

# Points on circle

## Problem

$N$  distinct points, numbered from 0 onwards, are located on a circle (in the rest of this problem all point numbers are taken **mod** $N$ ). Point  $i + 1$  is the clockwise neighbor of point  $i$ . An integer array,  $dist[0 \dots N)$ , is given such that  $dist.i$  is the distance (along the circle) between points  $i$  and  $i + 1$ . Derive a program to determine whether four of these points form a rectangle.

We adopt the same notation used in *Programming in the 1990s*<sup>1</sup> and *Programming, The Derivation of Algorithms*<sup>2</sup>: The notation of function application is the "dot" notation with name of function, followed by arguments, each separated by a dot. The notation of quantified expressions has the operator followed by the bounded variables, then a colon followed by the range for the bounded variables and ended with a colon and the actual expression. So

$$(\sum k : i \leq k < j : x_k)$$

corresponds to the more classical mathematical notation  $\sum_{k=i}^{j-1} x_k$ .

For our derivation steps in predicate calculus we will use the following notation:

$$\begin{array}{l} A \\ = \quad \{ \text{reason why A equals B} \} \\ B \\ \leq \quad \{ \text{reason why B is less than C} \} \\ C \end{array}$$

We are asked to solve  $S$  in

```
||[
  con N : int; {N ≥ 4}
    dist(i : 0 ≤ i < N) : int; {∀i : 0 ≤ i < N : dist.i > 0}
```

<sup>1</sup> Edward Cohen. *Programming in the 1990s, An Introduction to the Calculation of Programs*. Springer-Verlag, 1990

<sup>2</sup> A. Kaldewaij. *Programming, The Derivation of Algorithms*. Prentice Hall, 1990

```

var  $r : \text{bool};$ 
   $S$ 
   $\{r : r \equiv (\exists 4 \text{ points that form a rectangle})\}$ 
 $\} \} \}$ 

```

Let's first develop a more manageable postcondition. Evidently four points that form a rectangle is equivalent to two pairs of diametral opposing points. We introduce a function for the set of all indices from point  $x$  to point  $y$  in clockwise direction along the circle:

$$I : [0, \dots, N) \rightarrow [0, \dots, N) \rightarrow 2^{[0, \dots, N)}$$

$$I.x.y := \begin{cases} [x, \dots, y) & , x \leq y \\ [x, \dots, N) \cup [0, \dots, y) & , x > y \end{cases}$$

Let  $C$  be the circumference of the circle. We define function

$$f : [0, \dots, N) \rightarrow [0, \dots, N) \rightarrow \text{int}$$

$$f.x.y := C - 2(\sum i : i \in I.x.y : \text{dist}.i)$$

We want to find the number of diametral opposing pairs of points:

```

 $\} \{$ 
  con  $N : \text{int}; \{N \geq 2\}$ 
     $\text{dist}(i : 0 \leq i < N) : \text{int}; \{\forall i : 0 \leq i < N : \text{dist}.i > 0\}$ 
  var  $r : \text{int};$ 
     $S$ 
     $\{r : r = (\# x, y : 0 \leq x < N, 0 \leq y < N : f.x.y = 0)\}$ 
 $\} \}$ 

```

**Lemma 1.1.** *The function  $f$  is increasing in its first argument and decreasing in its second argument.*

*Proof.*  $f$  is increasing in its first argument:

$$\begin{aligned}
 & f.(x+1).y \\
 = & \{\text{definition of } f\} \\
 & C - 2(\sum i : i \in I.(x+1).y : \text{dist}.i) \\
 = & \{I.(x+1).y = I.x.y \setminus \{x\}\} \\
 & C - 2((\sum i : i \in I.x.y : \text{dist}.i) - \text{dist}.x) \\
 = & \{\text{definition of } f\} \\
 & f.x.y + 2\text{dist}.x \\
 > & \{\text{dist}.x > 0\} \\
 & f.x.y
 \end{aligned}$$

$f$  is decreasing in its second argument:

$$\begin{aligned}
 & f.x.(y+1) \\
 = & \{\text{definition of } f\} \\
 & C - 2(\sum i : i \in I.x.(y+1) : dist.i) \\
 = & \{I.x.(y+1) = I.x.y \cup \{y\}\} \\
 & C - 2((\sum i : i \in I.x.y : dist.i) + dist.y) \\
 = & \{\text{definition of } f\} \\
 & f.x.y - 2dist.y \\
 < & \{dist.y > 0\} \\
 & f.x.y
 \end{aligned}$$

□

Looking at the postcondition

$$\{r : r = (\# x, y : 0 \leq x < N, 0 \leq y < N : f.x.y = 0)\}$$

we define the function

$$G.a.b = (\# x, y : a \leq x < N, b \leq y < N : f.x.y = 0)$$

and we will maintain the invariants:

$$\begin{aligned}
 P_0 & : G.0.0 = r + G.a.b \\
 P_1 & : 0 \leq a \leq N \\
 P_2 & : 0 \leq b \leq N
 \end{aligned}$$

The initial values  $r, a, b := 0, 0, 0$  satisfy the invariants and

$$a = N \vee b = N \Rightarrow G.a.b = 0 \Rightarrow r = G.0.0$$

establishes the postcondition, so we can stop when  $a = N \vee b = N$ .

