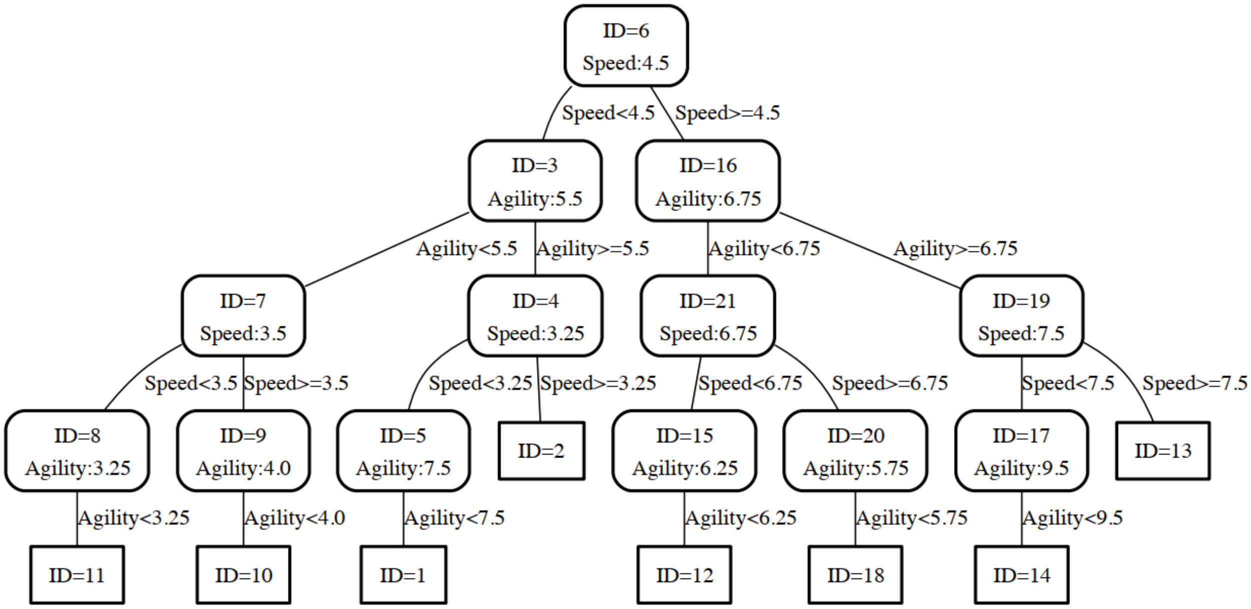
Course example

Table 5.4
The extended version of the college athletes dataset.

ID	Speed	Agility	Draft
1	2.50	6.00	no
2	3.75	8.00	no
3	2.25	5.50	no
4	3.25	8.25	no
5	2.75	7.50	no
6	4.50	5.00	no
7	3.50	5.25	no
8	3.00	3.25	no
9	4.00	4.00	no
10	4.25	3.75	no
11	2.00	2.00	no
12	5.00	2.50	no
13	8.25	8.50	no
14	5.75	8.75	yes
15	4.75	6.25	yes
16	5.50	6.75	yes
17	5.25	9.50	yes
18	7.00	4.25	yes
19	7.50	8.00	yes
20	7.25	5.75	yes
21	6.75	3.00	yes

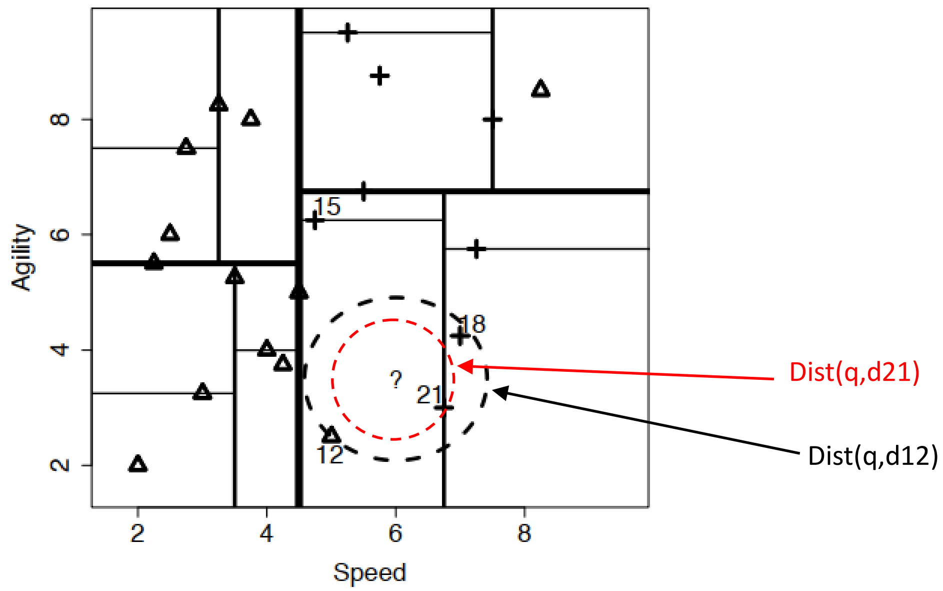
Query: Speed = **6.00**, Agility = **3.50**



Algorithm 3 Pseudocode description of the *k-d* tree nearest neighbor retrieval algorithm.

