Brute Force

Closest Point &

Convex Hull & KnapSack

Closest Pair Problem

- finding the two closest points in a set of *n* points.
- One of the important applications of the closest-pair problem is cluster analysis in statistics. Based on *n* data points, hierarchical cluster analysis seeks to organize them in a hierarchy of clusters based on some similarity metric. For numerical data, this metric is usually the Euclidean distance; for text and other nonnumerical data, metrics such as the Hamming distance are used.
- A bottom-up algorithm begins with each element as a separate cluster and merges them into successively larger clusters by combining the closest pair of clusters.
- For simplicity, we consider the two-dimensional case of the closest-pair problem.
- We assume that the points in question are specified in a standard fashion by their *(x, y)* Cartesian coordinates and that the distance between two points *pi(xi, yi)* and *pj(xj, yj)* is the standard Euclidean distance

$$d(p_i, p_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}.$$

Idea

- Find every pair and compute distance

PseudoCode

ALGORITHM *BruteForceClosestPair*(P)

//Finds distance between two closest points in the plane by brute force
//Input: A list P of n ($n \geq 2$) points $p_1(x_1, y_1), \ldots, p_n(x_n, y_n)$
//Output: The distance between the closest pair of points
$d \leftarrow \infty$
for $i \leftarrow 1$ **to** $n - 1$ **do**
    for $j \leftarrow i + 1$ **to** n **do**
        $d \leftarrow \min(d, sqrt((x_i - x_j)^2 + (y_i - y_j)^2))$ //*sqrt* is square root
return d

Definition

DEFINITION A set of points (finite or infinite) in the plane is called ***convex*** if for any two points p and q in the set, the entire line segment with the endpoints at p and q belongs to the set.

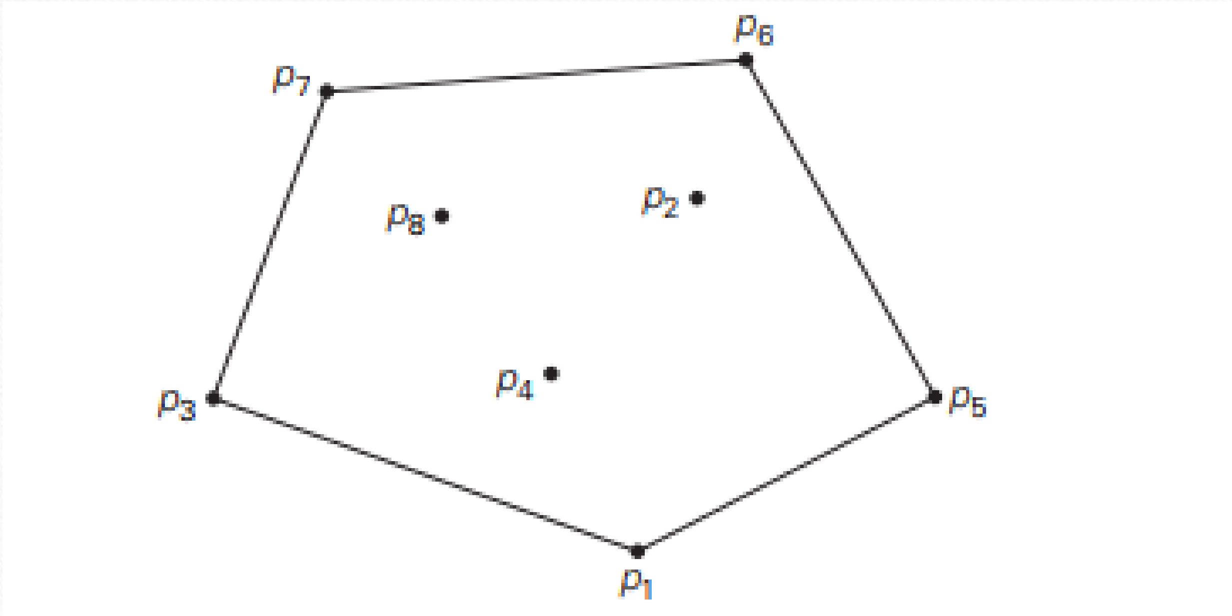
Convex?

