Sablon

Problema se rezolvă relativ simplu ținând cont de următoarele observații:

1. Deoarece întregul mesaj a fost codificat prin 4 rotiri ale textului, este clar că la o poziționare a textului sub şablon pot fi citite Lung(Mesaj)/4 caractere, deci întregul mesaj are 4*NumarGăuri caractere
2. Ca urmare a observației de la punctul 1, mesajul poate fi împărțit exact în 4 şiruri de lungimi egale `Mesaj1,...,Mesaj4`
3. Dacă o gaură se află în poziția `T[i,j]` din şablon, ei îi corespund pozițiile
   `T[j,N-i+1]` la rotire cu 90 grade
   `T[N-i+1,N-j+1]` la rotire cu 180 grade
   `T[N-j+1,i]` la rotire cu 270 grade
   de unde deducem că nu e nevoie să rotim textul!!!
4. Dacă lungimea unui şir este L4 (vezi în sursă), este suficient să parcurgem numai primul din cele 4 şiruri cu un `Index`. Atunci, parcurgând textul care ascunde mesajul, în poziția `(i, j)` există o gaură în şablon dacă şi numai dacă toate cele 4 caractere
   `Mesaj1[Index], Mesaj2[Index], Mesaj3[Index], Mesaj4[Index]`
   coincid cu cele 4 caractere obținute prin rotire (vezi observația 3)
5. "Cel mai bun pseudocod este... PASCAL-ul", deci:

```
Program DeterminareSablon;
Const DimMax = 50;           { lungimea maxima a textului   }
Type Gauri  = Array [1..DimMax, 1..DimMax] Of Boolean;
     Textul = Array [1..DimMax, 1..DimMax] Of Char;

Var G    : Gauri;
    T    : Textul;
    Mesaj: Array [1..4] Of String;
    N, L4: Integer;

Procedure Afisare;
Var i, j: Integer;
    f   : Text;
Begin
   Assign(f, 'sablon.out'); ReWrite(f);
   For i := 1 To N Do
      Begin
        For j := 1 To N Do
          If G[i,j] Then Write(f, 'O')
          Else Write(f, 'X');
        Writeln(f);
      End;
   Close(f);
End;

Procedure Prelucrare;
Var Index, i, j, Lung, N2: Integer;
Begin
  FillChar(G, SizeOf(G), False);
  Index := 1;        {parcurg cu Index pana la length(mesaj)/4-pentru 4 rotiri}
  For i := 1 To N Do
    For j := 1 To N Do
      Begin
        If (Mesaj[1][Index] = T[i,j]) And         {primul caracter}
           (Mesaj[2][Index] = T[j,N-i+1]) And     {rotit cu 90ø}
           (Mesaj[3][Index] = T[N-i+1,N-j+1]) And {rotit cu 180ø}
           (Mesaj[4][Index] = T[N-j+1,i])         {rotit cu 270ø}
        Then          {daca toate 4 caractere coincid}
          Begin
            G[i,j] := True;   {e gaura}
            Inc(Index);       {trec la alt caracter}
            If Index > L4 Then exit;  {div 4 pentru 4 rotiri}
          End;
      End
End;
```

```
Procedure Citire;
Var f   : Text;
    i, j: Integer;
    Lung: Integer;
    M   : Array[1..1000] Of Char;
Begin
  Assign(f, 'sablon.in'); reset (f);
  Lung := 0;
  FillChar(M, SizeOf(M), #0);
  While Not Eoln(f) Do          {citesc mesajul in sirul de caractere}
    Begin                       {determinandu-i lungimea}
      Inc(Lung);
      Read(f, M[Lung]);
    End;
  ReadLn(f, N);                 {citesc N}
  For i := 1 To N Do            {citesc textul in T}
    Begin
      For j := 1 To N Do
         Read(f, T[i,j]);
      Readln(f);
    End;
  Close(f);
  If Lung Mod 4<>0 Then
    WriteLn('Mesajul nu are lungimea corecta')
  Else
    Begin
      L4 := Lung Div 4;         {determin numarul de caractere la o rotire}
      For i := 1 To L4 Do       {si impart mesajul in 4 parti egale}
        Begin
          Mesaj[1][i] := M[i];
          Mesaj[2][i] := M[L4+i];
          Mesaj[3][i] := M[2*L4+i];
          Mesaj[4][i] := M[3*L4+i]
        End
    End
End;

Begin
  Citire;
  Prelucrare;
  Afisare;
End.
```

Şir 100 puncte

Notând cu r diferența dintre doi tereni consecutivi constatăm că pentru r=1 se pot construi urmatoarele submulțimi, şiruri cu proprietea cerută, de lungime 3 :
{1,2,3},{2,3,4},...,{n-2,n-1,n}. Cele de lungime superioară se construiesc adăugând elemente pe cele deja obtinute. Numărul lor va fi $\sum_{i=1}^{n-2} i$.
Similar pentru r=2 obținem următoarele submulțimi, şiruri de lungime 3:
{1,3,5},{2,4,6},...,{n-4,n-2,n}sau{n-5,n-3,n-1}funcție de paritatea lui n.Cele de lungime superioară se construiesc adăugând elemente pe acestea. Numărul lor este o sumă te tipul precedent.
Se continuă astfel până la r= n div 2, valoarea maximă a lui r.

Snipers

Explicație snipers

La început fiecărui trăgător *i* îi asociem căpetenia *i*, după care vom lua în considerare toate perechile trăgător-căpetenie şi vom elimina încrucişările razelor, la câte două perechi prin interschimbarea țintelor trăgătorilor.

Se vor face interschimbări între câte două perechi trăgător-țintă până când nu vor mai exista intersecții.
Se observă că la eliminarea unei intersecții suma distanțelor dintre trăgători şi ținte scade, ceea ce asigură finitudinea algoritmului.

```
procedure rezolva;
 var i,j,aux:word; e_int,ok:boolean;
begin
  for i:=1 to n do t[i].k:=i;{tragatorul i trage in capetenia i, pentru inceput}
  if n>1 then
  repeat
    ok:=true;{presupunem ca am gasit o combinatie valida tragator-capetenie}
    i:=1;
    repeat
      j:=i;
      repeat
        j:=j+1;
        {verifica daca traiectoria de la sniper i la tinta sa nu se intersecteaza
        cu traiectoria de la sniper j la tinta sa }
        e_int:=inter(t[i],c[t[i].k],t[j],c[t[j].k]);
        {daca da le interschimbam tintele, si notam ca nu avem o combinatie buna}
        if e_int then
          begin aux:=t[i].k;t[i].k:=t[j].k;t[j].k:=aux;ok:=false; end;
      until (j=n)or e_int;
      if not e_int then inc(i);{daca nu jeneaza pe numeni trecem al urmatorul sniper}
    until i=n;{pana terminam lista de tragatori}
  until ok;
end;
```