Master in HPC

Problem Sheet 3b - smart Hilbert key

Here the problem is the same as in PS3. However, the procedure used to compute the Hilbert key is more hidden, since it is based on manipulating the bits. This exercise is not mandatory and it is left to the interested student to understand why the new procedure (Hilbert_2D) gives the same Hilbert key as in PS3. In the main code the only difference is that Hilbert_2D now requires as input integerized coordinates.

Here we give the code fragment of the main program with only the necessary modifications and the new pseudocode Hilbert_2D.

Assume again as input $levscan = 4, L = 2^{levscan}$.

Algorithm 1 new Hilbert test

1: **procedure** POINT LATTICE ▷
Global:
Require: long int $Hilbert\_2D$ ▷ new Hilbert key
Local:
Require: int ix, iy ▷ integerized coordinates
Require: int $levkey$ ▷ order H-key
Require: long int $keybit\_H$ ▷ H-key
Require: real x_to_int ▷ convert to integer
Begin

2: $levkey := levscan$ ▷ set key order, note that $levkey > levscan$ is not admitted
3: $x\_to\_key = 2^{levkey-levscan}$ ▷ set coordinate conversion $0 \leq x < L \rightarrow 0 \leq ix < 2^{levkey} - 1$
4: *print side*
5: **for** $i \leftarrow 1, npoints$ **do**
6: $ix := x\_to\_int * xpos[i]$
7: $iy := x\_to\_int * ypos[i]$
8: $keybit\_H = Hilbert\_2D(ix, iy)$
9: *print* $i,\ xpos[i],\ ypos[i],\ keybit\_H$
10: **end for**
11: **end procedure**

```
1: procedure HILBERT_2D(ix,iy,levkey)          ▷ compute the 2D H-key
2:     input integer ix, iy, levkey, levs
3:     local integer rx, ry, quad, sidelocal, swap
4:     local longinteger Hilbert_2D, num, k

5:     sidelocal := 2^levkey
6:     k := sidelocal/2
7:     num := 0
8:     levs := levkey − 1
9:     while k > 0 do                  ▷ proceed until all the levels are done
10:        rx := IBITS(ix, levs, 1)
11:        ry := IBITS(iy, levs, 1)
12:        quad = IEOR(3 ∗ rx, ry)
13:        num := num + k ∗ k ∗ quad
14:        if ry = 0 then
15:            if rx = 1 then
16:                ix = k − 1 − ix
17:                iy = k − 1 − iy
18:            end if
19:            swap = ix
20:            ix := iy
21:            iy := swap
22:        end if
23:        k := k/2                                   ▷ next k value
24:        levs : levs − 1
25:    end while                                      ▷ end k > 0
26:    return Hilbert_2D := num
27: end procedure
```