

# Design & Analysis of Information Systems

Mathematics in Computer Science.

This year's topic is

**Computational Geometry.**

# Mathematics in computer science

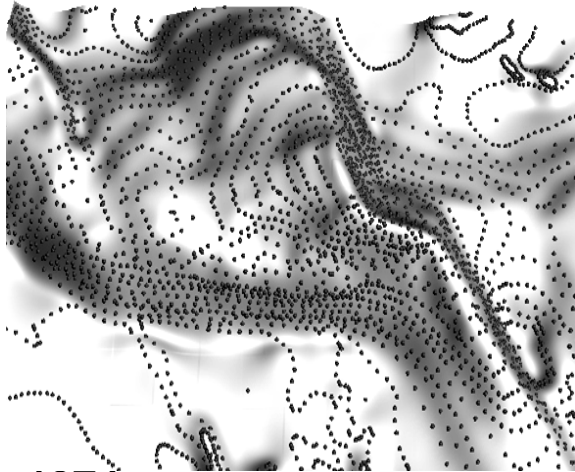
- Mathematics seeks for **elegant** solutions
  - “**Solution**” is not just “showing the answer”.
  - It is important to describe the solution process.
- Computer science seeks for **elegant** and **efficient** algorithms
  - **Algorithm**: Concrete description of process.
  - Algorithms lead to our modern life.
- Mathematics is vital in algorithm design
  - Solve seemingly-impossible tasks.

# Computational Geometry

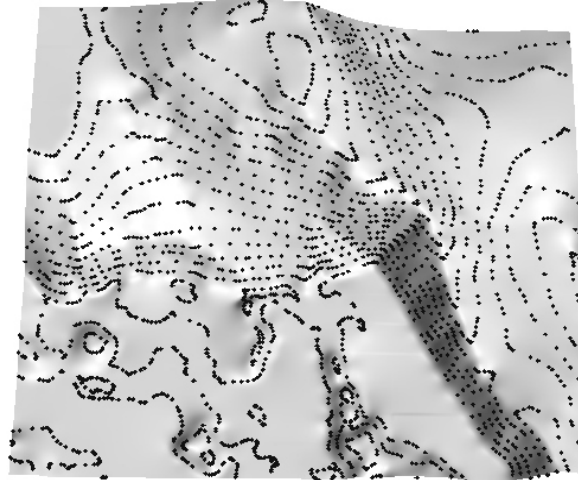
- Design algorithms for geometric problems
  - Many modern applications
    - GIS, Graphics, Geometric Modeling, Robotics, Multidimensional Database, Computer Vision.
  - Fast processing of massive data
    - Giga pixel data = 1000,000,000 data in a single digital picture
- **Elegant** geometry for algorithm design
  - Discrete geometry, etc.
  - Exciting intellectual puzzles (知的パズル)

# Diverse elevation point data: density, distribution, accuracy

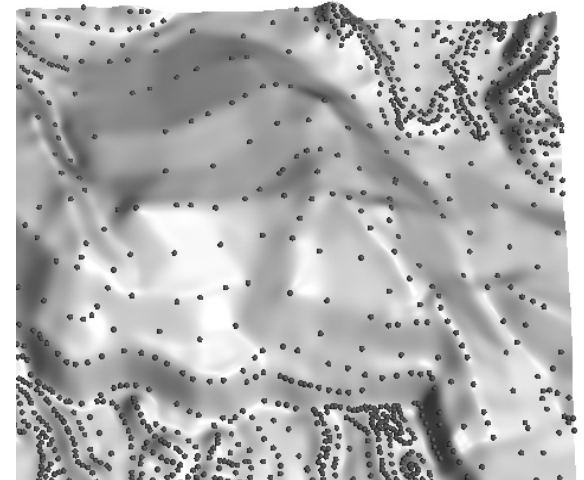
**Photogrammetry** 0.76m v. accuracy (5ft contours)



1974

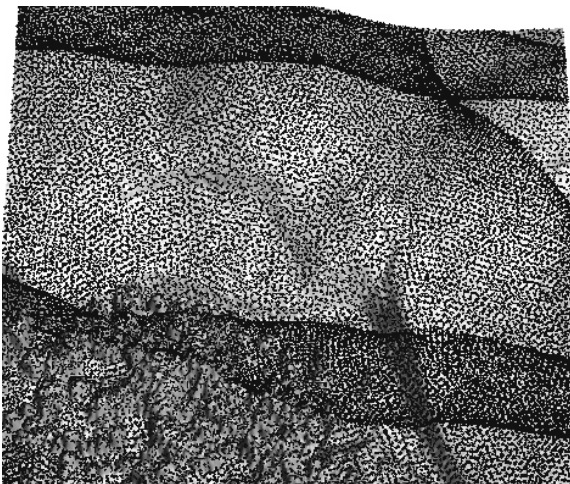


1995

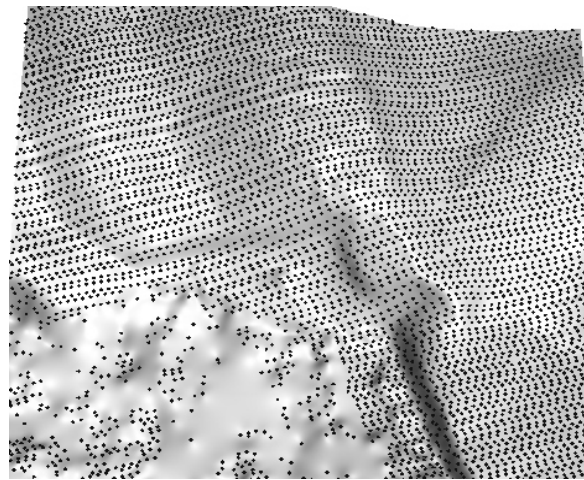


1998

**Lidar** 0.15m v. accuracy; altitude 700m and 2300m



1999



2001



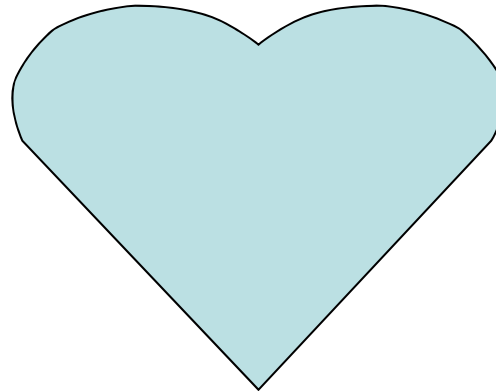
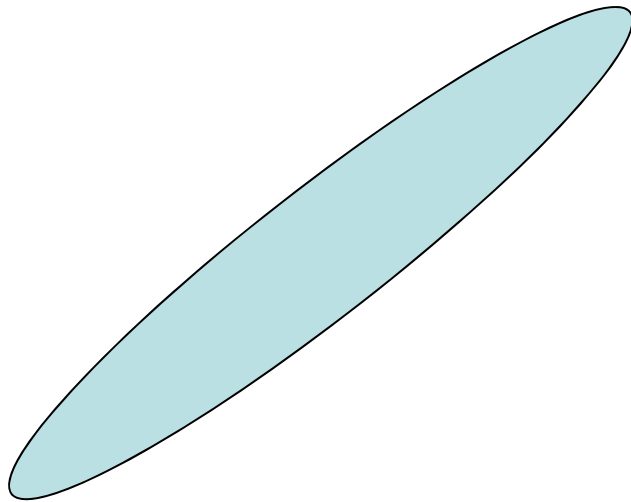
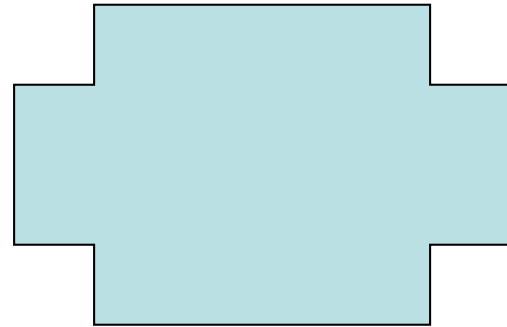
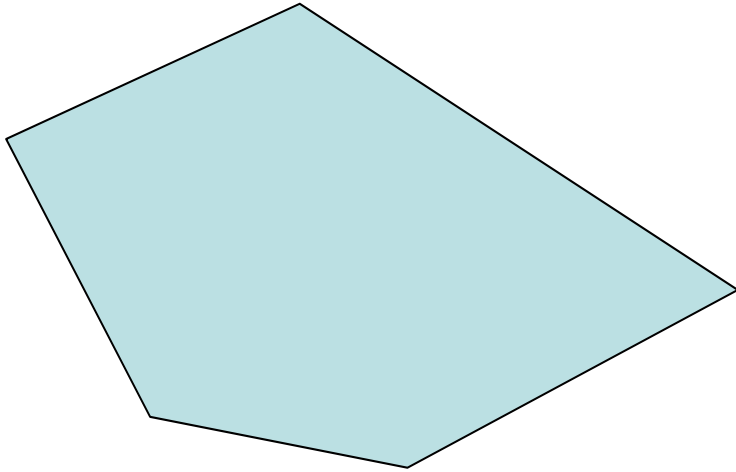
2004

