When do we use dynamic programming?

* When we want to optimize: maximize or minimize.

*Example)*

What is the maximum number of subsets, what is the shortest value of k for which there is a Hamiltonian path?

Find an optimal solution **we need**

1. An optimal solution for a sub problem
2. An optimal solution for overlapping problems.

How to attack dynamic programming problems?

1. What is the problem to be solved?
2. How would we solve it in a completely brute force attack and how expensive would it be?
3. Can we identify smaller sub problems and a way to optimize the whole problem?
4. Can we define an appropriate table structure to store already computed values?
5. Can we write pseudo-code to solve the problem?
6. What is the work involved?

EXAMPLE

For example, if *A* is a 10 × 30 matrix, *B* is a 30 × 5 matrix, and *C* is a 5 × 60 matrix, then

computing (*AB*)*C* needs (10×30×5) + (10×5×60) = 1500 + 3000 = 4500 operations, while

computing *A*(*BC*) needs (30×5×60) + (10×30×60) = 9000 + 18000 = 27000 operations.

A quicker solution to this problem can be achieved by breaking up the problem into a set of related subproblems. By solving subproblems once and reusing the solutions, the required run-time can drastically reduced. This concept is known as [dynamic programming](https://en.wikipedia.org/wiki/Dynamic_programming).