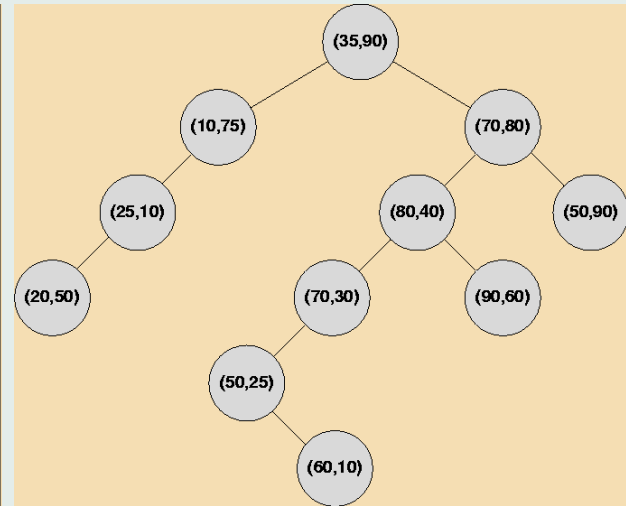
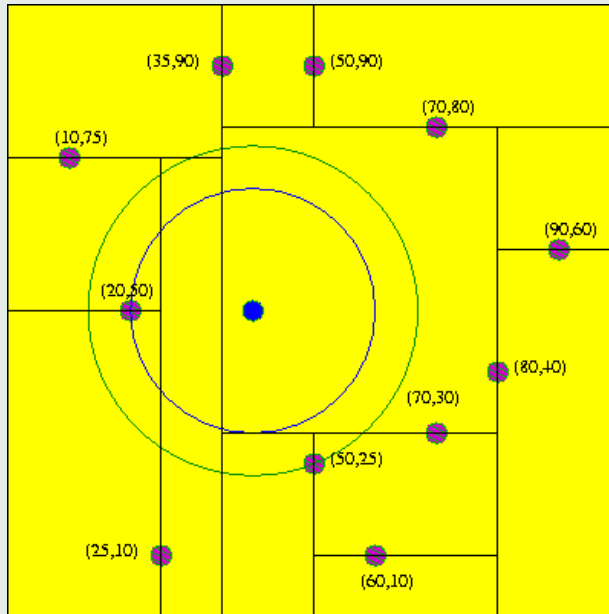


Nearest Neighbors: The scenario

- “Find the nearest Pizza Hut.” (Compare with the McDonald problem).
- Assume kd-tree T given, and C is the region associated with a node.
- Input p is a point
- Searching for point p in T helps
 - In one dimension, T is very useful: the closest neighbor is from the set of nodes visited (MANY nodes are pruned)
 - In higher dimensions, T is not as useful (the closest neighbor may be far away).
- Nevertheless, pruning is possible.
- General strategy: Collect partial results, judicious traversal, and prune.

What If We Locate Point?



We visit $(35, 90)$, $(70, 80)$, \dots , and fall off $(70, 30)$

Closest point is nowhere near this path. We must visit both subtrees.

Home Page

Title Page

Contents

◀▶

◀▶

Page 3 of 17

Go Back

Full Screen

Close

Quit

Nearest Neighbor: Naive version

```
class Result {int distance; Point point;}
init () {
    result.distance = infinity; result.point = null
}
Point query;
Result process(KDNode k, Result res) {
    if (null(k)) return res;
    int cost = distance (k.data, query);
    if (cost < res.distance) {
        res.point = node.data;
        res.distance = distance (node.data, query);
    }
    res = process(k.left, res);
    res = process (k.right, res);
    return res;
}
```

Nearest Neighbor: Pruned version

- Maintain the rectangle r associated with a node
- Compute a lower bound on the distance from the query q to the rectangle
 - Distance between q and *any* point in r is at least $\text{lowerbound}(r, q)$
 - Do not compute all distances between q and every point in r
- $\text{lowerbound}()$ helps because if the lower bound is larger than the distance computed so far, we do not consider many points
- Must compute $\text{lowerbound}()$ quickly

Nearest Neighbor: Pruned Version

```
float lowerbound(Rectangle r, Point p) {
    if (r.inside(p)) return 0;
    if (r.left(p)) return r.minX - p.x;
    ...
}
Result process(KDNode k, int cd, Rectangle r, Result res) {
    if (k == null) return res;
    if (lowerbound(r, query) >= res.distance) return res;
    ...
}
```

- If the lower bound is larger than the distance computed so far, exit!
- Otherwise compute the distance with the current node
- Process the two children in order!

