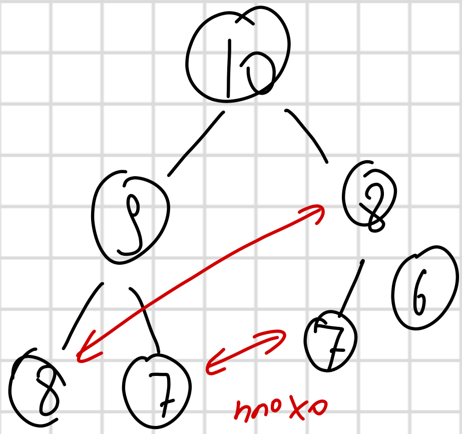
Динамическое программирование

Фиб. число.

$F_1=1 \quad F_2=1 \quad F_i = F_{i-1}+F_{i-2}$

```
f(i)
if (i ≤ 2):
    return 1
else: return f(i-1)+f(i-2)
```

$T(n) = 1 + T(n-1) + T(n-2)$



2-ой вариант

```
res = null
f(i):        O(n)
    if res[i] ≠ null: return res
    else:
        if (i ≤ 2): res[i] = 1
        else: res[i] = f(i-1) + f(i-2)
        return res[i]
```

Заходим один раз для каждого числа

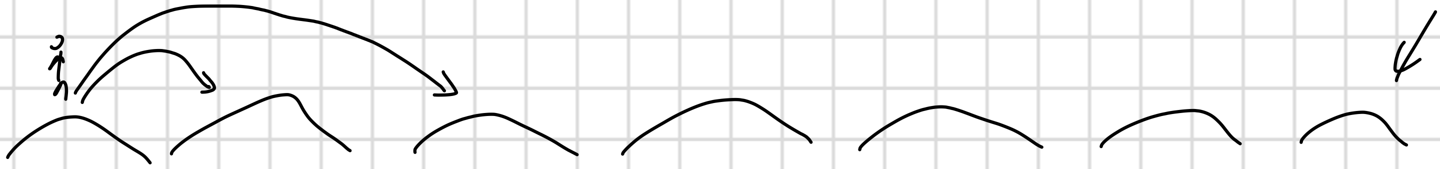
можно без рекурсии

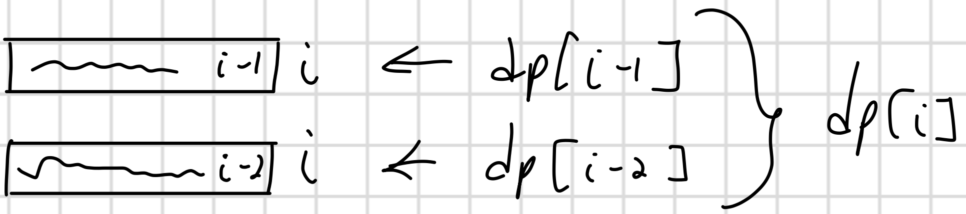
res[1] = res[2] = 1

for(i = 3...n):

  res[i] = res[i-1] + res[i-2]

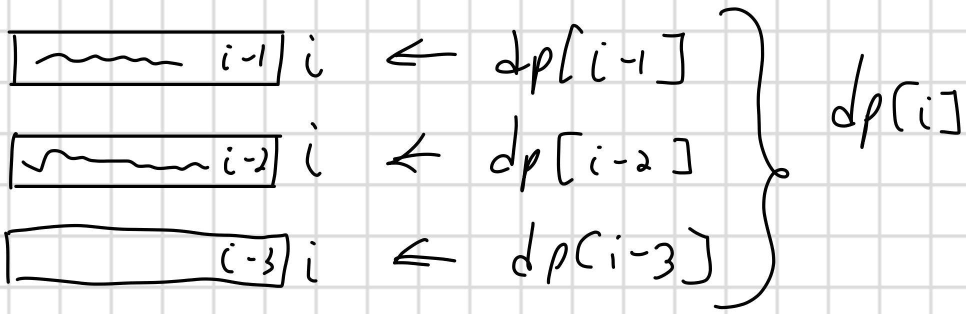
$O(n)$





dp[i] : # способов дойти до i

dp[i] = dp[i-1] + dp[i-2]



dp[i] = dp[i-1] + dp[i-2] + dp[i-3]

dp[1] = dp[2] = 1

dp[3] = 2 // 1→3
1→2→3

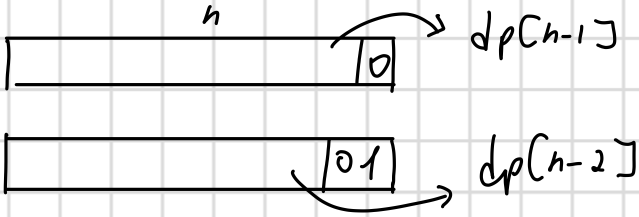
```
for (i=4...n):
    dp[i] = dp[i-1] + dp[i-2] + dp[i-3]
```

