```cpp
struct Point {
    long long x, y;
};

// squared distance between two points
long long dist2(const Point &a, const Point &b) {
    long long dx = a.x - b.x;
    long long dy = a.y - b.y;
    return dx*dx + dy*dy;
}

// recursive function
long long solve(vector<Point> &pts, int l, int r, vector<Point> &tmp) {
    if (r - l <= 3) {
        long long d = LLONG_MAX;
        for (int i = l; i < r; i++) {
            for (int j = i + 1; j < r; j++) {
                d = min(d, dist2(pts[i], pts[j]));
            }
        }
        // sort by y for merge
        sort(pts.begin() + l, pts.begin() + r, [](auto &a, auto &b) {
            return a.y < b.y;
        });
        return d;
    }

    int mid = (l + r) / 2;
    long long midx = pts[mid].x;

    long long d = min(solve(pts, l, mid, tmp), solve(pts, mid, r, tmp));

    // merge by y
    merge(pts.begin() + l, pts.begin() + mid,
          pts.begin() + mid, pts.begin() + r,
          tmp.begin(), [](auto &a, auto &b) {
              return a.y < b.y;
          });
    copy(tmp.begin(), tmp.begin() + (r - l), pts.begin() + l);

    // build strip of points within sqrt(d) of mid line
    vector<Point> strip;
    for (int i = l; i < r; i++) {
        if ((pts[i].x - midx) * (pts[i].x - midx) < d) {
            strip.push_back(pts[i]);
        }
    }

    // check neighbors in strip
    for (int i = 0; i < (int)strip.size(); i++) {
        for (int j = i + 1; j < (int)strip.size() && j <= i + 7; j++) {
            d = min(d, dist2(strip[i], strip[j]));
        }
    }
    return d;
}

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

    int n;
    cin >> n;
    vector<Point> pts(n);
    for (int i = 0; i < n; i++) {
        cin >> pts[i].x >> pts[i].y;
    }

    // sort by x first
    sort(pts.begin(), pts.end(), [](auto &a, auto &b) {
        if (a.x == b.x) return a.y < b.y;
        return a.x < b.x;
    });

    vector<Point> tmp(n);
    cout << solve(pts, 0, n, tmp) << "\n";
}
```

Mid euclidian(divide)

Min Euclidian (divide)

```cpp
struct Point {
    long long x, y;
};

// squared distance between two points
long long dist2(const Point &a, const Point &b) {
    long long dx = a.x - b.x;
    long long dy = a.y - b.y;
    return dx*dx + dy*dy;
}

// recursive function
long long solve(vector<Point> &pts, int l, int r, vector<Point> &tmp) {
    if (r - l <= 3) {
        long long d = LLONG_MAX;
        for (int i = l; i < r; i++) {
            for (int j = i + 1; j < r; j++) {
                d = min(d, dist2(pts[i], pts[j]));
            }
        }
        // sort by y for merge
        sort(pts.begin() + l, pts.begin() + r, [](auto &a, auto &b) {
            return a.y < b.y;
        });
        return d;
    }

    int mid = (l + r) / 2;
    long long midx = pts[mid].x;

    long long d = min(solve(pts, l, mid, tmp), solve(pts, mid, r, tmp));

    // merge by y
    merge(pts.begin() + l, pts.begin() + mid,
          pts.begin() + mid, pts.begin() + r,
          tmp.begin(), [](auto &a, auto &b) {
              return a.y < b.y;
          });
    copy(tmp.begin(), tmp.begin() + (r - l), pts.begin() + l);

    // build strip of points within sqrt(d) of mid line
    vector<Point> strip;
    for (int i = l; i < r; i++) {
        if ((pts[i].x - midx) * (pts[i].x - midx) < d) {
            strip.push_back(pts[i]);
        }
    }
```