

Пусть Х — задуманное число. По условито задачи известно значение S = ((Х + о) h + h) о, из которого найдём А = — о — 1. Поскольку задуманное число X— целое и положительное, необходимо проверить, что S делится нацело на oh (остаток равен 0) и Х > 0. Если ни выполняется хотя бы одно из этих условий, выводим —1.

(ГеіиенпеЗадачн—hопwяА.А.)

Количество секунд в сутках на планете Перо равно obc. Тогда число прошедшт суток равно целой части от деления t на obc, а остаток от деления на obc совпадает с количеством секунд, прошедших за последние, ещё незакончившиеся, сутки. Именно это время

t1 = t mod (o6c)

будет соответствовать показаниям часов. Количество секунд в одном часе на планете равно бс. Целая часть от деления t1 на (бс), то есть *hour —— t I* div (be), — это количество «полных» часов, которое прошло за последние сутки. Значение t2 = t1 mod (бс) — это число секунд, прошедших за последний, ещё незакончившийся, час. Значение min = t2 div с это число «полных» минут, истёкших за последний незакончившийся час. Значение *sec* = t2 mod с — количество секунд, прошедших за последний, ещё незакончившийся, час. В ответе вьтводим числа *hour, mix, see.*



# Блок из k-значнътх *р-ичных* чисел содержит числа из диапазона [р'—'; *р\* —* 1], следовательно, его длина *Lz —— Х (р —* 1) Х *р!—!.* Определим, в каком блоке и на какой позиции находится искомая цифра, последовательно вычитая *Le. Lz. Lz, ...* из заданного п, пока п не станет отрицательным, и затем, вернувшись в предыдущий блок:

k := 1; *stepeч :=* l;

while u > 0 do begin

/ := k \* *stepeч* \* @ — I); dec(u, /);

inc(k);

*stepen : —— stepeч • р,*

end;

inc(u, /); dec(k);

*stepen : —— stepeч* div *р,*

определим, в каком по счету числе и в каком его разряде находится искомая цифра: com ch*isla :—— (ч + h — I)* div k;

mc :—— п mod *h’,*

if *газ ——* 0 then inc(mc, *h),-*

# Найдем значение числа, содержащего искомук› цифру, в 10-ичной системе счисления, а затем и саму цифру:

if *h —— I* then cfiis/o 10 := nom\_c/tïs/n

else ch*islo* 10 *:= stepeii +* uom\_ch*isla —* 1; for i := 1 to *h — газ бо*

chislo 10 := cfiis/o 10 div р; cï/ro .—— chislo 10 mod р;

if cifro *<* 10 then writelп(/iдпre.out, ci/ro)

else writelп(/iдпre.out, Chr(Ord('A') + *ci/ro* — 10));

*(V от енп о зв вчп — Kttндeр М.И.)*

Є( Сть 31, *M2, Мз — zочнп,* соответствуіощие выходам из норки. Если треугольник *МіМ2Мз* остроугольный, то кошка *К* должна занять позициіо в центре описанного вокруг *МіМ2Мз* круга. Если этот треугольник — тупоугольный (или прямоугольный), то положение кошки в середине наибольшей стороны *МіМ2Мь.* Если точти *Mr,* 32, *Ме* лежат на одной прямой, то положение кошки будет по-прежнему в середине отрезка наибольшей длины. Во всех случаях положение кошки определяется однозначно.

(Автор задачи и решения— *R"uндер М.М.)*

Решение задачи основано на использовании динамического программирования, и основная идея такого решения заклточается в следуіощем. Подсчитаем m[п] — количество способов нарисовать непересекак›щиеся отрезки с вершинами в заданных п точках.

