Central European Olympiad in Informatics

28 July – 4 August 2005 Sárospatak, Hungary
http://ceoi.inf.elte.hu

# Electric Fence

Let $m$ be the number of intersections of the fence with the road line.
One can show by induction on $m$ that the total number of disjoint regions equals $m/2+1$. Hence, it is enough to compute the number of intersections and the number of regions located on the left side of the road.
Let $u$ and $v$ be two intersections such that going along the fence from $u$ to $v$, there is no other intersecting segment. Such part of the fence is called section and $u$ and $v$ are the endpoints of this section.
If the fence segment that connects fence post $i$ and $i \, mod \, n+1$ intersects with the road line then the intersection represented by post label $i$.
Note that the sections on each side are nested. Assign nesting level number to each section, such that the level of each outermost section is 1.
One can easily prove that the number of regions on the left side equals the number of the odd level sections on the left side.
It is clear, that if $p$ and $q$ are two consecutive sections, then $p$ is on the left side and $q$ is on the right side of the road, or conversely.
First compute the leftmost intersection (that is closest to the road endpoint a), denote it by $leftmost$. Starting at the fence post $leftmost$, visit the fence posts in increasing order if $leftmost$ is on the right side of the road, otherwise in decreasing order. A section $p$ on the left side is odd level iff and only if its starting intersection is closer to the road endpoint $a$ that its ending intersection.

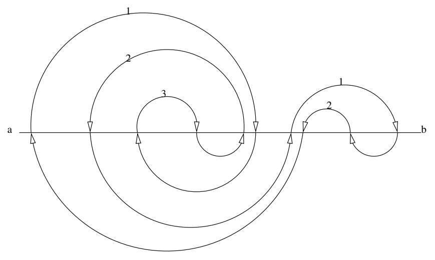

Figure 1:

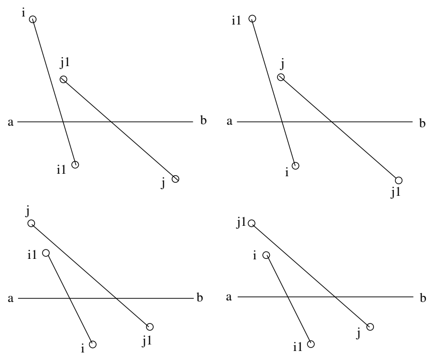

Figure 2: Situations when intersection of a segment $(i,i1)$ and the road line is closer to the road endpoint and $a$, then the intersection of a segment $(j,j1)$ and the road line. This relation can be computed using only the operation Drift.

**Complexity:**
**Time**: $O(n)$
**Memory**: $O(n)$
**Implementation**

```
Program fence;
uses lookup;
Const
  MaxN=100000;  {max # posts}
Type
  PointType=record
              left:boolean;    {post is on the left side of the road}
              cross:boolean;   {segment (i,i+1) crosses the road line}
            end;
Var
  N:longint;  {number of the posts}
  P:Array[1..MaxN] of PointType;
  i,i1:longint;
  side:boolean;
  ti,ti1:integer;
  leftpart:longint;
  leftmost:longint;
  iend:longint;
  d:integer;
  m:longint;

Function OnLeft(i,j:longint):boolean;
 {Returns true iff the intersection of the segment (i,i+1) is closer to the road
 endpoint "a" than the intercetion of the segment (j,j+1)}
var
  i1,j1:longint;
begin
  if i=n then i1:=1 else i1:=i+1;
  if j=n then j1:=1 else j1:=j+1;
  OnLeft:=(P[i1].left and(Drift(i,i1,j)<=0)and(Drift(i,i1,j1)<=0)) or
          (P[i].left and (Drift(i,i1,j)>=0)and(Drift(i,i1,j1)>=0)) or
          (P[j1].left and(Drift(j,j1,i)>=0)and(Drift(j,j1,i1)>=0)) or
          (P[j].left and (Drift(j,j1,i)<=0)and(Drift(j,j1,i1)<=0))
end {OnLeft};

Begin {Prog}
  n:=GetN;
  m:=0;      {number of intersections with the road line}
  leftmost:=0;{the leftmost intersecting segment}
  for i:=1 to n do begin
    if i=n then i1:=1 else i1:=i+1;
    ti:=Drift(n+1,n+2,i);
    ti1:=Drift(n+1,n+2,i1);
    if ti*ti1<0 then begin {segment (i,i1) intersects with the road line}
      inc(m);
      P[i].cross:=true;
      P[i].left:=ti>0;
      if leftmost=0 then
        leftmost:=i
      else begin
        if OnLeft(i,leftmost) then
          leftmost:=i;
      end;
    end else begin
      P[i].cross:=false;
      P[i].left:=ti>0;
    end;
  end {for i};

  if m=0 then begin
    if P[1].left then
      Answer(1, 0)
    else
      Answer(0, 1)
  end;

  side:=true;  {left side of the road}
  i:=leftmost;
  if P[leftmost].left then d:=-1 else d:=1;
  leftpart:=0; {number of regions on the left side}

  repeat
    iend:=i;
    repeat  {get the next crossing segment}
      iend:=iend+d;
      if iend>n then iend:=1;
      if iend<1 then iend:=n;
    until P[iend].cross;

    if side and OnLeft(i, iend) then {odd level fence section on the left side}
      inc(leftpart);
    i:=iend;
    side:=not side;
  until i=leftmost;

  Answer(leftpart, m div 2 +1-leftpart)

End.
```