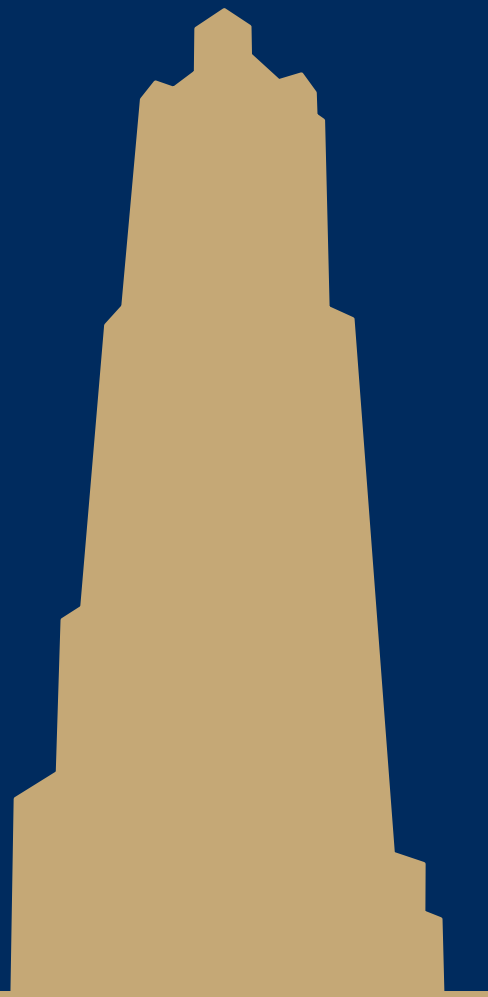
