AUG
Faculty of Computer and Information Sciences
Ain Shams University

Session 1
Genetic Distances
(Hamming Distance, Distance Matrix & DP)

Hamming Distance

Is to calculate the numbers of mismatched nucleotides between two strands.

GAGCCTACTAACGGGAT
CATCGTAATGACGGCCT

Figure 2. The Hamming distance between these two strings is 7. Mismatched symbols are colored red.

Why?

Two DNA strands taken from different organism or species genomes are homologous if they share a recent ancestor; thus, counting the number of bases at which homologous strands differ provides us with the minimum number of point mutations that could have occurred on the evolutionary path between the two strands , and similarity between two sequences.

Distance Matrix

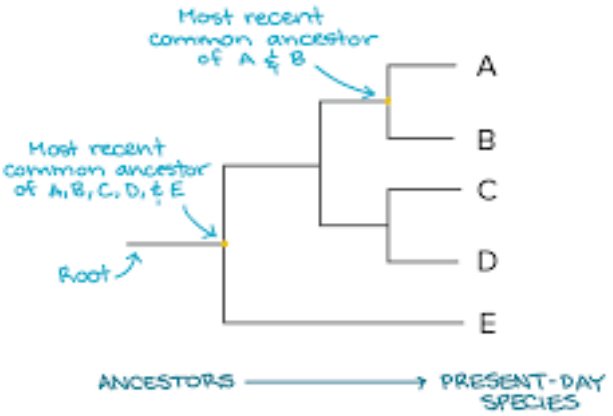
Show the sequence similarity between array of sequence.

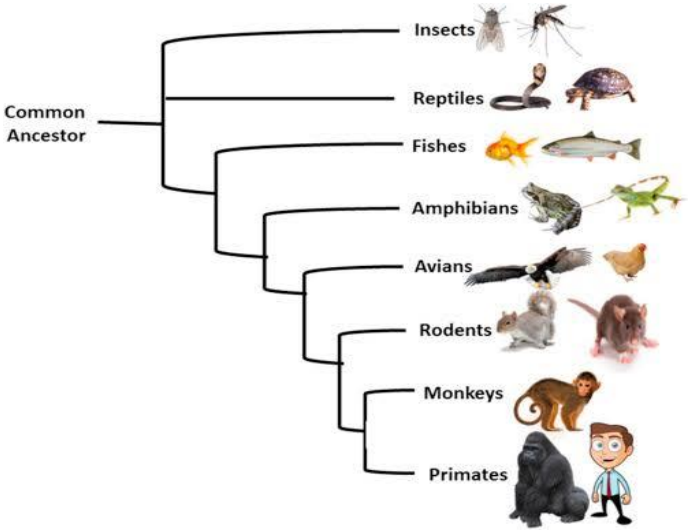
Distance Matrix

	A	B	C	D	E	F
A	0	16	47	72	77	79
B	16	0	37	57	65	66
C	47	37	0	40	30	35
D	72	57	40	0	31	23
E	77	65	30	31	0	10
F	79	66	35	23	10	0

Application

1 - Matrix visualization
2 - Phylogenetic trees (clustering the sequences)