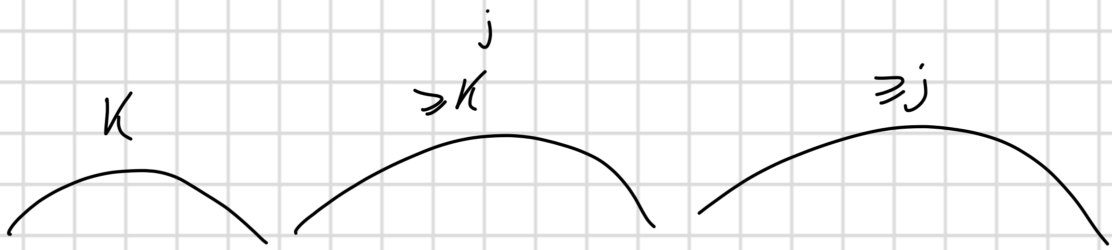
$\boxed{\ell \to K}$

```
dp[1] = dp[2]
for (i=3...n)
    for (j=1...K)
        if (i-j >= 0):
            dp[i] += dp[i-j]
```

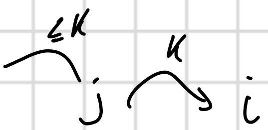
$O(nK)$

Для вектора без „11"





dp[len][last] # способов добрать до len ≤ last



for (i = 0 ... n)

  for (k = 1 ... i-1)

    for (j = 1 ... k)

      dp[i][k] += dp[i - k[j]]

$O(n^3)$

Пр.:

	1	2	3	4	5	6	7
1	1						
2	1						
3	1	1					
4	1	1	1				
5	1	2		1			
6					1		
7						1	

в 5 за прыжок 2

(5, 2)

3 → 5

1 → 3   2 → 3

dp[3][1] dp[3][2]

(6, 2)

4 → 6   dp[4][2]   dp[4][1]

(7, 3)

(4 → 7)

dp[4][2] dp[4][3] dp[4][1]

Текущий [k]

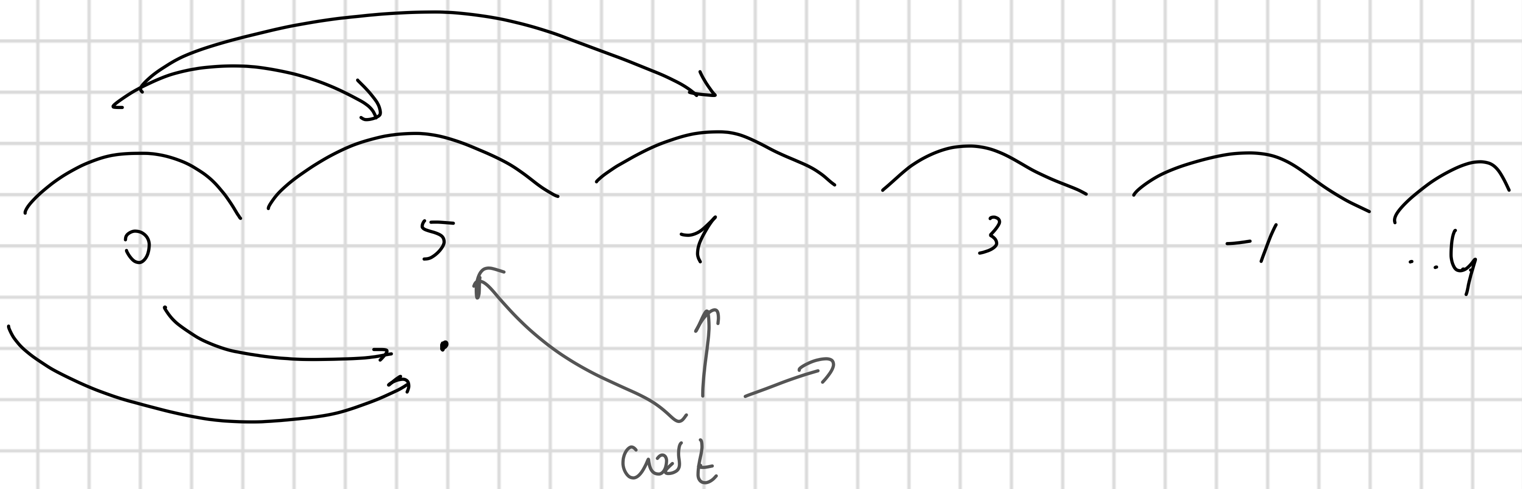
[k] = dp[i-k][k]

