Lab 4

* Hand-trace make\_tree function

struct kd\_node\_t \*n;

make\_tree(wp, sizeof(wp) / sizeof(wp[1]), 0, 2);

sizeof(wp) = 6, sizeof(wp[1]) = 1

sizeof(wp) / sizeof(wp[1]) = len = 1

len != 0 => go to if ((n = find\_median(t, t + len, i)))

if ((n = find\_median(t, t + len, i))) => sorting wp = {{{2, 3}, {4, 7}, {5, 4}, {7, 2}, {8, 1}, {9, 6}}}

n = &wp[2]

i = (0 + 1) % 2 => i = 1

n->left = make\_tree(t, n - t, i, dim); =>t = &wp[0], n - t = 2, i = 1, dim = 2

len!=0 => go to if ((n = find\_median(t, t + len, i)))

if ((n = find\_median(t, t + len, i))) => sorting wp = {{{2, 3}, {4, 7}, {5, 4}, {7, 2}, {8, 1}, {9, 6}}}

n = &wp[0]

i = (1 + 1) % 2 => i = 0

n->left = make\_tree(t, n - t, i, dim); =>NULL(!len, so make\_tree return 0)

n->right = make\_tree(n + 1, t + len - (n + 1), i, dim); => n+1=&wp[1], t + len - (n + 1)=1, i = 0, dim = 2

len!=0=> go to if ((n = find\_median(t, t + len, i)))

if ((n = find\_median(t, t + len, i))) => sorting wp = {{{2, 3}, {4, 7}, {5, 4}, {7, 2}, {8, 1}, {9, 6}}}

n = &wp[1]

i = (0 + 1) % 2 => i = 1

n->left=>NULL(!len, so make\_tree return 0)

n->right=>NULL(!len, so make\_tree return 0)

n->right = make\_tree(n + 1, t + len - (n + 1), i, dim); =>n+1=&wp[3], t + len - (n + 1) = 3, i=1, dim=2

len!=0 => go to if ((n = find\_median(t, t + len, i)))

if ((n = find\_median(t, t + len, i))) => sorting wp = {{{2, 3}, {4, 7}, {5, 4}, {8, 1}, {7, 2}, {9, 6}}}

n = &wp[4]

i = i = (1 + 1) % 2 => i = 0

n->left = make\_tree(t, n - t, i, dim);=>t = &wp[3], n - t = 1, i = 0, dim = 2

len!=0=> go to if ((n = find\_median(t, t + len, i)))

if ((n = find\_median(t, t + len, i))) => sorting wp = {{{2, 3}, {4, 7}, {5, 4}, {8, 1}, {7, 2}, {9, 6}}}

n = &wp[3]

i = (0 + 1) % 2 => i = 1

n->left=>NULL(!len, so make\_tree return 0)

n->right=>NULL(!len, so make\_tree return 0)

n->right = make\_tree(n + 1, t + len - (n + 1), i, dim); =>n + 1 = &wp[5], t + len - (n-1) = 1, i = 0, dim = 2

len!=0=> go to if ((n = find\_median(t, t + len, i)))

if ((n = find\_median(t, t + len, i))) => sorting wp = {{{2, 3}, {4, 7}, {5, 4}, {8, 1}, {7, 2}, {9, 6}}}

n = &wp[5]

i = (0 + 1) % 2 => i = 1

n->left=>NULL(!len, so make\_tree return 0)

n->right=>NULL(!len, so make\_tree return 0)

* Hand-trace nearest function

nearest(root, &testNode, 0, 2, &found, &best\_dist);

i = 0, dim = 2

double d, dx, dx2;

nd = &testNode

root = &wp[2]

d = dist(&wp[2], &testNode, 2) => (4-2)\*(4-2) + (5-9)\*(5-9) == 20

dx = root->x[i] - nd->x[i];=>dx = root->x[0] - nd->x[0] == 5-9 == -4

dx2 = dx \* dx; => dx2 = 16

visited ++;

if (!\*best || d < \*best\_dist) => !\*best is true

\*best\_dist = 20

\*best = &wp[2]

if (!\*best\_dist) return; =>false

if (++i >= dim) i = 0; =>false

nearest(dx > 0 ? root->left : root->right, nd, i, dim, best, best\_dist);

dx==-4, nearest(root->right, nd, i, dim, best, best\_dist)=>nearest(&wp[4], &testNode, 1, 2, &wp[2], 20)

i=1, dim=2, nd = &testNode, root = &wp[4]

double d = dist(&wp[4], &testNode, 2) => (2-2)\*(2-2) + (7-9)\*(7-9) == 4

double dx = root->x[i] - nd->x[i];=>dx = root->x[1] - nd->x[1] == 2-2 == 0

dx2 = dx \* dx; => dx2 = 0

visited ++;

if (!\*best || d < \*best\_dist) => d M \*best\_dist is true

\*best\_dist = 4

\*best = &wp[4]

if (!\*best\_dist) return; =>false

if (++i >= dim) i = 0;

nearest(dx > 0 ? root->left : root->right, nd, i, dim, best, best\_dist);