↑ 100 meters

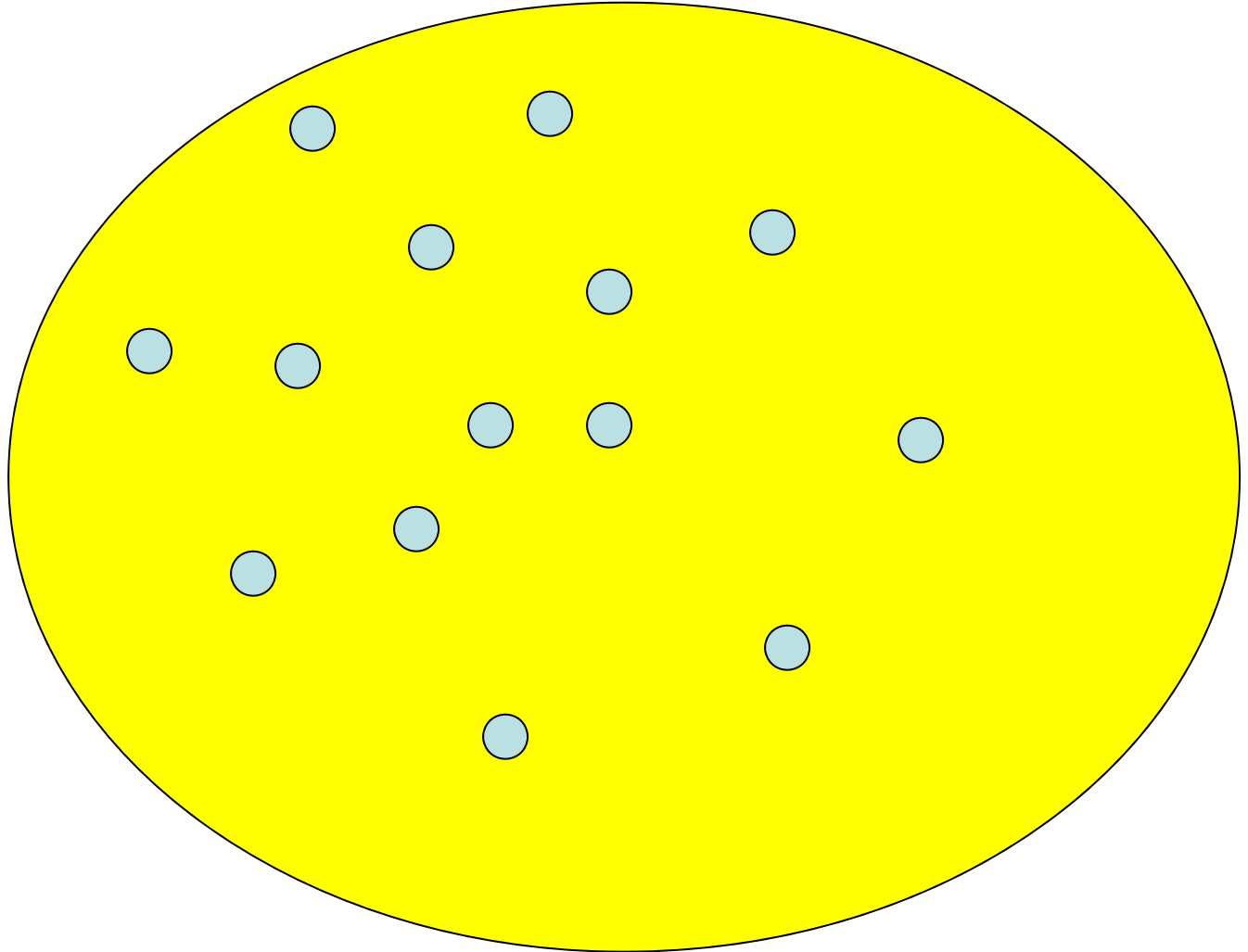
# Example of geometric computation

- **Convex hull** computation (凸包の計算)
  - A showcase of algorithmic techniques
- Given a set  $S$  of  $n$  points in a plane, compute its convex hull
  - Convex set: A set  $X$  such that for any two points  $p$  and  $q$  in  $X$ , the segment  $pq$  is in  $X$
  - Convex hull  $CH(S)$  of  $S$  : Minimum convex set containing  $S$ 
    - Question: Is convex hull well-defined (i.e., always uniquely exists) ?

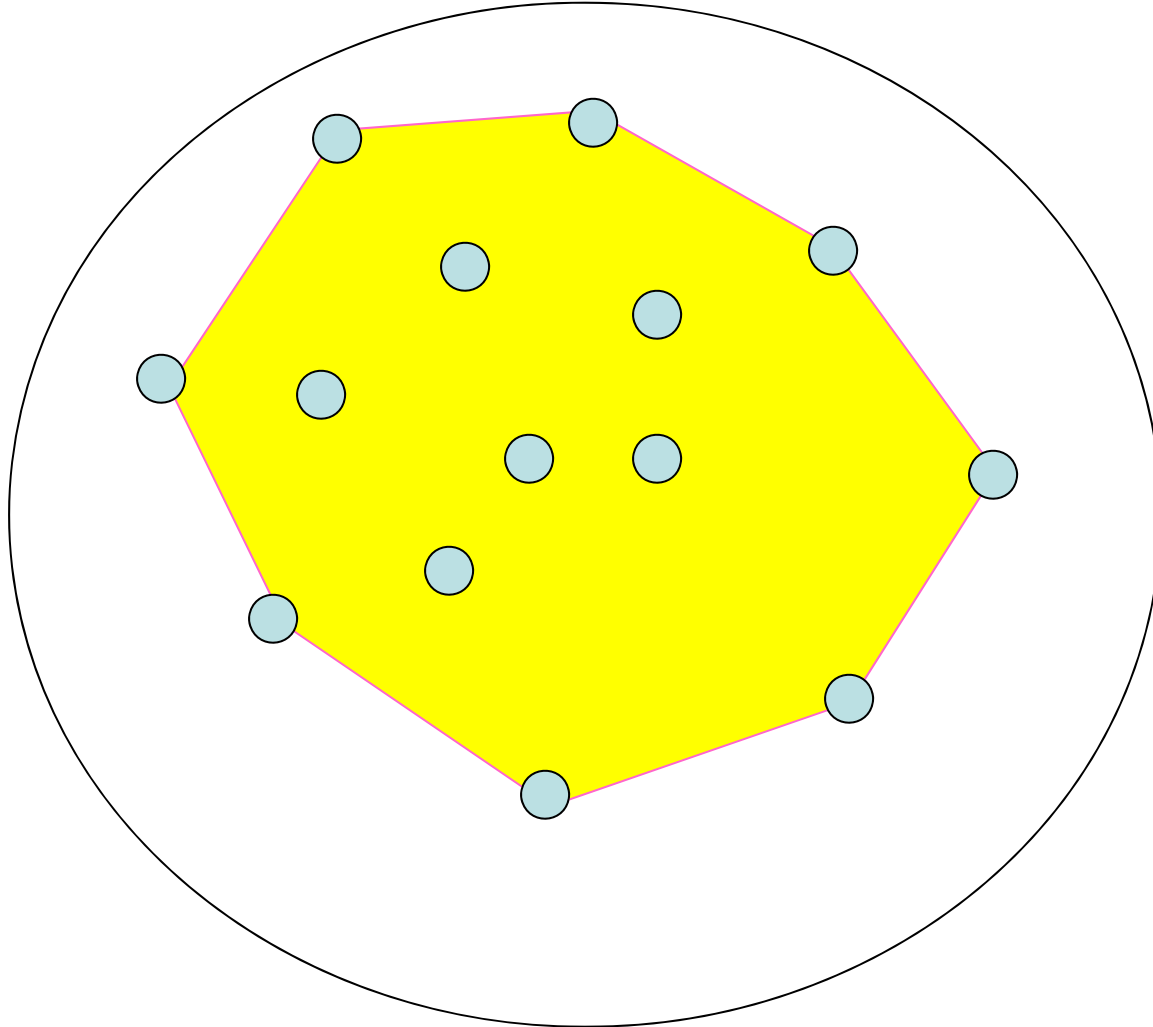
# Convex Sets ?



