Dynamic programming (DP) is a method of solving various problems. To apply this method we have to divide the problem to subproblems (and then somehow unite their solutions to get the solution to the problem). We defined brute force, did't we? We will remember how to solve subproblem and if we will solve this subproblem one more time we can use our memorised solution. Now we have brute force with memorization. But in DP we extend the idea of memorisation. It may happen that two or more different subproblems can be solved identicaly or give equal answers. If we could combine such subproblems and solve only one of them we would speed up our brute force. How can we understand which subproblems are similar? We have to determine attributes of subproblems which are really important for us. For example, there can be two subsegments of array with equal lengths and sums of elements, but if length and sum of elements are the only two important attributes they could be considered identical. So we should decide which attributes are important and which are not, and what is the "answer" to the subproblem. We will call them "state of DP" and "value of DP" respectively.

A. K-based numbers

State - (length of the number, is the last digit zero?)

Value - amount of numbers

Complexity - $O(n)$

B. Staircases

State - (number of bricks, the height of the highest step)

Value - number of staircases

Complexity - $O(n^3)$

C. Brackets sequence

State - segment of the given sequence

Value - how much symbols we need to add to make the sequence regular

Complexity - $O(n^3)$

D. Gentlemen

State - (prefix of the cards, sum of the weights)

Value - number of ways to reach this state

Complexity - $O(n^2C)$

E. False Mirrors

State - the subset (mask) of the broken balconies

Value - minimum amount of damage before this state

Complexity - $O(2^n n)$

F. Anniversary Firework

State - (the length of the segment, time of the last salvo)

Value - probability

Complexity - $O(n^3)$

G. Martian Army

State - subtree

Value - the answer to the problem for the subtree

Complexity - $O(n)$

H. One-two, one-two

State - (the length of the number, number modulo m)

Value - can we reach this state (bool)

Complexity - $O(len \cdot m)$

I. Staircases. Version 2

State - (number of bricks, number of steps)

Value - number of staircases

Complexity - $O(n\sqrt{n})$, $O(n)$ memory

J. K-based numbers. Version 2

Look at problem A. We should calculate DP faster.

Use matrix multiplication and binary power.

To multiply two long longs use binary multiplication.

Complexity - $O(\log n \log C)$

K.Classmates

State - subset (mask) of those who already know the news

Value - minimal time to reach this state

Transition - whom will we call to on the next step. We have to check can we call them all. For that purpose we will use Hungarian algorithm (maximal matching in bipartite graph)

Complexity - $O(3^n n^3)$