Математика для школьников 7 – 11 класса (заочный тур)
Решение задачи 6. ДНК для хранения информации: от теории к практике

Рис. 1

1. Поскольку существуют 4 буквы нуклеотида, то один нуклеотид будет кодировать 2 бита информации, 192 нуклеотида кодируют 192·2 = 384 бит информации.

   Сколько описанных в условии строк можно закодировать таким способом? - Столько же, сколько чисел можно закодировать 8·2 = 16 битами, т.е. суммарно 2^{16} = 65536 строк (или по-другому: максимальный номер строки будет 1111111111111111_2 = 65535; всего строк, с учетом нулевой, будет 65536). Следовательно, объем информации (в МБ) составит 384·65536/8/1024/1024 = **3 Мегабайта**.

2. Файл содержит **19** строк, из которых 18 полных (содержат, как указано в условии, 200 символов нуклеотидов, визуально их длина одинакова) и одна неполная из 76 нуклеотидов, следовательно, запись файла состоит из 200·18 + 56 = **3656** символов нуклеотидов, из которых информацию кодируют 3656 – 19·8 = 3504. Поскольку каждый символ кодирует 2 бита информации, то исходный файл имеет размер 3504·2/8 = **876 байт**.

3. Столько же, сколько вариантов сопоставить значения 1 бита (00 01 10 11) нуклеотидам (A C G T), т.е. 4! = **24**.

   Расшифруем код, зная, что в адресах строк закодированы цифры от нуля до 18 (19 строк). Самая короткая последовательность – это, очевидно, самая последняя строка. Поскольку всего 19 строк, то ее номер 18, следовательно:

   $$18 = 10010_2 \Rightarrow 00\ 00\ 00\ 00\ 00\ \underline{01\ 00\ 10}_2 \Leftrightarrow G\ G\ G\ G\ G\ \underline{C\ G\ T}$$

   Следовательно, C = 01, G = 00 и T = 10 (A = 11 методом исключения).

4. Алгоритм. Читаем последовательно строки из файла **image.txt**, раскодируем согласно найденной в предыдущем пункте таблице соответствия, разделяем строку на номер и данные. Записываем данные в ячейку массива с номером строки. После прочтения всех строк перебираем последовательно строки массива, записывая данные в двоичном виде в файл **image.png**, который можно открыть в любой программе, умеющей читать распространенные графические файлы. Это – упрощенный логотип 12-й олимпиады (см. заглавный рисунок).

Исходный код программы (PascalABC.NET http://pascalabc.net/):

```
var
  { для упрощения программы считаем, что строк не более 256 }
  P : array[0..255] of string;
  txt : text;
  png : file;
  s : string;
  {переменная с двоичным кодом, соответствующим последовательности нуклеотидов}
  b : byte;

procedure decode(nuc: char);
begin
{ (b shl 2) к двоичной записи b, добавляет справа 2 нуля, затем мы по сути «добавляем» (справа) двоичный код нуклеотида nuc. Важно отметить, что поскольку переменная b имеет тип byte, то в ней не может "храниться" более 1 байта = 8 бит, т.е. 4 нуклеотидов; последующее «добавление» нуклеотидов по сути "затирает" двоичный код крайнего нуклеотида слева и добавляет двоичный код нового нуклеотида справа: выполнение decode для 4-х нуклеотидов полностью «вытесняет» из нее информацию о предыдущих 4-х нуклеотидах, чем мы далее воспользуемся чтобы не обнулять каждый раз переменную b (поскольку информация о номерах строк и данные состоят из целого количества байт, следовательно закодированные номера строк и информация в строках записаны кратным 4 числом нуклеотидов). }
 case nuc of
  'A': b := (b shl 2) + 3; {A<=>11 т.е 3}
  'C': b := (b shl 2) + 1; {C<=>01 т.е 1}
  'G': b := (b shl 2) + 0; {G<=>00 т.е 0}
  'T': b := (b shl 2) + 2; {T<=>10 т.е 2}
 end;
end;

begin
    { построчно читаем файл image.txt и создаем массив P, в котором номер элемента равен номеру строки, а его значение - кодирующая данные последовательность }
assign(txt, 'image.txt');
reset(txt);
while not eof(txt) do
 begin
  readln(txt, s);          { читаем построчно 'image.txt' }
  for var n := 5 to 8 do decode(s[n]); { декодируем 4 нуклеотида номера }
  P[b] := copy(s, 9, 200); { заносим в массив P строку под декодированным №b }
 end;
assign(png, 'image.png');
rewrite(png);
       { перебираем массив P пока в нем есть непустые строки }
foreach s in P do
  for var n:= 1 to length(s) do  { перебираем символы нуклеотидов строки }
    begin
     decode(s[n]);
     {"заполненную" новой четверкой нуклеотидов переменную b пишем в файл}
     if n mod 4 = 0 then write(png, b);
    end;
end.
```