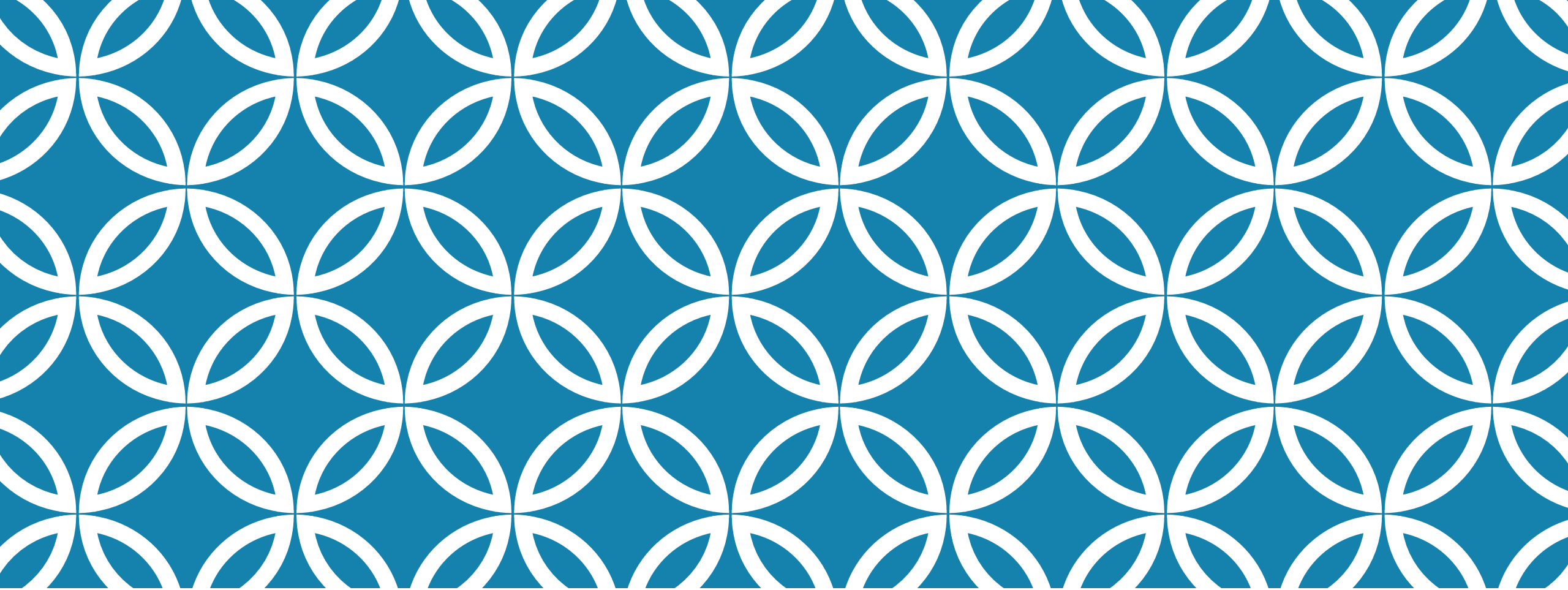


ALGORITHMS (COMP-2350)

Professor Salem Othman

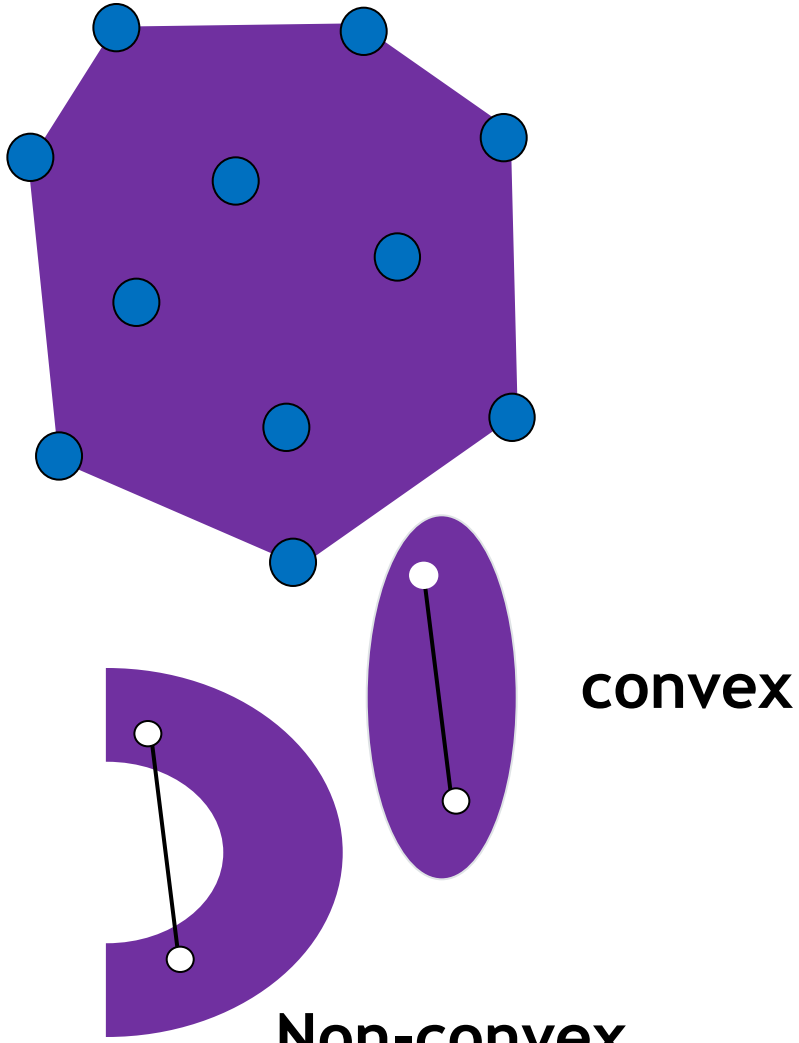
Lab 5 (**Convex-Hull by Brute Force**)

