Dynamic Programming (linear)

Step 1: Identify the sub-problem in words.

Step 2: Write out the sub-problem as a recurring mathematical decision.

Step 3: Solve the original problem using Steps 1 and 2.

Step 4: Determine the dimensions of the memoization array and the direction in which it should be filled.

Step 5: Code it! In recursive or iterative way.

E-OLYMP **1560. Decreasing number** There are three types of operations you can perform on an integer:

1. If it's divisible by 3, divide it by 3;
2. If it's divisible by 2, divide it by 2;
3. Subtract 1.

Given a positive integer *n*, find the minimal number of operations needed to produce the number 1.

► Let ***f***(*n*) contains the minimum number of operations to convert the number *n* to 1. For example,

- f(1) = 0, since we already have number 1;
- f(2) = 1, perform operations 2 → 1;
- f(5) = 3, perform operations 5 → 4 → 2 → 1;
- f(10) = 3, perform operations 10 → 9 → 3 → 1;

In the case of *n* = 10 it is better to subtract 1 first than to use the greedy idea and divide by 2.

Consider the process of calculating the function f(*n*).

- If we divide number *n* by 3 (if *n* is divisible by 3), then

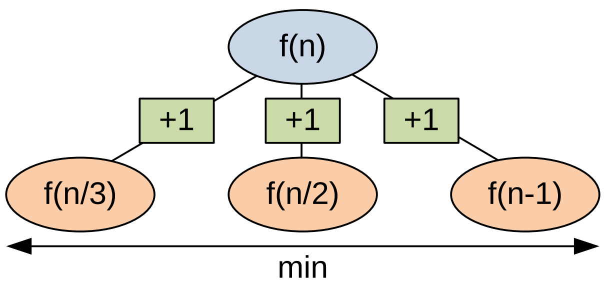
$$f(n) = f(n / 3) + 1$$

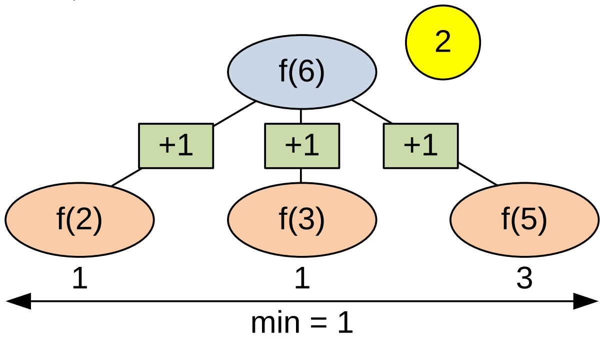
- If we divide number *n* by 2 (if *n* is divisible by 2), then

$$f(n) = f(n / 2) + 1$$

- If we subtract 1 from *n*, then

$$f(n) = f(n - 1) + 1$$





From the number n we can get one of three numbers: $n / 3$, $n / 2$ or $n - 1$. The number of operations for which each of these numbers we can be reduced to 1, equals to f(n / 3), f(n / 2) and f(i – 1) respectively. Since we are interested in the smallest number of operations, we have the relation:

$$f(n) = \min(f(n-1), f(n / 2), f(n / 3)) + 1,$$
$$f(1) = 0$$

Moreover, if n is not divisible by 2 (or by 3), then the corresponding element (f(n / 2) or f(n / 3)) is absent in the function *min*. For example, for $n = 8$ we have:

$$f(8) = \min(f(7), f(4)) + 1$$

For $n = 7$ we get:

$$f(7) = \min(f(6)) + 1 = f(6) + 1$$

The values of the function f(n) will be stored in the cells of array d[MAX], where MAX = $10^6 + 1$. Fill the cells of array d from 1 to 10^6 according to the given recurrence relation. For example, the following table shows the values of d[i] for $1 \le i \le 11$:

i	1	2	3	4	5	6	7	8	9	10	11
d[i]	0	1	1	2	3	2	3	3	2	3	4

For example, d[10] = min(d[9], d[5]) + 1 = min(2, 3) + 1 = 3. It means that it is more efficient to subtract 1 from 10, rather than divide it by 2.

