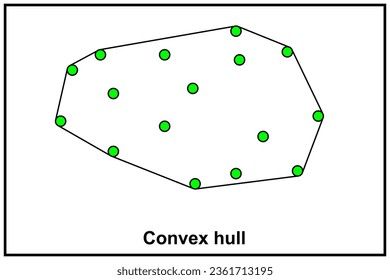
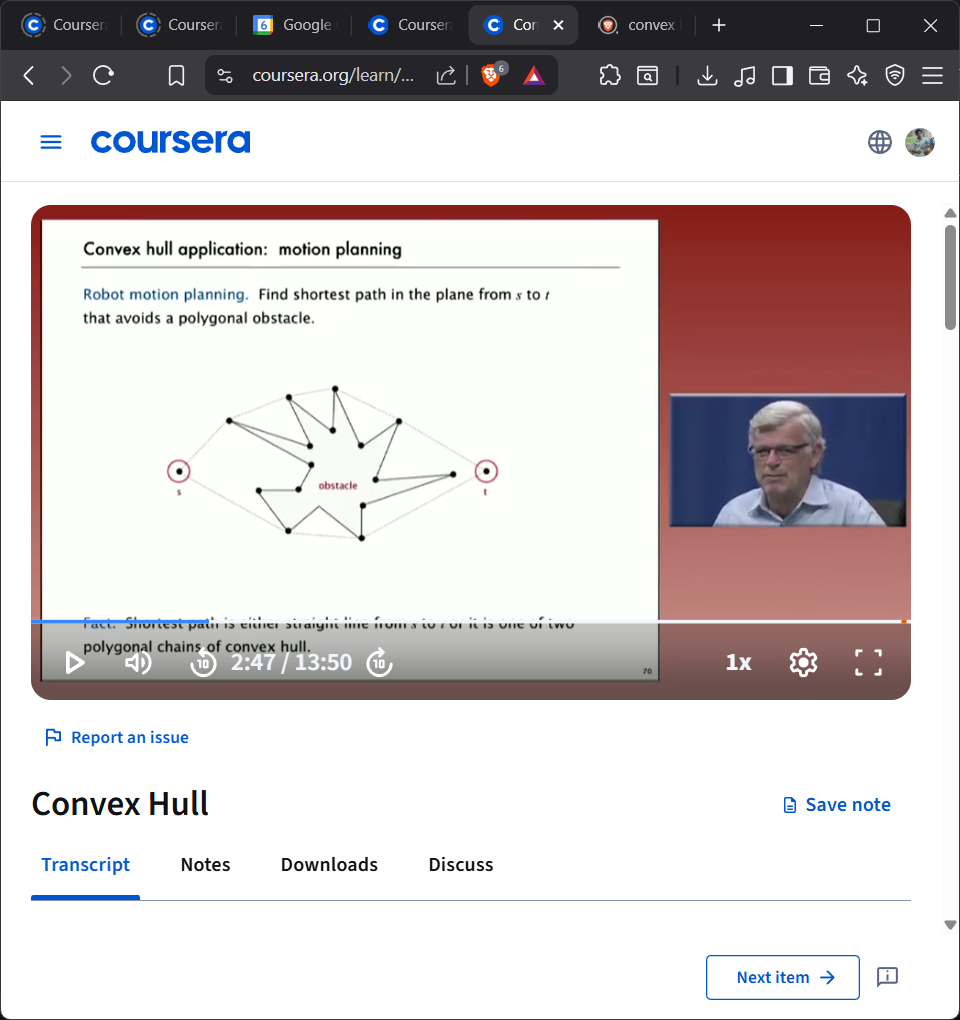
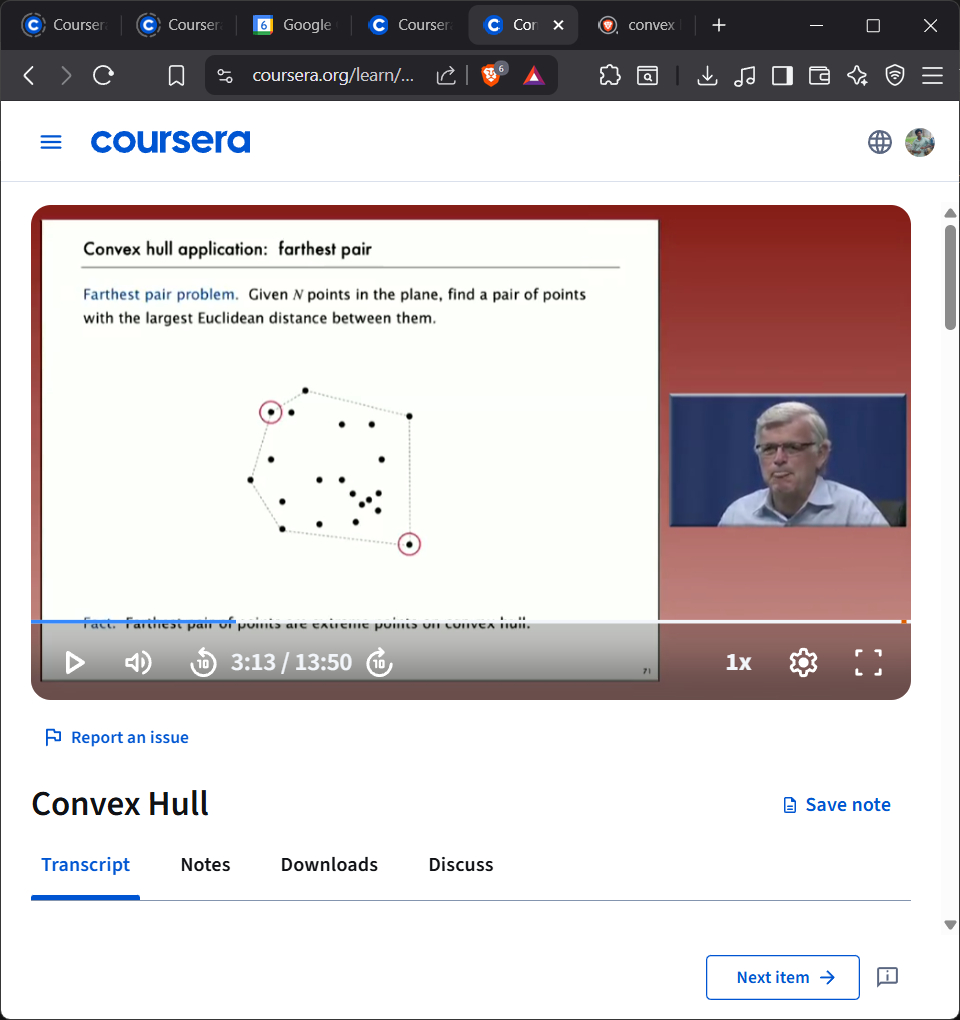
