**9 – 11 классы**

1. **Ферзь, ладья и конь (Простая математика)**

*(Время: 1 сек. Память: 16 Мб Сложность: 34%)*

*Тесты*

|  |  |  |
| --- | --- | --- |
| **№** | **INPUT.TXT** | **OUTPUT.TXT** |
|  | D1 D3 E5 | 29 |
|  | C1 B1 A1 | 25 |
|  | E5 A3 D5 | 37 |
|  | C3 D3 E3 | 32 |
|  | D4 C5 G4 | 37 |

*Разбор*

Для решения данной задачи нужно понять, каким образом определить, бьет ли та или иная фигура, находящаяся в клетке (x1,y1), некоторую клетку (x2,y2). При этом следует помнить, что нас интересуют только пустые клетки, т. к. именно их и нужно считать. Задача сильно упрощается тем, что фигуры могут «ходить» сквозь друг друга, поэтому, проверяя возможность достижения фигурой некоторой клетки, можно пренебрегать другими фигурами.

Проще всего эта задача решается для ладьи, т. к. некоторая клетка будет находиться под боем тогда и только тогда, когда одна из координат у рассматриваемых клеток совпадает, т. е. x1=x2 или y1=y2 (клетки расположены либо на одной горизонтали, либо на одной вертикали).

В случае с ферзем, помимо ограничений на ладью (ферзь может двигаться, как и ладья), добавляется условие расположения на одной из диагоналей, математически это происходит только тогда, когда модули разностей координат этих клеток совпадают, т. е. |x1-x2|=|y1-y2| (для слона это условие является единственным).

Случай с конем, пожалуй, самый сложный. Конечно, можно проверить все 8 клеток, в которые может сходить конь из клетки (x1,y1), и если окажется, что одной из таковых является клетка (x2,y2), то клетка «под боем», иначе – нет. Но запись в условии восьми проверок достаточно громоздка, и поэтому возможны ошибки. Данный вопрос можно решить более «хитро»: из того, что конь ходит буквой «Г» следует, что модуль произведения разностей координат этих двух клеток всегда равен двум.   
С другой стороны, это также является достаточным условием. Поэтому возможность достижения конем клетки математически эквивалентна выполнению условия |(x1-x2)\*(y1-y2)|=2.

Наиболее простой способ подсчета количества клеток «под боем» заключается в переборе всевозможных клеток и проверке их на достижимость одной из трех фигур. Это позволит избежать повторений клеток, которые бьют несколько фигур одновременно. Следует при этом не забывать, что клетки, на которых стоят фигуры, не должны учитываться.

1. **Годовой баланс (сортировка и последовательности)**

*(Время: 1 сек. Память: 16 Мб Сложность: 33%)*

*Тесты*

|  |  |  |
| --- | --- | --- |
| **№** | **INPUT.TXT** | **OUTPUT.TXT** |
|  | 14 -34 | 57 |
|  | 1203 251 | 2085 |
|  | 124578 90231 | 865182 |
|  | -254105 -54328 | -17023 |
|  | -2416098 5432786 | 1320989 |

*Разбор*

Для того чтобы значение **a-b** было максимально, необходимо из **a** получить максимально возможное число, а из **b**, наоборот, определить наименьшее. Если число неотрицательно, то для получения минимума цифры сортируются в порядке неубывания, а при вычислении максимума, соответственно, в порядке невозрастания. Для отрицательных чисел ситуация с сортировкой цифр меняется с точностью до наоборот. Причем, нужно заметить, что при сортировке цифр по возрастанию нужно избежать момента, где пришлось бы поставить **0** на первое место (этого можно достичь, например, при сортировке пузырьком, если при сравнении первых двух соседних элементов не менять их в том случае, когда второй - это ноль).

Сортировку цифр в числе проще проводить, если преобразовать само число в строку, и работать с обычным массивом символов. Причем, знак "-" не обязательно включать в этот массив. Отсортировав его, например пузырьком, можно сделать обратное преобразование строки в число. После нахождения максимального **a** и минимального **b** останется просто вывести значение **a-b**.