Nearest Neighbour Pruned Version

Home Page

Title Page

Contents



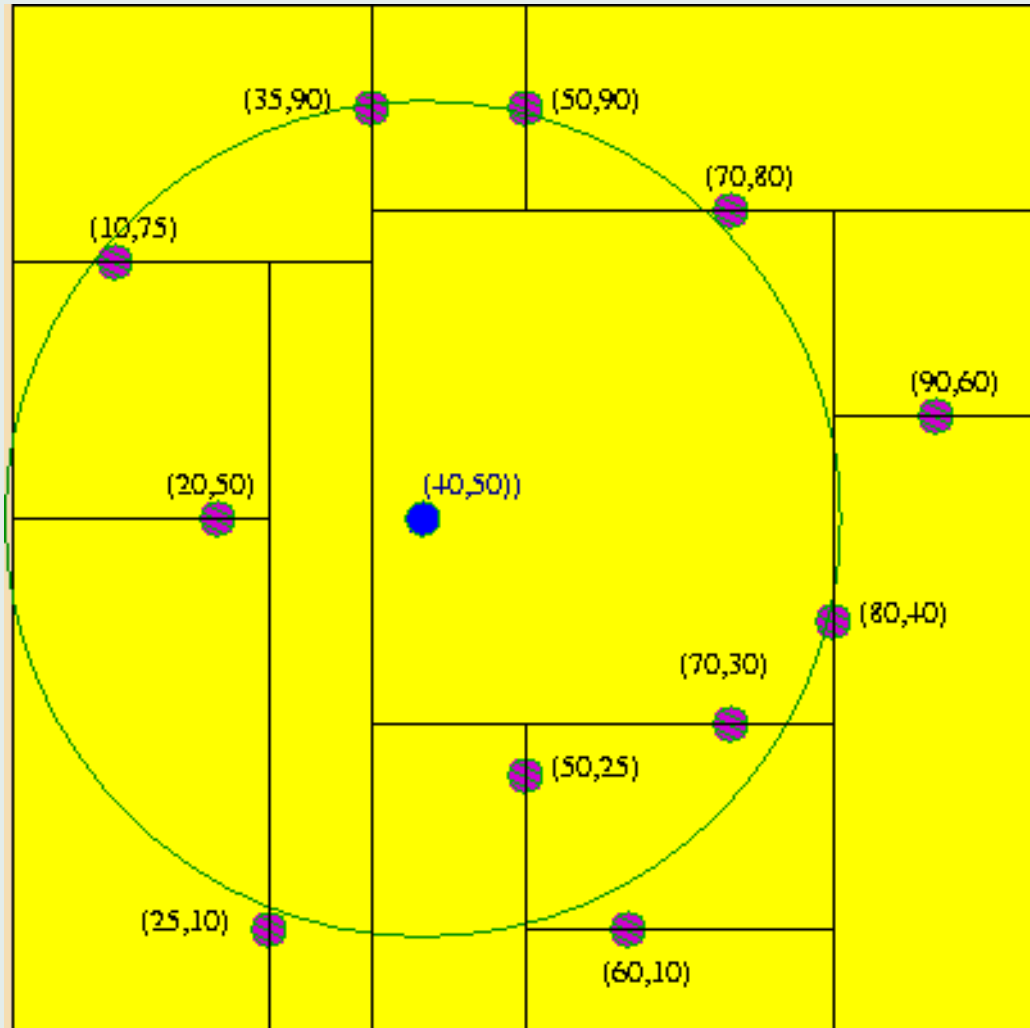
Page 6 of 17

Go Back

Full Screen

Close

Quit



Nearest Neighbour Pruned Version

Home Page

Title Page

Contents



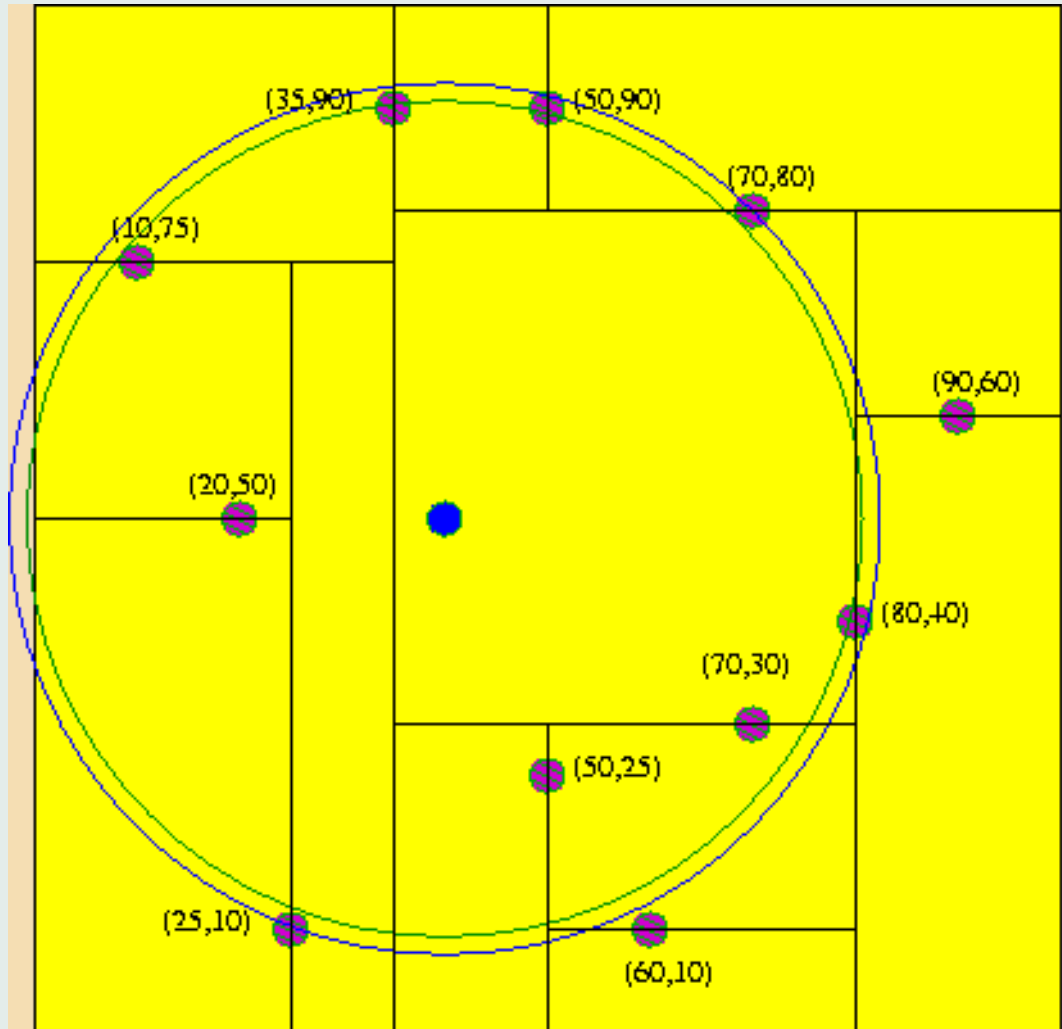
Page 7 of 17

Go Back

Full Screen

Close

Quit



Nearest Neighbour Pruned Version

Home Page

Title Page

Contents



