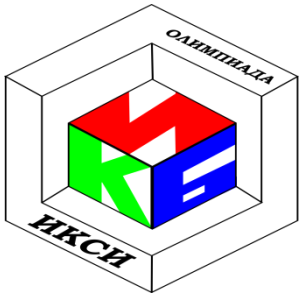

VIII ОЛИМПИАДА ШКОЛЬНИКОВ ПО ИНФОРМАТИКЕ И КОМПЬЮТЕРНОЙ БЕЗОПАСНОСТИ

11 КЛАСС

Задача 1. RAR

В результате анализа перехваченного сетевого трафика с компьютера Пятачка, Винни Пух обнаружил файл, который подозрительно похож на zip-архив. Архив intercept.zip оказался защищен паролем и содержит один единственный файл. Винни Пух считал Пятачка своим лучшим другом и поэтому очень сильно хотел узнать, что Пятачок может от него скрывать. Помогите Винни Пуху прочитать содержимое файла защищённого паролем архива, если известно, что:

1. Пятачок при создании файла использовал клавиатуру, на которой присутствуют только цифры от 0 до 9 включительно.
2. Для проверки целостности файлов zip-архив использует алгоритм CRC32, реализация которого приведена ниже, где:
   - *data* – указатель на буфер с данными длиной *length*, для которых считается результат работы алгоритма *CRC32*;
   - возвращаемое значение – результат работы алгоритма *CRC32*.

```
unsigned int crc32(unsigned char*data, int length)
{
      unsigned m_crc32;
      unsigned table[256];
      const unsigned CRC_POLY = 0xEDB88320;
      unsigned i, j, r;
      for (i = 0; i < 256; i++)
      {
      for (r = i, j = 8; j; j--)
            r = r & 1? (r>> 1) ^ CRC_POLY: r >> 1;
      table[i] = r;
      }
      m_crc32 = 0;

      const unsigned CRC_MASK = 0xD202EF8D;
      unsigned crc = m_crc32;
      while (length--)
      {
      crc = table[(unsigned char)(crc) ^ *data++] ^ crc>> 8;
      crc ^= CRC_MASK;
      }
      return crc;
}
```

Ход решения

Заметим, что файл, содержащийся в zip-архиве, имеет относительно небольшой размер (6 байт). Известно, что алфавит, символами которого заполнен данный файл, также сильно ограничен (10 символов). Следовательно, содержимое файла является одним из элементов множества *A* всех размещений с повторениями из 10 элементов по 6. Мощность множества *A* равна $10^6 = 1\,000\,000$. Поэтому, для того, чтобы найти содержимое файла, необходимо написать программу, которая достаточно быстро будет вычислять результат работы алгоритма *CRC32* на всех элементах множества *A* до тех пор, пока этот результат не совпадёт со значением, извлеченным из архива.

Задача 2. SMS

Молодой студент хотел бы порадовать любимую стихотворением Пушкина "Я помню чудное мгновение...", передав его набором SMS-сообщений. Однако, как у всех студентов у него ограниченные денежные ресурсы, поэтому хотелось бы передать стихотворение как можно меньшим количеством сообщений.

Являясь студентом технического ВУЗа, наш герой сообразил, что в 8-битной кодировке можно передавать сообщения длиной до 140 символов, тогда как в кодировке Unicode (2 байта) всего лишь 70, но можно использовать русские буквы.

Помогите студенту сэкономить бюджет, предложив вариант передачи стихотворения, если в одном сообщении могут быть либо символы 8 битной кодировки, либо Unicode.

Комментарий

По условию задачи считаем, что подруга студента осведомлена о различных схемах кодирования, но не признаёт транслитерацию (запись русских слов английскими буквами). Можно использовать символы латинского алфавита и цифры (кодировка 8 бит) или символы русского алфавита (кодировка Unicode). Также в переписке необходимо передать параметры, необходимые для декодирования сообщения.

К примеру, если студент хотел бы заменить слово "без" на цифру "1", то в последнем сообщении он должен передать: "Алфавит: без-1, как-2, ...". Позволяется передавать в 8 битной кодировке символы русского алфавита, совпадающие по начертанию с латинским (буквы а, о, н, р и пр.).

