# ДИНАМИЧЕСКОЕ ПРОГРАММИРОВАНИЕ

# КОНЦЕПЦИЯ

Метод динамического программирования используется для задач, обладающих следующим свойством:

Имея решения некоторых подзадач (для меньшего числа *N*), можно найти решение исходной задачи, т.е. **оптимальное решение подзадачи большего размера можно построить из оптимальных решений подзадач**.

**Иначе говоря, ДП – решение ряда простых задач со своими входными данными с целью нахождения решения сложной задачи.**

# ВИДЫ

Одномерное – последовательности и т.д.

Многомерное – например, решение задач на расположение элементов на площади и т.д.

# ПРИМЕР ОДНОМЕРНОЙ ДИНАМИКИ

Последовательность Фибоначчи задается формулами:

$F_1 = 1$, $F_2 = 1$, $F_n = F_{n-1}+F_{n-2}$ при $n>1$.

Необходимо найти $F_n$ по номеру n.

Неэффективное рекурсивное решение:

```
int F(int n){
    if (n<20 return 1;
    else return F(n-1)+ F(n-2);
}
```

Эффективное решение по методу ДП:

```
F[1] = 1;
F[2] = 1;
for (int i=2;i<n;i++)
    F[i] := F[i-1] + F[i-2];
```

# ПРИМЕР ОДНОМЕРНОЙ ДИНАМИКИ

И снова наступает очередной Конец Света! На этот раз во всем виноват календарь племени Июйля. Бобры этого племени знали толк в математике. Археологу Умному Бобру досталась священная скрижаль с магическим числом. Перевод со старобобруйского гласит:
«Да снизойдет на тебя благодать Великого Бобра, да раскроются чакры твои, да не ослепнет третий глаз твой от созерцания Истины! Возьми магическое число, вычти из него любую цифру, которая входит в написание этого числа и получи новое магическое число. Повтори эту операцию до тех пор, пока очередное магическое число не обратится в ноль. Сколько вычитаний сделаешь, столько и будет Земля стоять на Трех Бобрах!»
Очевидно, что при разной последовательности вычитаний можно получить различное количество операций. Но Умный Бобер готовится к худшему и просит рассчитать наименьшее количество операций, которое потребуется для обращения магического числа в ноль.

# ПРИМЕР ОДНОМЕРНОЙ ДИНАМИКИ

Входные данные

В единственной строке задано целое магическое число n, $0 \le n$.

для получения 20 баллов требуется решить задачу при $n \le 106$ (подзадача C1);

для получения 40 баллов требуется решить задачу при $n \le 1012$ (подзадачи C1+C2);

для получения 100 баллов требуется решить задачу при $n \le 1018$ (подзадачи C1+C2+C3).

Выходные данные

Выведите одно число — наименьшее количество вычитаний, которое обратит магическое число в ноль.

# ПРИМЕР ОДНОМЕРНОЙ ДИНАМИКИ

```
#include<bits/stdc++.h>
using namespace std;
bool cmp(string a, string b)
{
    return (a+b)<(b+a);
}
int main()
{
   string s;
   long long t,res=0;
   cin>>s;
   t=stoll(s);
   while (t>0){
```

# ПРИМЕР ДВУМЕРНОЙ ДИНАМИКИ

Дано прямоугольное поле размером n на m клеток. Можно совершать шаги длиной в одну клетку вправо и вниз. Посчитать, сколькими способами можно попасть из левой верхней клетки (с координатами (1,1)) в правую нижнюю (с координатами (n, m)).

- В некоторую клетку с координатами (i,j) можно прийти только сверху или слева, т.е. из клеток с координатами (i-1, j) и (i, j-1).
- Таким образом, для клетки (i, j) число маршрутов
- A[i,j] = A[i-1,j] + A[i,j-1], т.е. задача сводится к двум подзадачам.
- Необходимо последовательно пройти по строкам (или столбцам), находя число маршрутов для текущей клетки по формуле.
- Тривиальный случай: A[1,1] = 1
- Ответ находится в элементе A[n,m]

# ПРИМЕР ДВУМЕРНОГО ДП

Дано прямоугольное поле размером n на m клеток. Можно совершать шаги длиной в одну клетку вправо, вниз или по диагонали вправо-вниз.
В каждой клетке записано некоторое натуральное число.
Необходимо попасть из левой верхней клетки (с координатами (1,1)) в правую нижнюю (с координатами (n,m)).
Вес маршрута вычисляется как сумма чисел со всех посещенных клеток.
Необходимо найти маршрут с минимальным весом.

# ПРИМЕР ДВУМЕРНОГО ДП

В некоторую клетку с координатами (i,j) можно прийти из клеток с координатами (i-1, j), (i, j-1) и (i-1, j-1).

Допустим, что для каждой из этих трех клеток уже найден маршрут минимального веса, а сами эти веса находятся в W[i-1,j], W[i,j-1] и W[i-1,j-1].

Чтобы найти минимальный вес для (i,j), необходимо выбрать минимальный из весов W[i-1,j], W[i,j-1], W[i-1,j-1] и прибавить к нему число, записанное в текущей клетке:

$$W[i,j] = W[i-1,j] + W[i,j-1] + W[i-1,j-1].$$