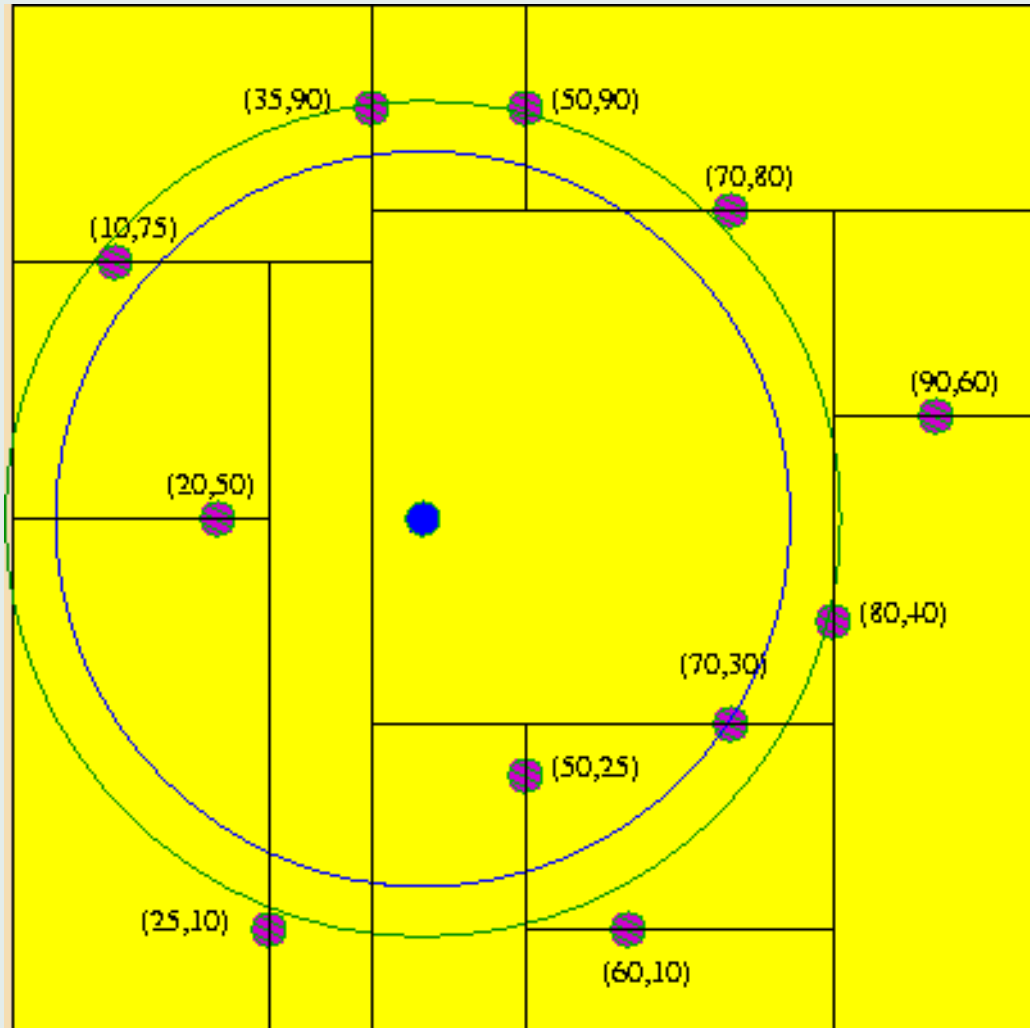
Page 8 of 17

Go Back

Full Screen

Close

Quit



Nearest Neighbour Pruned Version

Home Page

Title Page

Contents



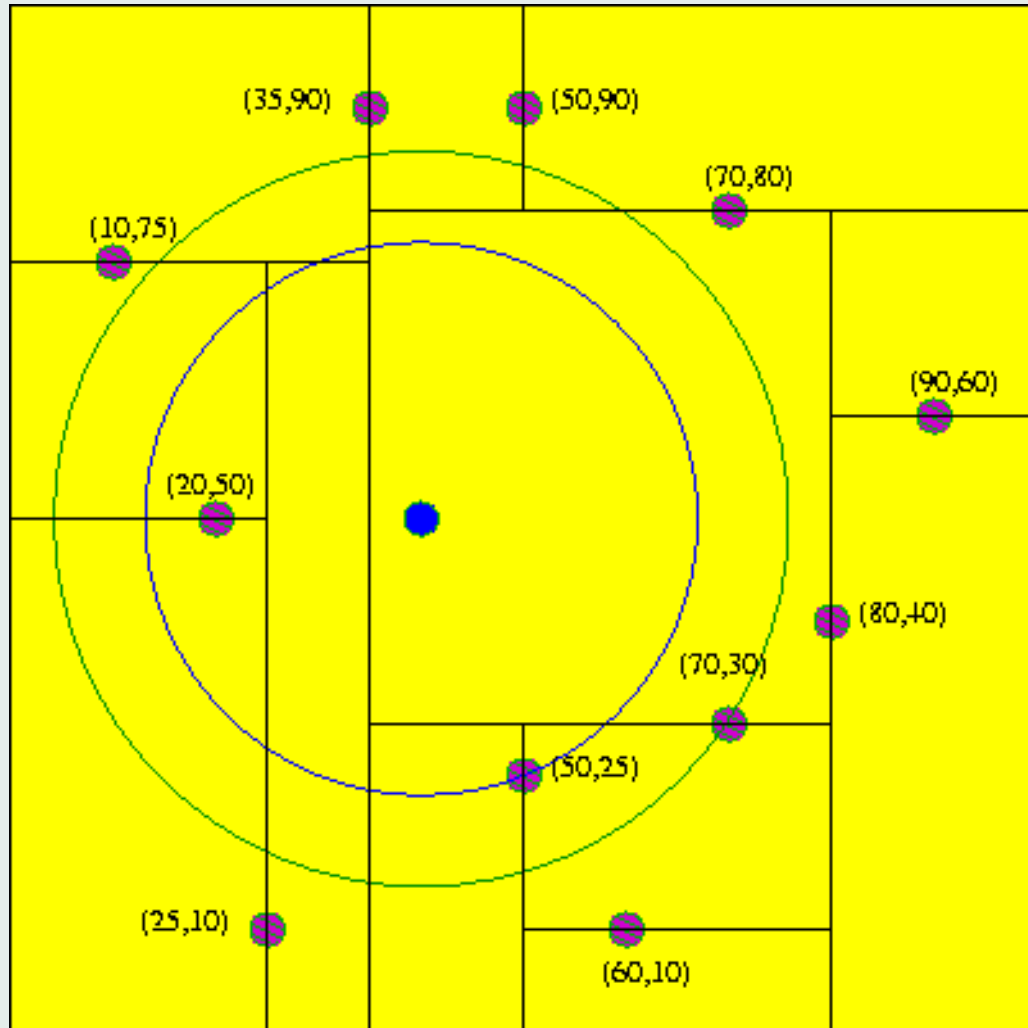
Page 9 of 17

Go Back

Full Screen

Close

Quit



Nearest Neighbour Pruned Version

Home Page

Title Page

Contents



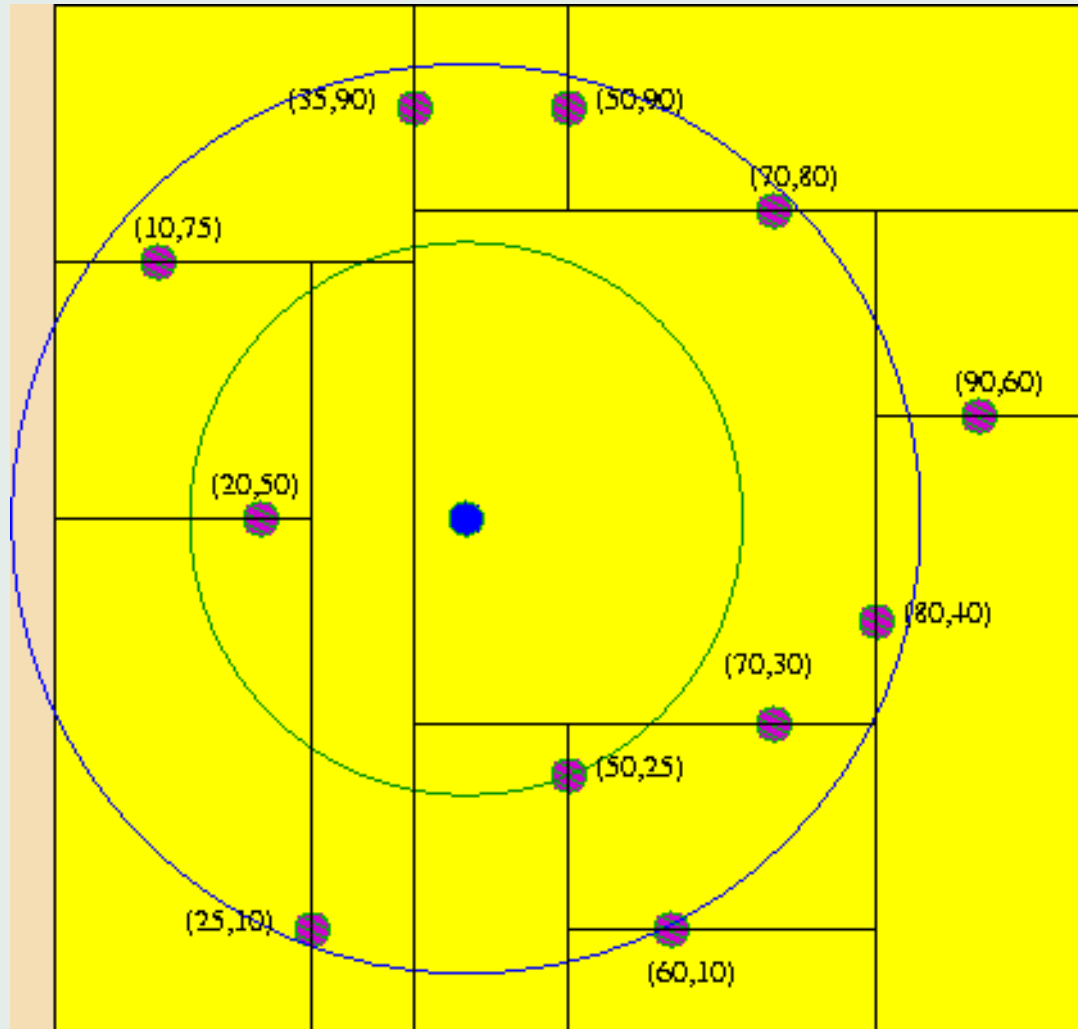
Page 10 of 17

Go Back

Full Screen

Close

Quit