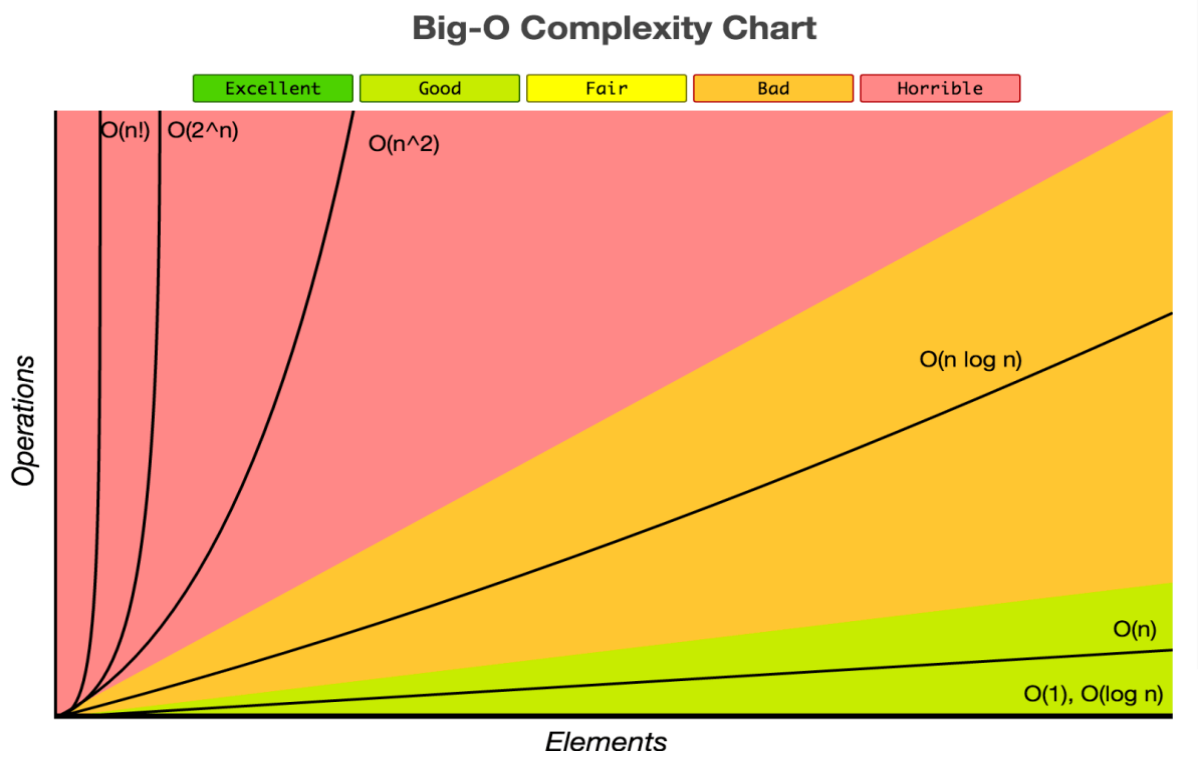


DP (Dynamic Programming)

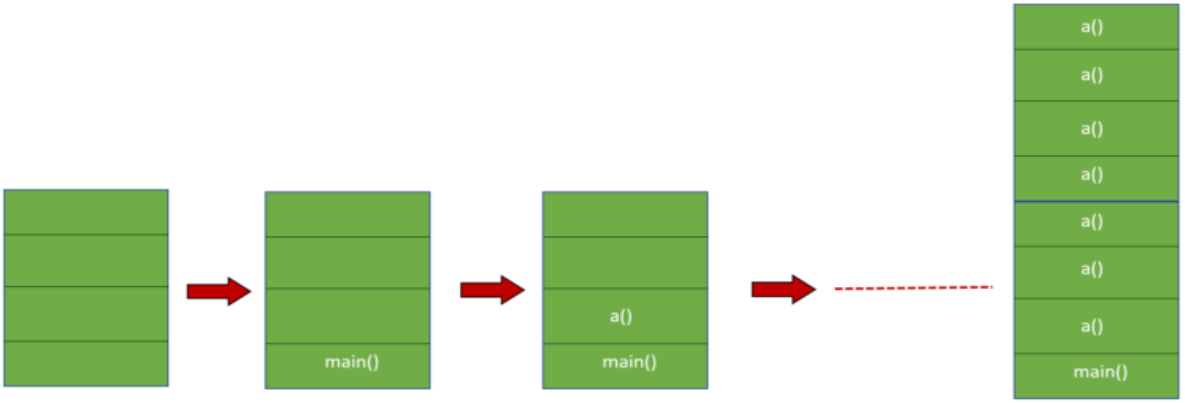
Dynamic Programming is an algorithmic technique for solving a problem by breaking it down into simpler subproblems where the optimal solution to the overall problem depends upon the optimal solution to it's individual subproblems. It's mainly an optimization for recursion.