Exercise. Find the values of d[i] for the next i:

i	12	13	14	15	16	17	18	19	20
d[i]									

E-OLYMP **1619. House robber** You are a professional robber planning to rob houses along a street. Each house has a certain amount of money stashed, the only constraint stopping you from robbing each of them is that adjacent houses have security system connected and it will automatically contact the police if two adjacent houses are broken into on the same night.

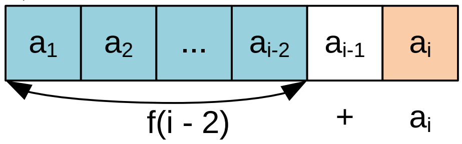
Given a list of non-negative integers representing the amount of money of each house, determine the maximum amount of money you can rob tonight without alerting the police.

► Let's number the houses starting from index one (i-th house contains a_i money). Let f(i) be the maximum amount of money that can be robbed from houses with numbers from 1 till i-th.

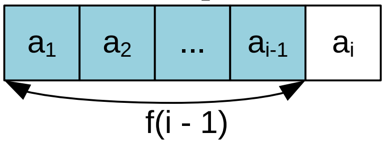
Then $f(1) = a_1$, $f(2) = \max(a_1, a_2)$.

To calculate f(i) we consider two cases:

- If the i-th house is robbed, then one can't rob the $(i - 1)$-th house. In this case profit will be $f(i - 2) + a_i$.



- if the i-th house is not robbed, the profit will be $f(i - 1)$.



So we have

$$f(i) = \max(f(i - 2) + a_i, f(i - 1))$$

Store the values of $f(i)$ in array res. The answer to the problem is the value of $f(n)$ = res[n].

i	1	2	3	4	5
a_i	6	1	2	10	4
f(i)	6	6	8	16	16

6
6

f(1) = 6

6	1
6	6

f(2) = 6

6	1	2
6	6	8

$f(3) = \max(f(2), f(1)+a_3) = \max(6,8) = 8$

6	1	2	10
6	6	8	16

$f(4) = \max(f(3), f(2)+a_4) = \max(8,16) = 16$

Let's find f(3). If house 3 is not robbed, the income is f(2) = 6. If house 3 is robbed, we can rob first house with income f(1) = 6 plus income for the third house that equals to 2 (the total profit is 6 + 2 = 8).

Let's find f(4). If fourth house is not robbed, the income is f(3) = 8. If fourth house is robbed, we can rob the first two houses with an income of f(2) = 6 plus income for the fourth house which equals to 10. Equating the total profit to 16 (6 + 10 = 16).

Exercise. Find the values of $f(i)$ for the next input data:

i	1	2	3	4	5	6	7
a_i	6	3	2	8	4	1	7
f(i)							

E-OLYMP **9628. Frog** There are n stones, numbered 1, 2, ..., n. For each i ($1 \le i \le n$), the height of stone i is h_i. There is a frog who is initially on stone 1. It will repeat the following action some number of times to reach stone n: if the frog is currently on stone i, jump to stone $i + 1$ or stone $i + 2$. Here, a cost of $| h_i - h_j |$ is incurred, where j is the stone to land on.

Find the minimum possible total cost incurred before the frog reaches stone n.

