Разбор задач Пятой Интернет-олимпиады

Автор: Максим Буздалов

Введение

В базовой номинации Пятой Интернет-олимпиады сезона 2006-2007 участникам было предложено для решения 8 задач. В олимпиаде приняло участие 101 команда, из них 86 решили хотя бы одну задачу.

Наиболее простой оказалась задача «H. Кипячение чая» — ее решили 71 команда. Наиболее сложной — задача «E. Обходчик лабиринтов» — которую решило всего 3 команды.

Условия задач, результаты олимпиады, тесты и решения жюри можно найти на сайте интернет-олимпиад `http://neerc.ifmo.ru/school/io`.

Задача A. Шахматная мастерская

Заметим, что способов правильной раскраски даже нестандартной шахматной доски всего 2.

Введем описанную в условии задачи систему координат на шахматной доске, то есть каждому полю соответствует пара чисел: номер горизонтали и номер вертикали, на которых находится поле.

Тогда одна из правильных раскрасок соответствует условию «если сумма координат четна, то поле имеет белый цвет, иначе черный», вторая правильная раскраска ей противоположна.

Пусть число отличий данной раскраски от первой правильной раскраски составляет A_1, а от второй — A_2. Тогда выберем минимально отличающуюся правильную раскраску и выведем в выходной файл сначала число полей, которые надо перекрасить, а затем сами поля.

Задача B. Сжатие последовательности

Задача решается методом динамического программирования [1].

Пусть $C[i]$ — число способов сжать последовательность из i единиц, используя числа от 1 до A. Тогда очевидно $C[0] = 1$, и справедливо следующее рекуррентное соотношение:

$$C[i] = \sum_{j=1}^{max(i,A)} C[i-j]$$

Оно получается при рассмотрении приписывания какого-нибудь из чисел $1 \dots A$ к уже имеющейся последовательности.

Ответом на задачу является $C[N]$.

При заданных в условии ограничениях ($1 \le A \le N \le 1000$) результат вычислений может превзойти рамки любого стандартного целочисленного типа, поэтому требуется реализовать операцию сложения длинных чисел.

Например, возможна такая реализация.

```
program compress;

const
    SIZE = 1000;

procedure print(var a: array of integer);
var
    i: integer;
    j: integer;
begin
    j := -1;
```

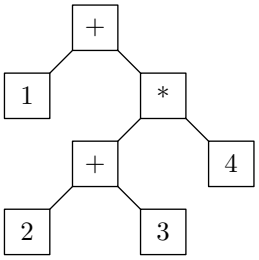
```
    for i := SIZE - 1 downto 0 do begin
        if (j = -1) and (a[i] <> 0) then begin
            j := i;
        end;
    end;
    if j = -1 then begin
        write(0);
    end else begin
        for i := j downto 0 do begin
            write(a[i]);
        end;
    end;
end;

procedure add(var a, b : array of integer);
var
    i: integer;
begin
    for i := 0 to SIZE - 1 do begin
        a[i] := a[i] + b[i];
        if a[i] >= 10 then begin
            a[i] := a[i] - 10;
            a[i + 1] := a[i + 1] + 1;
        end;
    end;
end;

var
    ans: array[0..1000, 1..SIZE] of integer;
    i, j, a, n: integer;
begin
    reset(input, 'compress.in');
    rewrite(output, 'compress.out');
    read(n, a);

    ans[0][1] := 1;

    for i := 1 to n do begin
        for j := 1 to a do begin
            if i >= j then begin
                add(ans[i], ans[i - j]);
            end;
        end;
    end;
    print(ans[n]);
    writeln;
end.
```

Задача C. Игра в 24

Рассмотрим выражение, полученное произвольной расстановкой знаков операций и скобок, примененных к числам из карточки.

$$\boxed{1} + \ (\ \boxed{2} + \boxed{3} \ ) \ * \ \boxed{4}$$

По определенным правилам выражение графически изображается в виде двоичного дерева. Будем называть его *деревом выражения*. Операции располагаются в вершинах дерева, а операнды образуют левое и правое поддеревья вершины с данной операцией. На рисунке приведен пример дерева выражения.