Nearest Neighbour Pruned Version

Home Page

Title Page

Contents



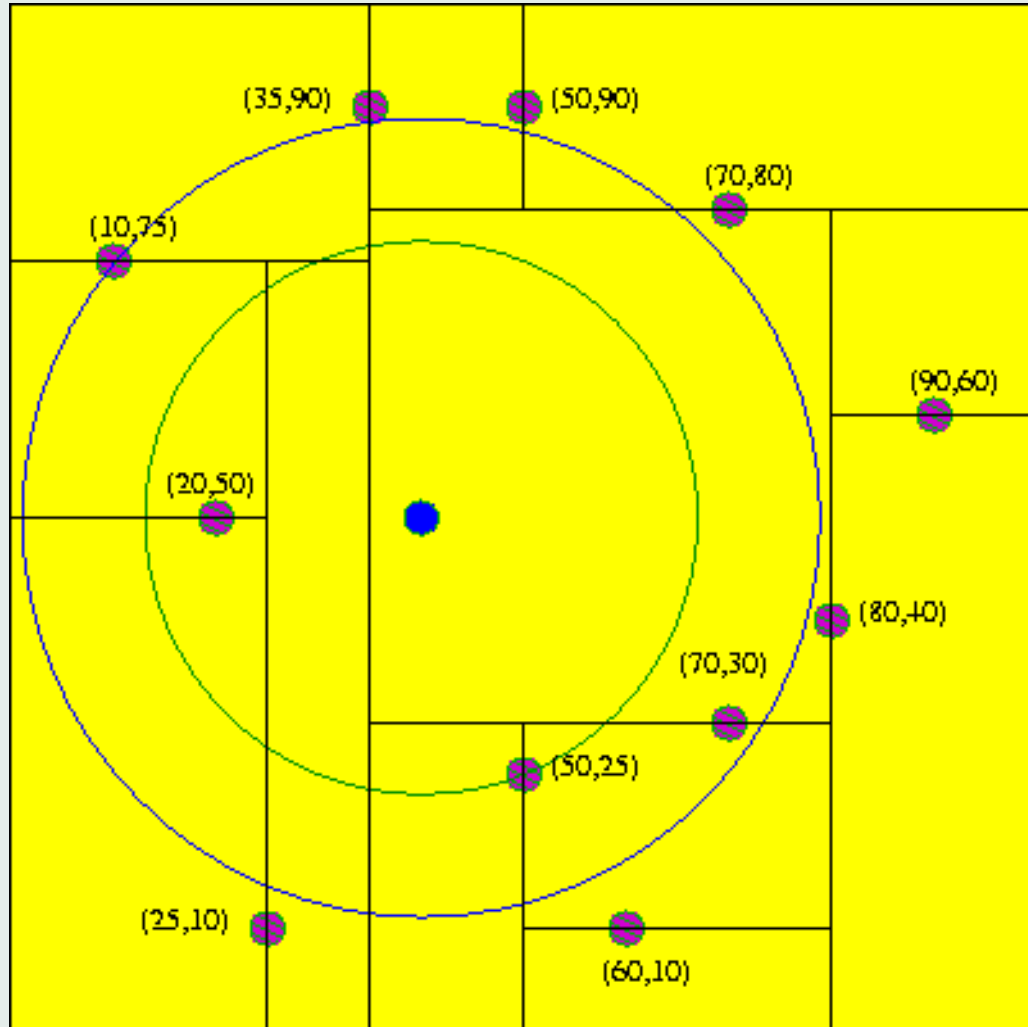
Page 11 of 17

Go Back

Full Screen

Close

Quit



Nearest Neighbour Pruned Version

Home Page

Title Page

Contents

◀ ▶

◀ ▶

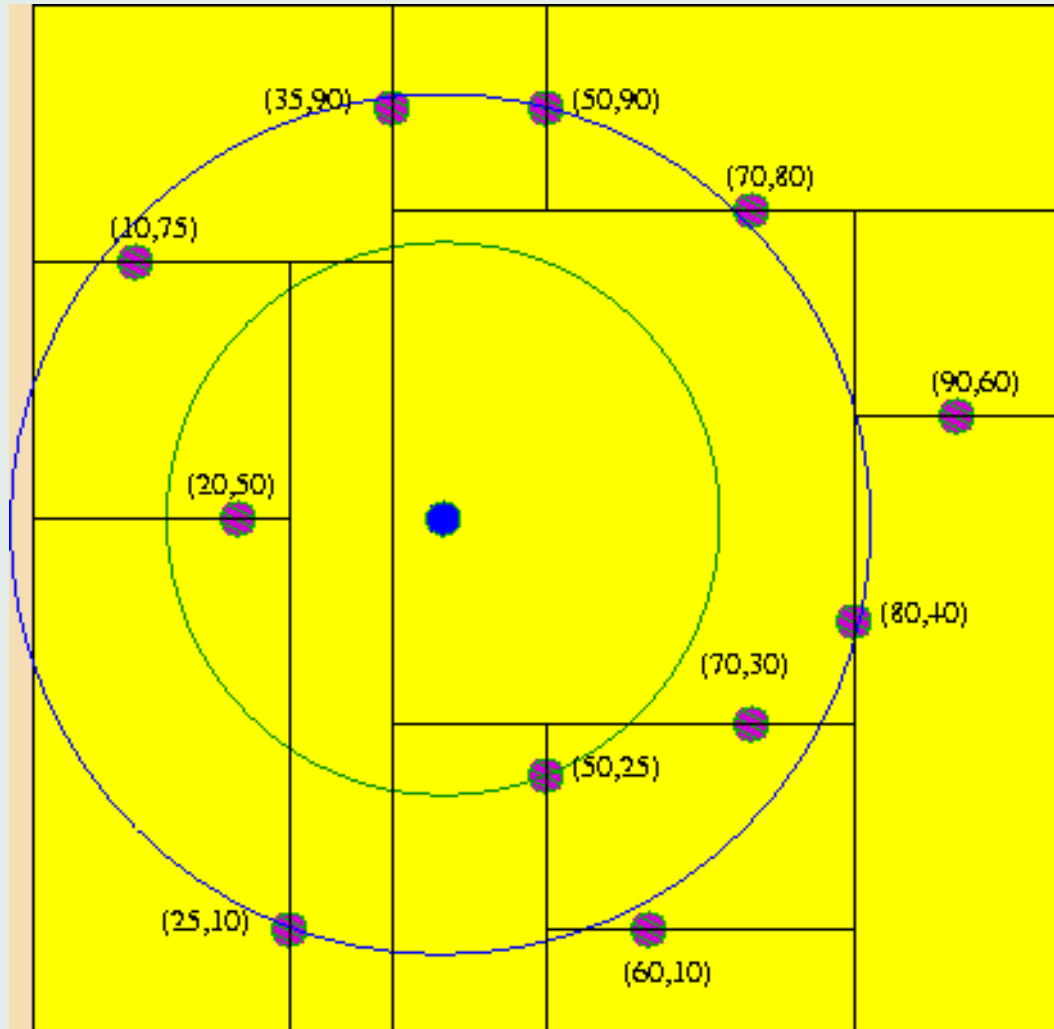
Page 12 of 17

Go Back

Full Screen

Close

Quit



Nearest Neighbour Pruned Version

Home Page

Title Page

Contents