► Let dp[i] be the smallest cost of moving the frog to stone i. Then:

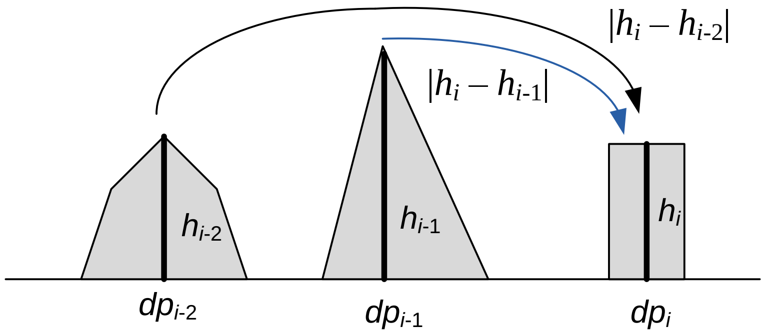
- dp[1] = 0 (do not need to move anywhere),
- dp[2] = $| h_2 - h_1 |$;

The frog can jump onto the i-th platform ($i \geq 3$):

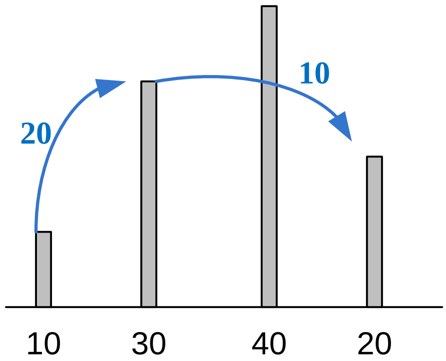
- either from (i – 1)-th with cost $|h_i - h_{i-1}|$;
- or from (i – 2)-th with cost $|h_i - h_{i-2}|$;

Hence

$$dp[i] = \min(dp[i-1] + |h_i - h_{i-1}| , dp[i-2] + |h_i - h_{i-2}|)$$



Example. For the given example, the path of a frog with cost 20 + 10 = 30 looks like this:



E-OLYMP **987. Nails** Some nails are hammered on a straight plank. Any two nails can be joined by a thread. Connect some pairs of nails with a thread, so that to each nail will be tied with at least one thread, and the total length of all threads will be minimal.

► Sort the nail's coordinates in array a. Let dp[i] equals to the minimal total length of all thread, when any two nails starting from the first one (the nails are numbered starting from 1) till i-th are connected with the thread.

If n = 2, both nails must be joined with the thread, so

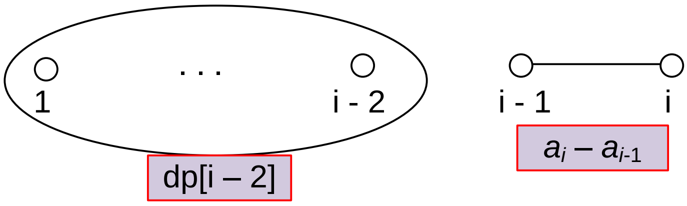
$$dp[2] = a_2 - a_1$$

If n = 3, we must connect first nail with the second, and second with the third. So

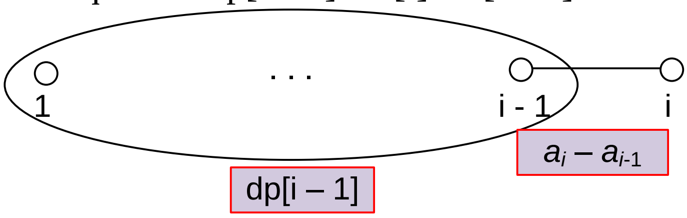
$$dp[3] = a_3 - a_1$$

To add i-th nail one has two possibilities to join it with the thread:

1) connect first i – 2 nails among themselves, the (i – 1)-th nail connect to the i-th. The total length of the thread for such connection equals to dp[i – 2] + a[i] – a[i – 1].



2) connect first i – 1 nails among themselves, the i-th nail we connect to the (i – 1)-th. The length of the thread equals to dp[i – 1] + a[i] – a[i – 1].