# Convex set containing $S$

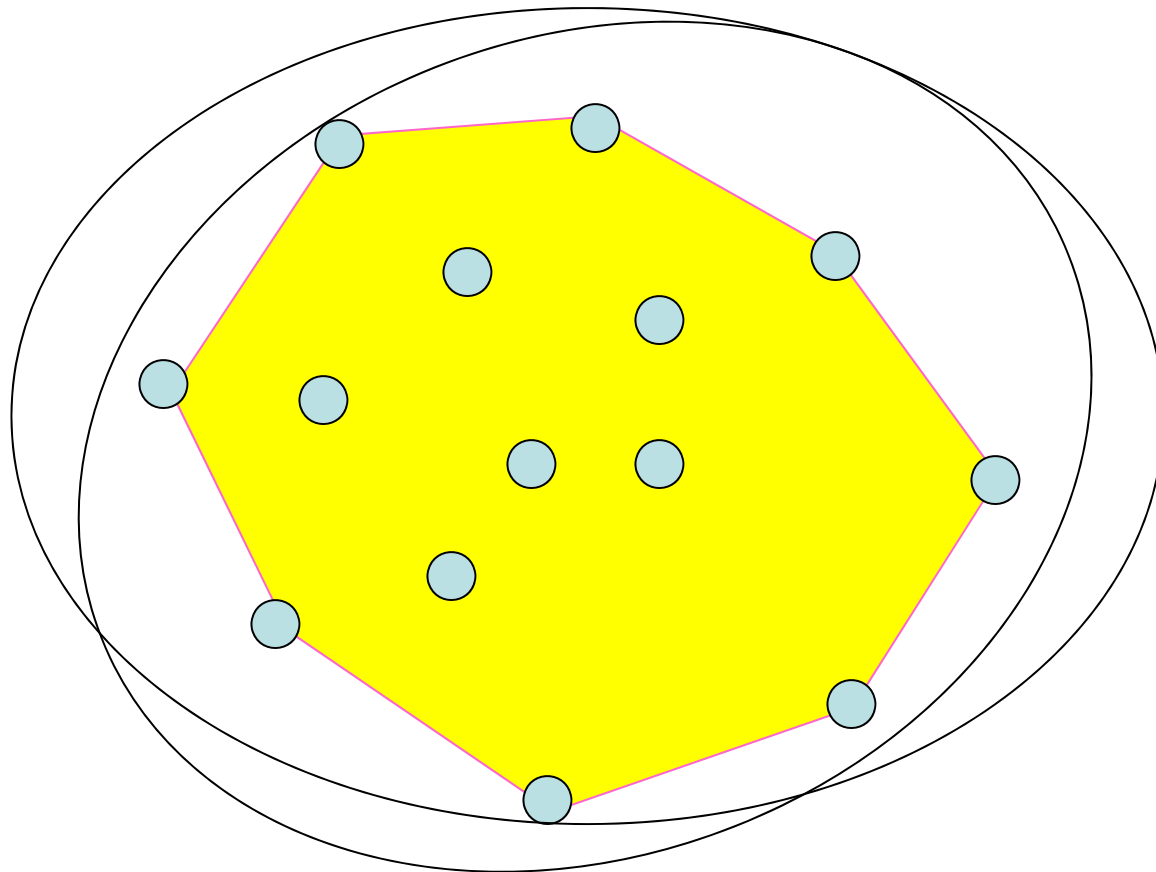


# Convex hull of $S$

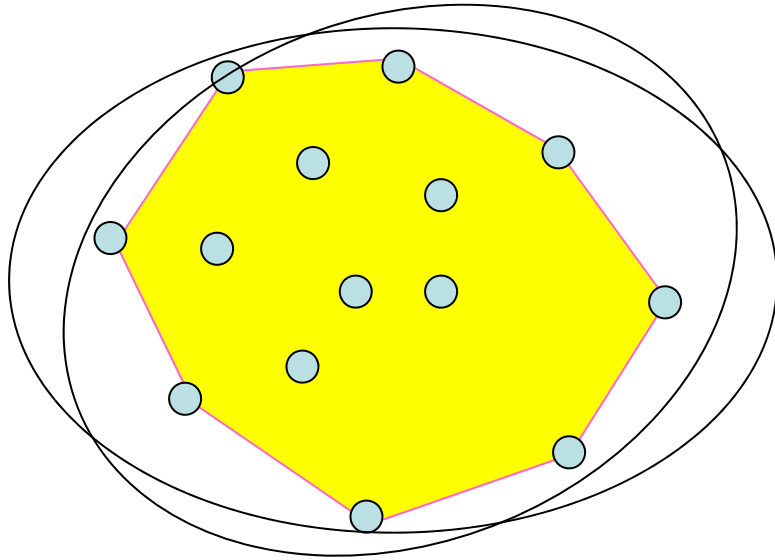




Every convex set containing  $S$   
must also contain  $\text{CH}(S)$



# Convex hull exists



$$CH(S) = \bigcap_{X : \text{convex}, X \supset S} X$$

- Nice mathematical representation.
- But it is hard to use in computation.

# Convex Hull Computation

- Given a set  $S$  of  $n$  points, compute  $CH(S)$
- Can you design an algorithm?

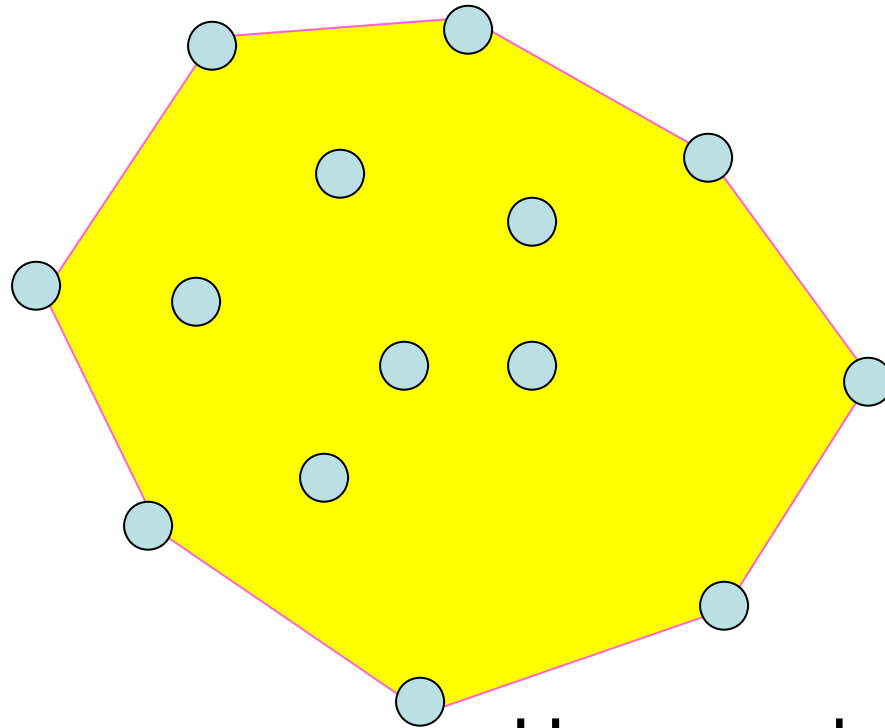
$$CH(S) = \bigcap_{X: \text{convex}, X \supset S} X$$

**Very nuisance formula. The right hand side is intersection of infinite number of convex sets**

We should transform this “cold” formula into more “friendly” one.

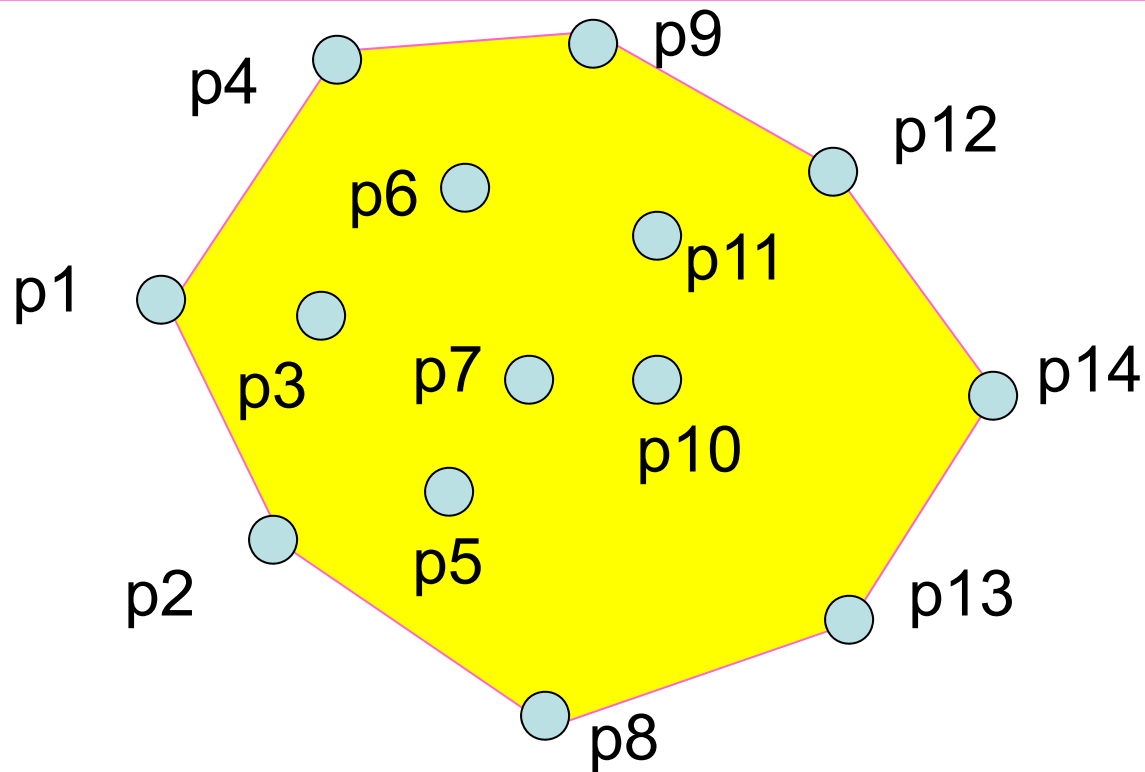
# Characterization.