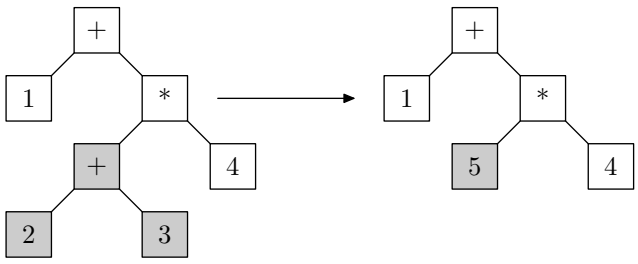
Дерево выражения соответствует порядку применения операций к операндам. Предполагается стандартный приоритет операций (умножение и деление выполняется раньше сложения и вычитания). Операции с одинаковым приоритетом ($+$ и $-$, $*$ и $/$) выполняются в порядке слева направо. Например, для выражения $1 + 2 * 3 - 4 - 5$ порядок выполнения операций (показанный дополнительными скобками) следующий: $((1 + (2 * 3)) - 4) - 5$.

С учетом вышесказанного можно легко показать, что каждому выражению соответствует ровно одно дерево выражения, и каждому дереву выражения соответствует ровно одно выражение (если считать одинаковыми выражения, в которые добавляются или удаляются скобки, не влияющие на порядок выполнения операций).

Каждый узел дерева выражения, соответствующий бинарной операции (сложение, вычитание, умножение), имеет двух потомков.

Заметим, что числа из карточки являются листьями дерева выражения, соответствующего любой расстановке знаков операций и скобок.

Рассмотрим следующую операцию над деревом выражения. Возьмем какой-нибудь узел дерева, чьими потомками будут листья этого дерева. В этом узле записана некоторая операция. Применим эту операцию к числам, записанным в потомках узла, и получим некоторое число как результат. Затем удалим узел, и на его место подвесим лист с записанным в нем результатом.

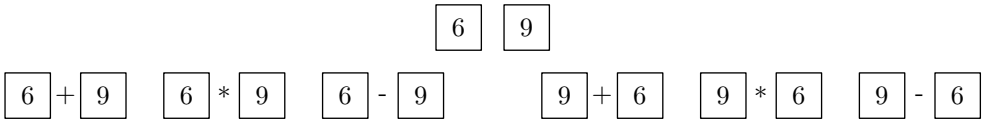


В результате данной операции получается новое дерево выражения. Назовем эту операцию *сокращением* дерева выражения.

Можно показать, что любое сокращение дерева выражения не меняет результат вычисления этого выражения.

Если некоторое дерево сократить нельзя, то это дерево состоит из одного листа, а соответствующее ему выражения — из одного числа.

Пусть S — множество листьев дерева выражения (размер его больше единицы). Сгенерируем все множества листьев деревьев выражения, получающихся из данного путем выполнения одной операции сокращения. Это можно сделать, перебирая все пары чисел из S (порядок следования учитывается), все операции, и применяя выбранную операцию к выбранной паре чисел. Сами числа из этой пары удаляются из множества, а результат применения к ним операции — добавляется. Таким образом, из множества S размера n получаем $3n(n-1)$ множеств размера $n-1$.



Выполняя эту процедуру несколько раз, начиная с множества чисел, данного во входном файле, получим набор множеств из одного элемента — возможные результаты вычисления выражения, полученного произвольной расстановкой знаков операций и скобок. Остается вывести в выходной файл `YES`, если в этом наборе присутствует число 24, и `NO`, если это не так.

Задача D. Гипотеза Гольдбаха

Для ограничений, данных в условии ($4 \le x \le 10000$) нетрудно каким-либо методом, например, методом *решета Эратосфена* [4], выяснить для каждого натурального числа, меньшего x, простое ли оно или составное.

После этого достаточно рассмотреть все пары чисел, дающих в сумме x, и найти число таких пар, в которых оба числа будут простыми.