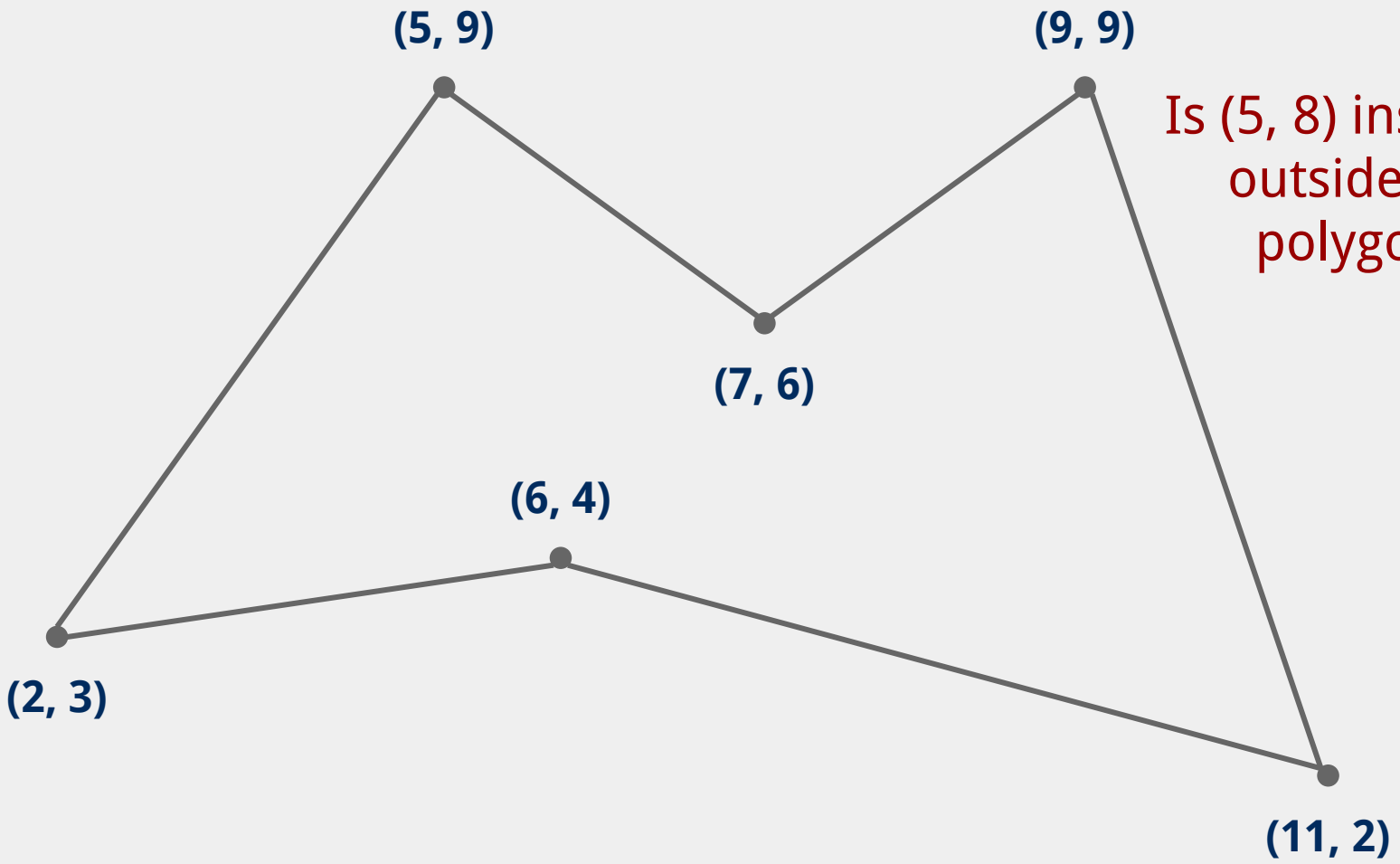
CS/COE 1501