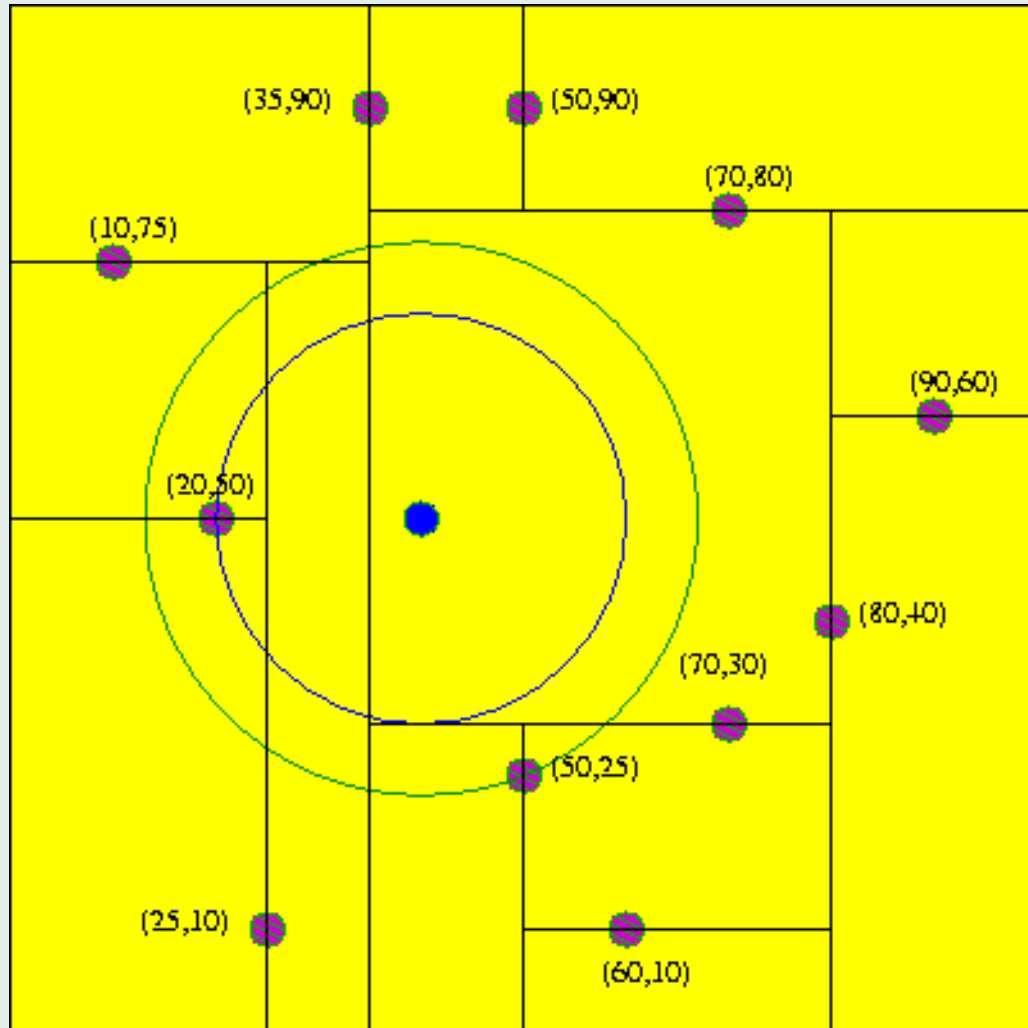
Page 13 of 17

Go Back

Full Screen

Close

Quit



Home Page

Title Page

Contents

◀▶

◀▶

Page 14 of 17

Go Back

Full Screen

Close

Quit

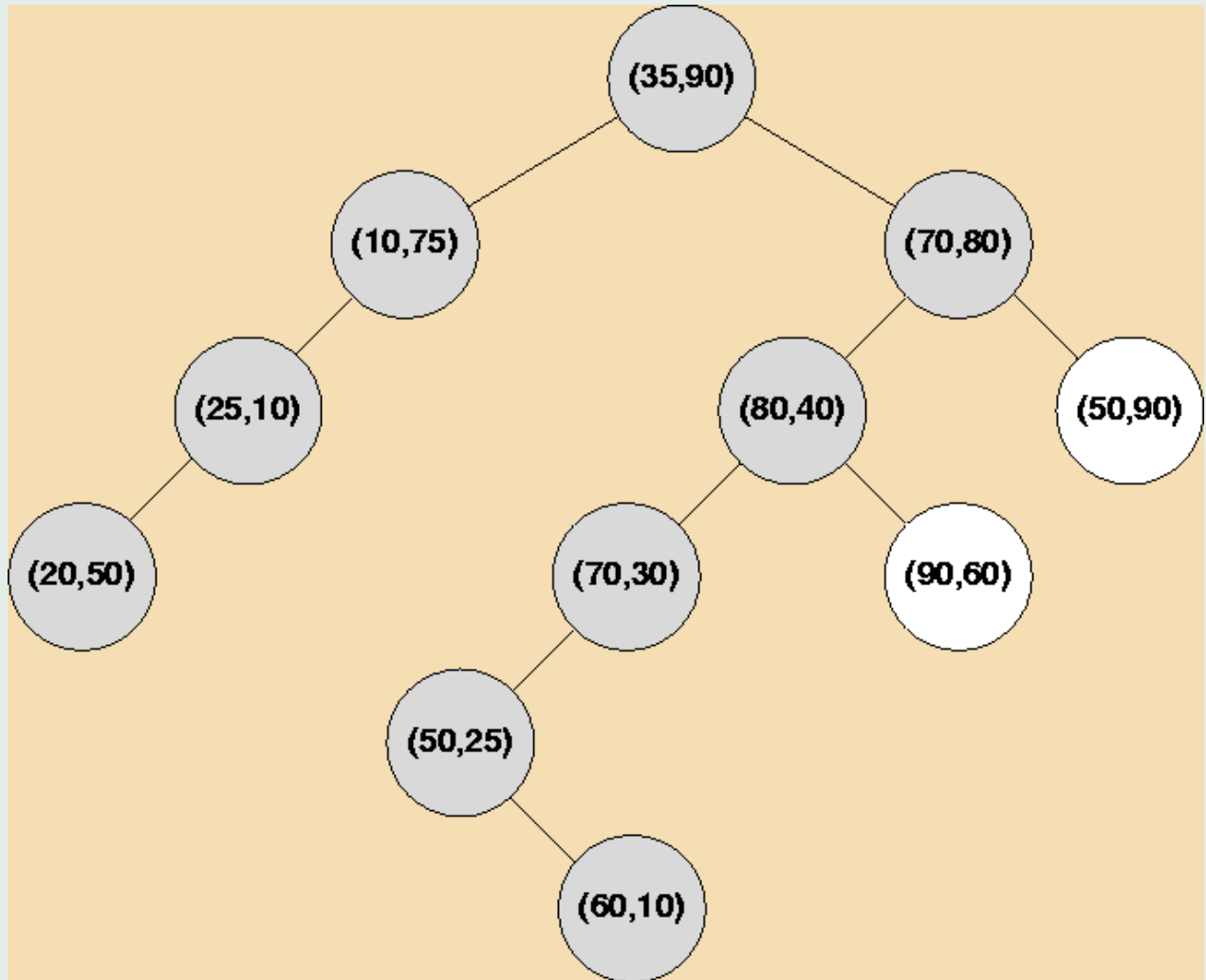
Nearest Neighbor: Pruned version

```
float lowerbound(Rectangle r, Point p) {
    if (r.inside(p)) return 0;
    if (r.left(p)) return r.minX - p.x;
    if (r.right(p)) return p.x - r.maxX;
