Разбор задач Шестой Интернет-олимпиады

Автор: Александр Торопов

Введение

В базовой номинации Шестой Интернет-олимпиады сезона 2006-2007 участникам было предложено для решения 8 задач. В олимпиаде приняло участие 85 команд, из них 55 решили хотя бы одну задачу.

Наиболее простыми оказались задачи «A. Снова простые числа» и «F. Поместье» — их решили 35 и 34 команды соответственно. Наиболее сложной — задача «C. Преобразование ДНК» — которую решила только 1 команда.

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

Задача A. Снова про простые числа

Покажем, что проверить число x на простоту можно за время $O(\sqrt{x})$. Действительно, пусть x — составное, то есть у него есть хотя бы один делитель. Покажем, тогда, что у x есть делитель, не превосходящий $\sqrt{x}$.

Пусть $x = a \cdot b$. Если $a \le \sqrt{x}$, то требуемый делитель найден. Иначе ($a > \sqrt{x}$), рассмотрим $b = x/a \le x/\sqrt{x} = \sqrt{x}$.

Таким образом, для проверки простоты числа за время $O(\sqrt{x})$ необходимо перебрать все числа от 2 до $\sqrt{x}$. Если хотя бы для одного i ($2 \le i \le \sqrt{x}$) i делит x, то число x не простое.

Приведем пример написания процедуры $isPrime(x)$, которая возвращает $true$, если число простое, и $false$ иначе.

```
function isPrime(x: Integer):boolean;
var
  i: Integer;
begin
  if x = 1 then
  begin
    result := false;
    exit;
  end;
  i := 2;
  while i * i <= x do
  begin
    if x mod i = 0 then
    begin
      result := false;
      exit;
    end;
    i := i + 1;
  end;
  result := true;
end;
```

Теперь воспользуемся тем, что $|b - a| \le 1000$. Из этого следует, что мы можем найти все простые числа в этом диапазоне за время $O(|b - a| \cdot \sqrt{b})$. Найдя все числа (а их не более 1000), переберем их и найдем простое число с максимальной суммой цифр. Заметим, что хранить все простые числа от a до b нет необходимости, т.к. найдя очередное простое число, мы можем сравнить его сумму цифр с текущим максимумом.

Задача B. Игра в фишки

Сопоставим каждой фишке первого цвета число 3, каждой фишке второго цвета — число 5, третьего — 6. Теперь определим число S, как:

$$S = \underbrace{3 \oplus 3 \ldots \oplus 3}_{A \text{ раз}} \oplus \underbrace{5 \oplus 5 \ldots \oplus 5}_{B \text{ раз}} \oplus \underbrace{6 \oplus 6 \ldots \oplus 6}_{C \text{ раз}},$$

где $\oplus$ — это битовая операция «xor» Одним из свойств операции «xor», которое нам понадобится, является следующее:

$a \oplus a = 0$. Заметим, что числа $3, 5, 6$ были выбраны не случайно, так как

$$3 \oplus 5 = 5 \oplus 3 = 6,$$

$$3 \oplus 6 = 6 \oplus 3 = 5,$$

$$6 \oplus 5 = 5 \oplus 6 = 3.$$

Пусть теперь мы заменили две фишки одного цвета на фишку другого цвета. Учитывая вышесказанное, понятно, что число S от этого не изменилось.

Теперь рассмотрим позиции, когда игра «сыграна». В этих позициях S принимает три возможных значения: 3, 5, 6. Таким образом задача свелась к нахождению числа S (игра может быть сыграна, если $S = 3$, 5 или 6), которое легко посчитать используя свойство операции «xor» $a \oplus a = 0$. Однако, в решении есть одна тонкость. Все дело в том, что данное решение не учитывает тот факт, что количество фишек в промежуточных ситуациях не может стать отрицательным. Поэтому нужно рассмотреть особый случай, когда фишек каких-либо двух цветов нет, а количество фишек третьего цвета не равно 1. Доказательство того, что это единственный случай, нарушающий закономерность, мы оставляем читателю в качестве упражнения.