The idea is to simply solve each subproblem just once and then store its answer, avoiding re-computation of the answer for the same subproblem every time.
This simple optimization reduces:

➔ Time complexity from exponential to polynomial.



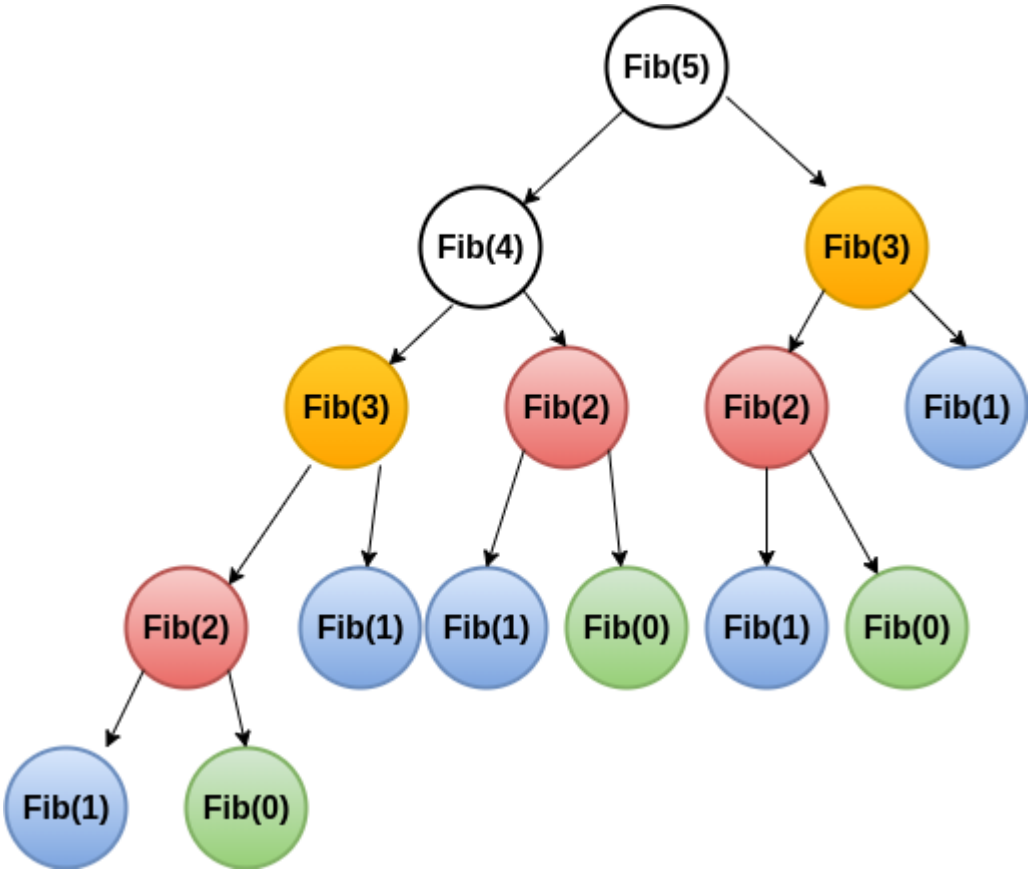
➔ Memory complexity to avoid stack overflow due to recursive function.



For example, if we write simple recursive solution for Fibonacci Numbers, we get exponential time complexity O(2^n) and if we optimize it by storing solutions of subproblems, time complexity reduces to linear O(n).

➔ Using recursion:

```
def recur_fib(n):
   if n <= 1:
       return n
   else:
       return recur_fib(n-1) + recur_fib(n-2)
```



➔ Using DP:

```
def dp_fib(n):
    f = [0, 1]
    for i in range(2, n+1):
        f.append(f[i-1] + f[i-2])
    return f[n]
```