Алфавит 8-битной кодировки: aA, bB, cC, dD, eE, fF, gG, hH, iI, jJ, kK, lL, mM, nN, oO, pP, qQ, rR, sS, tT, uU, vV, wW, xX, yY, zZ, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, точка (.), запятая (,), двоеточие (:), восклицательный знак (!), вопросительный знак (?).

Алфавит кодировки Unicode: любые известные печатные символы.

Приведённый стих занимает 696 символов в кодировке Unicode (10 sms).

Я ПОМНЮ ЧУДНОЕ МГНОВЕНЬЕ…

Я помню чудное мгновенье:
Передо мной явилась ты,
Как мимолетное виденье,
Как гений чистой красоты.

В томленьях грусти безнадежной
В тревогах шумной суеты,
Звучал мне долго голос нежный
И снились милые черты.

Шли годы. Бурь порыв мятежный
Рассеял прежние мечты,
И я забыл твой голос нежный,
Твой небесные черты.

В глуши, во мраке заточенья
Тянулись тихо дни мои
Без божества, без вдохновенья,
Без слез, без жизни, без любви.

Душе настало пробужденье:
И вот опять явилась ты,
Как мимолетное виденье,
Как гений чистой красоты.

И сердце бьется в упоенье,
И для него воскресли вновь
И божество, и вдохновенье,
И жизнь, и слезы, и любовь.

Решение

Наилучший результат студент может быть получен при использовании символов русского алфавита, совпадающих по начертанию с латинским, и цифровых кодов символов для оставшихся букв. В последнем сообщение требуется передать таблицу кодирования. При этом будет затрачено 6 сообщений:

1 2ОМН3 4У5НОЕ М6НОВЕН7Е...
1 2омн3 4у5ное м6новен7е:
2ере5о мно8 1в9897ас7 9592,
Как м98мо97е95ное в985ен7е,
Как 6ен988 498с95о8 красо9592.

В 95ом97ен71х 6рус9598 94е99на5е93но8
В 95ревоб6ах 91умно8 суе9592,
99ву4а97 мне 5о976о 6о97ос не93н928
98 сн989798с7 м989792е 4ер9592.

919798 6о592. 94ур7 2ор92в м195е93н928
Рассе197 2ре93н98е ме49592,
98 1 99а949297 95во8 6о97ос не93н928,
95во8 не94есн92е 4ер9592.

В 697у9198, во мраке 99а95о4ен71
951ну9798с7 9598хо 5н98 мо98
94е99 94о93ес95ва, 94е99 в5охновен71,
94е99 с97е99, 94е99 939899н98, 94е99 97394в98.

5у91е нас95а97о 2ро94у935ен7е:
98 во95 о21957 1в9897ас7 9592,
Как м98мо97е95ное в985ен7е,
Как 6ен988 498с95о8 красо9592.

98 сер596е 947е95с1 в у2оен7е,
98 5971 не6о воскрес9798 внов7
98 94о93ес95во, 98 в5охновен7е,
98 939899н7, 98 с97е9992, 98 97394ов7.

Я1, П2, Ю3, Ч4, Д5, Г6, Ь7, й8, ш91, ы92, ж93, б94, Т95, ц96, л97, и98, з99

<u>Задача 3.</u> Ошибка

В ходе разработки сложного программного проекта произошел сбой в системе контроля версий. В результате в коде функции heapSort произошли изменения. Известно, что в качестве параметров подается массив целых чисел и размер массива этого массива. В результате выполнения функции должен быть получен отсортированный массив.

Найдите ошибку, которая была внесена в исходный код в результате сбоя.

