Разбор задач Восьмой Интернет-олимпиады

Введение

В базовой номинации Восьмой Интернет-олимпиады сезона 2008-2009 участникам было предложено для решения восемь задач. В олимпиаде приняло участие 49 команд, из них 47 решили хотя бы одну задачу.

Наиболее простой оказалась задача «G. Фигурное катание» — ее решили 46 команд. Наиболее сложной — задача «B. Электрическая схема» — ее решили 6 команд.

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

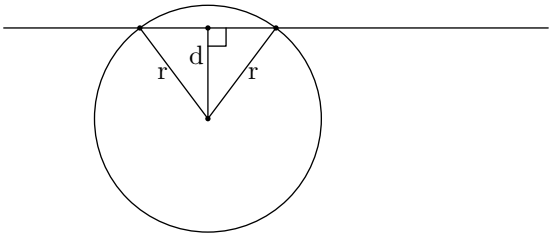
Задача A. Прямая и окружность

Автор задачи: Федор Царев
Автор разбора: Роман Сатюков

В этой задаче требуется найти длину общей части прямой и круга. Определим сначала число точек пересечения. Для этого вычислим расстояние от центра круга до прямой по формуле $d = \frac{|Ax+By+C|}{\sqrt{A^2+B^2}}$. Возможны три случая:

- если $d > r$, то число точек пересечения равно нулю;
- если $d = r$, то есть ровно одна точка пересечения;
- если $d < r$, то точек пересечения две.

Если точек пересечения меньше двух, то ответ равен нулю.



Если число точек пересечения равно двум, то длина общей части равна $2 \cdot \sqrt{r^2 - d^2}$ по теореме Пифагора.

Фрагмент решения на языке Pascal:

```
d := abs(a * x + b * y + c) / sqrt(a * a + b * b);
if (d >= r) then begin
  writeln(0);
  exit;
end;
writeln(2 * sqrt(r * r - d * d));
```

Задача B. Электрическая схема

Автор задачи: Федор Царев
Автор разбора: Роман Сатюков

Переберем все возможные способы подать напряжения от 1 до n вольт на n транзисторов так, чтобы на разные транзисторы было подано разное напряжение — этим способам соответствуют все перестановки чисел от 1 до n. Всего таких вариантов будет $n!$

Проверим для каждого такого способа, является ли он разумным — выполняются ли при таком способе подачи напряжения указанные в условии задачи условия для всех проводов и для всех пар транзисторов, не соединенных проводами.

Фрагмент решения на языке Pascal:

```
// g[i][j] = true тогда и только тогда, когда в исходной схеме i-ый и j-ый
// транзисторы соединены проводом
var g: array [1..8, 1..8] of boolean;

// Число транзисторов
var n: integer;

// Перестановка транзисторов
var p: array [1..8] of integer;

// u[i] = true тогда и только тогда, когда i-ый транзистор уже использован
// в перестановке
var u: array [1..8] of boolean;

function bt(i: integer): integer;
var
    j, k, ans: integer;

begin
    if (i = n + 1) then begin
        // Сгенерирована очередная перестановка
        // Проверяем - является ли она разумной
        for j := 1 to n do begin
            for k := 1 to n do begin
                if (g[j][k] and (not g[p[j]][p[k]])) then begin
                    // Перестановка не разумна
                    bt := 0;
                    exit;
                end;
            end;
        end;
        // Перестановка разумна
        bt := 1;
        exit;
    end else begin
        // Перебираем очередной элемент перестановки
        ans := 0;
        for j := 1 to n do begin
            if (u[j]) then
                continue;

            p[i] := j;
            u[j] := true;
            ans := ans + bt(i + 1);
            u[j] := false;
```

```
        end;
        bt := ans;
    end
end;

// Ответ на задачу - bt(1)
```

Время работы алгоритма — $O(n! \cdot n^2)$.

Задача C. Ключ

Автор задачи: Владимир Ульянцев
Автор разбора: Антон Банных

Решим задачу сначала для наборов из двух чисел ($n = 2$). Докажем, что ответом для данного p будет $p + 1$, то есть максимальную сумму имеет набор $(1, p)$. Действительно, пусть оптимальным является набор (a, b), где $2 \le a \le b$, $a \times b = p$. Тогда $a + b \le 2 \times b \le a \times b = p < p + 1$.

Аналогично для $n > 2$ можно показать, что ответом является $p + n - 1$. Для этого в оптимальном наборе $(a_1, a_2, \ldots a_n)$ будем последовательно заменять соседние пары чисел (a_i, a_{i+1}) на пары $(1, a_i \times a_{i+1})$, не изменяя произведение и не уменьшая сумму, пока не получим набор $(1, 1, \ldots p)$.

Фрагмент решения на языке Паскаль.

```
read(n, p);
writeln(p + n - 1);
```

Время работы алгоритма — $O(1)$.

