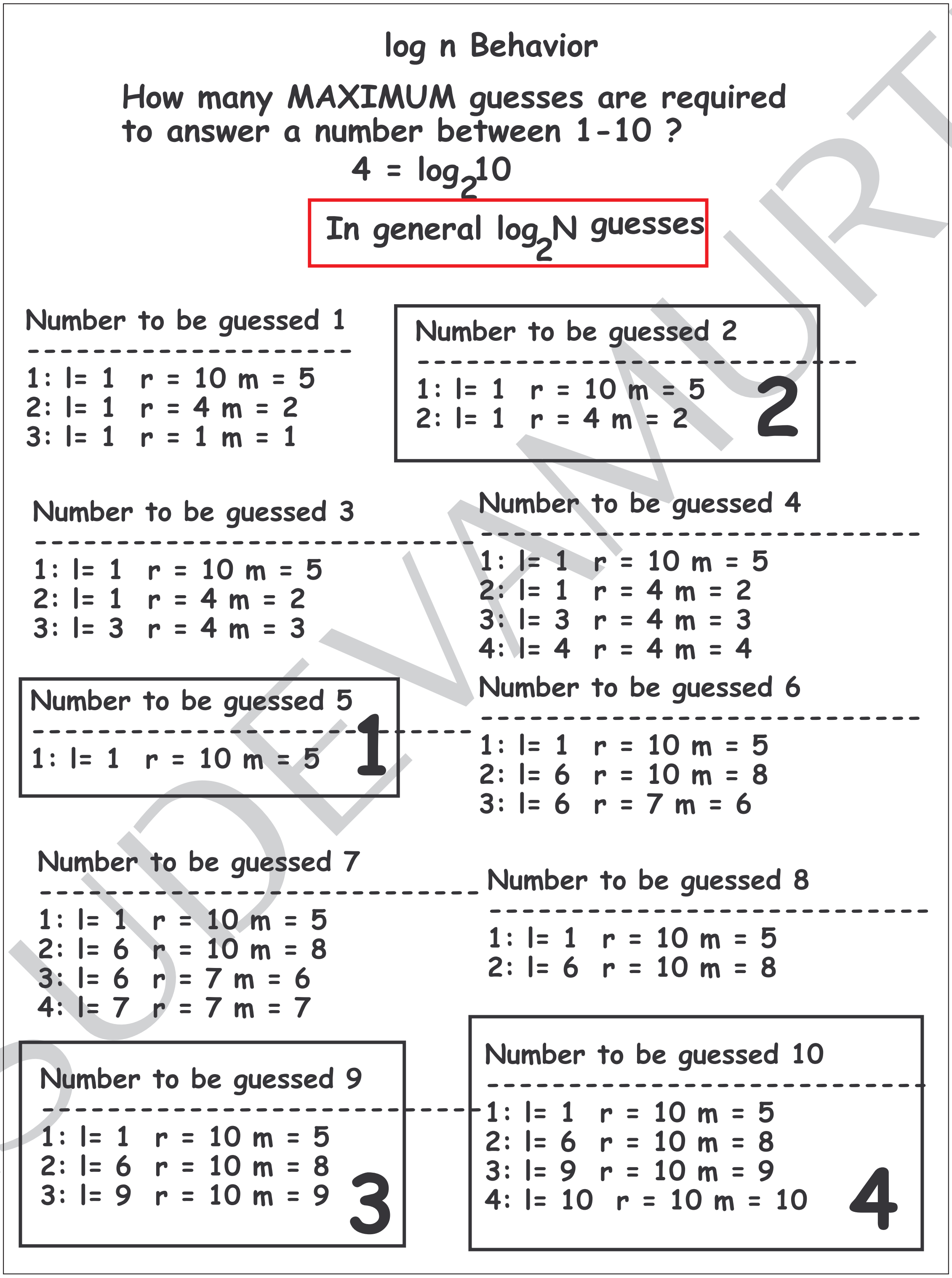


Figure 2.11: Guessing a number between 1 to 10

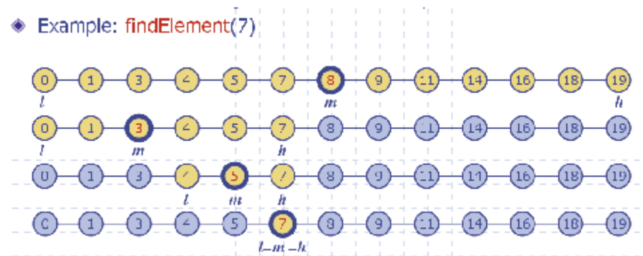
O(log n) Algorithm

```
int iter = 0 ;
for (int i = 1; i < 1000; i = i * 2) {
      iter++ ;
      some_func() ;
}
```

```
int iter = 0 ;
for (int j = 1000; j >=1 ; j = j / 2) {
      iter++ ;
      some_func() ;
}
```