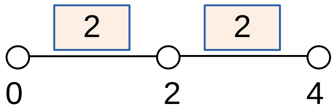


Select the connection method where the total length of the thread is smallest. So

$$dp[i] = \min(dp[i – 2], dp[i – 1]) + a[i] – a[i – 1]$$

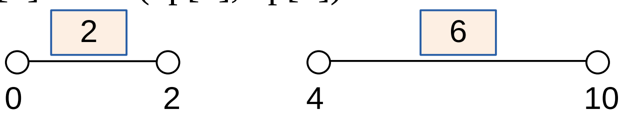
Sort the nail's coordinates for the sample input: 0, 2, 4, 10, 12. For two nails dp[2] = 2 – 0 = 2. For three nails we must connect them all with the thread (each nail must be tied with at least one thread), so

$$dp[3] = (4 – 2) + (2 – 0) = 4$$



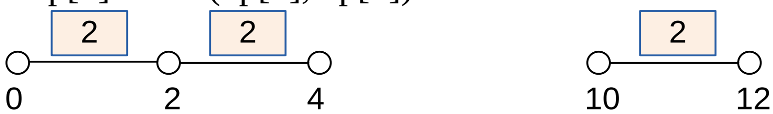
Let's calculate the optimal length of the thread for 4 nails:

$$dp[4] = \min(dp[2], dp[3]) + 10 – 4 = 2 + 6 = 8$$



For 5 nails the minimum possible length of the thread equals to

$$dp[5] = \min(dp[3], dp[4]) + 12 – 10 = 4 + 2 = 6$$



i	1	2	3	4	5
a_i	0	2	4	10	12
dp[i]	0	2	4	8	6

Exercise. Find the values of dp[i] for the next input data:

i	1	2	3	4	5	6
a_i	2	4	5	8	10	13
dp[i]						

E-OLYMP **44. The number of ones** In arithmetic expression you are allowed to use the number 1, operations of addition, multiplication and parenthesis. What is the minimum number of ones you need to obtain the positive integer *n*?

► Let f(*n*) equals to the smallest number of ones, with which you can get a number *n*. Obviously that f(1) = 1.

Number 2 can only be represented as the sum of two ones: 2 = 1 + 1. Therefore, f(2) = 2. Number 3 can only be represented as the sum of three ones: 3 = 1 + 1 + 1. Therefore, f(3) = 3.

Number 4 can be represented either as 4 = 1 + 1 + 1 + 1, or 4 = (1 + 1) * (1 + 1). However, in both cases, 4 ones are used. Hence f(4) = 4.

Consider the required expression with result *n*.

1. Let the last executed operation be addition. Let for example, the first term *i* contains f(*i*) ones, and the second term *n* – *i* contains f(*n* – *i*) ones. The value of *i* must be chosen so that the sum f(*i*) + f(*n* – *i*) is minimized.

i	+	*n* - *i*
f(*i*)		*f*(*n* - *i*)

The value of *i* is enough to iterate up to *n* / 2, since it can be assumed that the first term is not greater than the second one. Thus we have the relation:

$$f(n) = \min_{1 \le i \le n/2} \left(f(i) + f(n-i) \right)$$

2. Let the last executed operation in expression was multiplication. Let for example, the first term *i* contains f(*i*) ones, and the second term *n* / *i* contains f(*n* / *i*) ones. This case occurs if *n* is divisible by *i*.

i	*	*n* / *i*
f(*i*)		*f*(*n* / *i*)

The value of *i* must be chosen so that the sum f(*i*) + f(*n* / *i*) is minimized. The value of *i* is enough to iterate up to $\lfloor \sqrt{n} \rfloor$. Thus we have the relation:

$$f(n) = \min_{1 \le i \le \sqrt{n}} \left(f(i) + f(n/i) \right)$$

So

$$f(n) = \min\left(\min_{1 \le i \le n/2} \left(f(i) + f(n-i) \right), \min_{1 \le i \le \sqrt{n}} \left(f(i) + f(n/i) \right) \right)$$

Example. Compute the answer for *n* = 7:

f(7) = f(6) + f(1) = (f(2) + f(3)) + f(1) = 2 + 3 + 1 = 6

Number 7 can be represented with 6 ones:

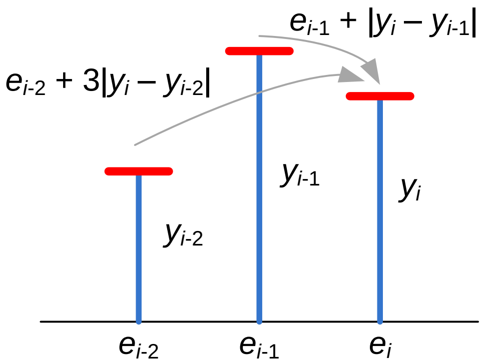
7 = (1 + 1) * (1 + 1 + 1) + 1

E-OLYMP **798. Platforms** In older games one can run into the next situation. The hero jumps along the platforms that hang in the air. He must move himself from one side of the screen to the other. When the hero jumps from one platform to the neighboring, he spends $|y_2 - y_1|$ energy, where y_1 and y_2 are the heights where these platforms hang. The hero can make a super jump that allows him to skip one platform, but it takes him 3 * $|y_3 - y_1|$ energy.

You are given the heights of the platforms in order from the left side to the right. Find the minimum amount of energy to get from the 1-st (start) platform to the n-th (last). Print the list (sequence) of the platforms that the hero must pass.

► Let e[i] contains the minimum amount of energy sufficient to get from platform 1 to platform i. Obviously, e[1] = 0 (to get from the first platform to the first is zero energy), and e[2] = $|y_2 - y_1|$, since the second platform can only be reached from the first one.

In cell p[i], we'll store the number of the platform from which we jumped to the i-th. Initially, we set p[1] = -1 (initially we are on the first platform), and also p[2] = 1.

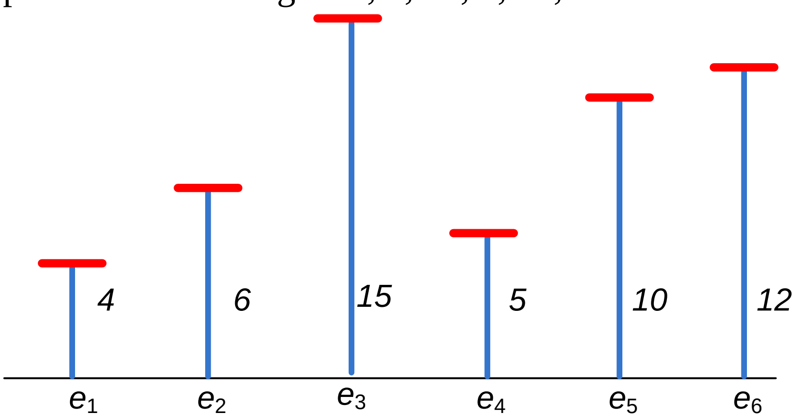


To the i-th platform ($i \geq 3$) you can jump either from ($i - 1$) -th, spending e[$i - 1$] + $|y_i - y_{i-1}|$ energy, or from ($i - 2$)-th, having made a super jump and spending e[$i - 2$] + $3 \cdot |y_i - y_{i-2}|$ energy. So

$$e[i] = \min(e[i-1] + |y_i - y_{i-1}| , e[i-2] + 3 \cdot |y_i - y_{i-2}|)$$

If to the i-th platform the jump is performed from ($i - 1$) -th, then set p[i] = $i - 1$. If from ($i - 2$) -th, then we set p[i] = $i - 2$. To find the number of platforms on which jumps from the first to the n-th were performed, one should walk from the n-th platform to the first one each time moving from the i-th platform to p[i] -th.