C	Pascal

`void downHeap(int a[],long k,long n)`
`{`
`int new_elem;`
`  long child;`
`new_elem = a[k];`
`  while(k <= n/2)`
`  {`
`    child = 2*k;`
`    if(child<n&&a[child]<a[child+1])`
`    child++;`
`    if( new_elem>= a[child] )`
`    break;`
`    a[k] = a[child];`
`    k = child;`
`  }`
`  a[k] = new_elem;`
`}`
`void heapSort(int a[], long size)`
`{`
	`procedure heapSort`
	`    (var a:array[0..n] of integer;`
	`var n: integer;)`
	`var`
	`  i: integer;`
	`  temp: integer;`
	
	`procedure downHeap`
	`     (var a:array[0..n] of integer;`
	`var n: integer;`
	`var k: integer);`
	`var`
	`new_element: integer;`
	`  child: integer;`
	`  label 1;`
	`begin`
	`new_element:=a[k];`
	`  while k <= n/2 do`
	`  begin`

```
  long i;
int temp;
  for(i=size/2-1; i >= 0; i--)
     downHeap(a, i, size-1);
  for(i=size-1; i > 0; i--)
  {
    temp=a[i];
    a[i]=a[0];
    a[0]=temp;
    downHeap(a, 0, i);
  }
}
```

```
   child:=2*k;
   if((child<n)&&(a[child]<a[child+1]))
   then goto 1;
   a[k]:=a[child];
   k:=child;
  end;
  1:a[k]=new_element;
end;

begin
  for i:=n/2-1 downto 0 do
downHeap(a,n,i);
  for i:=n-1 downto 0 do
  begin
    temp:=a[i];
    a[i]:=a[0];
    a[0]:=temp;
downHeap(a,0,i)
  end;
end;
```

Решение

Пронумеруем все строки алгоритма и структурируем его.

```
(1)void downHeap(int a[], long k, long n)
(2){
(3)  int new_elem;
(4)  long child;
(5)  new_elem = a[k];
(6)  while(k <= n/2)
(7)  {
(8)    child = 2*k;
(9)    if( child < n && a[child] < a[child+1] )
(10)    child++;
(11)    if(new_elem>= a[child] )
(12)      break;
(13)    a[k] = a[child];
(14)    k = child;
(15)  }
(16)  a[k] = new_elem;
(17)}
(18)void heapSort(int a[], long size)
(19){
(20)  long i;
(21)  int temp;
(22)  for(i=size/2-1; i >= 0; i--)
(23)    downHeap(a, i, size-1);
(24)  for(i=size-1; i > 0; i--)
(25)  {
(26)    temp=a[i];
(27)    a[i]=a[0];
(28)    a[0]=temp;
(29)downHeap(a, 0, i);
(30)  }
(31)}
```

Рассматриваемый исходный код представляет собой реализацию алгоритма пирамидальной сортировки.

Алгоритм на первом этапе строит пирамиду, которая имеет следующую структуру. На верху пирамиды находится один элемент. У каждого элемента есть два потомка, которые меньше или равны родителя. Каждая цепочка от вершины до основания имеет длину m или m+1. Построенная пирамида укладывается в массив. Для элемента массива с номером *i* потомки имеют индексы 2*i*+1 и 2*i*+2, а родитель – целая часть от деление *i* пополам.

На втором этапе алгоритм заменяет вершину пирамиды (максимальный элемент) на последний элемент в массиве. Далее на всех элементах пирамиды, кроме последнего, происходит построение пирамиды (первый этап). Затем заменяется вершина вновь построенной пирамиды на второй элемент с конца. Количество таких повторений равно количеству элементов в массиве.

Заметим, что в строке 29 после замены вершины пирамиды на очередной правый элемент происходит перестроение пирамиды с использованием замененного элемента *i*, что неверно. На некотором шаге получится ситуация, когда при применении функции downHeap *i*-ый элемент переместится в вершину пирамиды, что нарушит сортировку. Для правильной работы алгоритма необходимо перестроить пирамиду для элементов от 0 до *i*-1.