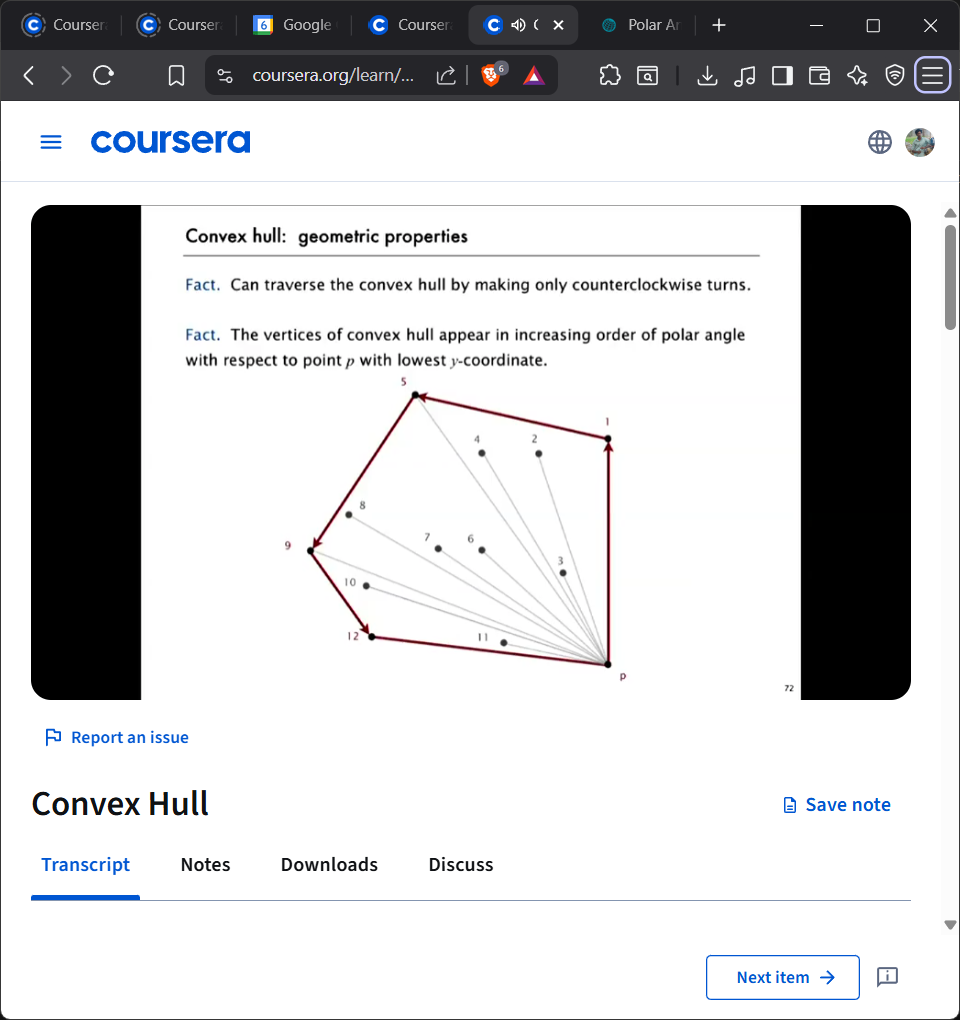
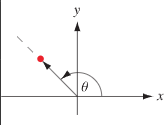
**Convex hull**

