DP的**核心思想**类似于我们高中学习的数学归纳法：

1. 把一个大问题(size == n)的解决方案用比他小的问题（问题们）来解决，也就是思考从问题size = n-1 增加到 size = n 的时候, 如何用小问题的solution构建大问题的solution。
2. 与recursion的关系：
    - 2.1. Recursion 从大到小来解决问题，不记录任何sub-solution只要考虑
        - 2.1.1. base case
        - 2.1.2. recursive rule
    - 2.2. DP 从小到大来解决问题，记录sub-solution
        - 2.2.1. 由size (< n) 的 subsolution(**s**) → size (n) 的solution
        - 2.2.2. base case
        - 2.2.3. Induction rule

DP 的解题常用方法：

1. 一维的original data (such as a rope, a word, a piece of wood)， 求MAX or MIN (cut, merge, etc..)
   - 1.1. if the **weight** of each smallest element in the original data is identical/similar
     - 1.1.1. e.g. **identical**: 1 meter of rope
     - 1.1.2. e.g. **similar**: a letter, a number

   Then this kind of problem is usually simple:

   Linear scan and look back to the previous element(s)

   For example:

   Longest Ascending Subarray (when at i, look back at i-1)
   Longest Ascending Subsequence (when at i, look back at 1….i-1)
   Cut rope
   Cut palindrome

   - 1.2. If the **weight** are not the same:
     - 1.2.1. e.g. DP1 课后题： 沙子归并
     - 1.2.2. e.g. 强化练习题： 切木头

   从中心开花， [index = 0.1.2.3. N-1], for each M[i, j], we usually need to try out all possible k that (i<k<j), M[i, j] = max (M[i, k] +/-/* M[k, j]) (for all possible k)

....**Method1: Recursion**

```
00 public int getMaxProduct(int n) {
01    if (n <= 1) {
02          return 0;    // base case;
03    }
04    int maxProduct = 0;
05    for (int i = 1; i < n; i++) { // i = meters of rope to cut off
06          int best = Math.max(n - i,     getMaxProduct(n - i));
                  //(no cut) rope remains,     cut== subproblem
07          maxProduct = Math.max(maxProduct, i * best);
08     }
09    return maxProduct;
10 }
```

1 meter long rope

__ **M[1] = invalid**

2 meter long rope

__ __ M[2] = ?

__ __ There is only 1 possible way to cut two meter long rope, that is

Case 1: __ | __

max (1, M[1]) x **max**(1,M[1]) = 1 x 1 = 1

左大段 右大段

所谓的大段，是指我们可以靠读取M[i] 序列，而得到的值 (it is actually a sub-problem that we have solved before.)

3 meter long rope

__ __ __ M[3] = 2

|

__ __ __ There are only 2 possible wasy to cut two meter long rope, that is

Case 1: __ | __ __

max (1, M[1]) x **max**(2,M[2]) = 1 x 2 = 2

Case 2: __ __ | __

max (2, M[2]) x **max**(1,M[1]) = 2 x 1 = 2

M[3] = max (Case1, Case2) = max(2, 2) = 2

Determine if you are able to reach the last index.

For example:
index 0 1 2 3 4
A = [2,3,1,1,4], return true.
B = [3,2,1,0,4], return false.

input= 0 1 2 3 4
A = [2,3,1,1, 4], return true.
M = t t t t t

Anonymous Iguana

Solution:

1. base case M[n-1] = true;
2. induction rule
    a. M[i] represents whether we could reach the target from i-th index
    b. M[i] = true iff there exists a j, where M[j] = true, AND **j -i <= input[i]**
       false otherwise

Time = O(n^2) or O(k* n) where k is the largest value in input, AND k < n