Master in HPC

Problem Sheet 4 - Quadtree

The aim of this problem is to construct a quadtree in two dimensions starting from a a given set of N points (input file 'tree.dat') with coordinates $0 \leq x_i,\ y_i < L$ with $i = 1, ...N$

For these points write a program which computes the quadtree defined by the array $pointers\_of\_tree[4, root : root + NC]$, here NC is the number of cells of the tree (typically $NC \simeq N$) and $root = N + 1$ is the cell starting address.

Let us consider the square of side length L which encloses all of the points. The quadtree is constructed by calling a recursive function (*tree_sort*) which has as input the memory address of a generic cell and the list of particles which are enclosed within the cell itself. Each call to *tree_sort* increases the tree level *itercell_tree* by a unity. The tree construction begins by a call to *tree_sort* with $itercell\_tree = 0$, and *root* as input cell together with the whole list of particles.

Before the first call the Morton keys of the particles are evaluated, setting $levorder = 10$.

A call to *tree_sort* finds first the subset of particles of the input list which fall within the boundaries of the four subcells. This is obtained by extracting for each particle the two bits of the Morton key which are distant $ndim \cdot itercell\_tree$ positions from the top most significant bits. By construction, the integer corresponding to these bits represent the subcell index j. Particles of the input list with the same subcell index j are appendend to a linked-list with header the subcell.

Let *Np_sub* be the number of points into each subcell. According to the value of *Np_sub* for the sub-quadrant j; $j = 1,\ 2,\ 3,\ 4$, the integer value *inext* of the array $pointers\_of\_tree[j, root] = inext$ takes the values :

$Np\_sub > 1$ - more than one particle in the sub-quadrant, the sub-cell needs to be further examined : $inext =$ address new cell $> root$.

$Np\_sub = 1$ - there is one particle in the sub-quadrant, : $inext = i$ address of the particle.

$Np\_sub = 0$ - there are no particles in the sub-quadrant, there is no need to open further the sub-quadrant : $inext = 0$.

At each iteration the subcells with more than one particle are opened by a new call to *tree_sort* which divides the subcell into four new subcells with size 1/2 of the parent cell. The procedure ends when there are no particles left to examine.

Write a program which reads as input file 'tree.dat' and for these points make a call to the function *tree_sort* which constructs the quadtree defined by $pointers\_of\_tree[4, root : root + NC]$.

At the end write on a file *treecell.dat*, for all the cells of the quadtree, the four coordinates (bottom, top, left,right).

Write also on a file *treecell_part.dat* all the cell coordinates which contain one particle, that is the finall cells of the quadtree.

Finally write on a file *tree_ord.dat* the particles coordinates, ordered according to their Morton key-value.

Make a plots of the particles coordinates, together with the cells of the tree.

Here are given the corresponding pseudocodes.

Algorithm 1 Quadtree test

procedure TREE STRUCTURE ▷

Global:

Require: Int $Np = 4096,\ ncells = 2 * Np,\ nbodcell = Np + ncells$

Require: int $ndim = 2,\ nsubcell = 2^{ndim}$

Require: real $pos[2, nbodcell],\ bottom[2, nbodcell],\ cellsize[nbodcell]$

Require: int $pointers\_of\_tree[nsubcell, nbodcell]$

Require: int $iback[2, Np]$

Require: int $pm1[0 : nsubcell - 1, ndim]$

Require: int $subindex[Np],\ bodlist[Np]$

Require: int $list\_of\_pointers[Np]$

Require: int $incells,\ root,\ N,\ itercell\_tree,\ levorder$

Require: real side, rmin, rsize

Local:

Require: int $jsub,\ ix,\ iy,\ ic,\ m,\ k,\ i,\ pcell$

Require: real $xcell[4],\ ycell[4]$

Require: real $conv\_to\_int$ ▷ integer conversion

Require: long int $key\_M,\ Morton\_2D$ ▷ Morton key

Begin

Open $tree.dat$ ▷ read data file

read side,N ▷ read box side and number of points

if $N > Np$ **then**

print N,Np

STOP

end if

for $i \leftarrow 1, N$ **do**

read $pos[1, i],\ pos[2, i]$ ▷ particles positions are stored in $pos[1 : 2, i]$

end for

▷ Now set the the sub-quadrant positions in pm1

$pm1[0, 1] := -1$

$pm1[0, 2] := -1$

$pm1[1, 1] := +1$

$pm1[1, 2] := -1$

$pm1[2, 1] := -1$

$pm1[2, 2] := +1$

$pm1[3, 1] := +1$

$pm1[3, 2] := +1$

```
                                        ▷ now compute the Morton keys
    levorder := 10
    conv_to_int := 2^levorder/side
    for i ← 1, N do
        ix = conv_to_int * pos[1, i]
        iy = conv_to_int * pos[2, i]
        key_M[i] = Morton_2D(ix, ix, levorder)
    end for
    rmin := 0                                   ▷ set tree boundaries
    rsize := side
    incells := 1
    root := nbodsmax + 1
    pointers_of_tree := 0
    cellsize := 0
    bottom := 0
    for k ← 1, ndim do                          ▷ set up position of root cell
        pos[k, root] := rmin[k] + rsize/2
        bottom[k, root] := rmin[k]
    end for
    cellsize[root] = side                       ▷ set up root cellsize
    for i ← 1, n do                             ▷ now initialize particle list
        bodlist[i] := i                         ▷ all particles need to be examined
    end for
    itercell_tree := 0                          ▷ set the level
    nbodlist := N                               ▷ actual value of the size of the particle list
    CALL tree_sort(nbodlist, bodlist, root)     ▷ call the tree function
```

```
    Open treecell.dat                                          ▷ write cell file
    print incells
    for ic ← 1, incells do
        pcell := ic + root − 1
        xcell[1] := bottom[1, pcell]
        ycell[1] := bottom[2, pcell]
        xcell[2] := bottom[1, pcell] + cellsize[pcell]
        ycell[2] := bottom[2, pcell]
        xcell[3] := bottom[1, pcell] + cellsize[pcell]
        ycell[3] := bottom[2, pcell] + cellsize[pcell]
        xcell[4] := bottom[1, pcell]
        ycell[4] := bottom[2, pcell] + cellsize[pcell]
        print ic, (xcell[m], ycell[m], m = 1, 4)
    end for

    CALL sorti(key_M, subindex, N)  ▷ this call returns in subindex the
  list of the particles ordered by the value of key_M
    Open tree_ord.dat                                      ▷ write particle file
    print side
    for m ← 1, N do
        i=subindex[i]
        print pos[1, i], pos[2, i], key_M[i]
    end for
end procedure
```

```
procedure TREE_SORT(NBLIST,LISTBODIES,OLDCELL)
  Local:
Require: int jsub, ju, nsubc, m, i, k , kpast, key_of_point
Require: int pbody, newcell, oldcell, levscan ,lpos
Require: int listbodies[nblist], hoc[0 : 3]
Require: int sublist[nblist], llj[nblist]

    itercell_tree := itercell_tree + 1              ▷ monitor the level
    if itercell_tree > levorder then
        print itercell_tree, levbit         ▷ level higher than Morton order
        STOP
    end if
    levscan := levbit − itercell_tree          ▷ bit positions for this level
    HOC := 0                                  ▷ reset the local linked-list
    LLJ := 0

    for k ← 1, nblist do            ▷ find the particle in each sub quadrant
        i = listbodies[k]                              ▷ which particle ?
        key_of_point = key_M[i]                ▷ Morton key of the particle
        lpos := ndim ∗ levscan ▷ this is the starting addres of the bits in
  the Morton key
        ju := IBITS(key_of_point, lpos, 2)   ▷ this is the sub-quadrant
        kpast := HOC(ju)                               ▷ append to the list
        LLJ(k) := kpast
        HOC(ju) := k
    end for
    for jsub ← 0, nsubcell − 1 do               ▷ now loop on the subcells
        k := HOC(jsub)                        ▷ first particle in the subcell
                                                       ▷ subcell empty
        if k := 0  then CYCLE
        end if
        nsubc := 0
        while  k > 0  do
            nsubc := nsubc + 1                       ▷ copy the linked-list
            i = listbodies[k]
            sublist[nsubc] = i
            k = llj[k]
        end while                                             ▷ end list
```

```
        if nsubc > 1 then                                          ▷ this is a cell
            incells := incells + 1                                 ▷ add the new cells
            if incells > ncells then
                print incells, ncells
                STOP
            end if
            newcell := incells + root − 1                          ▷ address new cell

            pointers_of_tree[jsub + 1, oldcell] := newcell         ▷ → pointer to the new cell
            cellsize[newcell] := cellsize[oldcell]/2
            for m ← 1, ndim do                                     ▷ new cell coordinates ( geometric center )
                pos[m, newcell] := pos[m, oldcell] + pm1[jsub, m] ∗ 0.5 ∗ cellsize[newcell]
                bottom[m, newcell] := pos[m, newcell] − 0.5 ∗ cellsize[newcell]
            end for

            iback[1, newcell] := jsub                              ▷ helpful to trace back the parent cell
            iback[2, newcell] := oldcell

            CALL tree_sort(nsubc, sublist, newcell)                ▷ go another level

        else if nsubc = 1 then                                     ▷ one particle in the subcell

            pbody := sublist[nsubc]
            pointers_of_tree[jsub + 1, oldcell] := pbody           ▷ because pbody < root this indicates that pointers_of_tree[jsub + 1, oldcell] points to a particle

            iback[1, pbody] := jsub
            iback[2, pbody] := oldcell
        end if
    end for                                                        ▷ end subcells
end procedure
```