www.cs.pitt.edu/~nlf4/cs1501/

Point in Polygon

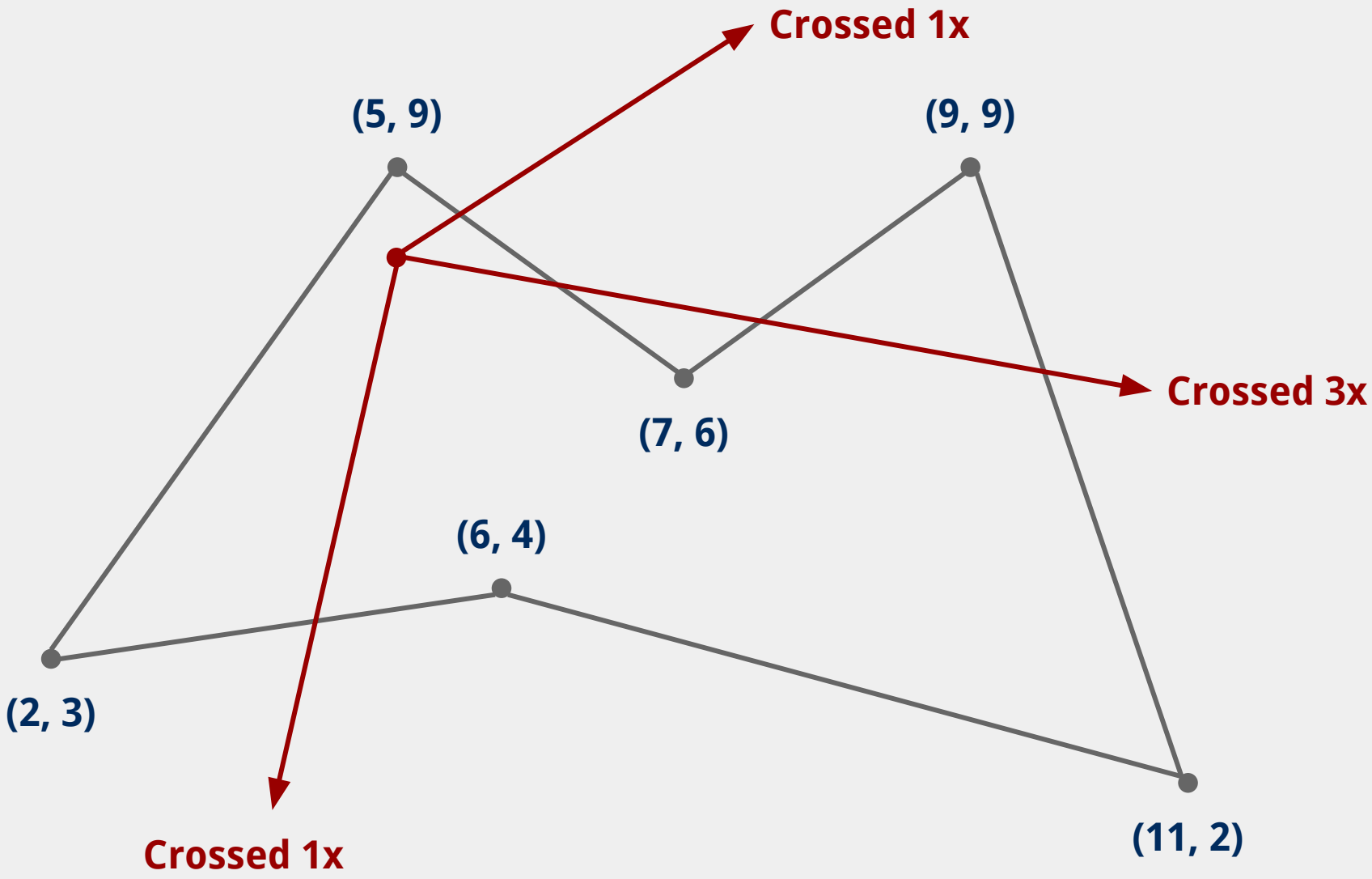


Determining if a point is in/out of a polygon