Definition

DEFINITION The ***convex hull*** of a set S of points is the smallest convex set containing S. (The "smallest" requirement means that the convex hull of S must be a subset of any convex set containing S.)

THEOREM The convex hull of any set S of $n > 2$ points not all on the same line is a convex polygon with the vertices at some of the points of S. (If all the points do lie on the same line, the polygon degenerates to a line segment but still with the endpoints at two points of S.)

Example



How to solve it?

- Find the pairs of points need to be connected to form the boundary of the convex hull
- a line segment connecting two points pi and pj of a set of n points is a part of the convex hull's boundary if and only if all the other points of the set lie on the same side of the straight line through these two points
- Repeating this test for every pair of points yields a list of line segments that make up the convex hull's boundary

Math behind

- the straight line through two points $(x1, y1)$, $(x2, y2)$ in the coordinate plane can be defined by the equation $ax + by = c$, where
  - $a = y2 - y1$,
  - $b = x1 - x2$,
  - $c = x1y2 - y1x2$.
- 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, $ax + by < c$

Brute Force Idea

- check whether certain points lie on the same side of the line
  - $ax + by - c$ has the same sign for each of these points
- Test each line segment to see if it makes up an edge of the convex hull
- If the rest of the points are on one side of the segment, the segment is on the convex hull.
  - else the segment is not.

Complexity

- Finding edges?
- Tests?
- All together

Assignment

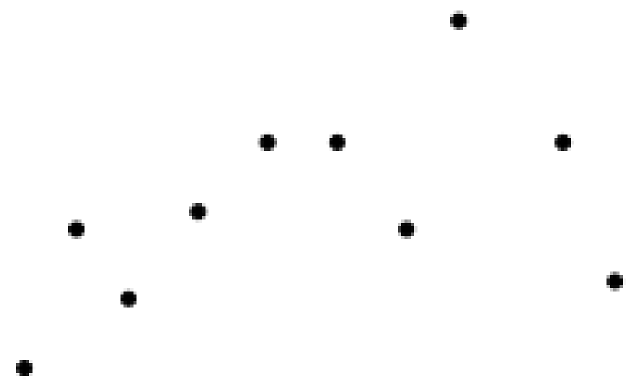
- http://pontus.digipen.edu/cgi-bin/submission.cgi
- Deadline: 2017-11-06 23:59:59
- 1) hullBruteForce: for each pair points determine whether all other points are one side of the line formed by the pair of points. If it does - add the points (or rather indices to the hull). Since hull is represented by a std::set, you do not have to worry about duplicates.
- 2) hullBruteForce2: find the first point that is in the hull (smallest or biggest x or y coordinate),then find the next vertex of the hull in counter-clockwise order by considering all lines through the previous vertex and requiring that there are no points to the right of it.
- To submit: hull-bruteforce.cpp

Second Approach

- Another approach: incremental, from left to right
- Let's first compute the upper boundary of the convex hull this way (property: on the upper hull, points appear in x-order)
- Main idea: Sort the points from left to right (= by x-coordinate).
- Then insert the points in this order, and maintain the upper hull so far

