MAX : total no of nodes

count_in : no of internal nodes
(= MAX * (1 - 0.9))

```
struct segment {
    population (n)
    boundary
} s;
```

algo :

Declare root [MAX] // for allocation of nodes to segments

partition square into count_in segments

Select internal nodes

Connect internal nodes using Prim's Algorithm

for each node
    nin = nearest internal node
    Mark edge (node, nin) in T

partition :

init root [ ] as 0

s[0]. n = MAX

s[0]. boundary = unit square under consideration

for count_in - 1 times

Bubble the most populated segment i.e.
    Specify new boundaries
    reallocate nodes as per new boundaries

select_in :

declare ax [count_in], ay [count_in]

Iterate through root, store mean of ordinates in ax
    mean of abscissae in ay

// s.t. (ax [i], ay [i]) represent centermost point of segment i

Identify internal nodes in inds [count_in]

// s.t. node inds[i] is nearest pt to centremost point of the segment i

prims_mst : (Modification)

declare parent[count_in]

~~// s.t. parent[i]~~ .

// s.t. wt(parent[i], i) = key[i]

Mark edge (parent[i], i) in T