# Geometry: Най-близък съсед

locked



Имплементирайте метода на KD дървото намиращ разстоянието до най-близкия съсед на дадена точка с целочислени координати в $K$-мерното пространство:

```
double closest_point(const vector<int>& point) {
    //TODO
}
```

Разстоянието между две точки в $K$-мерното пространство: $(x_1, x_2, \ldots, x_k)$ и $(y_1, y_2, \ldots, y_k)$ можете да намерите използвайки Питагоровата теорема:

$$\sqrt{(x_1 - y_1) \times (x_1 - y_1) + (x_2 - y_2) \times (x_2 - y_2) + \ldots + (x_k - y_k) \times (x_k - y_k)}$$

### Input Format

Вашият код не трябва да въвежда нищо от стандартния вход. Решението ви ще бъде тествано с $Q$ на брой заявки към дърво съдържащо $N$ на брой точки с целочислени координати.

### Constraints

$0 \leq N \leq 10^5$

$0 \leq Q \leq 10^5$

$2 \leq K \leq 5$

$-10^6 \leq coordinate_{i_j} \leq 10^6$

### Output Format

Вашият код не трябва да извежда нищо на стандартния изход.

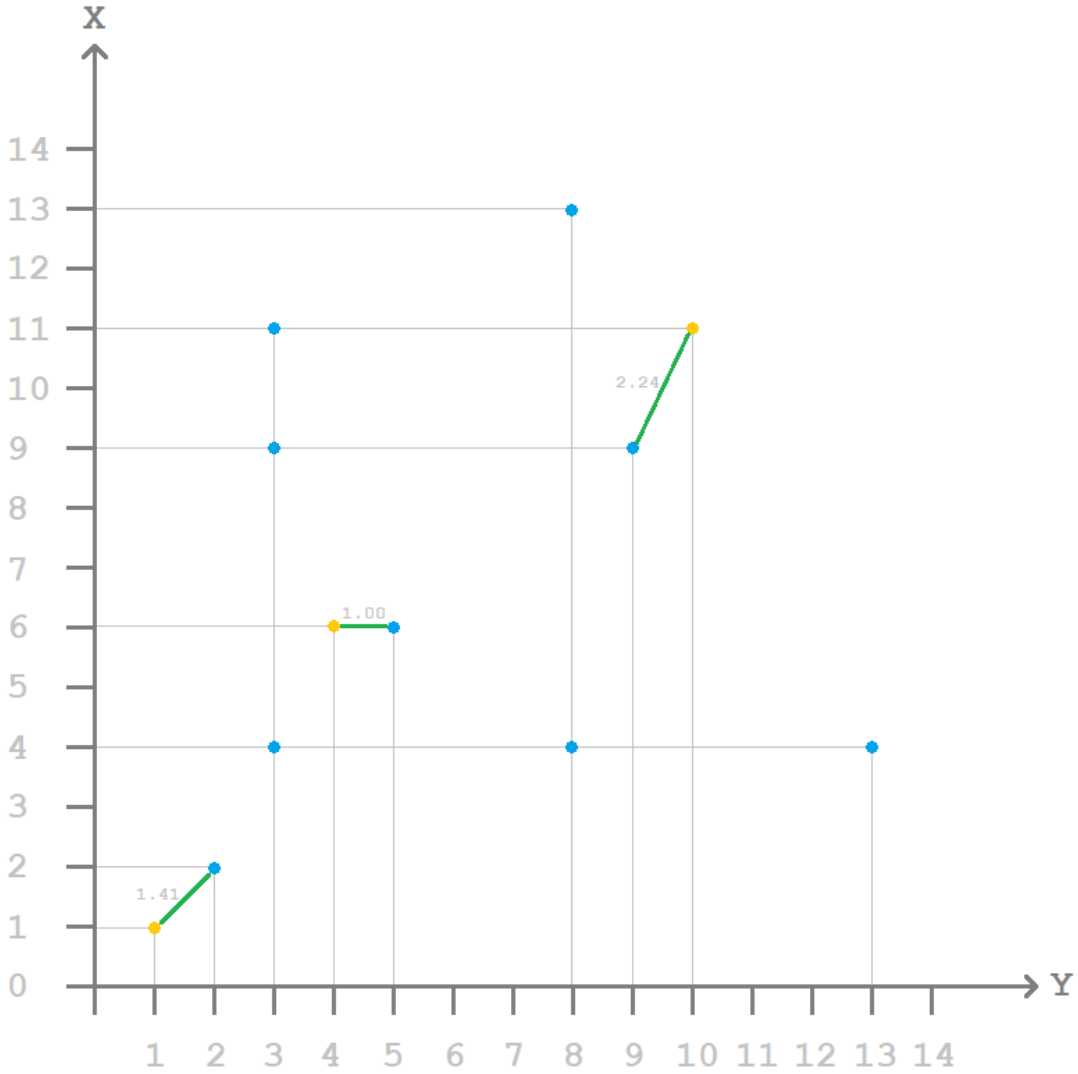
### Sample Input 0

```
9 2
2 2
3 4
3 9
3 11
5 6
8 4
8 14
9 9
13 4
3
1 1
10 11
4 6
```

