Задача A. Числа Фибоначчи

Эта задача разбиралась на лекции. Код на Python:

```
n = int(input())
f = [1, 1]
while len(f) < n:
    f.append(f[-1] + f[-2])
print(f[-1])
```

Задача B. Кузнечик

Введем $dp[i]$ как количество способов добраться на i-ый столбец. База: $dp[1] = 1, dp[2] = 1$. Далее делаем переход для i-ой позиции. Понятно, что мы пришли в этот столбец непосредственно с $i-1$-ой позиции. или с $i-2$-ой позиции. Поэтому, так как эти множества вариантов непересекаемы, $dp[i] = dp[i-1] + dp[i-2]$.

Код на эту задачу совпадает с кодом для предыдущей задачи :)

Задача C. Кузнечик-K

Та же идея, что с обычным кузнечиком, только переход у нас в столбик i из отрезка $[max(1, i-k), i-1]$. База динамики: $dp[1] = 1$.

Код на Python:

```
n, k = map(int, input().split())
dp = [1]
for i in range(1, n):
    if i < k:
        dp.append(sum(dp))
    else:
        dp.append(sum(dp[-k:]))
print(dp[-1])
```

Задача D. Кузнечик и лягушки

Пусть dp_i будет количество способов добраться на позицию i. База динамики — $dp_1 = 1$. Переходы на колонку i будет из колонок от $\max(1, i-k)$ до $i-1$, там, где нет лягушек. Если в этом отрезке на какой-то позиции нет лягушки, то прибавляем: $dp_i += dp_j$. Проще всего сначала посчитать очередное dp_i, а потом поставить туда 0 (если в клетке лягушка).

Код на Python:

```
n, k = map(int, input().split())
m = int(input())
a = set(map(int, input().split()))
dp = [1]
for i in range(1, n):
    if i < k:
        dp.append(sum(dp))
    else:
        dp.append(sum(dp[-k:]))
    if i + 1 in a:
        dp[-1] = 0
print(dp[-1])
```

Задача E. Лесенки

При решении этой задачи нужно было заметить, что когда мы уменьшаем число блоков в самой нижней строке, мы получаем лесенки, ответы для которых мы уже посчитали ранее. Используем эту идею для построения динамики!

- Параметрами динамики будут являться общее число блоков i и число j блоков, лежащих в основании лесенки. В качестве состояния динамики возьмём максимальное число лесенок, которое можно построить, используя i блоков и храня j блоков в основании
- Переходом динамики будет являться переиспользование ранее посчитанных ответов для лесенок – возьмём лесенку из $i - j$ блоков и, перебрав дополнительный параметр k, прибавим её ответы к ответам для лесенки из i блоков
- В качестве ответа возьмём сумму всех значений в n-й строке

Асимптотика решения: $O(N^3)$

Задача G. Калькулятор

В этой задаче надо сделать восстановление ответа, как разбиралось на лекции. Для каждого числа сохраним то число, из которого получили текущее. Потом будем прыгать по предкам, пока не дойдем до начала.

Код на Python:

```
n = int(input())
dp = [0, 0]
pred = [-1, -1]
for i in range(2, n + 1):
    best_pred = i - 1
    dp.append(dp[i - 1] + 1)
    if i % 2 == 0 and dp[i // 2] < dp[i]:
        best_pred = i // 2
        dp[i] = min(dp[i], dp[i // 2] + 1)
    if i % 3 == 0 and dp[i // 3] + 1 < dp[i]:
        best_pred = i // 3
        dp[i] = min(dp[i], dp[i // 3] + 1)
    pred.append(best_pred)
print(dp[-1])
ans, cur = [], n
while cur > 0:
    ans.append(cur)
    cur = pred[cur]
ans.reverse()
print(*ans)
```

Задача H. Покупка билетов

Пусть dp_i = сколько минимум времени надо, чтобы дать билеты первым i людям. Тогда есть следующие варианты для подсчета dp_i:

- $dp_{i-1} + a_i$
- $dp_{i-2} + b_{i-1}$
- $dp_{i-3} + c_{i-2}$

Понятно, что из этих вариантов берем минимальный.

Код на Python:

```
n = int(input())
dp = [0]
q = [()]
for i in range(1, n + 1):
```

```
    a, b, c = map(int, input().split())
    q.append((a, b, c))
    cur = dp[i - 1] + a
    if i >= 2:
        cur = min(cur, dp[i - 2] + q[-2][1])
    if i >= 3:
        cur = min(cur, dp[i - 3] + q[-3][2])
    dp.append(cur)
print(dp[-1])
```

Задача I. Весь мир театр

Поскольку ограничения невелики, давайте просто переберем все возможные числа b мальчиков в группе и посчитаем, сколькими способами мы можем образовать группу с b мальчиками.

Сначала рассмотрим только значения, где $b \geqslant 4$. При заданном b ясно, что количество девочек в группе должно быть $g = t - b$. Если $g < 1$, не рассматриваем этот случай. При заданных значениях b и g существует $C_n^b \cdot C_m^g$ способов сформировать группу, поскольку мы можем объединить мальчиков и девочек независимо. Просто просуммируйте $C_n^b \cdot C_m^g$ для каждой пары (b, g). Опять же, используйте 64-битные типы для выполнения математических вычислений.

Можно предварительно вычислить все значения C_i^j, используя треугольник Паскаля, но можно также вычислить его с помощью традиционной формулы, если ее реализация учитывает возможное переполнение (30! не помещается в 64-битный целочисленный тип).

Задача J. Печать вслепую

Давайте посчитаем $dp[i][0/1]$ — минимальное количество раз, когда Тимми придется поменять руку, если последний символ он напечатал левой/правой рукой.

Легко определить переходы. Мы или меняем руки, или остаемся писать той же рукой. Решение приведено в коде ниже

```
for (int i = 0; i < n; ++i) {
    if (s[i] == 'X' || s[i] == 'F') {
        dp[i + 1][1] = min(dp[i][0] + 1, dp[i][1]);
    }
    if (s[i] == 'O' || s[i] == 'F') {
        dp[i + 1][0] = min(dp[i][0], dp[i][1] + 1);
    }
}
```