Задача D. Число

Автор задачи: Владимир Ульянцев
Автор разбора: Антон Банных

Основная часть решения этой задачи состоит в том, чтобы написать функцию, определяющую, входит ли некоторая строка a как подпоследовательность в некоторую строку b.

Для решения этой подзадачи будем перебирать символы строки b, помня при этом, префикс какой длины i строки a уже встретился в качестве подпоследовательности в просмотренной части строки. При появлении символа, равного a_{i+1}, увеличиваем l на единицу. В конце проверим, что i совпадает с длиной строки.

Алгоритм проще понять из фрагмента решения на языке Паскаль — функции, определяющей, входит ли строка a как подпоследовательность в строку b.

```
function subseq(const a, b : string) : boolean;
var
        i, j : integer;
begin
        i := 0;
        for j := 1 to length(b) do
                if (i < length(a)) and (b[j] = a[i + 1]) then
                        inc(l);

        result := i = length(a);
end;
```

Теперь напишем решающую исходную задачу программу с использованием написанной функции.

```
read(n, b);
sb := IntToStr(b);
ans := 0;
for i := 1 to n do begin
        read(a);
        sa := IntToStr(a);
        if (subseq(sb, sa)) then begin
                inc(ans);
        end;
end;
writeln(ans);
```

Время работы алгоритма есть $O(n)$.

Задача E. Раскраска кубиков

Автор задачи: Владимир Ульянцев
Авторы разбора: Роман Сатюков, Федор Царев

Для решения этой задачи необходимо выписать явные формулы для ответов. Упорядочим размеры заданного параллелепипеда так, чтобы выполнялись неравенства $w \le h \le l$. Заметим, что формулы зависят от того, сколько из размеров параллелепипеда равны единице.

	$w,h,l>1$	$w=1,\ h,l>1$	$w=h=1,\ l>1$	$w=h=l=1$
$k=0$	$(w-2)\cdot(h-2)\cdot(l-2)$	0	0	0
$k=1$	$2\cdot((w-2)\cdot(h-2)+$ $(l-2)\cdot(h-2)+$ $(w-2)\cdot(l-2))$	0	0	0
$k=2$	$4\cdot(w+h+l-6)$	$(h-2)\cdot(l-2)$	0	0
$k=3$	8	$2\cdot(h+l-4)$	0	0
$k=4$	0	4	$l-2$	0
$k=5$	0	0	2	0
$k=6$	0	0	0	1

Эту задачу можно решить и более «программистским» методом. Введем систему координат так, чтобы координаты центров всех кубиков имели вид (x,y,z), где $1 \le x \le w$, $1 \le y \le h$, $1 \le z \le l$. Заведем трехмерных массив $a[x][y][z]$ — число покрашенных граней у кубика с координатами центра (x,y,z). Элементы этого массива вычисляются следующим образом: надо перебрать кубики на всех шести гранях параллелепипеда и прибавить к соответствующим элементам массива a единицу.

Фрагмент решения на языке Pascal:

```
// Грани z=1 и z=l
for i := 1 to w do
    for j := 1 to h do begin
        inc(a[i][j][1]);
        inc(a[i][j][l]);
    end;

// Грани y=1 и y=h
for i := 1 to w do
    for j := 1 to l do begin
        inc(a[i][1][j]);
        inc(a[i][h][j]);
```

```
    end;

// Грани x=1 и x=w
for i := 1 to h do
    for j := 1 to l do begin
        inc(a[1][i][j]);
        inc(a[w][i][j]);
    end;
```

После чего циклами по всем кубикам можно вычислить ответ:

```
ans := 0;
for i := 1 to w do
    for j := 1 to h do
        for p := 1 to l do begin
            if (a[i][j][p] = k) then
                inc(ans);
        end;
```

Время работы это алгоритма составляет $O(w \cdot h \cdot l)$.

Задача F. Железные дороги

Автор задачи: Федор Царев
Автор разбора: Федор Царев

Для решения этой задачи необходимо аккуратно реализовать системы вычисления стоимости, описанные в условии. Для удобства, напишем функции, реализующие это вычисление для каждой из систем.

Для того, чтобы вычислить стоимость проезда по зонной системе, достаточно знать номера зон, в которых находятся станции, и стоимость одной зоны. Номера зон, в которых находятся станции, будут передаваться в функцию как параметры, а стоимость c_1 одной зоны будет храниться в глобальной переменной.

```
function zoneCost(zone1, zone2 : integer) : integer;
begin
  if (zone1 = zone2) then begin
    zoneCost := c1;
  end else begin
    zoneCost := c1 * abs(zone1 - zone2);
  end;
end;
```

Функция вычисления стоимости для покилометровой системы принимает в качестве аргументов расстояния от станций до вокзала.

```
function kilometerCost(dist1, dist2 : integer) : integer;
var
  dist : integer;
begin
  dist := abs(dist1 - dist2);
  if (dist < R) then begin
    kilometerCost := c2 * dist;
  end else begin
    kilometerCost := c3 * dist;
```

```
  end;
end;
```

Величины R, c_2 и c_3 здесь также хранятся в глобальных переменных.

