BruteForce method:

```
vector<double> BruteForce(vector<pair<double,double> > vec, int size, int& count)
{
	double min = FLT_MAX;
	vector<double> v(5);
	for(int i=0; i<size; i++){
		for(int j=i+1; j<size; j++){
			count ++;
			if(dist(vec[i], vec[j]) < min){
				min = dist(vec[i], vec[j]);
				v[0] = (dist(vec[i], vec[j]));
				v[1] = (vec[i].first);
				v[2] = (vec[i].second);
				v[3] = (vec[j].first);
				v[4] = (vec[j].second);
			}
		}
	}
	return v;
}
```

Time complexity is O(n^2) because there are two for loops, which is n*n.

Plot: Compare point by point, record it as the shortest distance, min. If there is a d shorter than min, min = d. At the end of the comparison, min will be the shortest distance.

Divide and conquer method

```
vector<double> findmin(vector<pair<double,double> > v, int n, int& count)
{
	if (n <= 3)
	{
		return bruteForce(v, n, count);
	}
	int mid = n/2;
	pair<double,double> midPoint = v[mid];

	vector<pair<double,double> > v1(v.begin(), v.begin() + mid);
	vector<double> left = findmin(v1, mid, count);
	vector<pair<double,double> > v2(v.begin() + mid, v.end());
	vector<double> right = findmin(v2, n-mid, count);
	vector<double> d = min(left, right);
	double dis = d[0];
```

```
count ++;

vector<pair<double,double> > strip;//holds points in strip
for (int i=0; i<n; i++){
      if (abs(v[i].first - midPoint.first) < dis){
            strip.push_back(v[i]);
      }
}
sort(strip.begin(), strip.end(), myComparison);//strip sorted
count = count + strip.size();
 return min(d, stripClosest(strip, strip.size(), dis, count));
 }
```

Time complexity of divide and conquer is O(n(Logn)^2). The reason why there is a (Logn)^2 is because I use merge sort algorithm to sort y coordinate instead of directly merge(which takes O(n))

Plot: Divide all the points into half and keep doing so until groups have points less or equal than 3. Then use brute force find min distance d in each group. For each adjacent group, collect points which have distance < d(d is the min of two adjacent group) into a vector(strip). Then sort all the points in terms of y coordinate. Then compute shortest distance in this strip. If it is smaller than d, then make it min. Recursively doing so, we will get the shortest distance.