### Week 25: Geometry & Computational Geometry Advanced Problems

**Topics:** - Convex Hull Applications: Diameter, Width, and Area - Closest Pair of Points - Sweep Line for Intersections and Counting - Polygon Triangulation and Area Computations - Circle and Line Intersection, Tangents - Geometric Transformations (Rotation, Reflection, Scaling)

**Weekly Tips:** - Convex hull can be used with rotating calipers for diameter and width problems. - Use divide-and-conquer for closest pair of points to achieve O(n log n). - Sweep line helps in interval and intersection problems efficiently. - Polygon area via shoelace formula and centroid calculations. - Carefully handle floating point precision and integer coordinates.

**Problem 1: Closest Pair of Points** **Link:** [CSES Closest Pair](https://cses.fi/problemset/task/2167/) **Difficulty:** Advanced

**C++ Solution with Explanation Comments:**

#include <bits/stdc++.h>  
using namespace std;  
struct Point{ long long x,y; };  
long long dist(Point a, Point b){  
 long long dx=a.x-b.x, dy=a.y-b.y;  
 return dx\*dx+dy\*dy;  
}  
long long closestPair(vector<Point>& pts,int l,int r){  
 if(r-l<=3){  
 long long mn=LLONG\_MAX;  
 for(int i=l;i<r;i++) for(int j=i+1;j<r;j++) mn=min(mn,dist(pts[i],pts[j]));  
 sort(pts.begin()+l,pts.begin()+r,[](Point a,Point b){return a.y<b.y;});  
 return mn;  
 }  
 int m=(l+r)/2;  
 long long midx=pts[m].x;  
 long long d=min(closestPair(pts,l,m),closestPair(pts,m,r));  
 vector<Point> temp;  
 merge(pts.begin()+l,pts.begin()+m,pts.begin()+m,pts.begin()+r,temp.begin(),[](Point a,Point b){return a.y<b.y;});  
 copy(temp.begin(),temp.end(),pts.begin()+l);  
 vector<Point> strip;  
 for(int i=l;i<r;i++) if(abs(pts[i].x-midx)<d) strip.push\_back(pts[i]);  
 for(int i=0;i<strip.size();i++)  
 for(int j=i+1;j<strip.size() && (strip[j].y-strip[i].y)<d;j++) d=min(d,dist(strip[i],strip[j]));  
 return d;  
}  
int main(){  
 int n; cin>>n;  
 vector<Point> pts(n);  
 for(int i=0;i<n;i++) cin>>pts[i].x>>pts[i].y;  
 sort(pts.begin(),pts.end(),[](Point a,Point b){return a.x<b.x;});  
 cout<<closestPair(pts,0,n)<<endl;  
}

**Explanation Comments:** - Divide-and-conquer splits points into halves. - Merge step sorts by y-coordinate for cross-boundary check. - Strip contains points close to the division line. - Efficient O(n log n) solution compared to naive O(n^2).

**Problem 2: Polygon Area (Shoelace Formula)** **Link:** [CSES Polygon Area](https://cses.fi/problemset/task/2166/) **Difficulty:** Intermediate

**C++ Solution with Explanation Comments:**

#include <bits/stdc++.h>  
using namespace std;  
struct Point{ long long x,y; };  
int main(){  
 int n; cin>>n;  
 vector<Point> poly(n);  
 for(int i=0;i<n;i++) cin>>poly[i].x>>poly[i].y;  
 long long area=0;  
 for(int i=0;i<n;i++){  
 int j=(i+1)%n;  
 area+=poly[i].x\*poly[j].y - poly[j].x\*poly[i].y;  
 }  
 cout<<fixed<<setprecision(1)<<abs(area)/2.0<<endl;  
}

**Explanation Comments:** - Shoelace formula computes polygon area by summing cross products of consecutive vertices. - Absolute value divided by 2 gives area. - Works for convex and concave polygons.

**End of Week 25** - Practice geometric algorithms extensively, focusing on precision and edge cases. - Rotate calipers, closest pair, and polygon operations are common in ACM-ICPC geometry problems. - Sweep line and divide-and-conquer techniques optimize computational geometry tasks.