TASK

Implement Convex-Hull by Brute Force in Python

CONVEX-HULL BY BRUTE FORCE

http://kaikuroda.com/testjava/



convex

Non-convex

Convex hull Problem: Find smallest convex polygon enclosing n points on the plane

Convex:

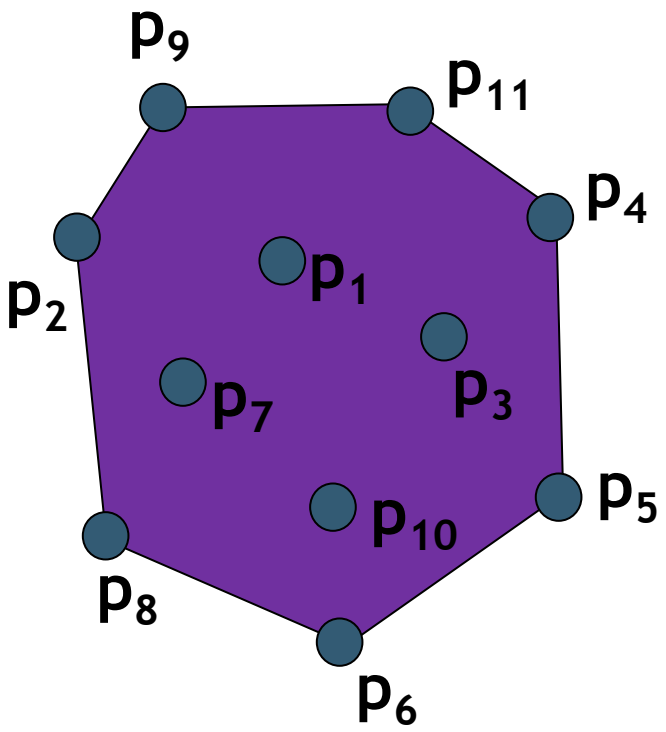
- A geometric figure with no indentations.
- Formally, a geometric figure is convex if every line segment connecting interior points is entirely contained within the figure's interior.

CONVEX-HULL BY BRUTE FORCE

Convex hull Problem: Find smallest convex polygon enclosing n points on the plane

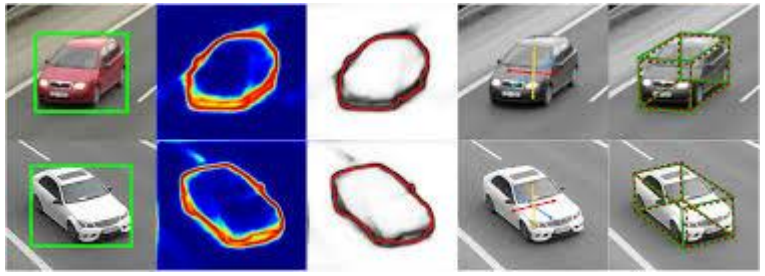
Input: $p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9, p_{10}, p_{11}$

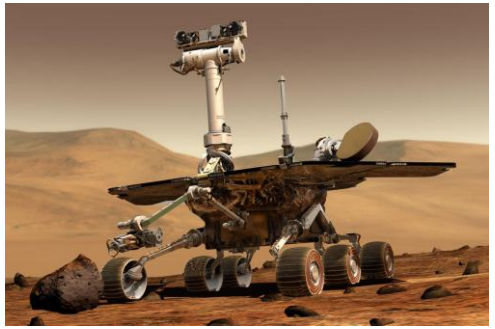
Output: $p_2, p_9, p_{11}, p_4, p_5, p_6, p_8, p_2$