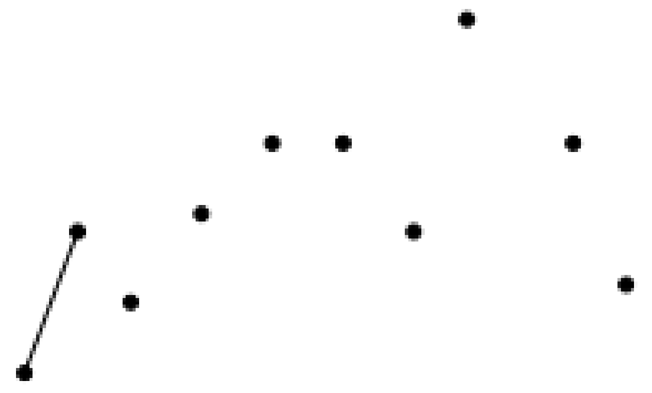
Example

Observation: from left to right, there are only right turns on the upper hull



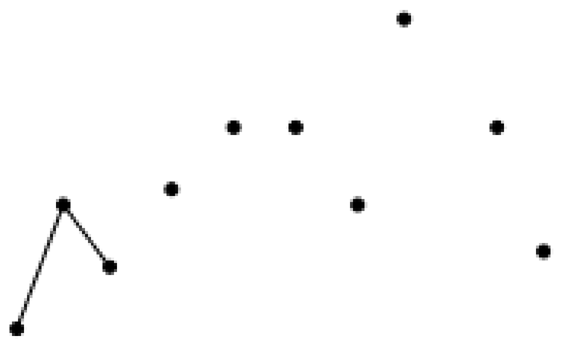
Example

Initialize by inserting the leftmost two points

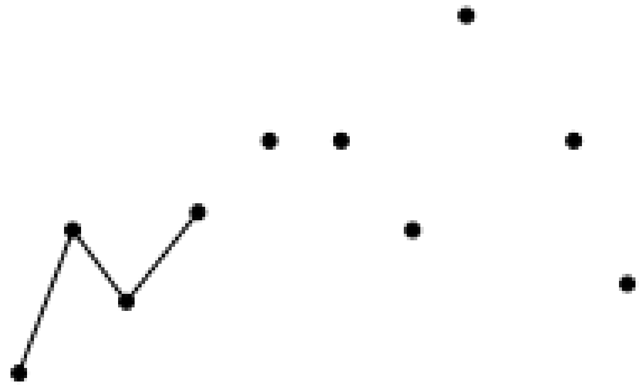


Example

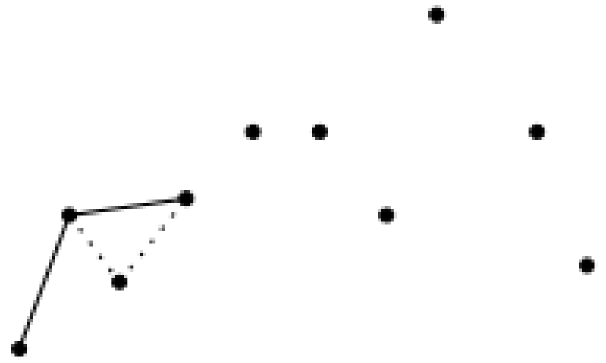
If we add the third point there will be a right turn at the previous point, so we add it



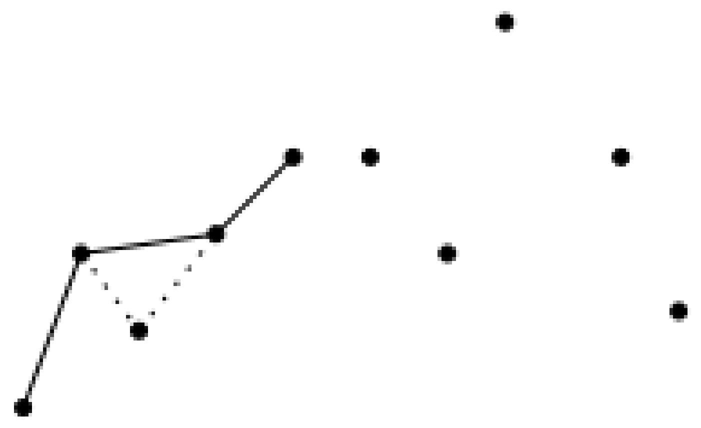
If we add the fourth point we get a left turn at the third point



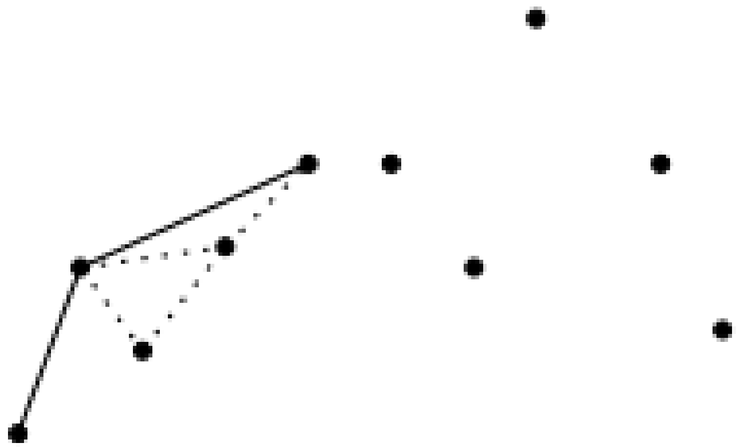
... so we remove the third point from the upper hull when we add the fourth