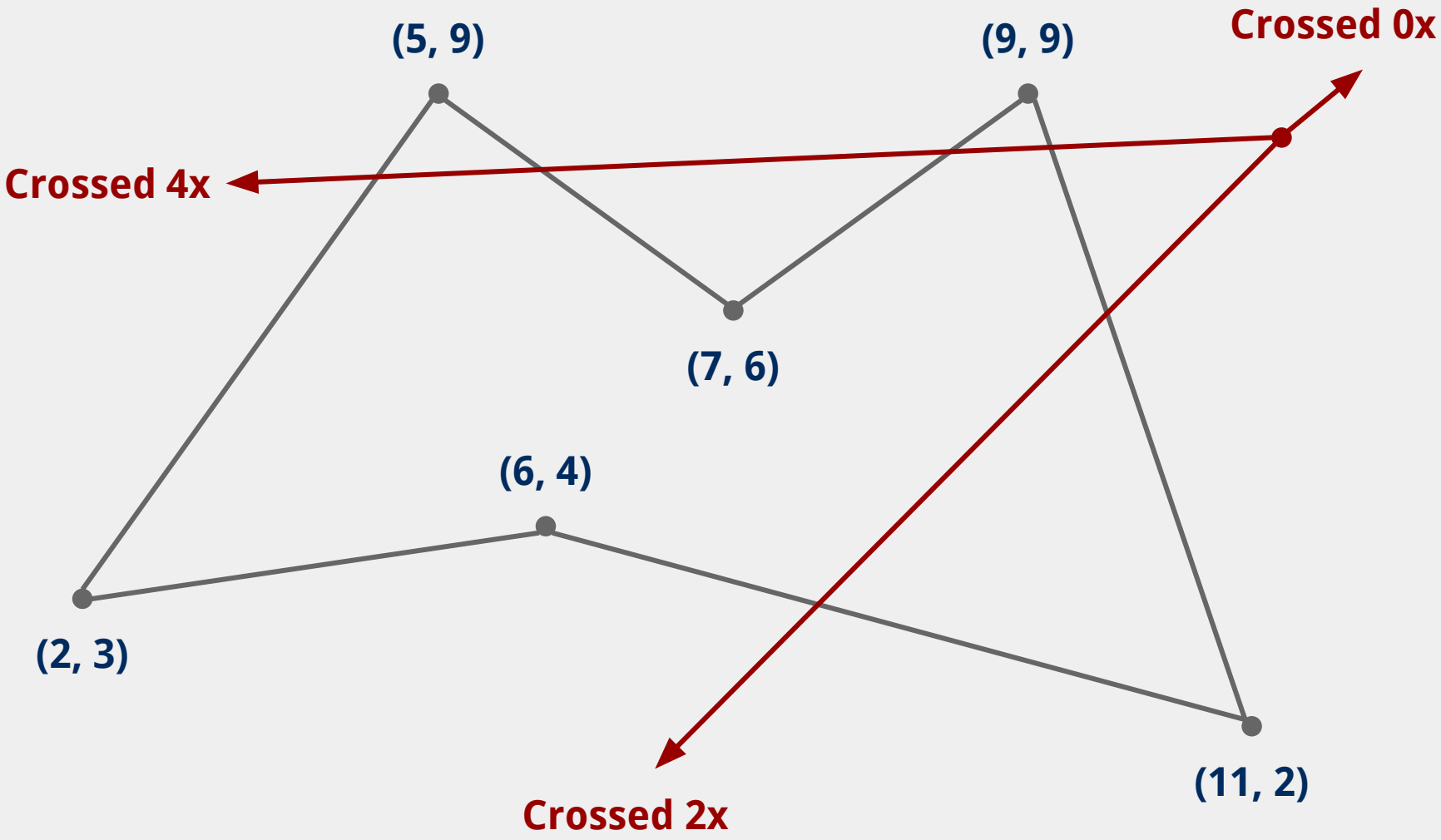


Is (5, 8) inside or outside the polygon?

Raytracing (Point inside example)