1.  $\text{CH}(S)$  is a convex polygon
2. Vertices of  $\text{CH}(S)$  are points of  $S$
3.  $\text{CH}(S)$  contains all points of  $S$



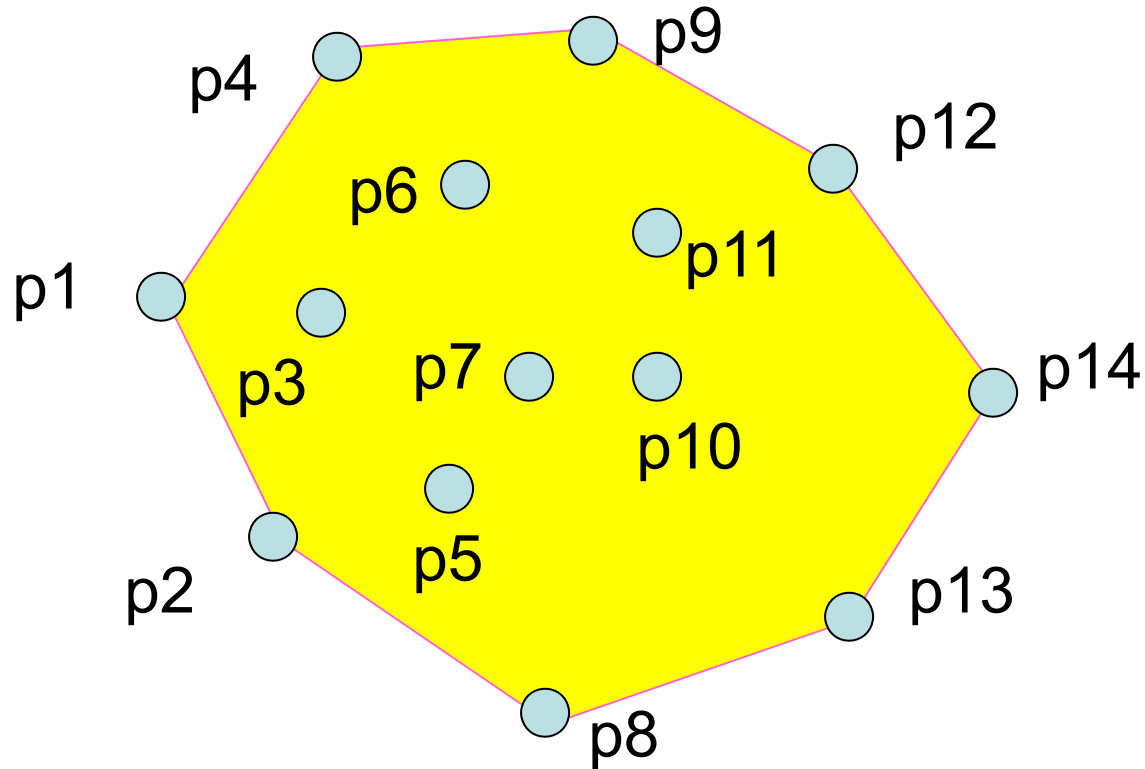
Homework: Prove it!

A representation of  $CH(S)$ :  
the list of vertices in a clockwise order  
starting from the leftmost one



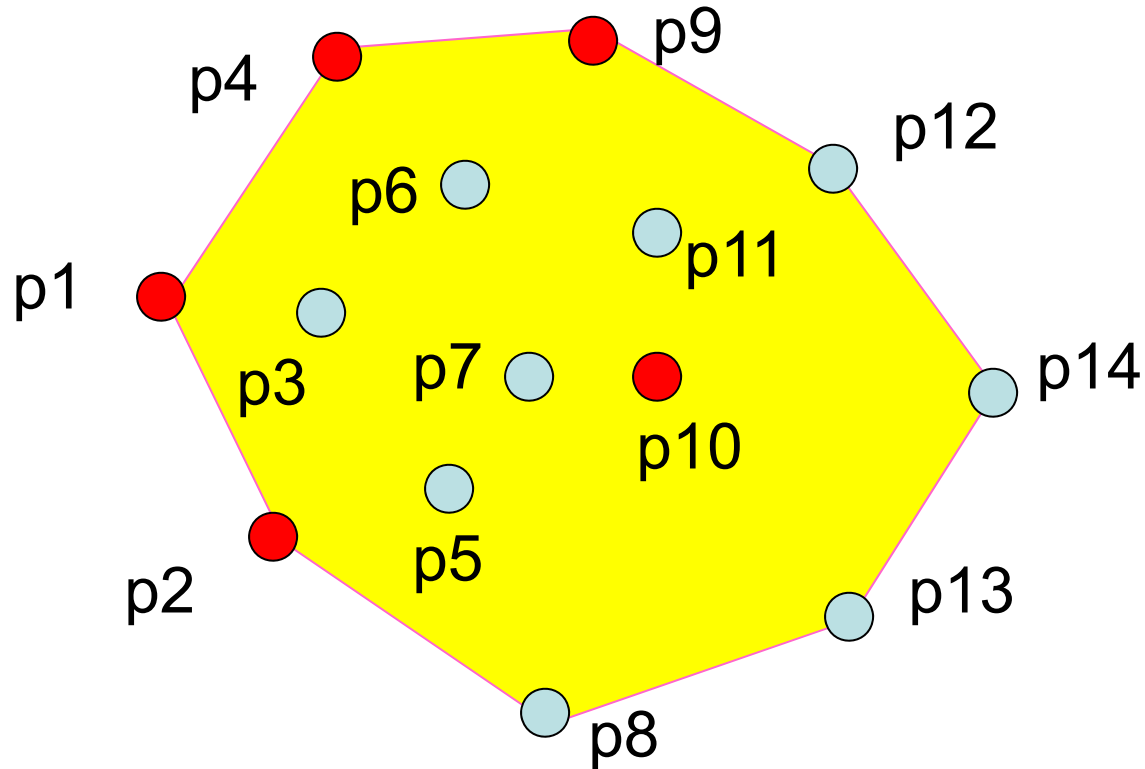
Question: Show  $CH(S)$  of the above picture  
in the above representation.