Правильная реализация с комментариями представлена ниже.

```
(1)void downHeap(int a[], long k, long n)
(2){
   //  процедура просеивания следующего элемента
   //  До процедуры: a[k+1]...a[n]  - пирамида
//  После:  a[k]...a[n]  - пирамида
(3)  int new_elem;
(4)  long child;
(5)  new_elem = a[k];
    // пока у a[k] есть дети
(6)  while(k <= n/2)
(7)  {
(8)    child = 2*k;
       //  выбираем большего сына
(9)    if( child < n && a[child] < a[child+1] )
(10)    child++;
(11)    if(new_elem>= a[child] )
(12)      break;
        // переносим сына наверх
(13)    a[k] = a[child];
(14)    k = child;
(15)  }
(16)  a[k] = new_elem;
(17)}
(18)void heapSort(int a[], long size)
(19){
(20)  long i;
(21)  int temp;
      // строим пирамиду
(22)  for(i=size/2-1; i >= 0; i--)
(23)    downHeap(a, i, size-1);
     // теперь a[0]...a[size-1] пирамида
(24)  for(i=size-1; i > 0; i--)
(25)  {
        // меняем первый с последним
(26)    temp=a[i];
```

```
(27)     a[i]=a[0];
(28)     a[0]=temp;
         // восстанавливаем пирамидальность a[0]...a[i-1]
(29)downHeap(a, 0, i-1);
(30)   }
(31)}
```

Задача 4. Арифметика больших чисел

Напишите программу, определяющую последнюю цифру числа a^{b^c}, где a, b, c–числа типа int, большие нуля.

Решение

Для того, чтобы найти последнюю цифру *x* числа a^{b^c} надо найти остаток от деления его на 10:

$$x = a^{b^c} \bmod 10$$

Представим число a в виде $a = m \cdot 10 + r$. Для любого натурального $n\, a^n = ((m \cdot 10)^n + n(m \cdot 10)^{n-1} r + \ldots + m \cdot 10 \cdot r + r^n$. Поскольку все слагаемые суммы делятся нацело на 10, последняя цифра числа a^n будет определяться последней цифрой числа r^n. То есть искомая последняя цифра $x = r^{b^c} \bmod 10$.

Нетрудно заметить, что при возведении в степень любого числа возникают циклические последовательности последних цифр. Таблице 1 приведены последовательности последних цифр.

Таблица 1

Цифры / степень	0	1	2	3	4	5	6	7	8	9
1	0	1	2	3	4	5	6	7	8	9
2	0	1	4	9	6	5	6	9	4	1
3			8	7	4			3	2	9
4			6	1				1	6	
5			2	3				7	8	

Поскольку последняя цифра повторяется с периодом *p*,очевидно, что $x = r^{b^c} \bmod 10 = (r)^y \bmod 10$, где $y = \begin{cases} b^c \bmod p & \text{если } b^c \bmod p \neq 0 \\ p & \text{если } b^c \bmod p = 0 \end{cases}$.

То есть необходимо возводить число a не в степень b^c, а в степень *y*. Рассмотрим отдельно два случая:

1. $b^c \bmod p \neq 0$.

$$y = \begin{cases} (b \bmod p)^c \bmod p & \text{если } b \bmod p \neq 0 \\ p & \text{если } b \bmod p = 0 \end{cases}$$

2. $b^c \bmod p = 0$.

$$y = \begin{cases} (b \bmod p)^c \bmod p & \text{если } b \bmod p \neq 0 \\ p & \text{если } b \bmod p = 0 \end{cases}$$

В итоге

$$y = \begin{cases} p & \text{если } b \bmod p = 0 \\ (b \bmod p)^c \bmod p & \text{в остальных случаях} \end{cases}$$

Поскольку p принимает 4 значения, то можно описать все варианты:

1. $P = 1$ (последние цифры 0, 1, 5, 6)
   В этом случае $y = 1$ и $x = r$.
2. $P = 2$ (последние цифры 4, 9)
   1) Если $b \bmod 2 = 0 \Rightarrow$ $y = 2$ и $x = r^2 \bmod 10$.
   2) Если $b \bmod 2 = 1 \Rightarrow$ $y = (b \bmod 2)^c = 1$ и $x = r$.