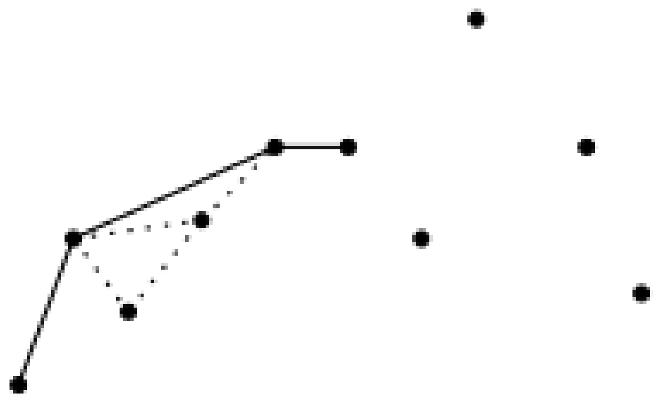
If we add the fifth point we get a left turn at the fourth point



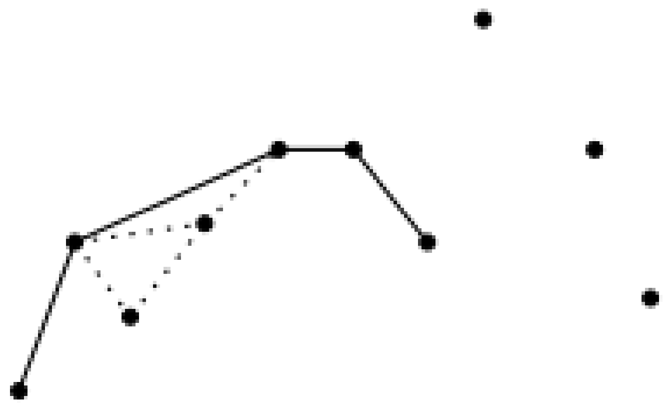
... so we remove the fourth point when we add the fifth



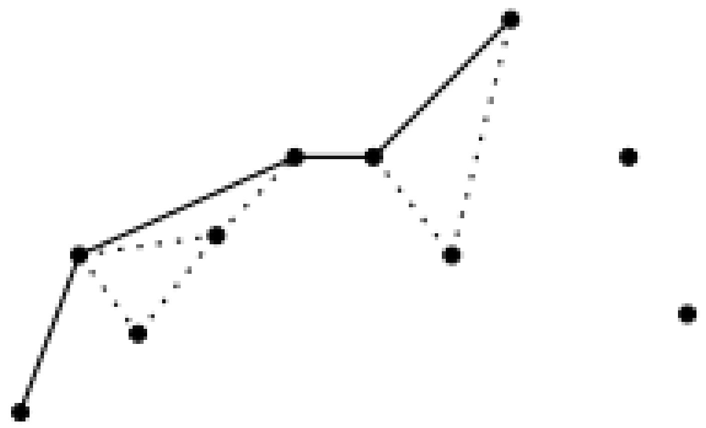
If we add the sixth point we get a right turn at the fifth point, so we just add it



