Brute Force

The brute-force method explores all possible combinations exhaustively. It quickly becomes inefficient as the problem size increases.

- **Performance:**
  - Time Complexity: Exponential O(2n)O(2^n)O(2n).
  - Space Complexity: Constant O(1)O(1)O(1).

BF is effective for small inputs, this approach becomes computationally impractical for larger datasets.

Dynamic Programming

Dynamic Programming optimizes the problem by storing intermediate results to avoid redundant computations.

- **Performance:**
  - Time Complexity: O(n2)O(n^2)O(n2).
  - Space Complexity: O(n)O(n)O(n) (or O(n2)O(n^2)O(n2) depending on implementation).

DP is ideal for large inputs, as it significantly reduces computational overhead compared to brute force.

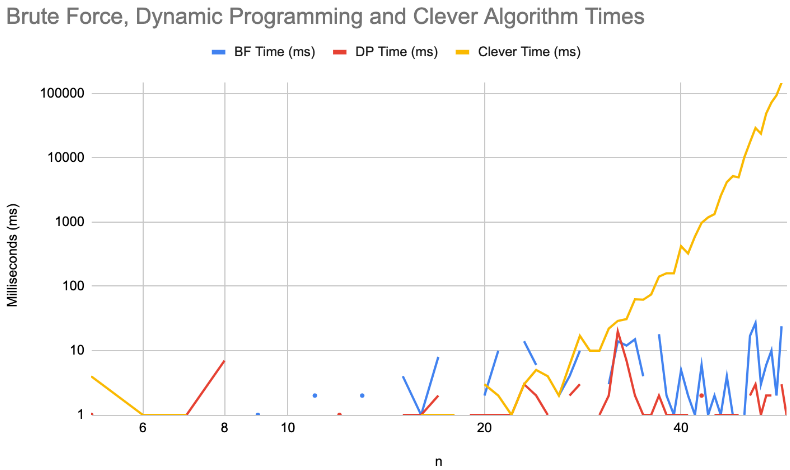
Clever Algorithm

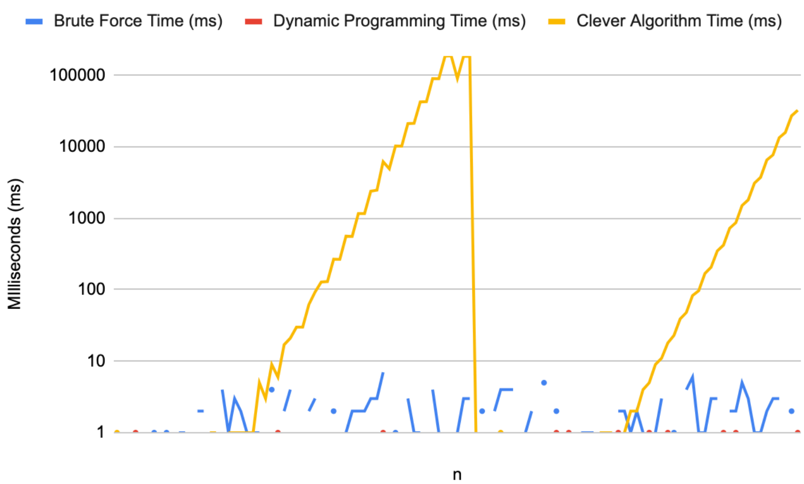
This algorithm isn’t a divide and conquer but a kind of form of direct computation.

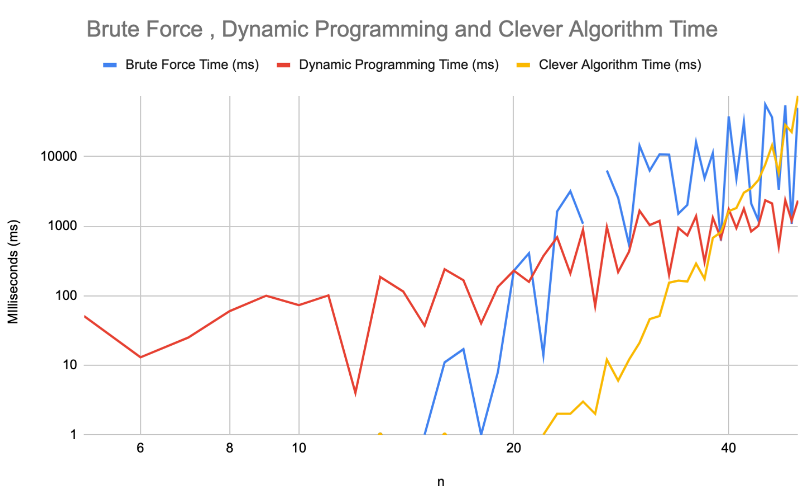
- **Performance:**
  - Time Complexity: O(n)O(n)O(n) or near-linear.
  - Space Complexity: O(1)O(1)O(1).

This is the most efficient method for large-scale inputs, as long as the problem's structure aligns with the algorithm.

Analysis


Brute Force, Dynamic Programming and Clever Algorithm Times
BF Time (ms)
DP Time (ms)
Clever Time (ms)
Milliseconds (ms)
100000
10000
1000
100
10
1
6
8
10
20
40
n


Brute Force Time (ms)
Dynamic Programming Time (ms)
Clever Algorithm Time (ms)
Milliseconds (ms)
100000
10000
1000
100
10
1
n


Brute Force , Dynamic Programming and Clever Algorithm Time
Brute Force Time (ms)
Dynamic Programming Time (ms)
Clever Algorithm Time (ms)
Milliseconds (ms)
10000
1000
100
10
1
6
8
10
20
40
n