Now, the problem is in the  
discrete and finite world



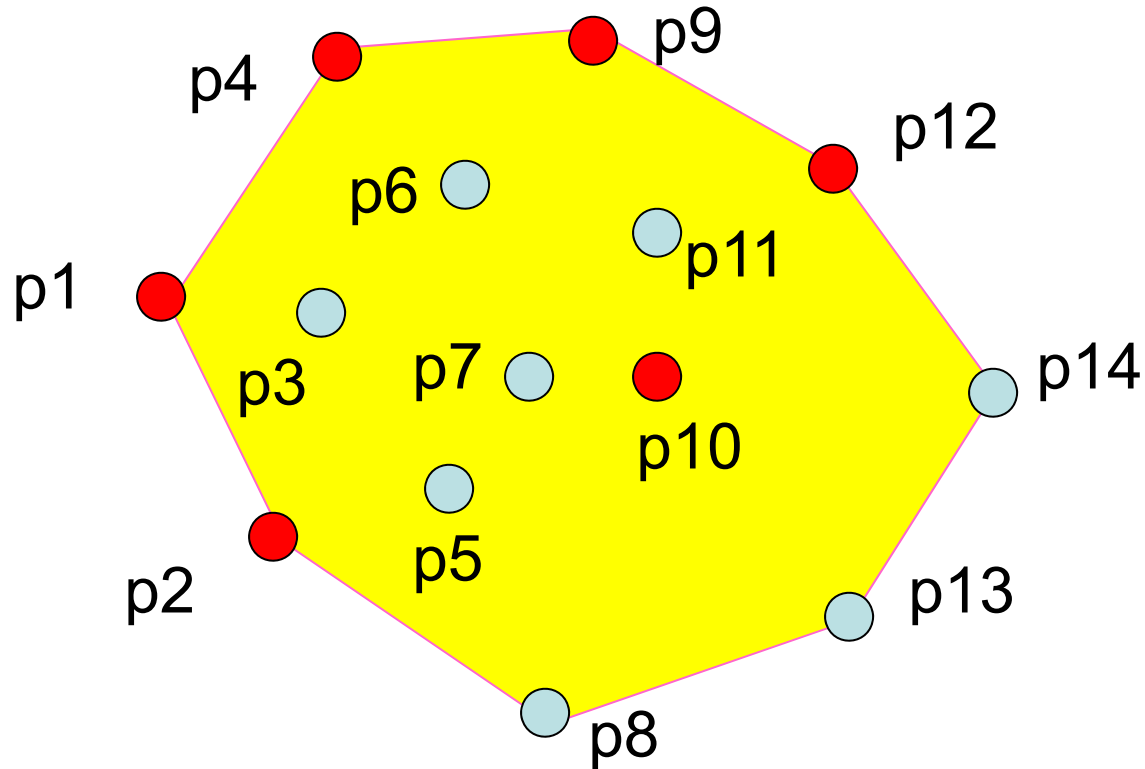
Find the (partial) permutation of  $S$  forming  
the convex hull.

# Verification problem



Given a list  $(p_1, p_2, p_{10}, p_{12}, p_9, p_4)$ , verify whether it is  $CH(S)$ .

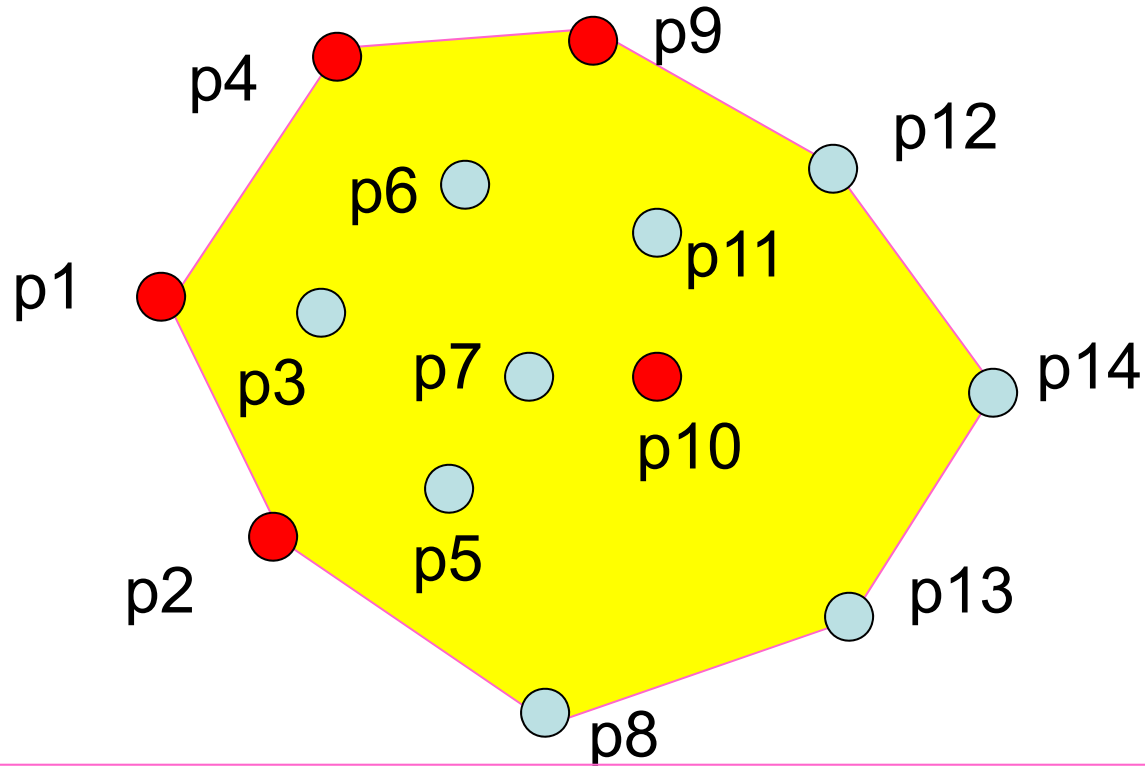
# Verification problem 1



Given a list  $(p1, p2, p10, p12, p9, p4)$ , verify whether it gives a convex polygon



## Verification problem 2



If the list  $(p_1, p_2, p_{10}, p_{12}, p_9, p_4)$  gives a convex polygon, show all other points are contained in it.

# A brute-force algorithm

- Algorithm 1:
  - Generate all possible partial permutations of  $S$
  - For each permutation  $P$ , verify it gives  $CH(S)$
- Questions
  - Is the above algorithm always correct?
  - How much time does it take if  $n = 1000$ .
- Conclusion: We need a better algorithm.
- Question: Please consider a better algorithm than this!

# Analysis of Algorithm

- Time Complexity
  - Given an input of size  $n$  (words/bits), how many basic steps are required in an algorithm?
    - Arithmetic operations
    - Comparisons, Data Access (read, write)
    - Floor/Ceiling  $[314,1592] = 314$
  - $T(n)$ : number of basic steps
  - Asymptotic time complexity
    - $T(n) < c f(n)$  for a suitable constant  $c$  and a familiar function  $f(n)$
    - We write  $T(n) = O(f(n))$
- Classification of time complexity
  - Polynomial time algorithm:  $f(n)$  is a polynomial in  $n$ 
    - Linear time algorithm:  $T(n) = O(n)$
    - Quadratic time algorithm:  $T(n) = O(n^2)$
  - Exponential time algorithm : e.g.,  $f(n) = 2^n$
  - Unbounded time algorithm : No such  $f(n)$

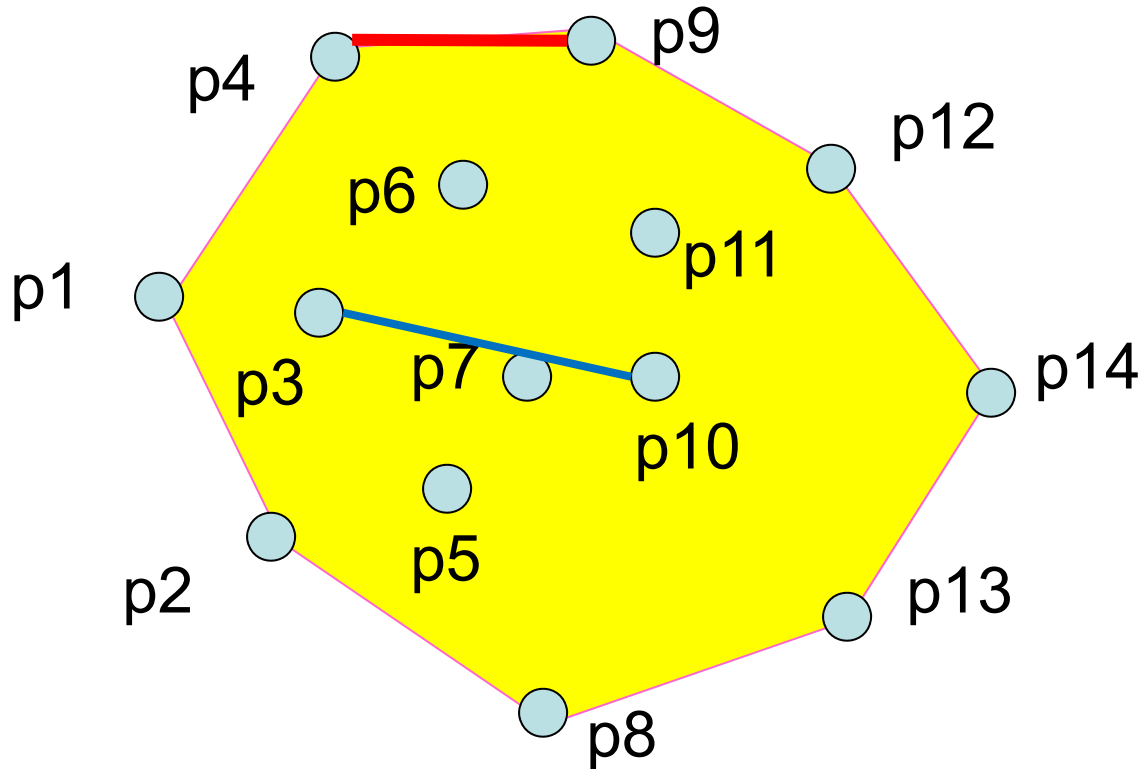
# Complexity of a problem

- Complexity of a problem  $X$ 
  - The complexity of  $X$  is  $O(f(n))$  if there is an algorithm to solve  $X$  in  $O(f(n))$  time
  - The complexity of  $X$  is  $\Omega(f(n))$  if there is no algorithm to solve  $X$  in  $o(f(n))$  time
    - $o(f(n))$ : strictly smaller than  $O(f(n))$
  - The complexity of  $X$  is  $\Theta(f(n))$  if it is both  $O(f(n))$  and  $\Omega(f(n))$
- Complexity class of a problem
  - A problem  $X$  is in class  $P$  if there is a polynomial time algorithm to solve  $X$ , that is, the complexity of  $X = O(f(n))$  for a polynomial  $f(n)$ .

