```
1 #include "../bits/stdc++.h"
     * kD-tree (k = 2)
 3
    * k次元二分探索木(?)
    * 構築までしか書いてない
 7 template <typename T>
 8 class KdTree
9 {
10
         static const int Dim = 2;
11
         struct Node
12
         {
             int idx; // 元々の頂点番号
Node *ch[2];
13
14
             int axis;
15
16
17
             Node() : idx(-1), axis(-1) { ch[0] = ch[1] = nullptr; }
18
         };
19
         Node *root;
20
         std::vector<T> points;
21
22
23
         // 構築
         // indices := 点の元々のインデックス
24
25
         Node *build(int *indices, int pointSize, int depth)
26
         {
27
             if (pointSize <= 0)</pre>
28
                  return nullptr;
29
             int axis = depth % Dim;
int mid = (pointSize - 1) / 2;
30
31
32
             std::nth_element(indices, indices + mid, indices + pointSize, [&](int lhs, int rhs) {
    return points[lhs][axis] < points[rhs][axis];</pre>
33
34
35
36
37
             Node *node = new Node();
38
             node->idx = indices[mid];
39
             node->axis = axis;
40
             \label{eq:node-ch} \begin{split} &\text{node->ch[0] = build(indices, mid, depth + 1);} \\ &\text{node->ch[1] = build(indices + mid + 1, pointSize - mid - 1, depth + 1);} \end{split}
41
42
43
             return node;
44
45
        }
46 };
47
48 struct Point
49
   {
50
         static const int Dim = 2;
51
         std::vector<double> ps;
52
         Point() {}
         Point(double x, double y) : ps(2)
53
54
             ps[0] = x;
55
             ps[1] = y;
56
57
        }
58
59
         double operator[](std::size_t t)
60
             assert(0 <= t && t <= 1);
62
             return ps[t];
63
64 };
65
66
```

localhost:4649/?mode=clike 1/1