iter	value of i	valueof j
1	1	1000
2	2	500
3	4	250
4	8	125
5	16	62
6	32	31
7	64	15
8	128	7
9	256	3
10	512	1
exit	1024	0

$$f(n) = \log_2 n$$



Binary search

Figure 2.12: log n algorithms

2.3.3 Linear O(n) algorithms

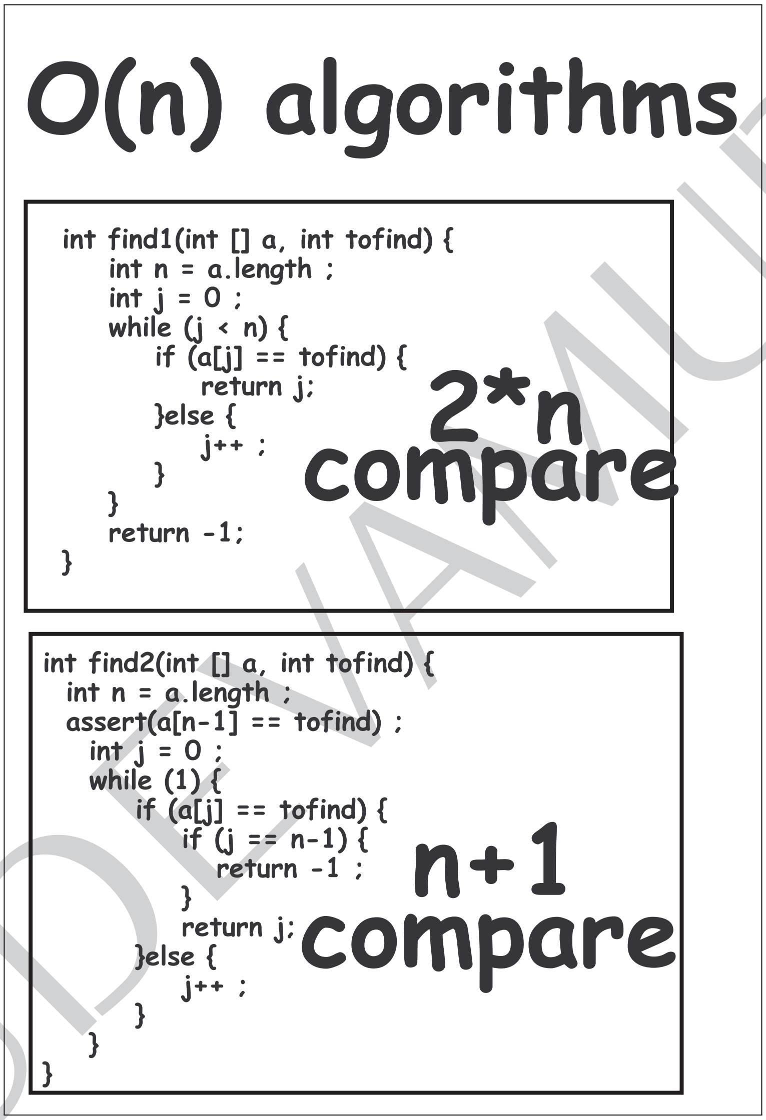


Figure 2.13: O(n) algorithms

2.3.4 O(n log n) algorithms

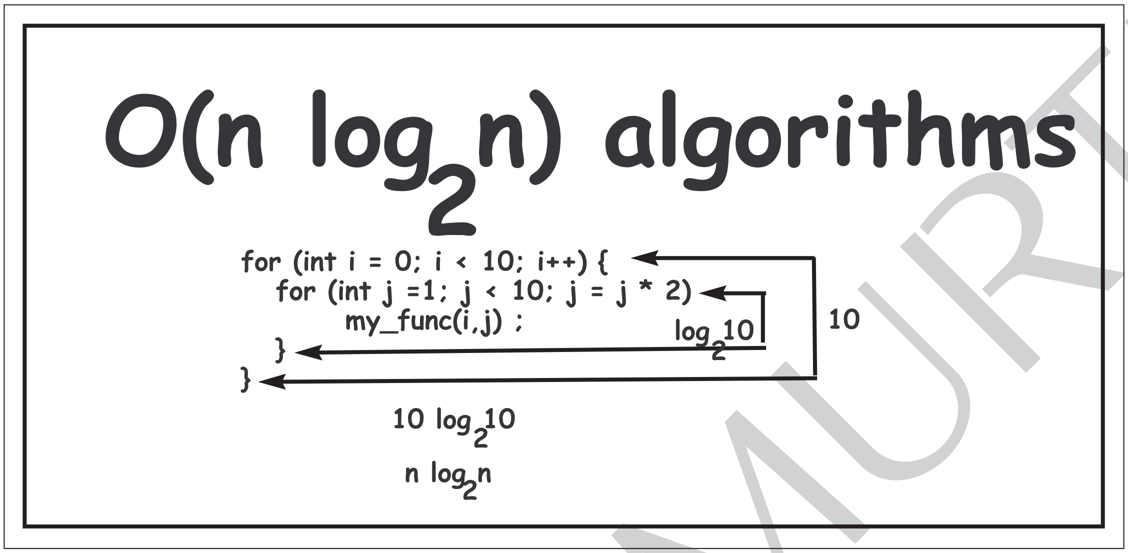


Figure 2.14: O(n log n) algorithms

2.3.5 Quadratic or O(n^2) algorithms

Quadratic algorithms
O(n^2 algorithm)

```
for( int i = 0; i < 10; i++) {
    for (int j = 0 ; j < 10; j++) {
        func(i,j) ;
    }
}
```

outerloop executed 10 times
each inner loop is executed 10 times

10 * 10 = 100

$f(n) = n * n = n^2$

```
for( int i = 0; i < 10; i++)
    for (int j = 0 ; j < i; j++) {
        func(i,j) ;
    }
}
```

outerloop executed 10 times
each inner loop is executed i times

i	outerloop executed	innerloop executed	total
0	1	0	1
1	1	1	2
2	1	2	3
3	1	3	4
4	1	4	5
5	1	5	6
6	1	6	7
7	1	7	8
8	1	8	9
9	1	9	10