# Typical problems in class P

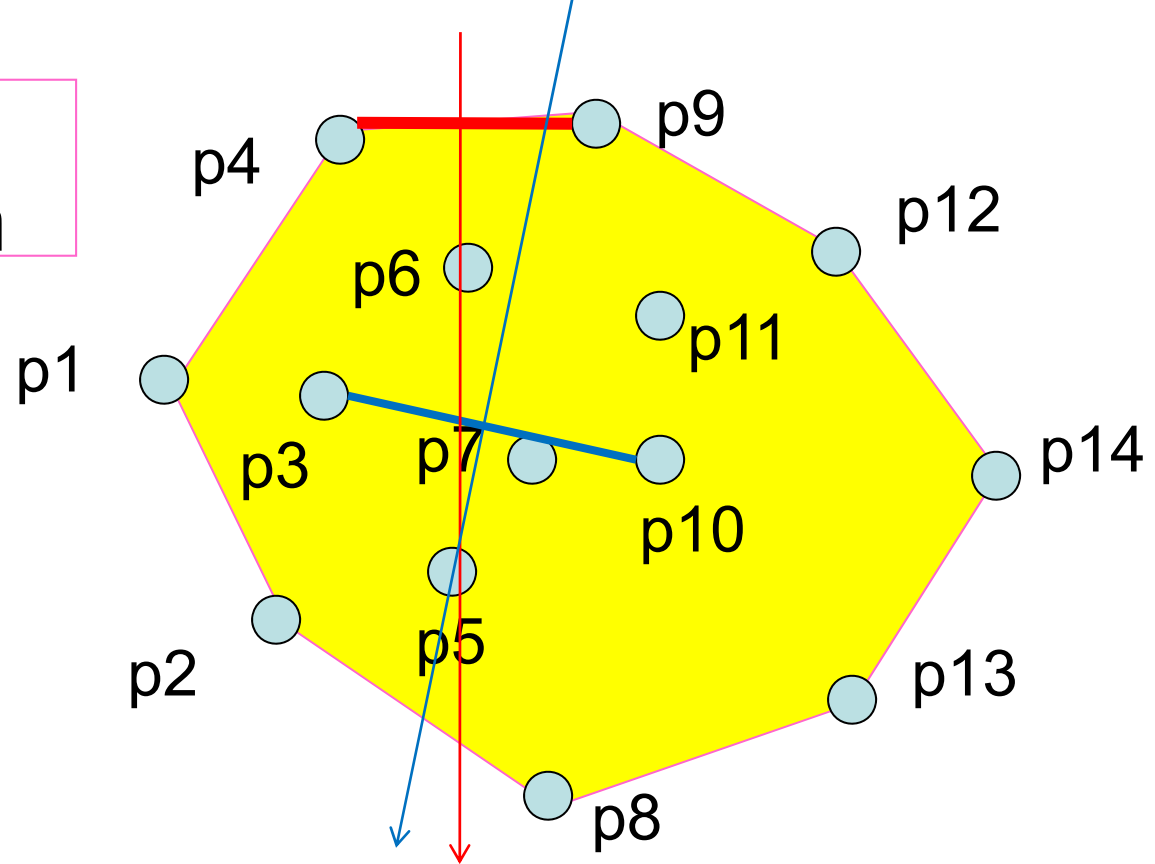
- Finding maximum element in a given set of  $n$  numbers :  $\Theta(n)$  time
- Sorting  $n$  numbers:  $O(n \log n)$  time
  - $\Theta(n \log n)$  if we restrict the computation model
- Computing the “distance” of two DNA sequences of length  $n$ :  $O(n^2)$  time
- **Computing convex hull of  $n$  points in the plane**
  - How to do it? What is the time complexity?

# Edge verification



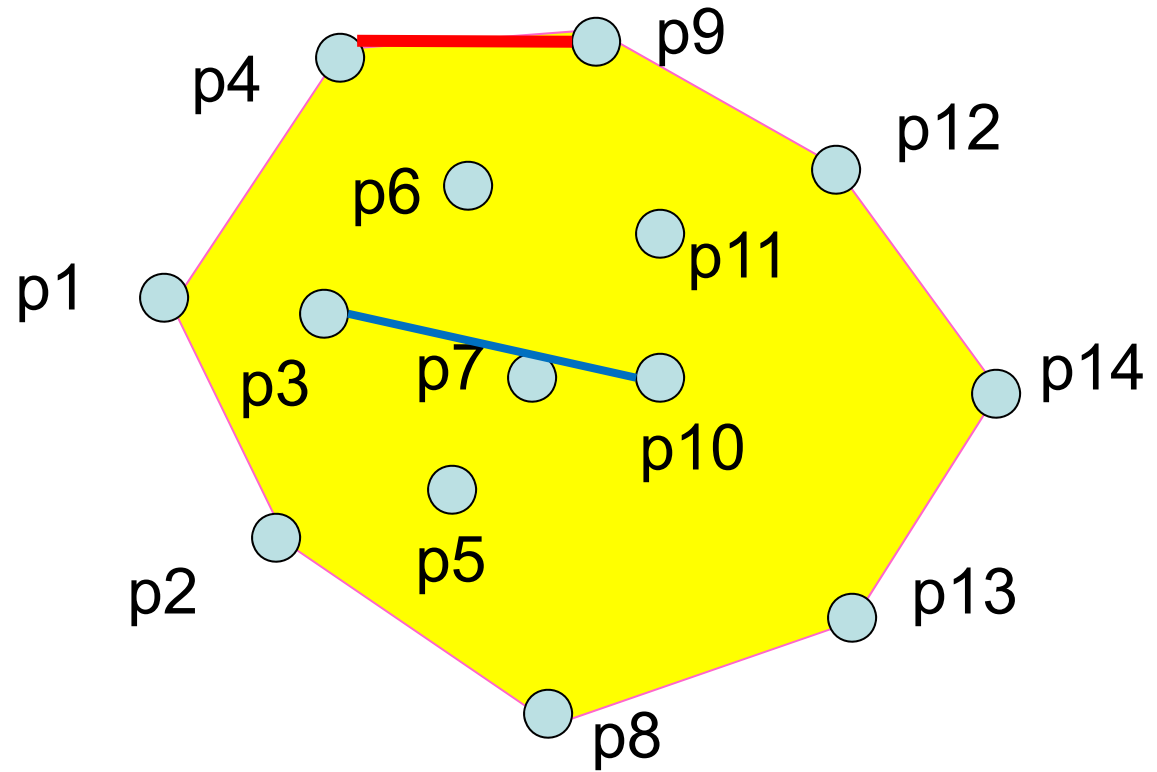
(p4, p9) is an edge of the convex hull, while (p3, p10) is not. How to distinguish them?

## Edge verification



- Order all points in the orthogonal direction to the edge we want to verify
- Convex hull edge if and only if its endpoints are both maximum (or minimum) in the ordering