3. $P = 4$ (последние цифры 2, 3, 7, 8).
   Если:

1) $b \bmod 4 = 0 \Rightarrow y = 4$ и $x = r^4 \bmod 10$.
2) $b \bmod 4 = 1 \Rightarrow y = 1$ и $x = r$.
3) $b \bmod 4 = 2 \Rightarrow y = \begin{cases} 2 & если\ c = 1 \\ 4 & если\ c > 1 \end{cases}$, следовательно $x = r^y \bmod 10$.
4) $b \bmod 4 = 3 \Rightarrow$
   $y = (2+1)^c \bmod 4 = (2^c + \ldots c \cdot 2 + 1) \bmod 4 = (2 \cdot (c \bmod 4) + 1) \bmod 4$
   и $x = r^y \bmod 10$.

Алгоритм возведения в степень

1. Находим последнюю цифру числа a: $r = a \bmod 10$.
2. Находим p период повторения последней цифры при возведении в степень числа r.
3. Если p = 1 , то $x = r$ – искомая цифра и переходим к шагу 6
4. Находим остаток от деления b на p: $y = \begin{cases} p & \text{если } b \bmod p = 0 \\ (b \bmod p) & \text{в остальных случаях} \end{cases}$
5. Если

1.1 $y = 1$, то $x = r$– искомая цифра и переходим к шагу 6
1.2 $y = 2$ если $p = 2$ то $x = r^2 \bmod 10$ – искомая цифра,
иначе если $c = 1$ то $x = r^2 \bmod 10$ – искомая цифра,
иначе – $x = r^4 \bmod 10$ – искомая цифра.
переходим в к шагу 6.
1.3 y =3, то $y = (2 \cdot (c \bmod 4) + 1) \bmod 4$ $x = r^y \bmod 10$
переходим в к шагу 6.
1.4 $y = 4$, то $x = r^4 \bmod 10$. Переходим в шагу 6.
6. Вывод значения x. Окончание работы алгоритма.

<u>Задача 5.</u> Бал

Перед отъездом со своими дочерями на королевский бал злая мачеха приказала к утру Золушке отделить «хорошие» гиперссылки от «плохих» внутри файла links.txt (каждая гиперссылка

содержится в отдельной строке этого файла). Чтобы отличить «хорошие» гиперссылки от «плохих», мачеха дала ей текстовый файл rules.txt с правилами, описывающими плохие гиперссылки. Каждое правило находится в отдельной строке и может принимать одну из двух форм:

– http://www.nsa.*, где * ноль и более символов;
– *.nsa.gov, где * ноль и более символов.

К утру мачеха ждет от Золушки файл bad_links.txt с выделенными «плохими» гиперссылками из файла links.txt. Напишите программу, которая поможет Золушке выделить «плохие» гиперссылки и попасть на бал при условии, что он начнется через 5 секунд.

Ход решения

Пусть в файле links.txt содержится M записей, а в файле rules.txt содержится N записей. Прямая проверка каждой гиперссылки с каждым правилом дает алгоритм сложности O(N*M), что не позволяет уложиться в ограничение 5 секунд при заданных файлах links.txt и rules.txt.

Для ускорения проверки каждой гиперссылки предлагается выполнить предварительную обработку файла rules.txt разделив все правила на два множества: заканчивающиеся на «*» и начинающиеся с«*».

Из всех правил, заканчивающихся «*», производится удаление конечной «*» и перевод результата к нижнему регистру. Полученные таким образом правила сортируются в лексикографическом порядке.

Из всех правил, начинающихся с «*» производится удаление начальной «*», перевести к нижнему регистру и записать в обратном порядке. Полученные таким образом правила сортируются в лексикографическом порядке.

Теперь для проверки того, является ли очередная гиперссылка из файла links.txt «плохой» достаточно привести ее к нижнему регистру и проверить с помощью алгоритма бинарного поиска ее наличие в первом множестве (при этом критерием равенство будет то, что гиперссылка начинается с преобразованного правила). Если гиперссылка не была найдена в первом множестве, то она записывается в обратном порядке, приводится к нижнему регистру и ищется аналогично предыдущему случаю во втором множестве. Если гиперссылка не была найдена ни в первом и ни во втором множестве, то она является «хорошей».