Приведем код программы на языке *Паскаль*:

```
read(test);
for i := 1 to test do
begin
        read(a, b, c);
        if (a = 0) and (b = 0) and (c <> 1) then
        begin
                writeln('No');
                continue;
        end;
        if (a = 0) and (b <> 1) and (c = 0) then
        begin
                writeln('No');
                continue;
        end;
        if (a <> 1) and (b = 0) and (c = 0) then
        begin
                writeln('No');
                continue;
        end;
        if (a mod 2 = 1) then a := 3;
        else a := 0;
        if (b mod 5 = 1) then b := 5;
        else b := 0;
        if (c mod 6 = 1) then c := 6;
        else c := 0;
```

```
          s := a xor b xor c;
          if (s = 3) or (s = 5) or (s = 6) then
                    writeln('Yes')
          else
                    writeln('No');
end;
```

Задача C. Преобразование ДНК

Для начала заметим, что поскольку, по условию, решение всегда существует, то количество букв каждого вида в первой строке равно количеству букв этого же вида во второй строке.

Будем решать задачу следующим образом. Пусть мы добились того, что первые k букв исходной последовательности ДНК после применения необходимых преобразований (обозначим ее s_1) совпадают с первыми k буквами требуемой последовательности ДНК (обозначим ее s_2).

Разберем, как добиться того, чтобы после применения одного преобразования совпадали первые $k+1$ букв. Для этого выберем букву стоящую на позиции j, что $j > k$ и $s_1[j] = s_2[k+1]$ и применим преобразование с $k+1$ по j-тый символ к строке s_1. Понятно, что после этого у строк s_1 и s_2 будут совпадать первые $k+1$ букв. Таким образом, мы сможем не более чем за $l-1$ (где l — это длина последовательности ДНК) операцию преобразовать исходную последовательность ДНК в требуемую.

Будем считать, что процедура *reverse(i, j: longint)* производит разворот фрагмента строки $s1$ с i-го по j-й символ и выводит в выходной файл соответствующую информацию. Приведем основной фрагмент программы, решающий нашу задачу:

```
for i := 1 to l - 1 do
begin
          for j := i to l do
          begin
                    if s1[i] = s2[j] then
                    begin
                              reverse(i, j);
                              break;
                    end;
          end;
end;
```

Задача D. Поместье

В задаче требовалось найти площадь выпуклой оболочки точки и окружности при условии, что путь, соединяющий центр окружности и точку, является отрезком. Если точка лежит внутри или на окружности, то ответом будет площадь окружности, равная πR^2, где R — радиус окружности.

Более интересным представляется случай, когда точка лежит вне окружности. Обозначим точкой O центр окружности, точкой A — место, где проходил исторический разговор, Точки B и C будут точками касания касательных из точки A к окружности, а точку пересечения AO и окружности обозначим D. Заметим, что площадь искомой «капли» равна $\pi R^2 + S_{\triangle ABO} + S_{\triangle ABO} - S_{\smile OCD} - S_{\smile OBD}$, где $S_{\smile OCD}$ и $S_{\smile OBD}$ — площади секторов OCD и OBD соответственно.

Легко понять, что

$$AO = \sqrt{(x_A - x_O)^2 + (y_A - y_O)^2}$$

$$OC = OB = R$$

$$AC = AB = \sqrt{R^2 - AO^2}$$

Теперь из треугольника $\triangle OAC$ получаем, что

$$\angle COB = 2 \cdot arctan(\frac{AC}{OC})$$

Теперь мы вычислили все необходимые величины, чтобы записать ответ:

$$S = \pi R^2 + 2 \cdot \frac{OC \cdot AC}{2} - \frac{\angle COB \cdot R^2}{\pi}$$

Задача E. Задача о рюкзаке

Как и сказано в условии задачи, данная задача является NP-полной (Подробнее о NP-полных задачах можно узнать из [1]) Для ее решения надо было лишь организовать перебор всех возможных подмножеств предметов, которые можно взять, и выбрать среди них лучшее. Наиболее удобный способ перебрать все подмножества — это перебрать числа от 0 до $2^n - 1$, где n — количество предметов. Пусть i $(0 \leq i \leq 2^n - 1)$ описывает подмножество. В таком случае j-й предмет входит в подмножество i, если j-й бит числа i равен 1. Приведем пример основного фрагмента реализации программы на языке *Паскаль*:

```
bestT := -1;
bestP := -1;
count := 1;
for i := 1 to n do
        count := count * 2;
for i := 0 to count - 1 do
begin
        curW := 0;
        curP := 0;
        for j := 0 to n do
        begin
                if (t and shl(1, i)) = 0 then
                begin
                        curP := curP + p[i];
                        curW := curW + w[i];
                end;
        end;
        if curW < W then
        begin
                CheckAndUpdate
        end;
end;
```

«CheckAndUpdate» — фрагмент программы, который проверяет, лучше ли наше решение текущего максимума.