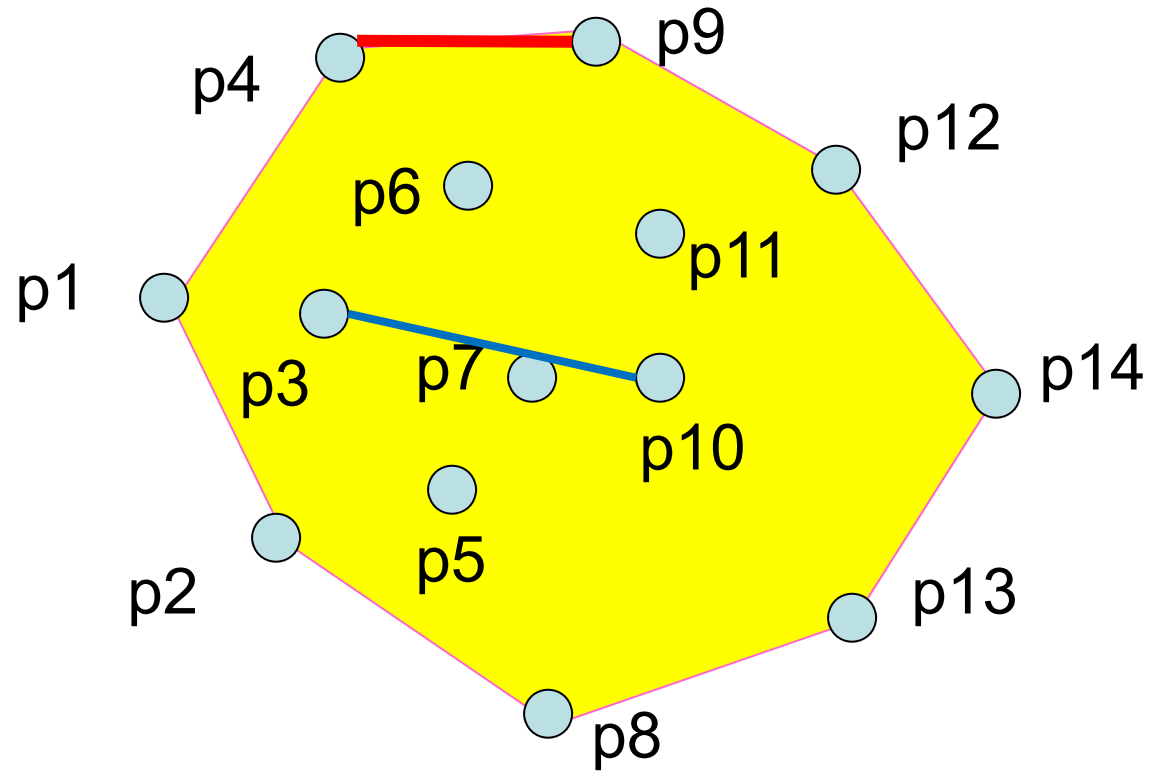
# Convex hull algorithm using edge verification



- Verify all  $n(n-1)/2$  candidate edges
- Collect all the convex hull edges
- Arrange them into convex hull
  - How to do it?

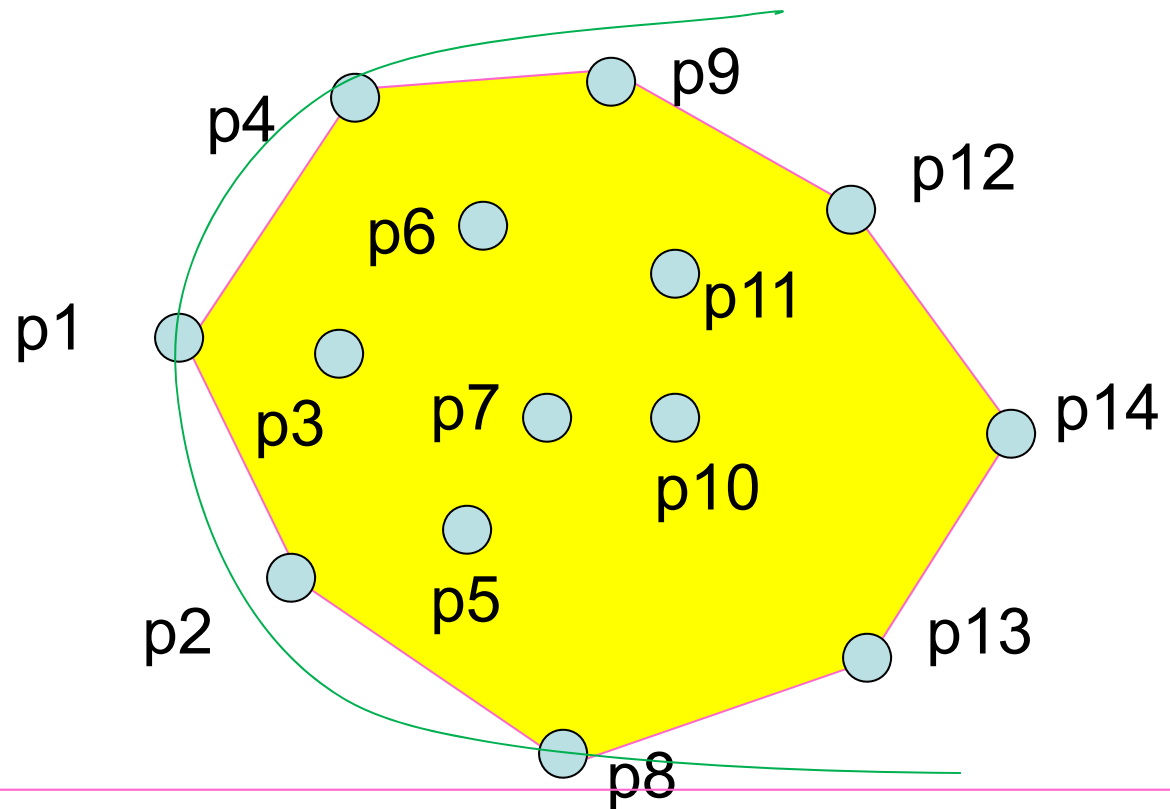


Time complexity of the algorithm using edge verification



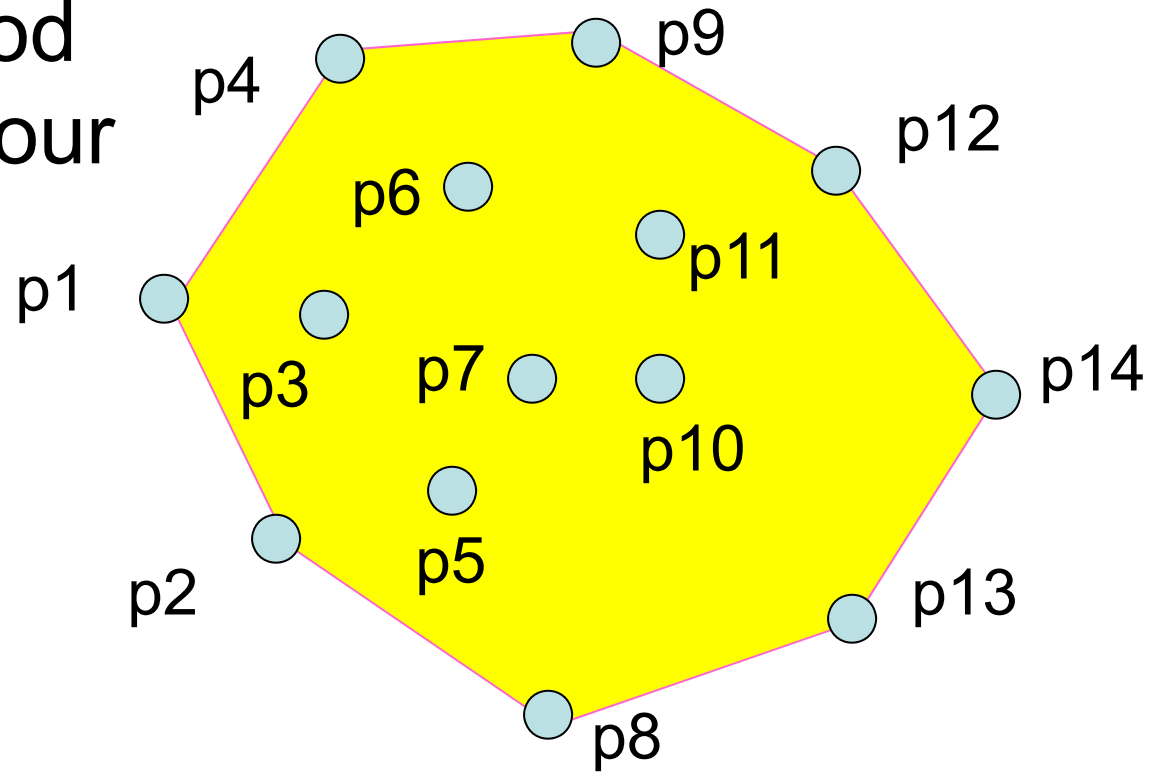
- $O(n^3)$  time algorithm
  - Edge verification =  $O(n)$  time for each candidate
  - $O(n^2)$  candidate edges
- Polynomial time. But very slow! How to improve?

Learn from  
our real life



- Consider points as pins on a board. (パチンコ台の釘だと思おう)
- You are given a string. How to realize the convex hull. (紐を使って凸包を計算しよう)
- “Gift wrapping” algorithm or.....