So far we have

```

|||
con N : int; {N ≥ 4}
    dist(i : 0 ≤ i < N) : int; {∀i : 0 ≤ i < N : dist.i > 0}
var a, b, r : int;
    a, b, r := 0, 0, 0;
do a ≠ N ∧ b ≠ N
    S
od
    {r : r = G.0.0}
|||

```

We need to increment  $a, b$  and maintain the invariants:

$$\begin{aligned}
& G.a.b \\
= & \{\text{definition of } G\} \\
& (\# x, y : a \leq x < N, b \leq y < N : f.x.y = 0) \\
= & \{\text{range split } x = a\} \\
& G.(a+1).b + (\# y : b \leq y < N : f.a.y = 0) \\
= & \{f \text{ is decreasing in second argument (1.1), and assume } f.a.b < 0\} \\
& G.(a+1).b
\end{aligned}$$

so  $f.a.b < 0 \Rightarrow G.a.b = G.(a+1).b$ . Similarly

$$\begin{aligned}
& G.a.b \\
= & \{\text{definition of } G\} \\
& (\# x, y : a \leq x < N, b \leq y < N : f.x.y = 0) \\
= & \{\text{range split } y = b\} \\
& G.a.(b+1) + (\# x : a \leq x < N : f.x.b = 0) \\
= & \{f \text{ is increasing in second argument (1.1), and assume } f.a.b > 0\} \\
& G.a.(b+1)
\end{aligned}$$

so  $f.a.b > 0 \Rightarrow G.a.b = G.a.(b+1)$ . Also for the case  $f.a.b = 0$  we have

$$\begin{aligned}
& r + G.a.b \\
= & \{\text{definition of } G\} \\
& r + (\# x, y : a \leq x < N, b \leq y < N : f.x.y = 0) \\
= & \{\text{range split } x = a\} \\
& r + G.(a+1).b + (\# y : b \leq y < N : f.a.y = 0) \\
= & \{f \text{ is decreasing in second argument (1.1), and assume } f.a.b = 0\} \\
& (r+1) + G.(a+1).b
\end{aligned}$$

Our program becomes

```

|||
  con N : int; {N ≥ 4}
    dist(i : 0 ≤ i < N) : int; {∀i : 0 ≤ i < N : dist.i > 0}
  var a, b, r : int;
  a, b, r := 0, 0, 0;
  do a ≠ N ∧ b ≠ N
    if
      □ f.a.b > 0 → b := b + 1
      □ f.a.b < 0 → a := a + 1
      □ f.a.b = 0 → a, r := a + 1, r + 1
    fi
  od
  {r : r = G.0.0}
|||

```

We cannot have  $f$  in the program text so the last thing we have to do is eliminate  $f$ . We do this by introducing a new variable  $c : \text{int}$  and

maintaining the additional invariant  $P_3 : c = f.a.b$ . Lemma 1.1 already showed us the expressions for  $f$  when the first or the second argument increase, so our final program looks like this<sup>3</sup>

```

|||
  con  $N : \text{int}; \{N \geq 4\}$ 
     $\text{dist}(i : 0 \leq i < N) : \text{int}; \{\forall i : 0 \leq i < N : \text{dist}.i > 0\}$ 
  var  $a, b, c, r : \text{int};$ 
   $a, b, c, r := 0, 0, C, 0;$ 
  do  $a \neq N \wedge b \neq N$ 
    if
       $\square c > 0 \rightarrow b, c := b + 1, c - 2\text{dist}.b$ 
       $\square c < 0 \rightarrow a, c := a + 1, c + 2\text{dist}.a$ 
       $\square c = 0 \rightarrow a, c, r := a + 1, 2\text{dist}.a, r + 1$ 
    fi
  od
   $\{r : r = G.0.0\}$ 
|||

```

<sup>3</sup> The program is bound by the function  $2N - a - b$  so it is  $O(N)$ . The solution is an example of the slope search technique.

## *Bibliography*

Edward Cohen. *Programming in the 1990s, An Introduction to the Calculation of Programs*. Springer-Verlag, 1990.

A. Kaldewaij. *Programming, The Derivation of Algorithms*. Prentice Hall, 1990.