k = 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10

$$f(n) = \frac{n(n-1)}{2} = g(n^2)$$

Figure 2.15: O(n^2) algorithms

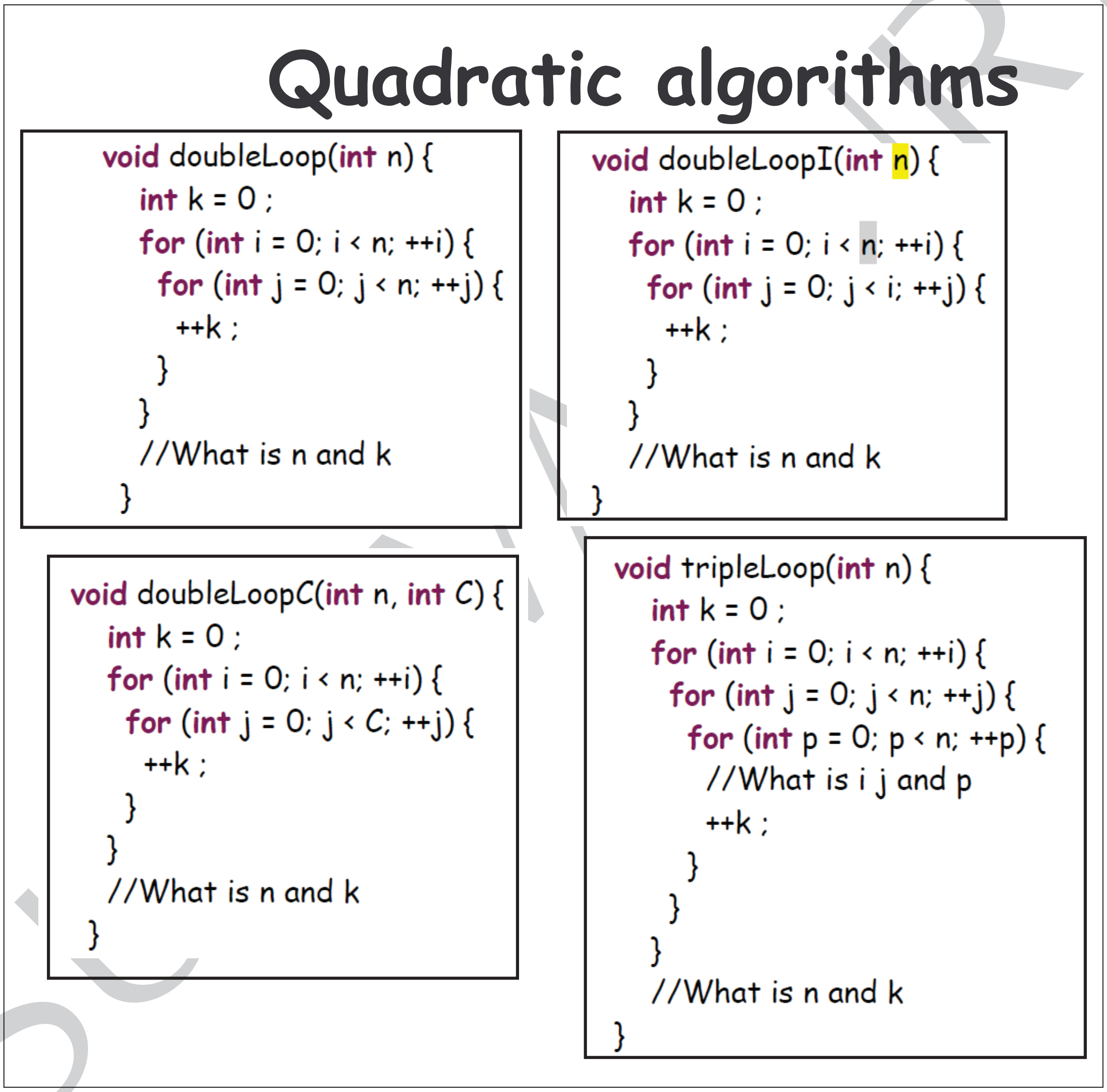


Figure 2.16: Quadratic algorithms

2.3.6 Exponential or O(2^n) algorithms

