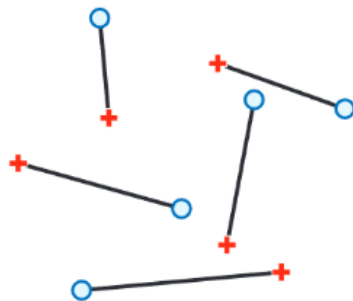


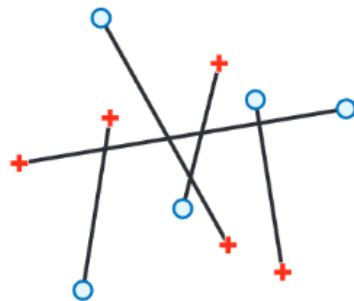
PROBLEM 1 (HAM SANDWICH HAM SANDWICH PROBLEM – INSPIRED BY DR. JEFF ERICKSON) The **Ham Sandwich Theorem** states, informally, that any ham sandwich can be cut into two equal pieces. In particular, if you have two pieces of bread with a slice of ham in between, there is a single straight slice you can make that cuts all of them into exact halves. Surprisingly, this is true under the most liberal conditions: the bread and ham can take any shape, they don't even need to be single pieces, and they can be anywhere in relation to one another, not necessarily adjacent.

You want to help people test this out by helping them create their own ham sandwiches. Let H be the set of ham shops in Chicago, and let B be the set of bread shops in Chicago, both represented as points on a 2D map. Say there are n of each type of shop. A **non-crossing matching** of H and B is a matching between ham shops and bread shops such that, when you draw straight lines connecting the matched shops, no lines intersect. A **ham-sandwich cut** is a line drawn through the map so that both the ham shops and the bread shops are evenly split into the two sides of the line. This cut should not pass through any points, and if the number of shops is odd then one side will have one more of each type of shop.

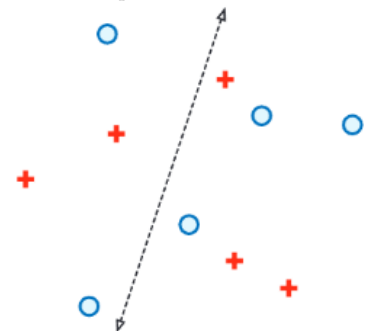
Below is an example, where blue circles are bread shops and red crosses are ham shops.



A non-crossing matching



A matching that is not non-crossing



A ham-sandwich cut

Design an algorithm to compute a **non-crossing matching** of H and B . You may call a function `HamSandwich(A, B)` that takes in two sets A and B of size n and returns (A_1, A_2, B_1, B_2) , where a **ham-sandwich cut** has split A into A_1, A_2 and split B into B_1, B_2 .

PROBLEM 2 (HAM SANDWICH LOCATOR) *You also want to make sure everyone knows which ham and bread shop to select. Chicago is modeled by a connected graph $G = (V, E)$ where some but not all vertices are labeled as either ham shops or bread shops, and edges are weighted by distance. For each vertex $v \in V$, you'd like to know both the nearest ham shop and the nearest bread shop. However, you'd like to find this all out in time that is not asymptotically larger than (one run of) Dijkstra's algorithm. Design an algorithm that does so, and prove that it is correct and runs in the time desired. Remember that there could be $O(|V|)$ of these types of shops.*

Extra Challenge: Taking the pairings from the prior problem, design an algorithm that finds the shop pair whose sum of distances is closest to each point v .