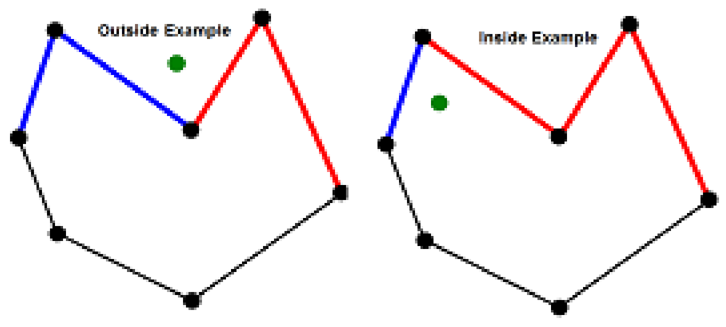
How to check if a given point lies inside or outside a polygon?

Given a polygon and a point ‘p’, find if ‘p’ lies inside the polygon or not. The points lying on the border are considered inside.



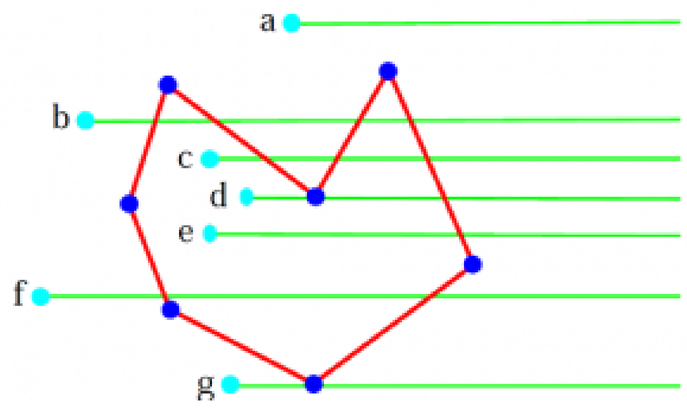
We strongly recommend to see the following post first.
How to check if two given line segments intersect?

Following is a simple idea to check whether a point is inside or outside.

```
1) Draw a horizontal line to the right of each point and extend it to infinity

1) Count the number of times the line intersects with polygon edges.

2) A point is inside the polygon if either count of intersections is odd or
   point lies on an edge of polygon.  If none of the conditions is true, then
   point lies outside.
```



How to handle point ‘g’ in the above figure?
Note that we should returns true if the point lies on the line or same as one of the vertices of the given polygon. To handle this, after checking if the line from ‘p’ to extreme intersects, we check whether ‘p’ is colinear with vertices of current line of polygon. If it is coliear, then we check if the point ‘p’ lies on current side of polygon, if it lies, we return true, else false.

Following is C++ implementation of the above idea.

```
// A C++ program to check if a given point lies inside a
// Refer http://www.geeksforgeeks.org/check-if-two-given
// for explanation of functions onSegment(), orientation
#include <iostream>
using namespace std;

// Define Infinite (Using INT_MAX caused overflow proble
#define INF 10000

struct Point
{
    int x;
    int y;
};

// Given three colinear points p, q, r, the function che
// point q lies on line segment 'pr'
bool onSegment(Point p, Point q, Point r)
{
    if (q.x <= max(p.x, r.x) && q.x >= min(p.x, r.x) &&
            q.y <= max(p.y, r.y) && q.y >= min(p.y, r.y))
        return true;
    return false;
}

// To find orientation of ordered triplet (p, q, r).
// The function returns following values
// 0 --> p, q and r are colinear
// 1 --> Clockwise
// 2 --> Counterclockwise
int orientation(Point p, Point q, Point r)
{
    int val = (q.y - p.y) * (r.x - q.x) -
              (q.x - p.x) * (r.y - q.y);

    if (val == 0) return 0;  // colinear
    return (val > 0)? 1: 2; // clock or counterclock wis
}

// The function that returns true if line segment 'p1q1'
// and 'p2q2' intersect.
bool doIntersect(Point p1, Point q1, Point p2, Point q2)
{
    // Find the four orientations needed for general and
    // special cases
    int o1 = orientation(p1, q1, p2);
    int o2 = orientation(p1, q1, q2);
    int o3 = orientation(p2, q2, p1);
    int o4 = orientation(p2, q2, q1);

    // General case
    if (o1 != o2 && o3 != o4)
        return true;
    // Special Cases
    // p1, q1 and p2 are colinear and p2 lies on segment
    if (o1 == 0 && onSegment(p1, p2, q1)) return true;

    // p1, q1 and p2 are colinear and q2 lies on segment
    if (o2 == 0 && onSegment(p1, q2, q1)) return true;

    // p2, q2 and p1 are colinear and p1 lies on segment
    if (o3 == 0 && onSegment(p2, p1, q2)) return true;

     // p2, q2 and q1 are colinear and q1 lies on segmen
    if (o4 == 0 && onSegment(p2, q1, q2)) return true;

    return false; // Doesn't fall in any of the above ca
}

// Returns true if the point p lies inside the polygon[]
bool isInside(Point polygon[], int n, Point p)
{
    // There must be at least 3 vertices in polygon[]
    if (n < 3)  return false;

    // Create a point for line segment from p to infinit
    Point extreme = {INF, p.y};

    // Count intersections of the above line with sides
    int count = 0, i = 0;
    do
    {
        int next = (i+1)%n;

        // Check if the line segment from 'p' to 'extrem
        // with the line segment from 'polygon[i]' to 'p
        if (doIntersect(polygon[i], polygon[next], p, ex
        {
            // If the point 'p' is colinear with line se
            // then check if it lies on segment. If it l
            // otherwise false
            if (orientation(polygon[i], p, polygon[next]
               return onSegment(polygon[i], p, polygon[n

            count++;
        }
        i = next;
    } while (i != 0);

    // Return true if count is odd, false otherwise
    return count&1;  // Same as (count%2 == 1)
}

// Driver program to test above functions
int main()
{
    Point polygon1[] = {{0, 0}, {10, 0}, {10, 10}, {0, 1
    int n = sizeof(polygon1)/sizeof(polygon1[0]);
    Point p = {20, 20};
    isInside(polygon1, n, p)? cout << "Yes \n": cout <<

    p = {5, 5};
    isInside(polygon1, n, p)? cout << "Yes \n": cout <<

    Point polygon2[] = {{0, 0}, {5, 5}, {5, 0}};
    p = {3, 3};
    n = sizeof(polygon2)/sizeof(polygon2[0]);
    isInside(polygon2, n, p)? cout << "Yes \n": cout <<

    p = {5, 1};
    isInside(polygon2, n, p)? cout << "Yes \n": cout <<

    p = {8, 1};
    isInside(polygon2, n, p)? cout << "Yes \n": cout <<

    Point polygon3[] =  {{0, 0}, {10, 0}, {10, 10}, {0,
    p = {-1,10};
    n = sizeof(polygon3)/sizeof(polygon3[0]);
    isInside(polygon3, n, p)? cout << "Yes \n": cout <<

    return 0;
}
```

Output:

```
No
Yes
Yes
Yes
No
No
```

Time Complexity: O(n) where n is the number of vertices in the given polygon.

Source:
http://www.dcs.gla.ac.uk/~pat/52233/slides/Geometry1x1.pdf