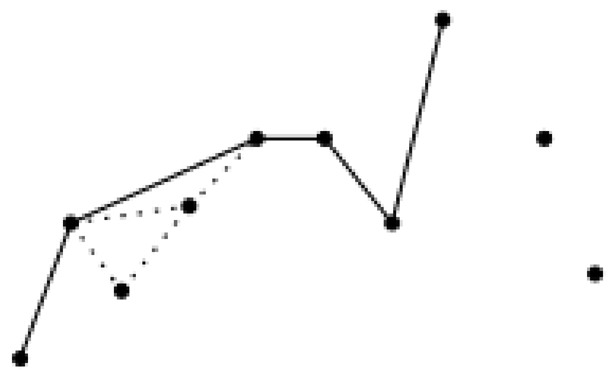
We also just add the seventh point



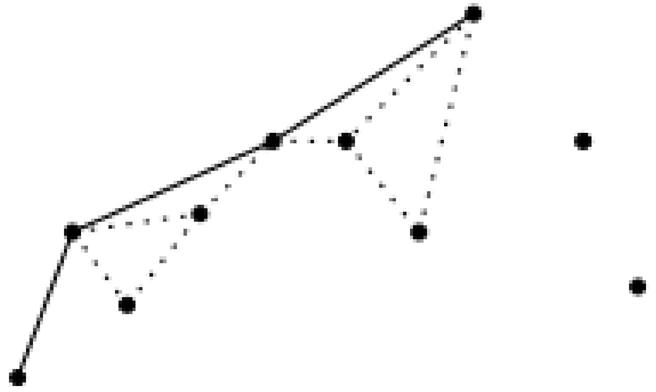
... we must remove the seventh point



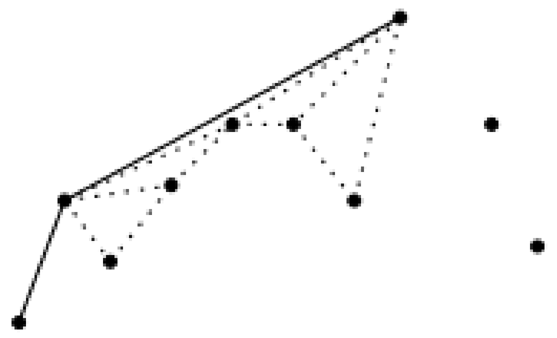
When adding the eight point
… we must remove the
seventh point



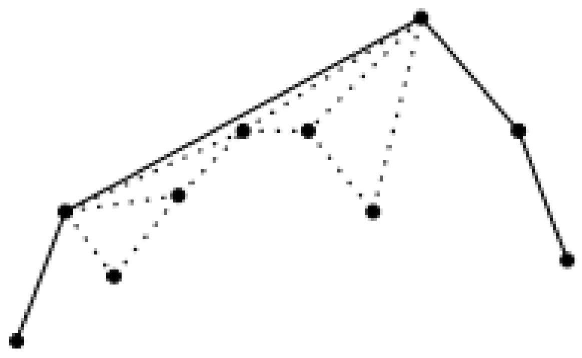
… and also the sixth point



. . . and also the fifth point



After two more steps we get:



Algorithm CONVEXHULL(P)
Input. A set P of points in the plane.
Output. A list containing the vertices of $CH(P)$ in clockwise order.

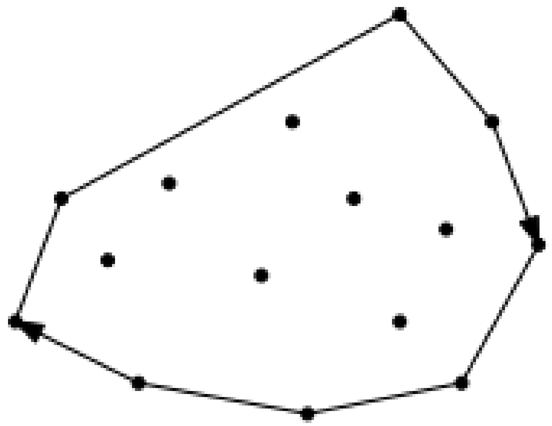
1. Sort the points by x-coordinate, resulting in a sequence $p_1, \ldots, p_n$.
2. Put the points p_1 and p_2 in a list L_{upper}, with p_1 as the first point.
3. **for** $i \leftarrow 3$ **to** n
4.   **do** Append p_i to L_{upper}.
5.   **while** L_{upper} contains more than two points **and** the last three points in L_{upper} do not make a right turn
6.   **do** Delete the middle of the last three points from L_{upper}.

