The Farthest-Pair Assignment, Level 4 version  
 ( / 60 marks)

In this assignment, you will design and code an algorithm for the following problem:

*Given a set of points S in 2-D, find the two that are the farthest apart.*

In the field of [computational geometry](http://www.cs.umd.edu/~mount/754/Lects/754lects.pdf), this maximum distance is called the *diameter* of S.

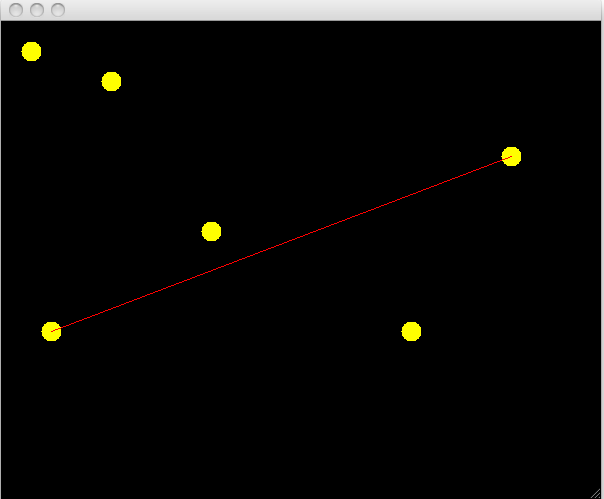
# Example

Suppose S has six points, stored as an array of ordered pairs:  
  
 [ [20,20], [100, 50], [200, 200], [500, 125], [400, 300], [40, 300] ]

The picture below shows the points of S plotted in screen coordinates. The two that are the farthest apart are [40, 300] and [500, 125]. They are shown connected by the red line.

Thus, the diameter of S is .

But how do we find those two points?



# A simple but inefficient algorithm

The simplest approach (called the “brute force” approach) is to measure the distance between every pair of points using a nested for-loop and find the one that’s maximum, much like finding the maximum value in an array of plain numbers.

Write the pseudocode for the brute-force algorithm below. ( / 10 marks )

|  |  |
| --- | --- |
| **Input**: An array *S* of *n* ordered pairs | **Output**: The pair of points that are the farthest apart |
| **Brute-force algorithm**  let maxD = 0  **for i = 0 to n-1:**  let p1 = S[i]  for j = 0 to n-1:  let p2 = S[j]  let d = getDistance(p1, p2)  if d > maxD:  let maxD = dist  let best = {p1, p2}  return best | |

Write the above algorithm in condensed form (i.e. Stuff 1, Stuff 2, etc.) ( / 4 marks )