[≤k-1] = dp[i][k-1]

dp[i][k] - # способов прыгнуть ≤ k

$O(n^2)$

dp[i][k] = dp[i][k-1] + dp[i-k][k]



$dp[1] = 0$

$dp[2] = cost[2]$

for ($i = 3 \dots n$)

$\quad dp[i] = \min(dp[i-2], dp[i-1]) + cost[i]$

$dp = [\ 0\ \ 5\ \ 1\ \ 4\ \ 0\ \ 4\ ]$

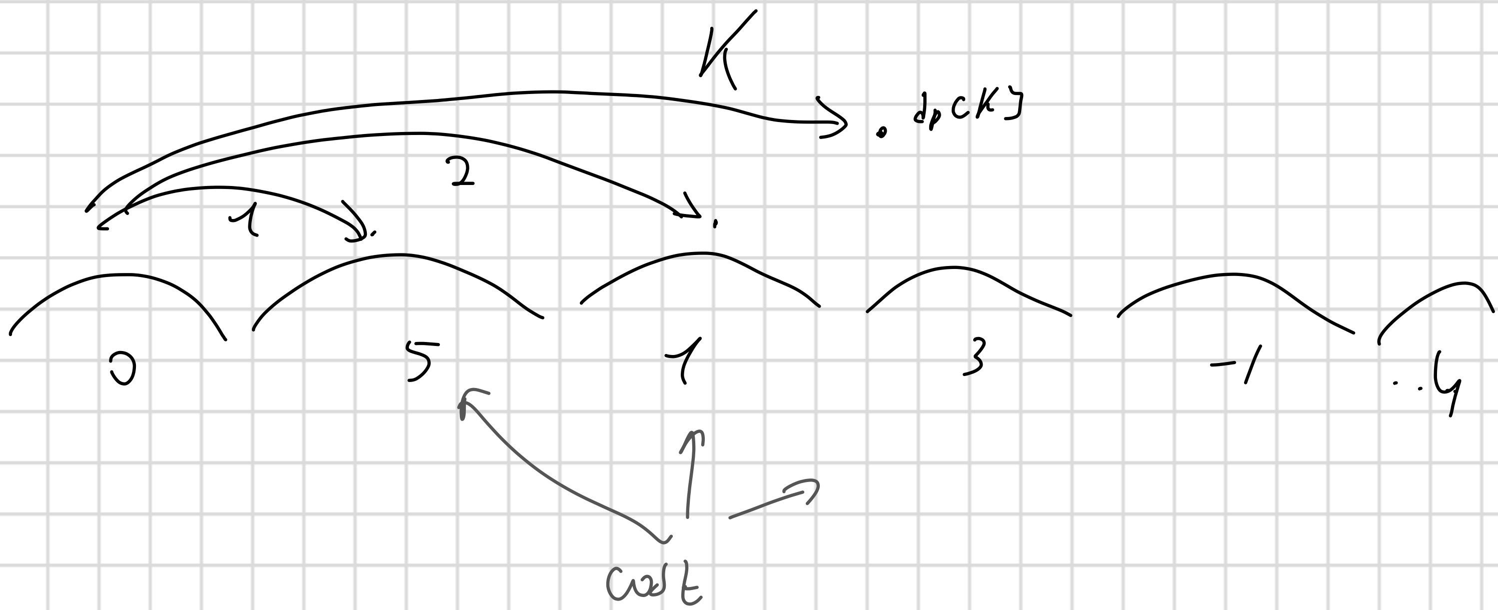
int end = n $\qquad dp[0] = \infty$

while (end > 1)

$\quad$ if (dp[end-1] > dp[end-2])

$\qquad$ end -= 2

$\quad$ else: end - 1



$dp[1] = 0$

for ($i = 2 \dots n$)

$\quad dp[i] = \infty$

$\quad$ for ($j = 1 \dots K$)

```
if (i-j < 1) break
dp[i] = min(dp[i], dp[i-j] + cost[i])
```