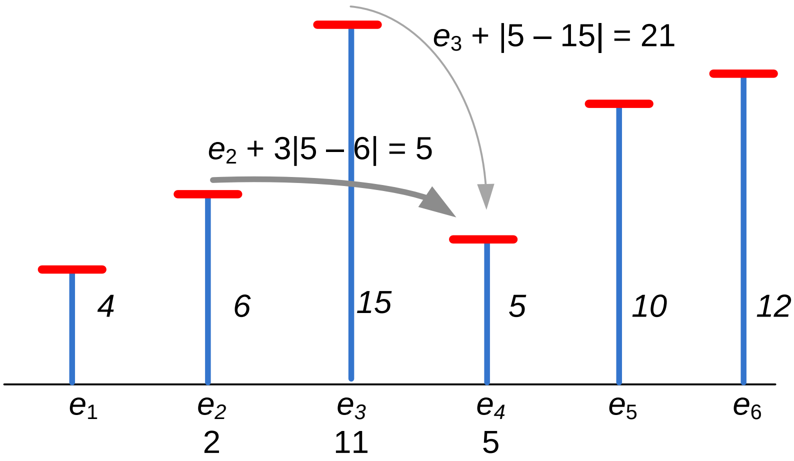
Let we have 6 platforms with heights 4, 6, 15, 5, 10, 12.



Calculate the spended energy when jumping:

e[1] = 0, p[1] = -1;

e[2] = |6 – 4| = 2, p[2] = 1;
e[3] = min(e[2] + |15 – 6|, e[1] + 3 * |15 – 4|) = min(11, 33) = 11, p[3] = 2;



e[4] = min(e[3] + |5 – 15|, e[2] + 3 * |5 – 6|) = min(21, 5) = 5, p[4] = 2;
e[5] = min(e[4] + |10 – 5|, e[3] + 3 * |10 – 15|) = min(10, 26) = 10, p[5] = 4;
e[6] = min(e[5] + |12 – 10|, e[4] + 3 * |12 – 5|) = min(12, 26) = 12, p[6] = 5;

i	1	2	3	4	5	6
y_i	4	6	15	5	10	12
e_i	0	2	11	5	10	12
p_i	-1	1	2	2	4	5

To restore the path, move from the final (6-th) platform back along the indexes of p[*i*]:

$$\boxed{6} \xrightarrow{p[6] = 5} \boxed{5} \xrightarrow{p[5] = 4} \boxed{4} \xrightarrow{p[4] = 2} \boxed{2} \xrightarrow{p[2] = 1} \boxed{1}$$

The list of platforms to go through is as follows:

1, 2, 4, 5, 6

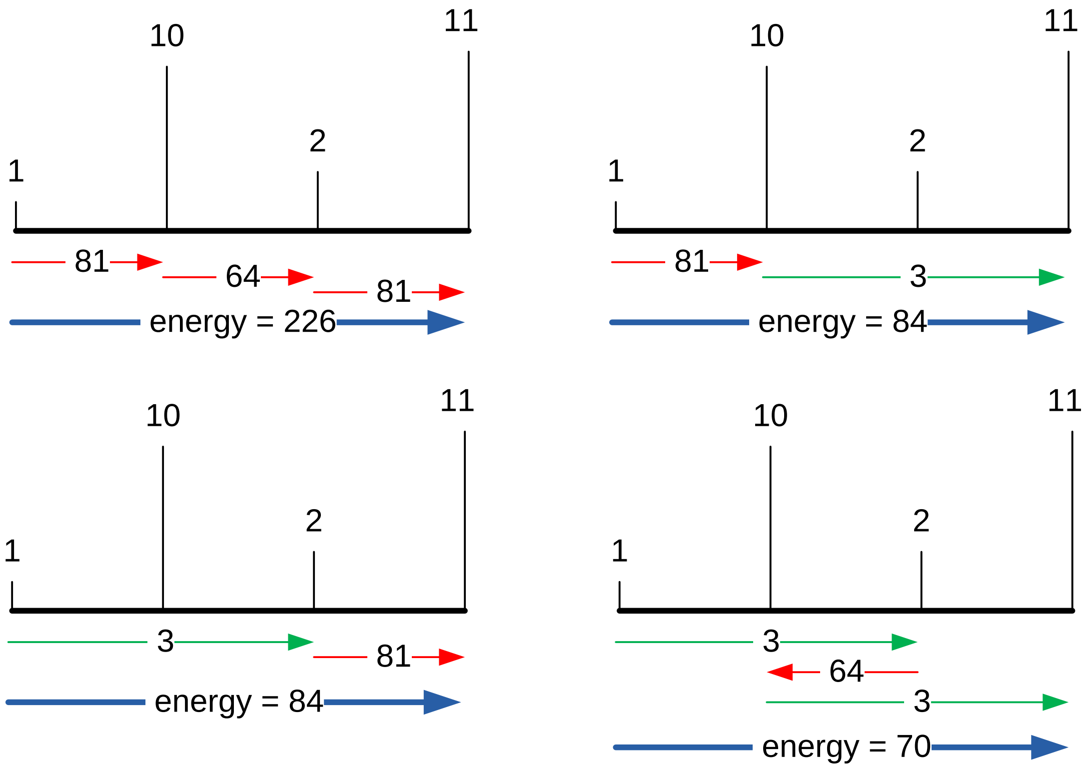
Exercise. Fill the arrays with the next input data.

