1. **Problem Definition and example :** 0-1 Knapsack = বস্তা বা থলে
   Profit Maximization within a certain capacity. 0-1 : taken/not taken.

2. **Brute force :** Brute force way to solve this Knapsack problem.

   brute(n) = max{ value[n] + brute(n-1) }

	1	2	3	4	
Value :	200	250	100	80	Capacity : 13
Weight :	3	11	5	4	

   i. States : knapsack(4, 13) (weight, value)
   /(Yes) with 4th \(No)
   knapsack(3, 9) knapsack(3, 13)

   ii. Recurrence Case will be :

   knapsack(n, cap) = max{ val[n] + knapsack(n-1, cap-w[n])
   0 + knapsack(n-1, cap)
   }

   iii. Base Case :
   Knapsack(0, cap) = 0;

   Time Complexity : $O(2^n)$

3. **DP formulation :**
   Knapsack(n, cap) = $O(2^n)$ → O(n * cap), if we use DP formulation.
   Space Complexity = dp[n][cap] = O(n * cap)

4. **Coding with Memoization : Implementation**
   atcoder Task-D : Knapsack-1 : *https://atcoder.jp/contests/dp/tasks/dp_d*
5. **Coding with Tabulation : Same Task from 4.**