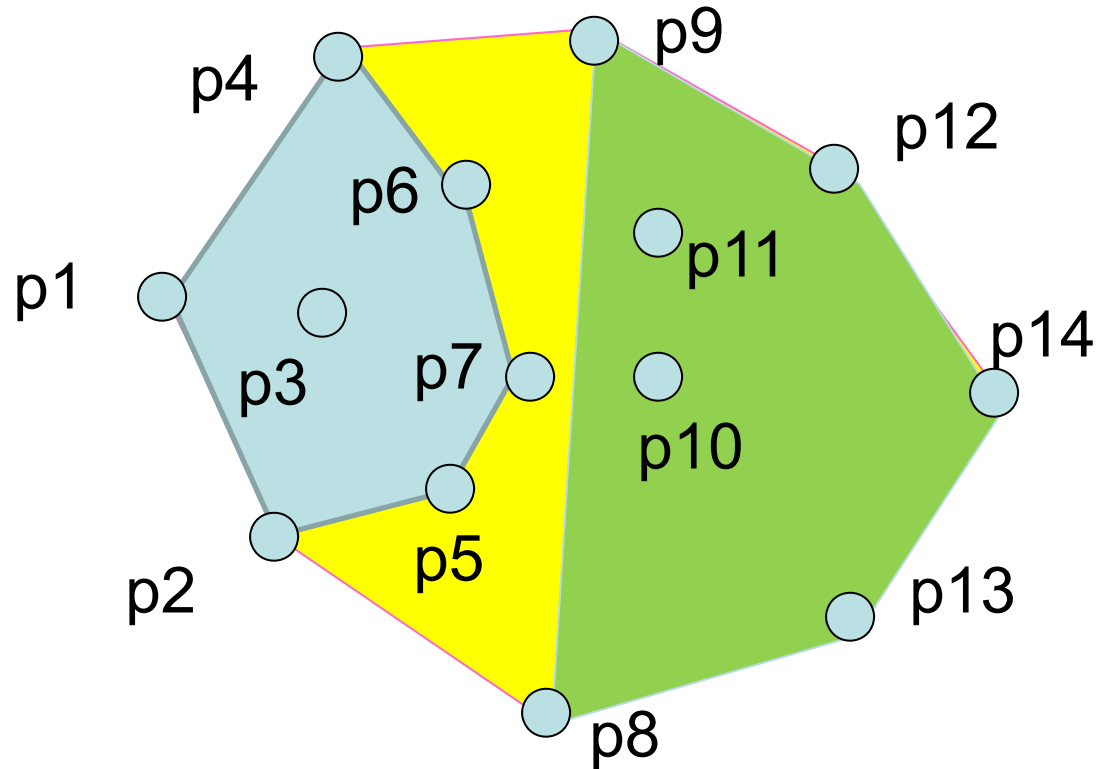
Another method  
learning from our  
real life



- Consider points as pins on a board.
- You can touch from any given direction by hand. How to realize the convex hull.
- “hand probing” algorithm

# Algorithmic paradigms

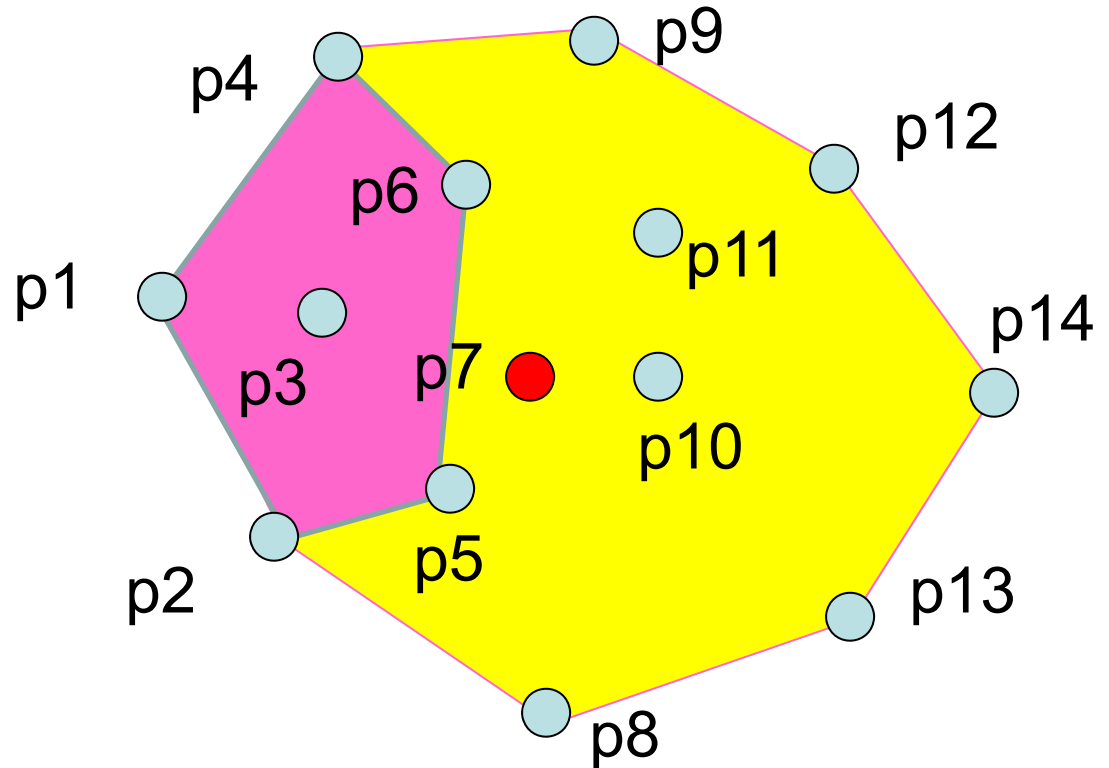
## 1. Divide and conquer



- Process “left half” and “right half” independently
- Merge two outputs

# Algorithmic paradigms

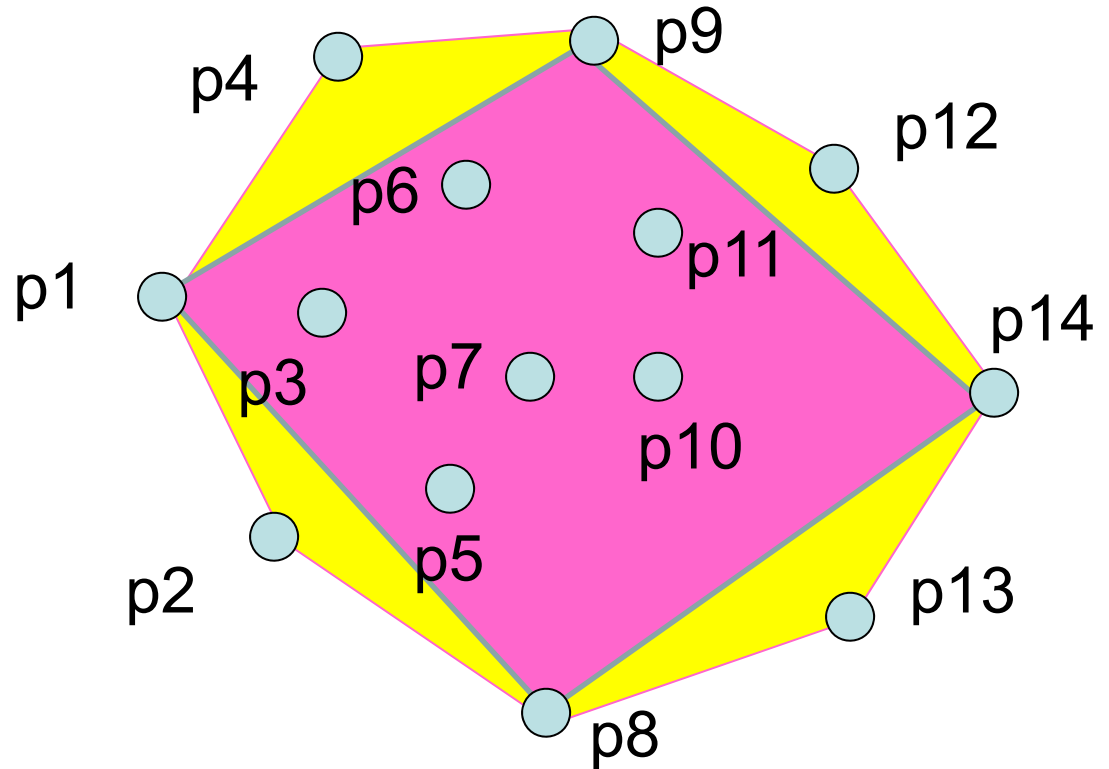
## 2. Incremental method



- Process points left-to-right
- Start from the triangle formed by p1,p2,p3, and add points one by one updating the convex hull

# Algorithmic paradigms

## 3. Prune by preprocessing



- Remove all points in the pink quadrangle
- Run an algorithm discussed before

# Time complexities of algorithms

- Brute-force:
- Edge verification:
- Gift wrapping:
- Probing:
- Divide & Conquer:
- Left to right incremental:
- Pruning + Gift wrapping

# Applications

- Classical use of convex hulls
  - Diameter computation
    - Rotating caliper
  - Fast collision detection
- Tokuyama's original use of convex hull algorithms
  - Statistics
  - Data mining (!)
  - Image processing (!!)