i	1	2	3	4	5	6
y_i	8	4	1	5	12	3
e_i						
p_i						

E-OLYMP **806. Platforms - 3** In older games one can run into the next situation. The hero jumps along the platforms that hang in the air. He must move himself from one side of the screen to the other. When the hero jumps from one platform to the neighboring, he spends $|y_2 - y_1|^2$ energy, where y_1 and y_2 are the heights where these platforms hang. The hero can make a super jump that allows him to skip one platform, but it takes him 3 * $|y_3 - y_1|^2$ energy.

You are given the heights of the platforms in order from the left side to the right. Find the minimum amount of energy to get from the 1-st (start) platform to the n-th (last).

► With given energy functions sometimes it is optimal for the hero to make one move backwards. Consider the following example. Suppose we have 4 platforms with heights of 1, 10, 2, 11. We calculate the energy that hero should spent to reach the last platform in different ways.



As we can see, the optimal way is to jump from 1-st platform to the 3-rd using super jump, then return to the 2-nd and using super jump once more, to appear on the last, 4-th platform. The total spent energy equals to 3 + 64 + 3 = 70.

Let dp[i] be the minimal amount of energy enough to get from the 1-st platform to the i-th. Let dp[1] = 0, because initially we are on the first platform. We can get to the second platform either from the first only (if $n = 2$), or in two ways:

- from 1-st to the 2-nd using $|y_2 - y_1|^2$ energy;
- from 1-st to the 3-rd and then to the 2-nd spending 3 * $|y_3 - y_1|^2 + |y_2 - y_3|^2$ energy;

So if $n > 2$ then dp[2] = min($|y_2 - y_1|^2$, 3 * $|y_3 - y_1|^2 + |y_2 - y_3|^2$).

Now consider the calculation of dp[i]. One can get into i-th platform either from (i – 1)-th or from (i – 2)-th using super jump. But when $i < n$, one can get into the i-th platform from the (i + 1)-th, where we jumped from the (i – 1)-th. So dp[i] equals to minimum among the values:

- dp[i – 1] + $|y_i - y_{i-1}|^2$: normal jump from the (i – 1)-th platform;

- dp[i – 2] + 3 * $|y_i - y_{i-2}|^2$: super jump from the (i – 2)-th platform;
- dp[i – 1] + 3 * $|y_{i+1} - y_{i-1}|^2$ + $|y_i - y_{i+1}|^2$: one jumps from the (i – 1)-th to (i + 1)-th, and then to i-th platform. This movement possible only if $i < n$.