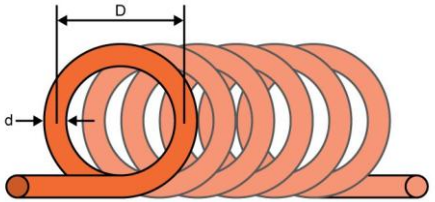


CONVEX-HULL BY BRUTE FORCE: APPLICATION DOMAIN*

- Computer visualization, ray tracing
  - (e.g. video games, replacement of bounding boxes)
- Path finding
  - (e.g. embedded AI of Mars mission rovers)
- Geographical Information Systems (GIS)
  - (e.g. computing accessibility maps)
- Visual pattern matching
  - (e.g. detecting car license plates)
- Verification methods
  - (e.g. bounding of Number Decision Diagrams)
- Geometry
  - (e.g. diameter computation)





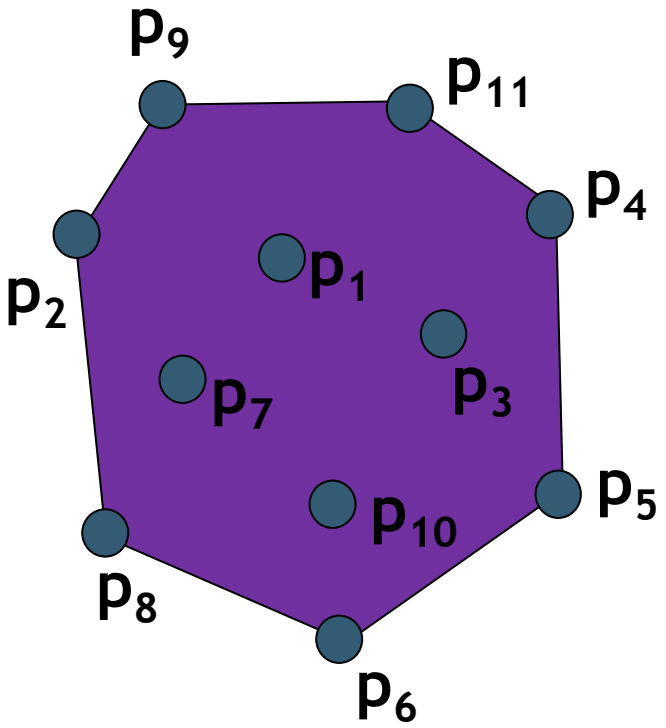


*slide refer to http://www.montefiore.ulg.ac.be/~briquet/algo3-chull-20070206.pdf

CONVEX-HULL BY BRUTE FORCE: EXTREME POINTS

Extreme points of the convex polygon consider all the points in the polygon as a set. An extreme point is a point of the set that is not a middle point of any line segment with end points in the set.

Which pairs of extreme points need to be connected to form the boundary of the convex hull?



CONVEX-HULL BY BRUTE FORCE

- A line segment connecting two points Pi and Pj of a set of n points is a part of its convex hull's boundary if and only if all the other points of the set lies on the same side of the straight line through these two points.
- The straight line through two points (x1, y1), (x2, y2) in the coordinate plane can be defined by the following equation
  - $ax + by = c$

    where $a = y_2 - y_1$, $b = x_1 - x_2$, $c = x_1y_2 - y_1x_2$
- Such a line divides the plane into two half-planes: for all the points in one of them: $ax + by > c$, while for all the points in the other, $ax + by < c$.

