Problem Tutorial: "Distance Between Two Closest Points (on a Line)"

Sort all the points and check each pair of neighboring (adjacent) points calculating distance between them.

```
sort(x.begin(), x.end());
int result = INT_MAX;
for (int i = 0; i + 1 < n; i++)
    result = min(result, x[i + 1] - x[i]);
cout << result << endl;
```

Problem Tutorial: "Intersection of Segments (on a Line)"

An intersection (if any) of segments in 1D is such a segment that its left bound is maximum among all left bounds and its right bound is minimum among all right bounds. The following code solves the problems:

```
int n;
cin >> n;
vector<int> l(n), r(n);
for (int i = 0; i < n; i++) {
    cin >> l[i] >> r[i];
    if (l[i] > r[i])
        swap(l[i], r[i]);
}
int L = INT_MIN;
int R = INT_MAX;
for (int i = 0; i < n; i++) {
    L = max(L, l[i]);
    R = min(R, r[i]);
}
cout << max(0, R - L) << endl;
```

Problem Tutorial: "Two Rectangles Intersection"

You can solve the problem independently by coordinates: check that the horizontal projections of the rectangles intersect and check that the vertical projections of the rectangles intersect. If both projections intersect then the rectangles intersect (and vice versa).

```
int x11, y11, x12, y12;
cin >> x11 >> y11 >> x12 >> y12;
int l1 = min(x11, x12);
int r1 = max(x11, x12);
int a1 = min(y11, y12);
int b1 = max(y11, y12);

int x21, y21, x22, y22;
cin >> x21 >> y21 >> x22 >> y22;
int l2 = min(x21, x22);
int r2 = max(x21, x22);
int a2 = min(y21, y22);
int b2 = max(y21, y22);

bool intersect = (max(l1, l2) <= min(r1, r2) && max(a1, a2) <= min(b1, b2));
cout << (intersect ? "Yes" : "No") << endl;
```

Problem Tutorial: "Covering Rectangle"

Just calculate minimal x among all the x_i, it will be the left bound of the minimal covering rectangle. Similarly, calculate maximal x among all the x_i, it will be the right bound of the minimal covering rectangle. Use the same strategy to calculate the top and bottom bounds.

```
int n;
cin >> n;
int min_x = INT_MIN;
int max_x = INT_MAX;
int min_y = INT_MIN;
int max_y = INT_MAX;
for (int i = 0; i < n; i++) {
    int x, y;
    cin >> x >> y;
    min_x = max(min_x, x);
    max_x = min(max_x, x);
    min_y = max(min_y, y);
    max_y = min(max_y, y);
}
cout << (max_x - min_x) * (max_y - min_y) << endl;
```

Problem Tutorial: "Segment Middle"

Take the middle point of the given segment. It is exactly average between coordinates of the the endpoints.

```
double x1, y1, x2, y2;
cin >> x1 >> y1 >> x2 >> y2;
cout << (x1 + x2) / 2.0 << " " << (y1 + y2) / 2.0 << endl;
```

Problem Tutorial: "K-point"

Calculate vector from the first point to the second point, say (dx, dy). Multiply it by $\frac{k}{k+1}$ to scale it in the required ratio. Move the first point to the resulting vector to get the answer. Example code is:

```
double x1, y1, x2, y2;
cin >> x1 >> y1 >> x2 >> y2;
int k;
cin >> k;
double dx = x2 - x1, dy = y2 - y1;
dx = dx / (k + 1) * k;
dy = dy / (k + 1) * k;
printf("%.10f %.10f\n", x1 + dx, y1 + dy);
```

Problem Tutorial: "Triangle Type"

Suppose the given triangle is ABC. Consider three dot products $\vec{AB} \cdot \vec{AC}$, $\vec{BA} \cdot \vec{BC}$ and $\vec{CA} \cdot \vec{CB}$. If any of them is negative, the triangle is obtuse. If any of them is zero, the triangle is right. Otherwise, the triangle is acute.

Problem Tutorial: "Is on Segment?"

Suppose the point is P and the segment is AB. At first, check P is inside the bounding box of AB: $\min(A_x, B_x) \le P_x \le \max(A_x, B_x)$ and $\min(A_y, B_y) \le P_y \le \max(A_y, B_y)$. Also check that the line AB contains P: $\vec{AB} \times \vec{AP} = 0$.