### Sample Output 0

```
1.414
2.236
1.000
```

Explanation 0




Sample Input 1

```
11 4
-3 -9 9 -5
8 -4 5 -10
7 -5 -4 -6
0 -9 -5 10
10 10 -2 10
-7 8 3 -2
3 -5 5 -9
-9 9 -5 -4
3 5 0 -6
5 6 9 -6
3 -5 5 -9
5
-9 9 -9 -1
7 2 -8 7
6 5 -9 -8
7 2 -8 7
0 -9 -5 10
```

Sample Output 1

```
5.000
10.863
9.695
10.863
0.000
```

Submissions: 38
Max Score: 100
Difficulty: Hard

Rate This Challenge:

More

Current Buffer (saved locally, editable) C++14

```cpp
#include <bits/stdc++.h>
using namespace std;

struct point_comparator {
    int dimension;

    point_comparator(int dimension = 0) : dimension(dimension) {}

    const bool operator()(const vector<int>& point1, const vector<int>& point2) const {
        return point1[dimension] < point2[dimension];
    }
};

class kd_tree {
    private:
        struct kd_node {
            vector<int> point;

            kd_node* left;
            kd_node* right;

            kd_node(const vector<int>& point) : point(point) {}
        };

        int k;
        kd_node* root;

        kd_node* build(vector<vector<int>>& points, int from, int to, int axis) {
            if (from > to) {
                return nullptr;
            }

            int mid = (from + to) / 2;
            nth_element(points.begin() + from,
                        points.begin() + mid,
                        points.begin() + to + 1,
                        point_comparator(axis));

            kd_node* node = new kd_node(points[mid]);
            node->left = build(points, from, mid - 1, (axis + 1) % k);
            node->right = build(points, mid + 1, to, (axis + 1) % k);

            return node;
        }

    public:
        kd_tree(vector<vector<int>> points, int dimensions) {
            k = dimensions;
            root = build(points, 0, points.size() - 1, 0);
        }

        /*
         * Each node in the KD tree is represented as follows:
         *
         *      struct kd_node {
         *          vector<int> point;
         *
         *          kd_node* left;
         *          kd_node* right;
         *      };
         *
         * The root of the tree is stored in the variable:
         *
```

```
         *      kd_tree* root;
         *
         * Each point in the tree has k coordinates (one for each dimension):
         *
         *      int k;
         *
         * The following method finds the distance to the point in the KD tree
         * that is closest to the given point.
         *
         * The method returns a single real number (of type dobule) that is
         * the distance to the closest point in the tree.
         */
        double closest_point(const vector<int>& point) {
            //TODO
        }

};

vector<int> read_point(int k) {
    vector<int> point(k);
    for (int i = 0; i < k; i++) {
        cin >> point[i];
    }
    return point;
}

void solve() {
    int n, k;
    cin >> n >> k;

    vector<vector<int>> points;
    for (int i = 0; i < n; i++) {
        points.push_back(read_point(k));
    }

    kd_tree tree(points, k);

    int q;
    cin >> q;
    cout << fixed << setprecision(3);

    for (int i = 0; i < q; i++) {
        cout << tree.closest_point(read_point(k)) << "\n";
    }
}

int main() {
    ios_base::sync_with_stdio(false);
    cin.tie(nullptr);

    solve();

    return 0;
}
```

Line: 1 Col: 1

Upload Code as File  Test against custom input  Run Code  Submit Code