CONVEX-HULL BY BRUTE FORCE

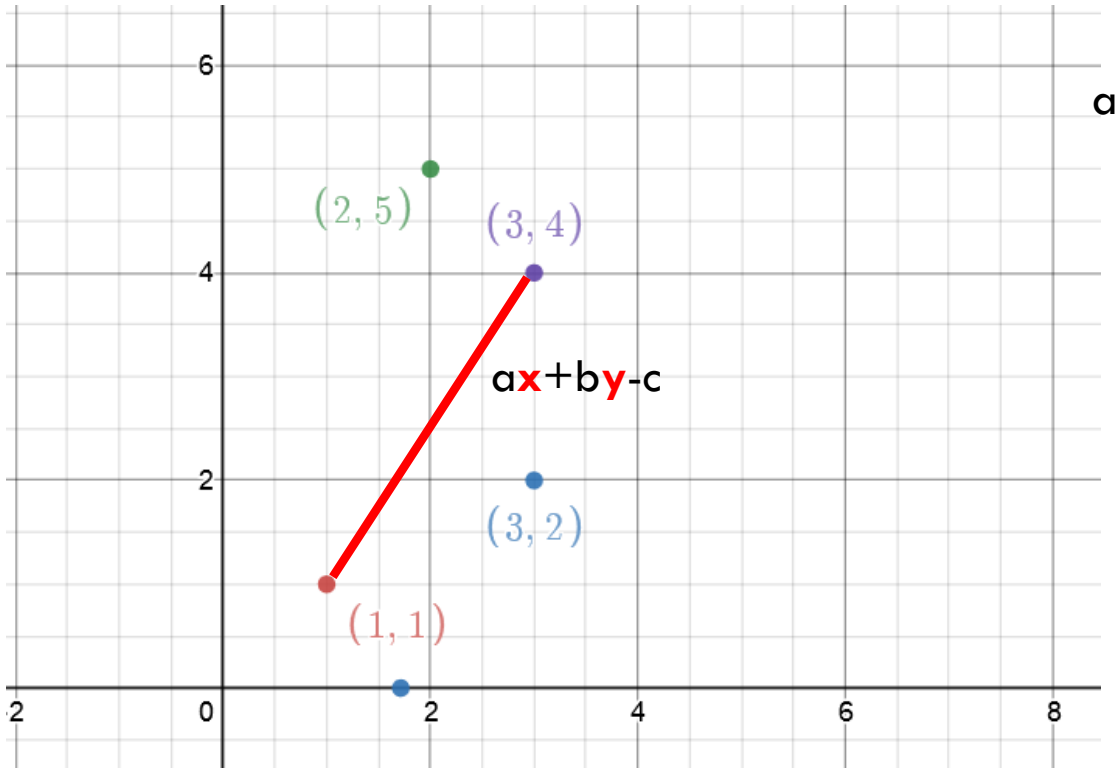
- **Algorithm:** For each pair of points p1 and p2 determine whether all other points lie to the same side of the straight line through p_1 and p_2, i.e. whether ax+by-c all have the same sign.

```
brute_force_convex_hull(a set of n points, P) {
        create empty set of line segments L
        for (each point p1 in P)
          for (each point p2 in P after p1)
            a = p2.y - p1.y; b = p1.x - p2.x;
            c = p1.x*p2.y - p1.y*p2.x
            foundProblem = false
            for (each point p3 in P (not p1 or p2))
              check = a*p3.x + b*p3.y - c
              if (check does not match others)
                foundProblem=true
                break
            if (!foundProblem) add segment p1,p2 to L
        extract and return list of points from L
}
```

$a = y_2 - y_1$, $b = x_1 - x_2$, $c = x_1y_2 - y_1x_2$

ax+by-c

EXAMPLE



$a = y_2 - y_1$, $b = x_1 - x_2$, $c = x_1y_2 - y_1x_2$

```
brute_force_convex_hull(a set of n points, P) {
   create empty set of line segments L
   for (each point p1 in P)
     for (each point p2 in P after p1)
       a = p2.y - p1.y; b = p1.x - p2.x;
       c = p1.x*p2.y - p1.y*p2.x
       foundProblem = false
       for (each point p3 in P (not p1 or p2))
         check = a*p3.x + b*p3.y - c
         if (check does not match others)
           foundProblem=true
           break
       if (!foundProblem) add segment p1,p2 to L
   extract and return list of points from L
}
```

Pick(x_1=1,y_1=1) and (x_2=3,y_2=4)

a= $y_2 - y_1$= 4-1=**3**

b= $x_1 - x_2$= 1-3=**-2**

c= $x_1y_2 - y_1x_2$=(1*4)-(1*3)=4-3=**1**

So, now we have the values of a, b, and c then we should use ax+by-c

check(x=2,y=5)

3*2+(-2*5)-1=6-10-1=-5

Check(x=3,y=2)

3*3+(-2*2)-1=9-4-1=+4

Pick (1,1) and (3,2).....

CONVEX-HULL BY BRUTE FORCE: ANALYSIS

```
brute_force_convex_hull(a set of n points, P) {
        create empty set of line segments L
        for (each point p1 in P)
          for (each point p2 in P after p1)
            a = p2.y - p1.y; b = p1.x - p2.x;
            c = p1.x*p2.y - p1.y*p2.x
            foundProblem = false
            for (each point p3 in P (not p1 or p2))
              check = a*p3.x + b*p3.y - c
              if (check does not match others)
                foundProblem=true
                break
            if (!foundProblem) add segment p1,p2 to L
        extract and return list of points from L
      }
```

Efficiency: $\Theta(n3)$

BRUTE-FORCE STRENGTHS AND WEAKNESSES

- Strengths
  - Wide applicability
  - Simplicity
  - Yields reasonable algorithms for some important problems (e.g., matrix multiplication, sorting, searching, string matching)
- Weaknesses
  - Rarely yields efficient algorithms
  - Some brute-force algorithms are unacceptably slow
  - Not as constructive as some other design techniques

REFERENCES

1. https://en.wikipedia.org/wiki/Brute-force_search


STAY SAFE
AND STAY
POSITIVE