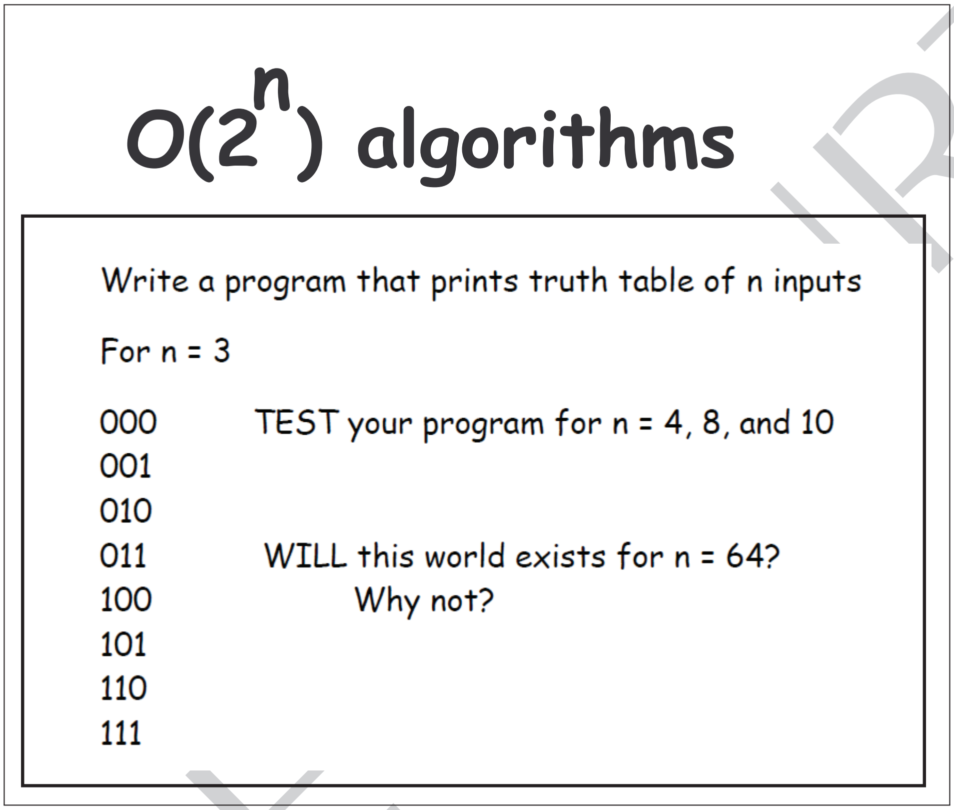


Figure 2.17: Printing a truth table

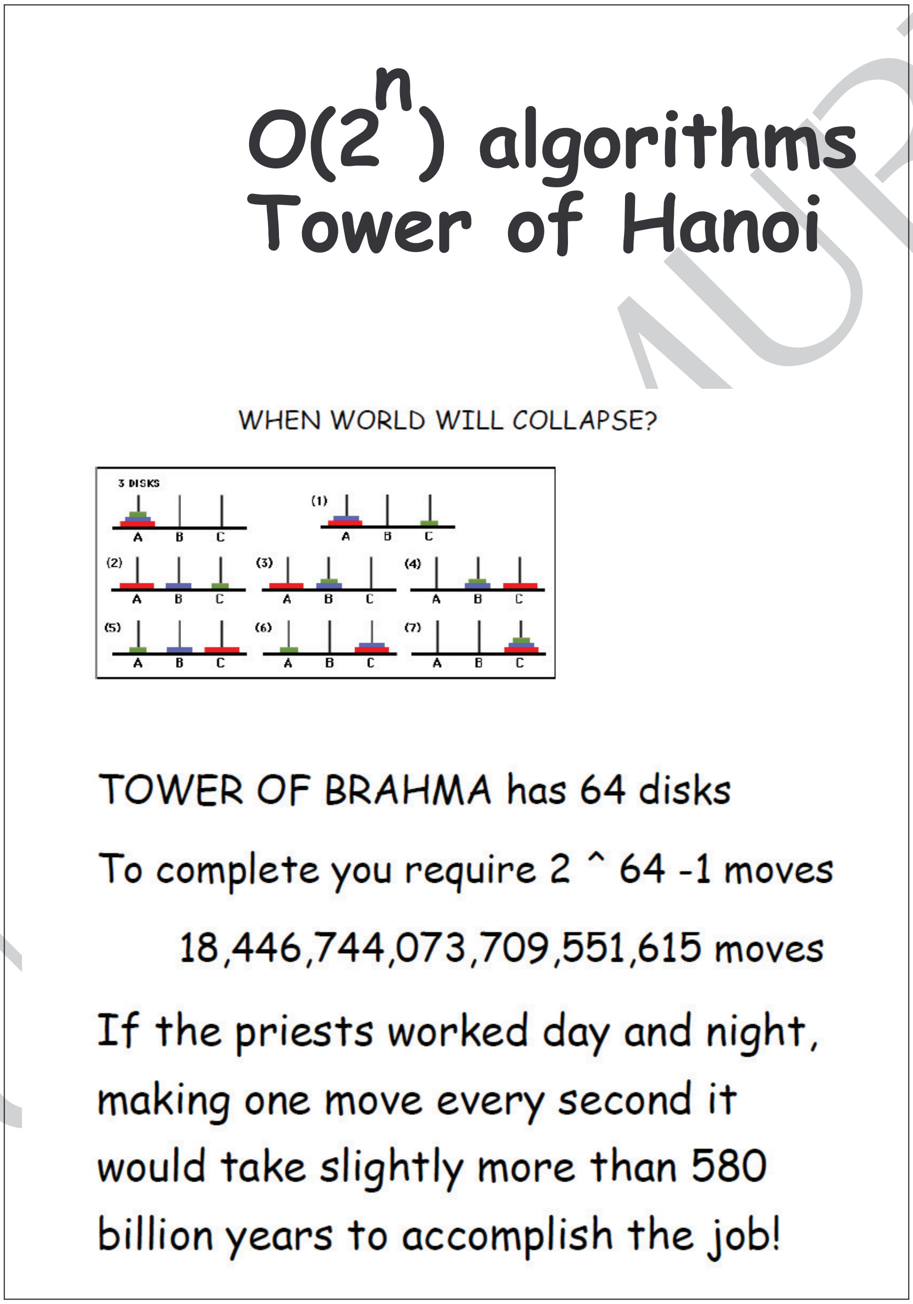