* Convex hull of a set of N points is the smallest perimeter fence enclosing the points.



* **Convex hull output:** Sequence of vertices in counterclockwise order
* **Applications:**
* **Robot motion planning:**
  + ****Finding the shortest path in the plane from s to t that avoids a polygonal obstacle.
  + Shortest path is either a straight line from s to t or it is one of two polygonal chains of convex hull.
* **Farthest pair problem:** 
  + Given N points in plain, find a pair of points with the largest Eculidean distance between them.
  + Farthest points are the extreme points of convex hull.
* **Convex hull: Geometric Properties**
  + Can traverse the convex hull by making only counterclockwise turns.
  + ****The vertices of convex hull appear in increasing order of polar angle with respect to point p with lowest y-coordinate.
  + In the plane, the polar angle  is the counterclockwise angle from the x-axis at which a point in the  plane lies.

// Graham Scan convex hull algorithm: case study

*The****Graham scan algorithm****is a simple and efficient algorithm for computing the convex hull of a set of points. It works by iteratively adding points to the convex hull until all points have been added.*

* *The algorithm starts by finding the point with the smallest y-coordinate. This point is always on the convex hull. The algorithm then sorts the remaining points by their polar angle with respect to the starting point.*
* *The algorithm then iteratively adds points to the****convex hull****. At each step, the algorithm checks whether the last two points added to the convex hull form a right turn. If they do, then the last point is removed from the convex hull. Otherwise, the next point in the sorted list is added to the convex hull.*

// Grahm Scan: Implementation challenges

* How to find p with smallest -coordinate.?
* How to sort points by polar angle with respect to p?
* How to determine whether p1->p2->p3 is a counterclockwise turn?
* How to sort efficiently?
* How to handle degeneracies (three or more points on a line)?

// Implementing Counterclockwise