Problem Tutorial: "Arrangement of Lines"

Find two lines going through the each pair of points, say A_1, B_1, C_1 and A_2, B_2, C_2. A determinant $det(a, b, c, d) = ad - bc$.

If $det(A_1, B_1, A_2, B_2) = 0$ it means that lines parallel or coincide:

- if $det(C_1, B_1, C_2, B_2) = det(A_1, C_1, A_2, C_2) = 0$, the lines are coincide;
- if $det(C_1, B_1, C_2, B_2) \neq 0$ or $det(A_1, C_1, A_2, C_2) \neq 0$, the lines are parallel.

If $det(A_1, B_1, A_2, B_2) \neq 0$ exactly one intersection point exists: $x = -det(C_1, B_1, C_2, B_2)/det(A_1, B_1, A_2, B_2)$, $y = -det(A_1, C_1, A_2, C_2)/det(A_1, B_1, A_2, B_2)$.

Problem Tutorial: "Distance from Point to Segment"

Assume the given point is P and the given segment is AB. Calculate $d_A = |PA|$ and $d_B = |PB|$. If the projection of P to the line AB is outside the segment AB, then the expected distance is $\min(d_A, d_B)$. If the projection of P to the line AB is inside the segment AB, then the expected distance is distance to the line AB.

To check if the projection of P to the line AB is inside the segment AB you should check the both angles PAB and PBA to be non-obtuse. Use dot products to check it.

Problem Tutorial: "Polygon Area"

If the vertices of a polygon are given in clockwise or counterclockwise order you may use any of the following formulas:

$$A = |\frac{\sum_{0 \le i < n} (p_i.x - p_{i+1}.x)(p_i.y + p_{i+1}.y)}{2}|$$

$$A = |\frac{\sum_{0 \le i < n} \overrightarrow{p_0 p_i} \times \overrightarrow{p_0 p_{i+1}}}{2}|$$

Here assume $p_n = p_0$.

Problem Tutorial: "Triangle Perimeter"

Intersect each pair of lines with Cramer's rule. After it just print a perimeter of the triangle.

Problem Tutorial: "Symmetrical Point"

Assume the given point is F and the points on the line are P and Q. Find line coefficients A, B and C. If $A \cdot F_x + B \cdot F_y + C = 0$, then the point F is on the line.

Normalize line equation to make normal vector (A, B) to be of length 1. To do it calculate $l = \sqrt{A^2 + B^2}$ and divide A, B and C by l.

After if calculate signed distance from F to the line: $d = A \cdot F_x + B \cdot F_y$. Move F on distance d twice to the direction opposite to normal vector: $G_x = F_x - 2 \cdot d \cdot A$, $G_y = F_y - 2 \cdot d \cdot B$. The point G is the required point.

Problem Tutorial: "Two Segments"

1. Check that segment bounding boxes intersect with exactly 4 lines of code like "`if (max(x11, x12) < min(x21, x22)) return false;`".
2. Build lines $L_1 = (A_1, B_1, C_1)$ and $L_2 = (A_2, B_2, C_2)$.

3. Find $det = A_1B_2 - A_2B_1$ ($det = 0$ if and only if some line degenerates or lines are parallel/coincide).

4. Check if lines coincide (do not test determinants, but test an endpoint of the second segment to lie on the first line AND an endpoint of the first segment to lie on the second line). If $det = 0$ and lines don't coincide then no intersections.

5. If lines are coincide then they intersect by segment AB, there $A = \max(\min(p_1, q_1), \min(p_2, q_2))$ and $B = \min(\max(p_1, q_1), \max(p_2, q_2))$. If $A = B$ then there is exactly 1 intersection else segments intersect by segment AB.

6. The only case has left: $det \neq 0$. Find intersection of lines and check if it lies on the both segments (use bounding boxes) like this: `in(c.X, p1.X, q1.X) && in(c.Y, p1.Y, q1.Y) && in(c.X, p2.X, q2.X) && in(c.Y, p2.Y, q2.Y)`. Here `in(t,a,b)` tests that t is between a and b: `in(t,a,b) { return min(a,b) <= t && t <= max(a,b); }`.