Выделим одну из точек, например, первую, и разобьем способы на два непересекающихся класса: те, которые содержат отрезок, выходящий из точки 1, и те, в которых такого отрезка нет. Количество способов первого класса, очевидно, равно m[п — 1]. Все способы второго класса содержат некоторый отрезок, выходящий из точки 1. Пусть, это будет отрезок, соединяіощий точки с номерами 1 и k + 1. По разные стороны от этого отрезка расположено k — 1 и п — k — 1 точек. Количество способов провести непересекающиеся отрезки на первой дуге равно in[k — 1]; количество таких способов на второй дуге равно in[n — *h I ]. Ухо* правилу произведения общее число всех таких комбинаций равно m[k — 1] in[n — *h I ].* Суммируя по всем k от 1 до п — 1, получим окончательнуіо формулу для подсчета чисел in[n]:

с начальным условием m[0] = 1.

Приведем несколько первых значений последовательности in[п]:

1, 1, 2, 4, 9, 21, 51, 127, ... .

*Замечание.* Последовательность m[п] играет важнукі роль в дискретной математике. Эти числа называкітся чп п *Моцкина* (http://oeis.on/A001006), они отісывак›т количество способов провести непрерывнуто ломануіо в верхней полуплоскости, составленнук› из векторов (1; 1), (1; 0) и (1; —1), начинакіщукіся в начале координат и заканчиваіощуіося на оси абсцисс. Приведем явнукі формулу через биномиальные коэффиіщенты:



(Аптор задачи и решения *Киндс• р М.И.)*

Пусть попугай Пi выдрал пepo у Пz, тот — у Пз и так далее. Такая цепочка, в которуто, возможно, войдут не все попугаи, замкнётся. Аналогично составил цепочку из части оставшихся попугаев, и так далее до конца. Теперь можно указать, как поступить удаву. Он должен рассаживать попугаев из цепочек поочередно в разные клетки: Пi , Пз, ... — в одну клетку, П2, 34, ... — в другуто. Если цепочка состоит из *иечётиого* числа попугаев, то одного оставшегося попугая придётся проглотить. (При лк›бом способе рассаживания попугаев по клеткам нам не удастся рассадить нечётное число попугаев из этой цепочки по двум клеткам в соответствии с условием задачи.) Если же в цепочке *чётиое* число попугаев, то попугаев этой цепочки удаётся рассадить требуемым образом: всех попугаев с нечётными номерами размещаем в одну клетку, с чётными номерами — в другую. Таким образом, наименьшее число попугаев, которое необходимо съесть удаву, совпадает с *колпчестаож нечётньtх циклов в ориентированном графе*

Реализовать предложенный алгоритм можно, например, с помощьто обхода графа «в глубину». Сначала всем вершинам графа присвоим метку 0. Затем первую вершина графа полетим

меткой 1 (покрасим её в первый цвет). Вершина графа, связанная с ней ребром, получит метку

2. Смешная с ней вершина снова получит метку 1, и так далее до тех пор, пока не попадёт в вершину, которая была уже помечена раньше. В этом случае помеченную вершину пропускаем и переходим к следуіощей непомеченной вершине графа. В результате этого шага все вершины графа будут покрашены (помечены) в первый или второй цвет.

procedure find((, с: integer); begin

end;

if со/[(] 0 then exit; со/[i] := с;

find(P[(] , 3 — с);

fi11char(co/, sizeof(co/), 0); for i := 1 to п do

if со/[i] = 0 then find((,1);

На следуіощем этапе алгоритма заново просматриваем все вершины графа. Все вершины первот цвета помещаем в клетку 1, все вершины второго цвета — в клетку 2. Если *смежные* вершины оказались одного цвета, то одну из двух вершин удаляем (соответствутощего попугая удав съедает), увеличивая при этом счётчик «съеденных» попугаев: inc(eat).

cmm[1] := 0; cmm[2]:= 0; eot:= 0; for i := 1 to ч do begin

if со/[i] = со/[Р[i]] then

begin inc(eat); continue; end; inc(unm [co/[i]]);

Cage[со/[i], com [со/[i]]] := i;

**end;**

*Председаіпель жюри М.И.Киидер*