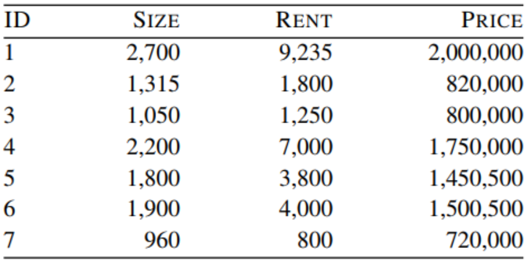
```
Require: query instance q and a k-d tree kdtree
 1: best = null
 2: best-distance = ∞
 3: node = descendTree(kdtree,q)
 4: while node! = NULL do
 5:     if distance(q,node) < best-distance then
 6:         best = node
 7:         best-distance = distance(q,node)
 8:     end if
 9:     if boundaryDist(q, node) < best-distance then
10:         node = descendtree(node,q)
11:     else
12:         node = parent(node)
13:     end if
14: end while
15: return best
```

Applying the algorithm for: **Query**: SPEED = **6.00**, AGILITY = **3.50**

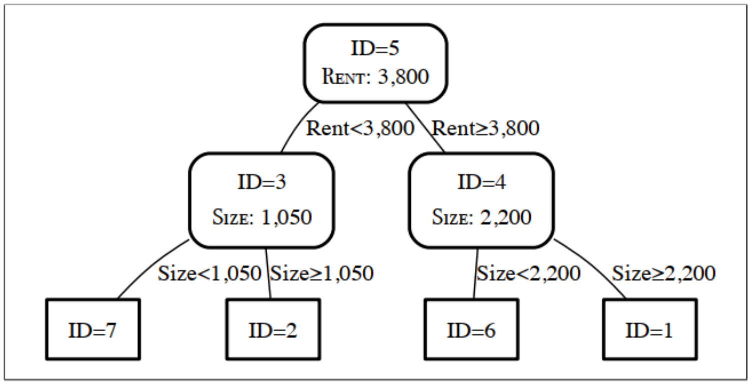


- Start algo: Best dist. = infinity, node = descent Tree (KdTree, q) ➔ search leads to leaf node d12
- Dist. (q, **d12**) = **1.4142** (**Best dist. updated**) ➔ line 9 leaf node has no boundary dist. so we execute the “else” line 11 and we go to its parent which is node d15.
- Dist. (q, d15) = 3.0208 (Line 5 fails no update for best dist.) || Boundary_Dist (q, d15/AGILITY) =2.75 ➔bigger than best don’t descend subtree (Line 9 fails thus we go to its parent node with d21)
- Dist. (q, **d21**) = **0.9014** (**Best dist. updated**) || Boundary_Dist (q, d21/SPEED) =0.75 ➔smaller than best, **thus descend subtree (Line 9 succeed we traverse the subtree ➔ the right subtree of d21 which leads to the leaf node d18)**
- Dist. (q, d18) = 1.2500 (Line 5 fails no update for best dist.) || line 9 leaf node no boundary so we do the else line 11 and we go to its parent which is d20
- Dist. (q, d20) = 2.75 (Line 5 fails no update for best dist.) || Boundary_Dist (q, d20/ AGILITY) =2.25 ➔ bigger than best don’t descend subtree (Line 9 fails thus we go to its parent which is d16)
- Dist. (q, d16) = 3.2882 (Line 5 fails no update for best dist.) || Boundary_Dist (q, d16/ AGILITY) =3.25 ➔ bigger than best don’t descend subtree (Line 9 fails thus we go to its parent which is the root node d6)
- Dist. (q, d6) = 2.1213 (Line 5 fails no update for best dist.) || Boundary_Dist (q, d6/SPEED) =1.5 ➔ bigger than best don’t descend subtree (Line 9 fails thus we go to its parent) ➔ no parent node (parent node == NULL) for the root, thus exits the while loop and return the best which is **d21**

Tutorial example



ID	Size	Rent	Price
1	2,700	9,235	2,000,000
2	1,315	1,800	820,000
3	1,050	1,250	800,000
4	2,200	7,000	1,750,000
5	1,800	3,800	1,450,500
6	1,900	4,000	1,500,500
7	960	800	720,000



Query: Size = 1000, Rent = 2200.

- Start algo: Best dist. = infinity, node = descent Tree (KdTree, q) ➔ search leads to leaf node d7
- Dist. (q, **d7**) = **1400.57** (**Best dist. updated**) ➔ line 9 leaf node has no boundary dist. so we execute the "else" line 11 and we go to its parent which is node d3.
- Dist. (q, **d3**) = **951.31** (**Best dist. updated**) || Boundary_Dist (q, d3/RENT) = 50 ➔smaller than best, **thus descend subtree (Line 9 succeed we traverse the subtree ➔ the right subtree of d3 which leads to the leaf node d2)**
- Dist. (q, **d2**) = **509.14** (**Best dist. updated**) ➔ line 9 leaf node has no boundary dist. so we execute the "else" line 11 and we go to its parent which is node d5 (the root node).
- Dist. (q, d5) = 1788.85 (Line 5 fails no update for best dist.) || Boundary_Dist (q, d5/ RENT) =1600 bigger than best don't descend subtree (Line 9 fails thus we go to its parent) ➔ no parent node (parent node == NULL) for the root, thus exits the while loop and return the best which is **d2.**