Figure 2.18: Tower of Hanoi

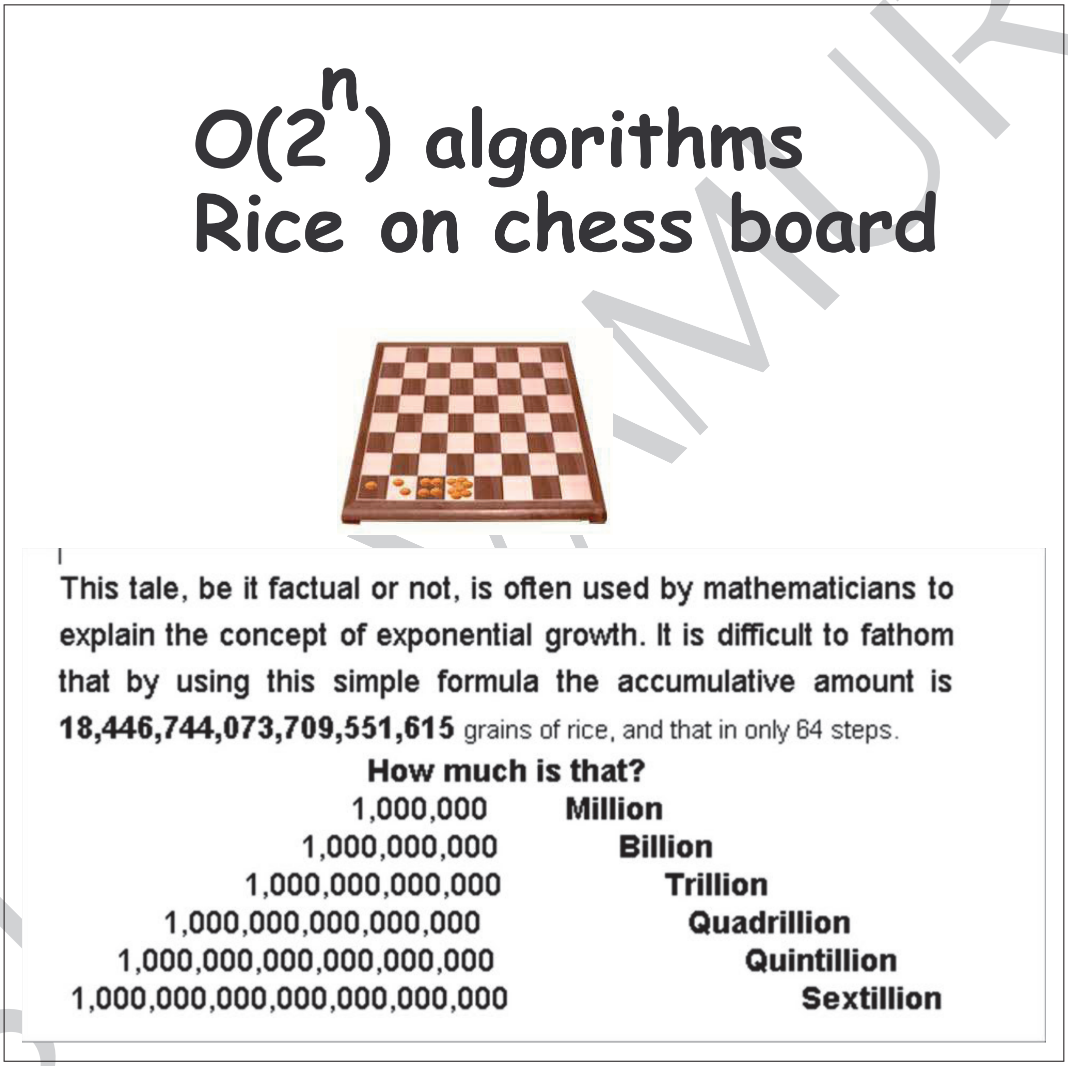
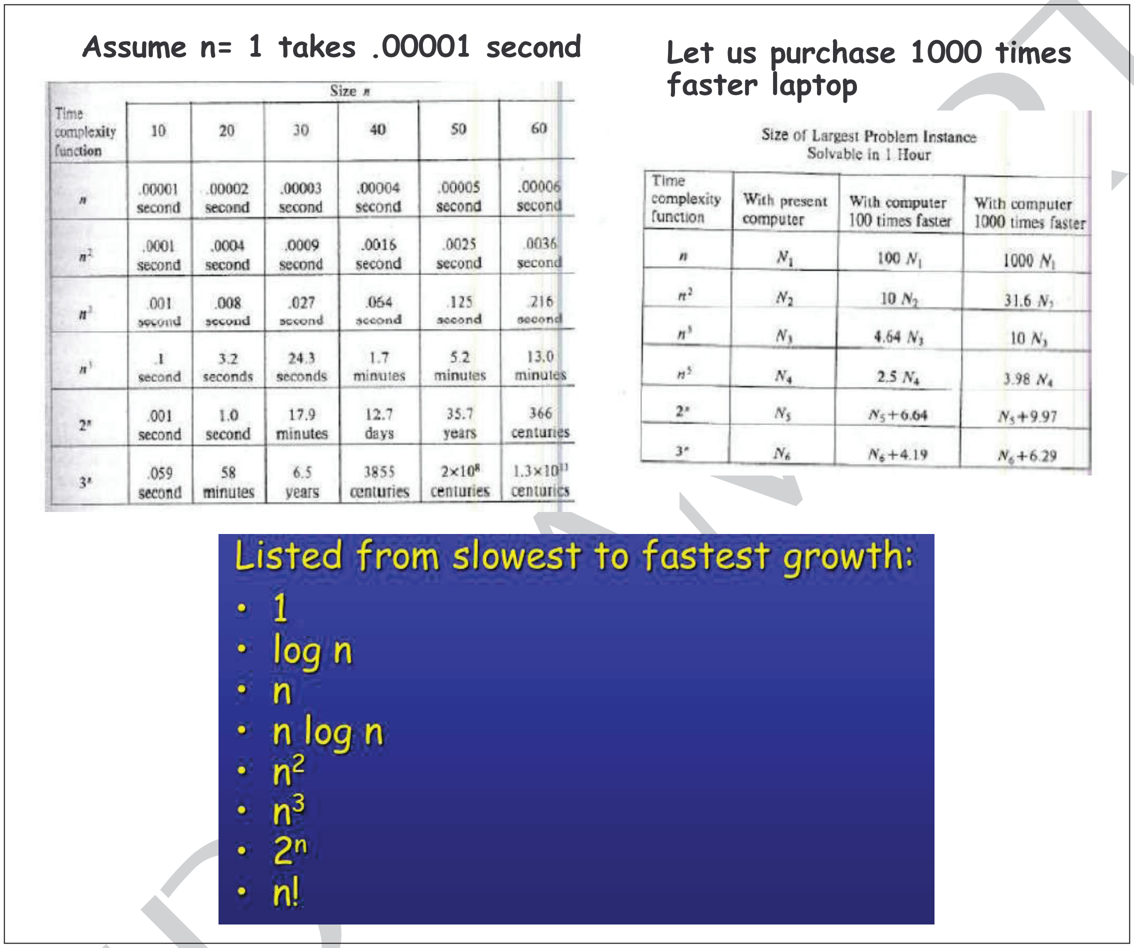