Задача F. Четно-нечетная задача

Представим число a в восьмеричной системе счисления $\overline{a_n a_{n-1} \dots a_1 a_0}$, тогда $a = a_1 \cdot 8^{n-1} + a_2 \cdot 8^{n-2} + \dots + a_{n-2} \cdot 8^2 + a_{n-1} \cdot 8 + a_n$. Теперь легко понять, что третья справа цифра (т.е. a_{n-2}) равна $[\frac{a}{64}]$ mod 8. Задача свелась к нахождению во входном файле всех чисел a, что a mod $2 = 0$ и $([\frac{a}{64}]$ mod $8)$ mod $2 = 1$ и выводу их в отсортированном порядке.

Следует отметить, что в данной задаче надо использовать быстрые сортировки (т.е. работающие за время $O(N \log N)$). О них можно почитать в [1].

Задача G. Точки и линии

В задаче требовалось вывести «Yes», если возможно построить неориентированный граф из N вершин и M ребер, так, чтобы он был несвязным. Разберем, какое максимальное количество ребер может быть в несвязном графе из N вершин.

Пусть G — это граф, имеющий N вершин, K компонент связности и максимально возможное количество ребер M. Покажем, что:

$$M = \frac{(N-K)\cdot(N-K+1)}{2}$$

Легко понять, что каждая компонента графа G является полным графом. Пусть n_i — количество вершин в компоненте с номером i. Без ограничения общности можно полагать, что $n_1 \geq n_2 \geq \ldots \geq n_k$. Докажем, что $n_2 = n_3 = \ldots = n_k = 1$. Предположим, что $n_2 > 1$, тогда если мы отцепим от компоненты 2 вершину u, то потеряем $n_2 - 1$ ребро. Прицепим u к первой компоненте, таким образом получая n_1 новых ребер. Так как $n_1 > n_2 - 1$, то получаем, что если $n_2 > 1$, то M не максимально, что противоречит выбору G. Таким образом, $n_2 = n_3 = \ldots = n_k = 1$.

Теперь из формулы максимального количества ребер для k компонент легко установить, что выгоднее всего брать $k = 2$. Таким образом, получаем решение:

```
read(n, m);
if ((n - 1) * (n - 2)) / 2 >= m then
        write('Yes')
else
        write('No');
```

Задача H. Шаблон программы

Решение состоит в простом суммировании описанных в условии задачи величин для символов входного файла. Для простоты реализации будем использовать массив, в качестве индекса в котором будет выступать символ.

Приведем пример реализации данного подхода:

```
var
  energy: array [char] of Integer;
  c: char;
  sum, i: Integer;

begin
  reset(input, 'template.in');
  rewrite(output,'template.out');

  for c := 'a' to 'z' do
  begin
    energy[c] := (ord(c) - ord('a') + 1) div 10 + (ord(c) - ord('a') + 1) mod 10;
    energy[chr(ord(i) - ord('a') + ord('A'))] := energy[i] + 10;
  end;
  energy[' '] := 4;
  for c := '0' to '9' do
    energy[c] := 13 - (ord(c) - ord('0'));
  energy['.'] := 5;
  energy[';'] := 7;
  energy[','] := 2;
```

```
  energy['='] := 3;
  energy['+'] := 3;
  energy['-'] := 3;
  energy[''''] := 3;
  energy['"'] := 3;

  energy[')'] := 1;
  energy['('] := 1;

  energy['{'] := 8;
  energy['}'] := 8;
  energy['['] := 8;
  energy[']'] := 8;
  energy['<'] := 8;
  energy['>'] := 8;

  energy[10] := 0;

  sum := 0;
  while not EOF do
  begin
    read(c);
    sum := sum + energy[c];
  end;
  write(sum);
end.
```

Список литературы

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