    if (r.southEastCorner(p)) ...
}
Result process(KDNode k, int cd, Rectangle r, Result res) {
    if (k == null) return res;
    if (lowerbound(r, q) >= res.distance) return res;
    float dist = distance (node.data, query);
    if (dist < res.distance) {
        result.point = node.data;
        result.distance = distance (node.data, query);
    }
    if (q[cd] < k.data[cd]) ... else ...
    return res;
}
```

Nearest Neighbor: Pruned version

```
Result process(KDNode k, int cd, Rectangle r, Result res) {
    float dist = distance (node.data, query);
    if (dist < res.distance) {
        result.point = node.data;
        result.distance = distance (node.data, query);
    }
    if (q[cd] < k.data[cd]) {
        res = process(k.left, cd+1, r.trimLeft(cd, k.data), res);
        res = process(k.right, cd+1, r.trimRight(cd, k.data), res);
    }
    else {
        res = process(k.right, cd+1, r.trimRight(cd, k.data), res);
        res = process(k.left, cd+1, r.trimLeft(cd, k.data), res);
    }
    return res;
}
```

Example



Home Page

Title Page

Contents

◀◀

▶▶

◀

▶

Page 16 of 17

Go Back

Full Screen

Close

Quit

Home Page

Title Page

Contents



Page 17 of 17

Go Back

Full Screen

Close

Quit