Then we do the same for the lower convex hull, from right to left

We remove the first and last points of the lower convex hull

... and concatenate the two lists into one

$p_1, p_2, p_{10}, p_{13}, p_{14}$



$p_{14}, p_{12}, p_8, p_4, p_1$

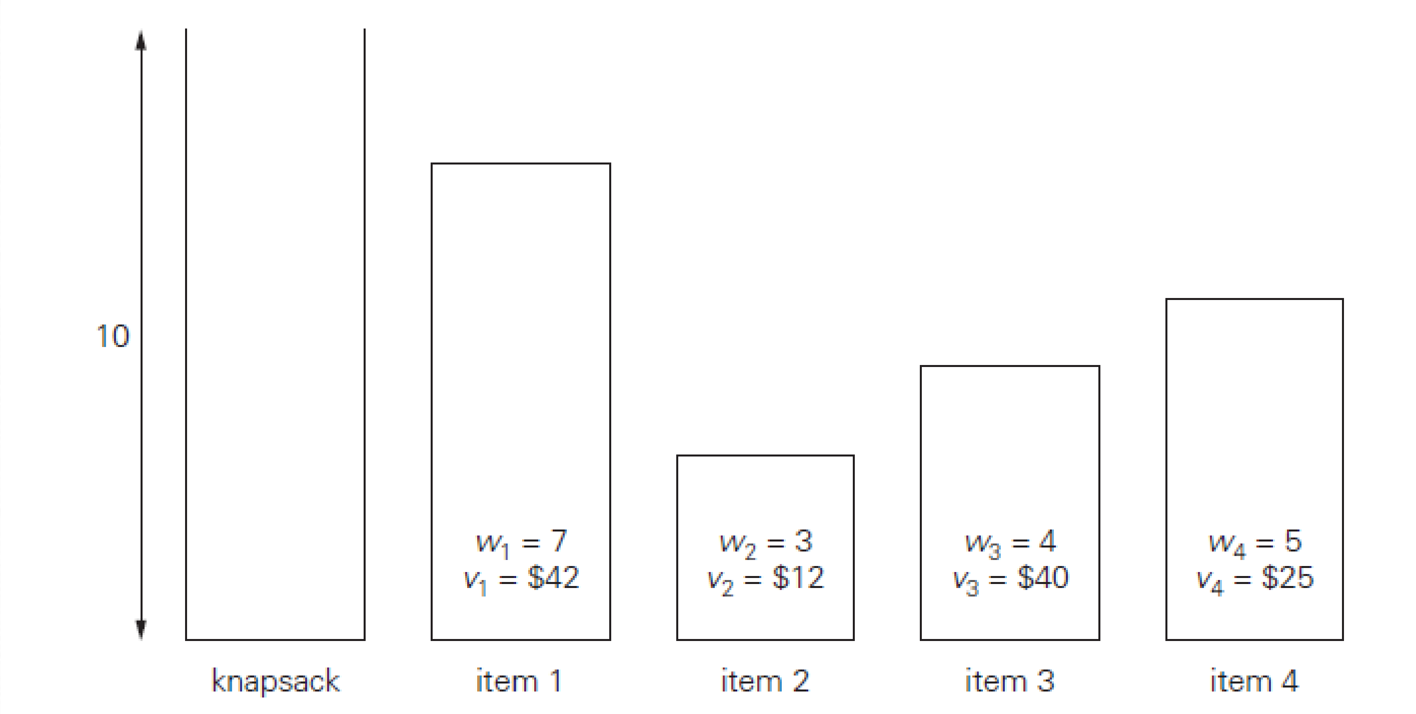
Knapsack Problem

- Given n items of known weights w1, w2, . . . , wn and values v1, v2, . . . , vn and a knapsack of capacity W,
- find the most valuable subset of the items that fit into the knapsack.

Idea

- Create all subsets and choose the one with the highest value

Example


10
$w_1 = 7$
$v_1 = \$42$
$w_2 = 3$
$v_2 = \$12$
$w_3 = 4$
$v_3 = \$40$
$w_4 = 5$
$v_4 = \$25$
knapsack
item 1
item 2
item 3
item 4