|  |
| --- |
|  |

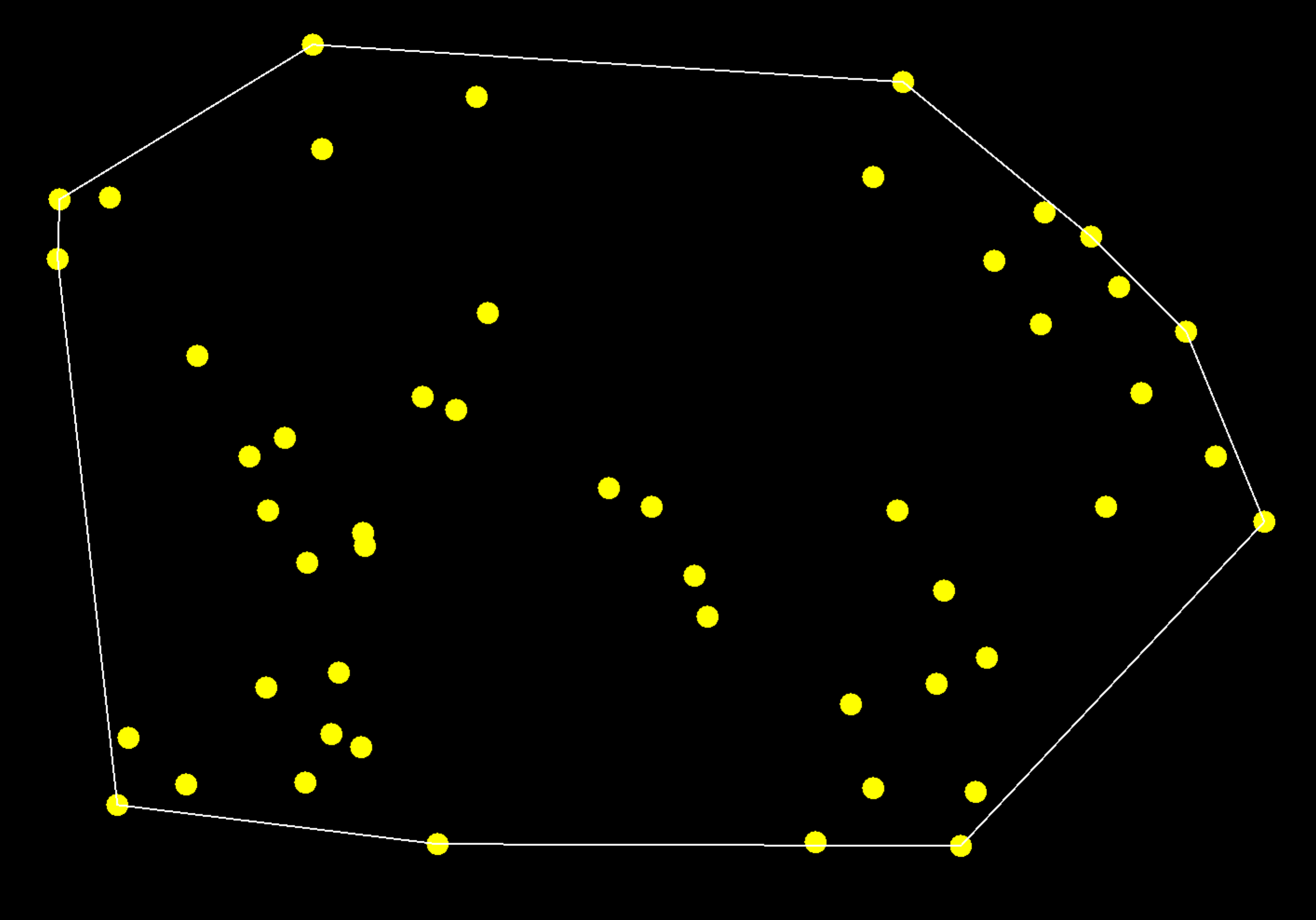
Write the worst-case runtime of your algorithm in Big-O notation. ( / 4 marks )

|  |
| --- |
|  |

# A more efficient algorithm

The *Diameter Theorem* of computational geometry states that the farthest pair will always lie on the set’s   
*convex hull*. The convex hull of a set S is defined as the smallest convex polygon that contains all the points in *S*.

In the example below, S is the set of yellow points, and their convex hull is the white polygon. Another theorem states that each vertex of the convex hull of S is also a point in S. (Can you argue why?)



Here’s the idea behind the more efficient algorithm:

1. Determine which points in S are vertices of the convex hull, and store them in a list in counterclockwise order. Call that list *CH*.
2. Find the two vertices in *CH* that are the farthest apart. Because of the Diameter Theorem, those two vertices are also the farthest pair for all of *S*.

Why might this approach be more efficient than the brute-force approach? ( / 2 marks )

|  |
| --- |
|  |

# Coding the algorithm in Processing ( / 40 marks )

Each of the two steps above is challenging in its own right! Your main job in this assignment is to research algorithms for carrying out those two steps and code them as a Processing program. The program should:

* Create an array of 100 random points with screen coordinates between 0 and 500.
* Determine which pair of points in the array is the farthest apart using the two steps above.
* Use the **draw**() method to
  + draw all the points on the screen as dots
  + draw the convex hull
  + draw a line between the farthest pair

Spend at least a day researching and reading, with no coding.

[Wikipedia’s page on Convex Hull Algorithms](http://en.wikipedia.org/wiki/Convex_hull_algorithms) is a useful starting point. At some point, you should encounter the term *antipodal pair*, which is used in step 2 above.

Spend another day planning ideas on paper.

Start coding on day 3. I’ve given you a skeleton program to start from. It contains classes for Points in 2-D and Vectors.