Для хранения чисел можно использовать обычный 4-байтовый целый тип (long или longint например), т. к. результат вычислений не может превзойти значения 1999999998 по абсолютной величине. Если бы ограничения на **a** и **b** были не до миллиарда, а до двух миллиардов, то итоговая разница могла бы достигать значения, большего 19 миллиардов, где пришлось бы использовать такие 8-байтовые целые типы, как int64 или \_\_int64.

1. **Дачники (геометрия)**

*(Время: 1 сек. Память: 16 Мб Сложность: 45%)*

Тесты

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **INPUT.TXT** | **OUTPUT.TXT** | **Баллы** |
|  | 3  6 6 3 6 6 9 8 7 5 4  13 5 9 2 9 8 12 8 12 2  3 2 2 1 2 3 6 3 6 1 | 2 | 50 |
|  | 7  -2 2 --4 4 -2 4 -2 2 -4 2  3 1 0 3 3 5 5 2 4 0  -4 1 -3 1 -1 1 -1 -1 -3 -1  -2 -2 -3 -2 -2 -3 -3 -4 -4 -3  -1 -3 -2 -4 -1 -2 0 -3 -1 -5  2 -3 0 -1 4 -1 4 -3 0 -3  0 0 0 1 2 1 2 0 0 0 | 5 | 50 |

*Разбор*

Решение этой задачи основывается на решении другой геометрической задачи.

Как по отношению некоторого вектора на плоскости расположена заданная точка:

на векторе – между его началом и концом;

впереди вектора;

позади вектора;

справа от линии вектора;

слева от вектора.

Слева

Впереди

ди

Между

Справа

Позади

Конец вектора

Начало вектора

Если решение этой задачи получено, то решение задачи «Дачники» заключается в следующем: дачник тогда приземлится на свой участок, если точка его приземления находится на одном из векторов или справа от каждого из векторов-сторон его участка.

То есть, пользуясь переменными исходных данных и представляя каждую сторону участка дачника как вектор, логическое выражение приземления дачника на свой участок будет выглядеть, например, следующим образом.

(YESМежду(X, Y, X1,Y1,X2,Y2) OR YESСправа (X, Y, X1,Y1,X2,Y2)) AND

(YESМежду(X, Y, X2,Y2,X3,Y3) OR YESСправа (X, Y, X2,Y2,X3,Y3)) AND

(YESМежду(X, Y, X2,Y2,X3,Y3) OR YESСправа (X, Y, X2,Y2,X3,Y3)) AND

(YESМежду(X, Y, X3,Y3,X4,Y4) OR YESСправа (X, Y, X3,Y3,X4,Y4))

Х2, Y2

Х, Y

Х1, Y1

Х4, Y4

Х3, Y3

Принадлежность точки вектору определить несложно, если записать уравнение прямой, проходящей через 2 точки и подставив в него координаты точки.

**4. Лесенка**

*(Время: 1 сек. Память: 16 Мб Сложность: 55%)*

*Тесты*

|  |  |  |
| --- | --- | --- |
| **№** | **INPUT.TXT** | **OUTPUT.TXT** |
|  | 10 | 10 |
|  | 13 | 18 |
|  | 16 | 32 |
|  | 20 | 64 |
|  | 23 | 104 |

*Разбор*

Идея решения сводится к вычислению числа вариантов разложения заданного целого числа на неповторяющиеся слагаемые, или, другими словами, сколькими способами данное число N можно представить в виде суммы чисел от 1 до N.

Как один из вариантов решения – это рекуррентная формула:

F(n, M) = F(n, M1) + F(n-Mk, M1), где

n – заданное число (число кубиков);

M – массив слагаемых от 1 до n;

k – текущее число элементов массива M;

M1 – массив слагаемых без элемента Mk (наибольшего).

Начальные условия:

F(0, M) = 1;

F(n, M) = 1 при n = M1 и k = 1;

F(n, M) = 0 при n ≠ M1 и k = 1.

Возможны и другие рекурсивные процедуры, например, набирать слагаемые, начиная с наибольшего и уменьшением его на 1 в каждом рекурсивном вызове, подсчитывая число таких вызовов. Например для n = 15 получаем ряд комбинаций слагаемых:

15/0, 14/1, 13/2, 12/3, 11/4, …,8/7, 7/6/2, 7/5/3 …, 5/4/3/2/1.