dx==-4, nearest(root->right, nd, i, dim, best, best\_dist) => nearest(&wp[5], &testNode, 0, 2, &wp[4], 4)

i=0, dim=2, nd=&testNode, root=&wp[5]

double d = dist(&wp[5], &testNode, 2) => (6-2)\*(6-2) + (9-9)\*(9-9) == 16

double dx = root->x[i] - nd->x[i];=>dx = root->x[0] - nd->x[0] == 0-0 == 0

double dx2 = 0\*0;=> dx2 = 0

visited ++;

if (!\*best || d < \*best\_dist) => false

if (!\*best\_dist) return; =>false

if (++i >= dim) i = 0; => false

nearest(dx > 0 ? root->left : root->right, nd, i, dim, best, best\_dist);

dx==0, nearest(root->right, nd, i, dim, best, best\_dist) => nearest(NULL, &testNode, 1, 2, &wp[4], 4)

if (!root) return;

if (dx2 >= \*best\_dist) return;=>false

nearest(dx > 0 ? root->right : root->left, nd, i, dim, best, best\_dist);

dx==0, nearest(root->left, nd, i, dim, best, best\_dist)=>nearest(NULL, &testNode, 1, 2, &wp[4], 4)

if (!root) return;

if (dx2 >= \*best\_dist) return;=>false

nearest(dx > 0 ? root->right : root->left, nd, i, dim, best, best\_dist);

dx==-4, nearest(root->left, nd, i, dim, best, best\_dist)=>nearest(&wp[3], &testNode, 0, 2, &wp[4], 4)

i=0, dim-2, nd=&testNode, root=&wp[3]

double d = dist(&wp[3], &testNode, 2) => (1-2)\*(1-2) + (8-9)\*(8-9) == 2

double dx = root->x[i] - nd->x[i];=>dx = root->x[0] - nd->x[0] == 8-9 == -1

double dx2 = (-1)\*(-1) = 1

visited ++;

if (!\*best || d < \*best\_dist) => d < \*best\_dist is true

\*best\_dist = 2

\*best = &wp[3]

if (!\*best\_dist) return; =>false

if (++i >= dim) i = 0; => false

nearest(dx > 0 ? root->left : root->right, nd, i, dim, best, best\_dist);

dx==0, nearest(root->right, nd, i, dim, best, best\_dist) => nearest(NULL, &testNode, 1, 2, &wp[3], 2)

if (!root) return;

if (dx2 >= \*best\_dist) return;=>false

nearest(dx > 0 ? root->right : root->left, nd, i, dim, best, best\_dist);

dx==0, nearest(root->left, nd, i, dim, best, best\_dist)=>nearest(NULL, &testNode, 1, 2, &wp[3], 2)

if (!root) return;

if (dx2 >= \*best\_dist) return;=>false

nearest(dx > 0 ? root->right : root->left, nd, i, dim, best, best\_dist);

dx==-4, nearest(root->left, nd, i, dim, best, best\_dist)=>nearest(&wp[0], &testNode, 1, 2, &wp[2], 20)

i=1, dim=2, nd = &testNode, root = &wp[0]

double d = dist(&wp[0], &testNode, 2) => (3-2)\*(3-2) + (2-9)\*(2-9) == 50

double dx = root->x[i] - nd->x[i];=>dx = root->x[1] - nd->x[1] == 3-2 == 1

dx2 = dx \* dx; => dx2 = 1

visited ++;

if (!\*best || d < \*best\_dist) => false

if (!\*best\_dist) return; =>false

if (++i >= dim) i = 0;

nearest(dx > 0 ? root->left : root->right, nd, i, dim, best, best\_dist);

dx==1, nearest(root->left, nd, i, dim, best, best\_dist) => nearest(NULL, &testNode, 0, 2, &wp[2], 20)

if (!root) return;

if (dx2 >= \*best\_dist) return; => false

nearest(dx > 0 ? root->right : root->left, nd, i, dim, best, best\_dist);

dx==1, nearest(root->right, nd, i, dim, best, best\_dist)=>nearest(&wp[1], &testNode, 0, 2, &wp[2], 20)

i=0, dim-2, nd=&testNode, root=&wp[1]

double d = dist(&wp[1], &testNode, 2) => (7-2)\*(7-2) + (4-9)\*(4-9) == 50

double dx = root->x[i] - nd->x[i];=>dx = root->x[0] - nd->x[0] == 4-9 == -5

double dx2 = (-5)\*(-5) = 25

visited ++;

if (!\*best || d < \*best\_dist) => false

if (!\*best\_dist) return; =>false

if (++i >= dim) i = 0; => false

nearest(dx > 0 ? root->left : root->right, nd, i, dim, best, best\_dist);

dx==-5, nearest(root->right, nd, i, dim, best, best\_dist) => nearest(NULL, &testNode, 1, 2, &wp[2], 20)

if (!root) return;

if (dx2 >= \*best\_dist) return;

nearest(dx > 0 ? root->right : root->left, nd, i, dim, best, best\_dist);

dx==0, nearest(root->left, nd, i, dim, best, best\_dist)=>nearest(NULL, &testNode, 1, 2, &wp[2], 20)

if (!root) return;