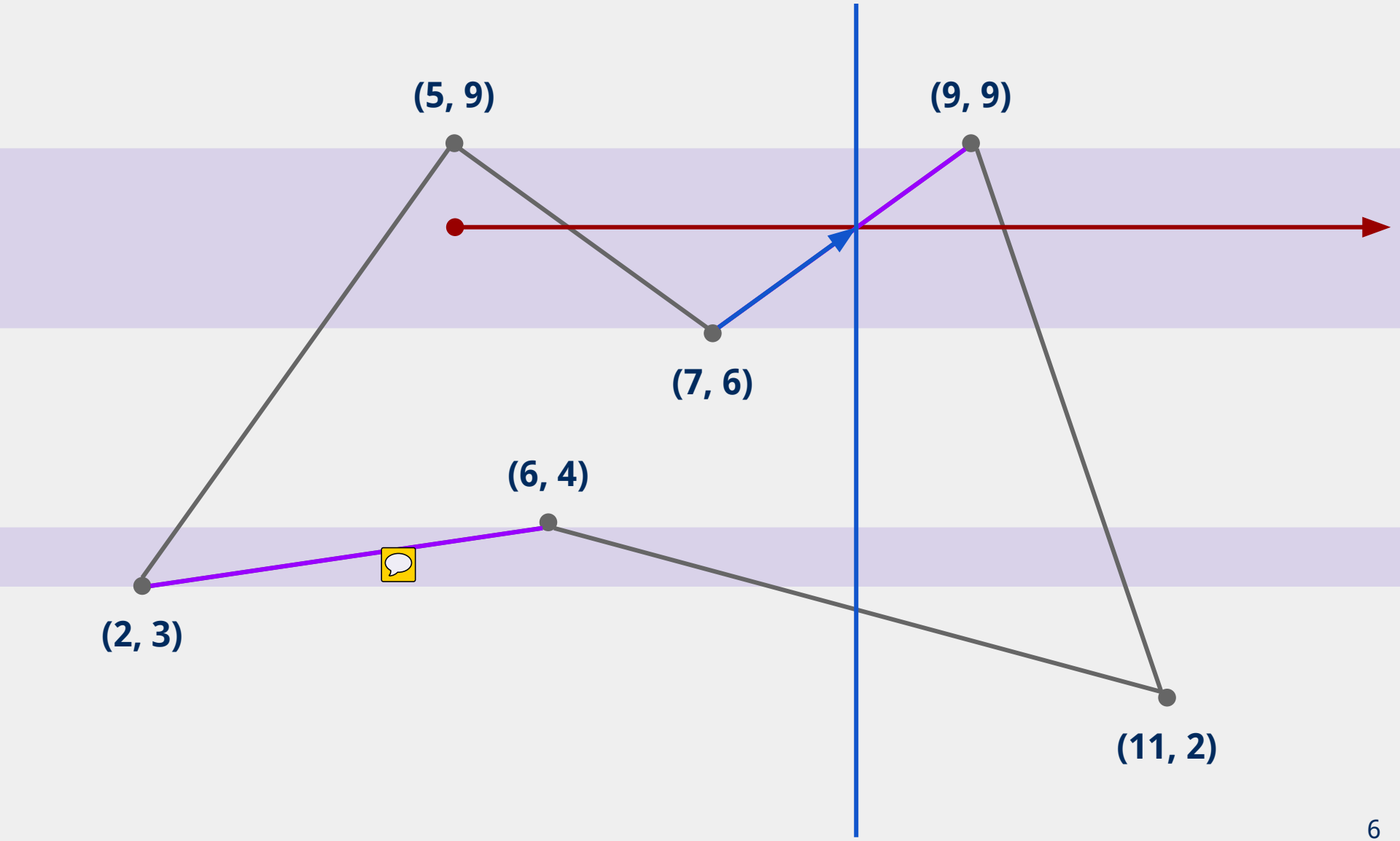
Raytracing (Point inside example)



How can we implement raytracing?

- Assume we are given:
  - The point to check
  - An array of points that make up the vertices of the polygon

Raytracing approach



Raytracing implementation

```
public boolean contains2(Point p) {
  int crossings = 0;
  for (int i = 0; i < N; i++) {
    int j = i + 1;
    boolean cond1 = (a[i].y <= p.y) && (p.y < a[j].y);
    boolean cond2 = (a[j].y <= p.y) && (p.y < a[i].y);
    if (cond1 || cond2) {
      if (p.x < (a[j].x - a[i].x) * (p.y - a[i].y)
                / (a[j].y - a[i].y) + a[i].x)
        crossings++;
    }
  }
  if (crossings % 2 == 1) return true;
  else                    return false;
}
```

- `Point` has defined x and y attributes
- a is an array of N-1 different vertices (as `Point` objects)