Теперь с помощью этих функций можно написать программу полностью.

```
read(n);
for i := 1 to n do begin
  read(d[i]);
end;
for i := 1 to n do begin
  read(z[i]);
end;
read(s, t);
read(c1, c2, c3, R);
writeln(zoneCost(s, t));
writeln(kilometerCost(s, t));
```

Задача G. Фигурное катание

Автор задачи: Федор Царев
Автор разбора: Антон Ахи

Прежде всего в этой задаче необходимо выяснить, какие же оценки будут вычеркнуты. Для этого необходимо найти максимум и минимум в массиве чисел-оценок:

```
min := 11;
max := 0;
for i := 1 to n do begin
  if (min > a[i]) then begin
    min := a[i];
  end;
  if (max < a[i]) then begin
    max := a[i];
  end;
end;
```

Здесь `a` — массив, содержащий оценки.

После этого найдем сумму элементов массива и вычислим итоговую оценку по формуле

$$a_{res} = \frac{\sum_{i=1}^{n} a_i - \max - \min}{n-2}$$

```
sum := 0;
for i := 1 to n do begin
  inc(sum, a[i]);
end;
writeln((sum - min - max) / (n - 2));
```

Задача H. Игра в слова

Автор задачи: Максим Буздалов
Автор разбора: Антон Ахи

Как можно понять из условия задачи, необходимо проверить вхождение каждого слова из списка как подстроки в каждую строку и в каждый столбец таблицы. Для этого можно перебирать все возможные позиции начала слова в таблице и направление написания, а затем посимвольно сравнивать символы искомого слова с символами в таблице. Такой алгоритм требует $O(n \cdot m \cdot k \cdot \text{maxlen})$, где maxlen — максимальная длина слова. При ограничениях, указанных в усвловии, на максимальном по размеру тесте этому алгоритму требуется менее 15 миллионов операций, что укладывается в ограничение по времени.

Фрагмент программы на языке Паскаль:

```
for w := 1 to k do begin
  readln(s);
  found := false;
  for x := 1 to n do
    for y := 1 to m do begin
      i := 0;
      while (x + i <= n) and (i < length(s)) and (s[i + 1] = a[x + i][y]) do
        inc(i);
      if (i = length(s)) then
        found := true;
      i := 0;
      while (y + i <= m) and (i < length(s)) and (s[i + 1] = a[x][y + i]) do
        inc(i);
      if (i = length(s)) then
        found := true;
    end;
  if (found) then
    writeln('YES')
  else
    writeln('NO');
end;
```

Здесь `a : array [1..50] of string` — массив строк, содержащий таблицу.

Также задача поиска образца в строке может быть решена за линейное время от суммы их длин алгоритмом Кнута-Морриса-Пратта, о котором можно прочесть в [**?**].

Применяя данный алгоритм ко всем парам слово-строка и слово-столбец, получаем алгоритм, работающий за $O(k \cdot (n \cdot (\text{maxlen} + m) + m \cdot (\text{maxlen} + n)) = O(k \cdot (n + m) \cdot \text{maxlen} + k \cdot n \cdot m)$. Это решение задачи более эффективно по времени.

Фрагмент программы на языке Паскаль:

```
  readln(n, m);
  for i := 1 to n do
    readln(a[i]);
  readln(k);
  for x := 1 to k do begin
    readln(s);
    found := false;
    j := 0;
    for i := 2 to length(s) do begin
      while (j > 0) and (s[i] <> s[j + 1]) do
        j := p[j];
      if (s[i] = s[j + 1]) then
```

```
        inc(j);
      p[i] := j;
    end;
    for y := 1 to n do begin
      j := 0;
      for i := 1 to length(a[y]) do begin
        while (j > 0) and (a[y][i] <> s[j + 1]) do
          j := p[j];
        if (a[y][i] = s[j + 1]) then
          inc(j);
        if (j >= length(s)) then
          found := true;
      end;
    end;
    for y := 1 to m do begin
      j := 0;
      for i := 1 to n do begin
        while (j > 0) and (a[i][y] <> s[j + 1]) do
          j := p[j];
        if (a[i][y] = s[j + 1]) then
          inc(j);
        if (j >= length(s)) then
          found := true;
      end;
    end;
    if (found) then
      writeln('YES')
    else
      writeln('NO');
  end;
```

Здесь `a` — таблица, `s` — текущие слово, поиск которого будет осуществляться, `p` — значения *префикс-функции* для искомого слова.

Список литературы

[1] *Кормен Т., Лейзерсон Ч., Ривест Р.* Алгоритмы: Построение и Анализ. — М.: МЦНМО, 1999.