Приведем пример решения на языке *Паскаль*:

```
program goldbach;
const MAX_SIZE = 10000;
var
    isPrime: array [1..MAX_SIZE] of boolean;
    x, a, i, j: integer;
begin
    reset(input, 'goldbach.in');
    rewrite(output, 'goldbach.out');
    readln(x);
    isPrime[1] := false;
    for i := 2 to MAX_SIZE do isPrime[i] := true;
    for i := 2 to MAX_SIZE do begin
        if isPrime[i] then begin
            j := 2 * i;
            while (j <= MAX_SIZE) do
            begin
                isPrime[j] := false;
                j := j + i;
            end;
        end;
    end;
    a := 0;
    i := 2;
    j := x - 2;
    while (i <= j) do begin
        if isPrime[i] and isPrime[j] then begin
                inc(a);
        end;
        inc(i);
        dec(j);
    end;
    writeln(a);
end.
```

Задача E. Обходчик лабиринтов

Решение задачи заключается в аккуратной реализации того, что описано в условии.

А именно, пусть $B[i][j]$ — значение логического типа, означающее, может ли обходчик находиться в комнате номер i после шага j. Вначале будет верно только $B[S][0]$, где S — комната, откуда начинается обход.

Будем также хранить коридоры лабиринта в виде массива записей вида «начало коридора; конец коридора; цвет коридора».

На каждом шаге переберем все коридоры. Если номер цвета коридора совпадает с инструкцией и обходчик мог после предыдущего шага находиться в комнате, являющейся началом коридора, то после текущего шага он может находиться в комнате, являющейся концом коридора.

Ответом на задачу будет количество таких $B[i][L]$, что $1 \le i \le N$ и $B[i][L] =$ `true`. Если этот ответ равен 0, то выводим `Hangs`.

Заметим, что при реализации можно легко обойтись одномерным массивом.

В целом эта задача является задачей на эмуляцию поведения недетерминированного конечного автомата (НКА), описание которого можно найти в [3].

Задача F. Система пересекающихся множеств

Будем хранить массив логических значений $S[i][j]$ — лежит ли число j в множестве i.

Операция `ADD a b` в этом случае устанавливает $S[a][b]$ в `true`.

Операция `LISTSET a` выводит такие i, что $S[a][i] =$ `true`, проверяя все возможные i в порядке возрастания.

Операция `LISTSETSOF b` выводит такие j, что $S[j][b] =$ `true`, проверяя все возможные j в порядке возрастания.

Если в одной из операций `LISTSET`, `LISTSETSOF` список пуст, то необходимо выводить -1.

Задача G. Манхеттенские улицы

Ответом на задачу является суммарная площадь улиц плюс суммарная площадь авеню минус суммарная площадь перекрестков. Площадь одной улицы, как и одного авеню, равна $d \cdot l$, площадь каждого перекрестка равна $d \cdot d$, так как улицы и авеню пересекаются под прямым углом. Перекрестков всего $n \cdot m$.

Итого, получаем формулу

$$(n+m)(d \cdot l) - (n \cdot m)(d^2)$$

Так как ограничения на числа в задаче достаточно большие, то необходимо вычислять ответ, используя целочисленные типы `int64` в Delphi, `long long` в C++, `long` в Java.

Задача H. Кипячение чая

Заметим, что вставляя тройник с k разъемами в свободный разъем, мы увеличиваем число свободных разъемов на $k-1$. Отсюда видно, что число свободных разъемов никак не зависит от того, как и в каком порядке соединять тройники, и равно

$$1 + \sum_{i=1}^{N} (a_i - 1)$$

Список литературы

[1] Кормен Т., Лейзерсон Ч., Ривест Р. Алгоритмы: построение и анализ. - М.: МЦНМО, 1999. - 960 с., 263 ил.

[2] Шень А. Программирование: теоремы и задачи. - М.: МЦНМО, 1995. - 264 с.

[3] Хопкрофт Дж., Мотвани Р., Ульман Дж. Введение в теорию автоматов, языков и вычислений. - Вильямс, 2002. - 528 с.

[4] Статья про решето Эратосфена в Википедии `http://ru.wikipedia.org/wiki/Решето_Эратосфена`