Figure 2.19: Rice on chess board

2.3.7 Execution time for algorithms with given time complexites



Time complexity function	Size n 10	20	30	40	50	60
n	.00001 second	.00002 second	.00003 second	.00004 second	.00005 second	.00006 second
n^2	.0001 second	.0004 second	.0009 second	.0016 second	.0025 second	.0036 second
n^3	.001 second	.008 second	.027 second	.064 second	.125 second	.216 second
n^5	.1 second	3.2 seconds	24.3 seconds	1.7 minutes	5.2 minutes	13.0 minutes
2^n	.001 second	1.0 second	17.9 minutes	12.7 days	35.7 years	366 centuries
3^n	.059 second	58 minutes	6.5 years	3855 centuries	2×10^8 centuries	1.3×10^{13} centuries

Size of Largest Problem Instance Solvable in 1 Hour

Time complexity function	With present computer	With computer 100 times faster	With computer 1000 times faster
n	N_1	$100\ N_1$	$1000\ N_1$
n^2	N_2	$10\ N_2$	$31.6\ N_2$
n^3	N_3	$4.64\ N_3$	$10\ N_3$
n^5	N_4	$2.5\ N_4$	$3.98\ N_4$
2^n	N_5	$N_5+6.64$	$N_5+9.97$
3^n	N_6	$N_6+4.19$	$N_6+6.29$

Figure 2.20: Execution time