|  |  |
| --- | --- |
| **Algorithm 1**: VP-Tree Concurrent Filtering | |
| **Global variables:**  Maxheap *result*, /\* *result* stores accepted points at a specific moment, ordered by distance to the *query* point, and *result* has a maximum capacity of *k*, further insertion implies replacement. \*/  Point *query*, Integer *k*  **Input:** Node *node* /\* *node* is the current node, which contains a vantage *point* and a *radius* \*/  **Output:** *result* | |
| 1. | /\*Identify the threshold *t* that determines whether new entries are accepted into *result* \*/ |
| 2. | **if** *result* is empty |
| 3. | *t* 🡨 |
| 4. | **else** |
| 5. | *t* 🡨max distance in *result* |
| 6. | /\* *d* is the distance between the point the current node contains and the query point \*/ |
| 7. | *d* 🡨 EuclideanDistance (*node*.*point*, *query*) |
| 8. | /\* *r* is the radius of the current node that separates the inner and outer subtree \*/ |
| 9. | *r* 🡨 *node*.*radius* |
| 10. | /\* determine if the current point should be added to the intermediate result. \*/ |
| 11. | **if** *d<=t* **and** *node.point* fulfills attribute filtering criteria |
| 12. | add *node.point* to *result* |
| 13. | /\* search the inner subtree if it overlaps with the query area \*/ |
| 14. | **if** *d<=r+t* |
| 15. | search(*inner*) |
| 16. | /\* search the outer subtree if it overlaps with the query area \*/ |
| 17. | **if** *d>=r-t* |
| 18. | search(*outer*) |
|  |
|  |
|  |
|  |