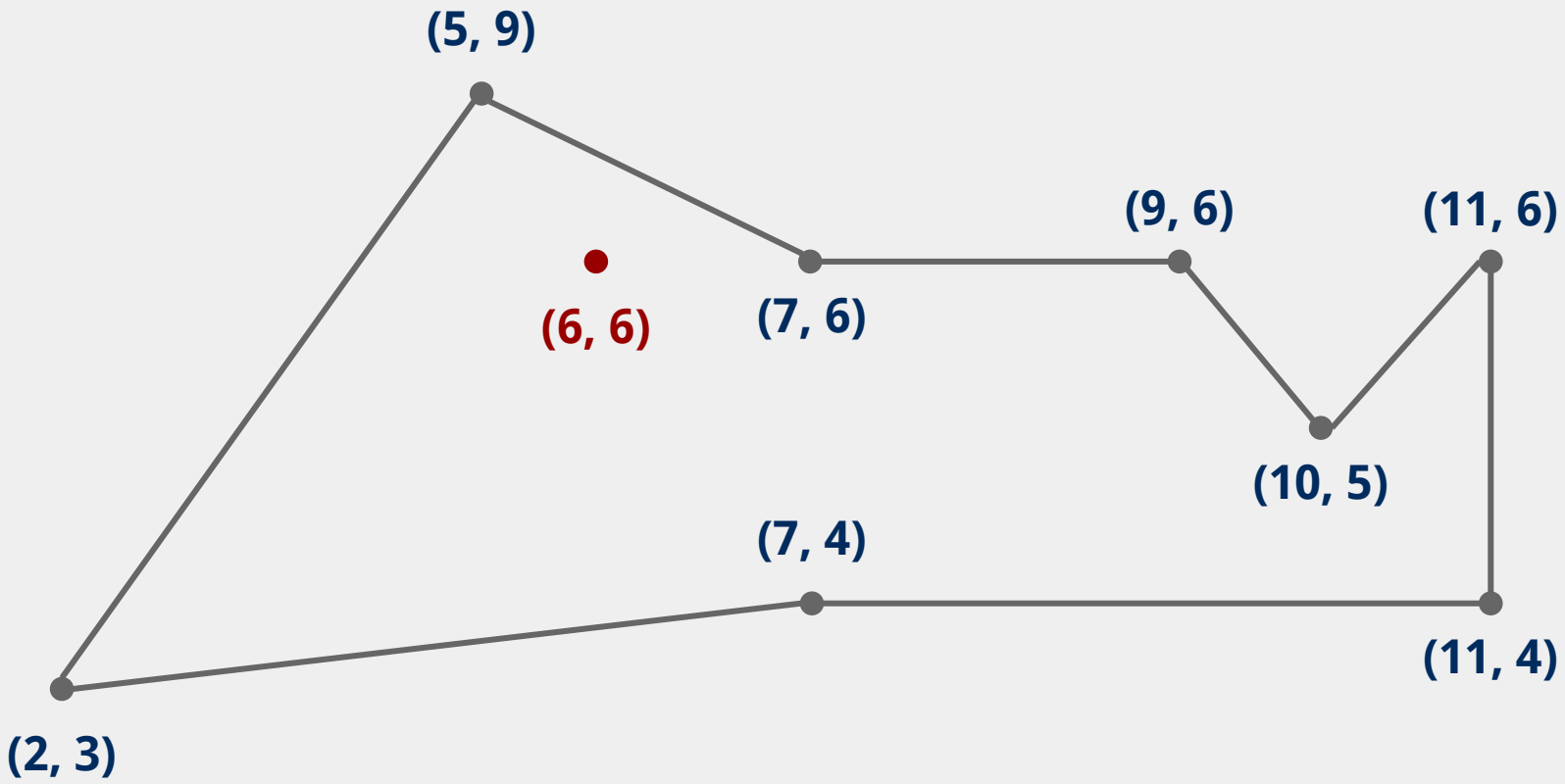
Line slope review

- Typically slopes are rise/run
- x * (rise/run) yields vertical distance line will travel after x horizontal until
  - Hence y = x * (rise/run) + y-intercept is the equation for a line
- y * (run/rise) yields horizontal distance line will travel after y vertical until
  - Hence (a[i].y - p.y) * (run/rise) + a[i].x gives x-intercept at y coordinate p.y

Another raytracing implementation (in C)

```
int pnpoly(int nvert, float *vertx, float *verty, float testx,
           float testy) {
  int i, j, c = 0;
  for (i = 0, j = nvert-1; i < nvert; j = i++) {
    if ( ((verty[i]>testy) != (verty[j]>testy)) &&
         (testx < (vertx[j]-vertx[i]) * (testy-verty[i])
                  / (verty[j]-verty[i]) + vertx[i]) )
      c = !c;
  }
  return c;
}
```

Raytracing oddity



(11, 2)