CS 577 hw02, Problem 2,

Haoyi Lu, William Shu

2(a) we create two arrays, A and B, each array contain n elements which is the order of p points and q points, for example, if the order of p is p3, p4, p2, p5, p1, Then A = [3,4,2,5,1] and B is same for q. For example if q is q2, q5, q4, q1, q3, then B = [2,5,4,1,3]. Then I create two arrays P and Q, we add elements in A to Q in sorted order, and add elements to B in the same position to Q as B to A. For example, if A = [3,4,2,5,1], B = [2,5,4,1,3], then P = [1,2,3,4,5] and Q = [3,4,2,5,1].

For value in P that i < j, two lines intersect if and only if Q[i] > Q[j]. This pair is an inversion. We need to find how many pairs of inversion and then we can get how many intersections will be.

CountAndSort(Q[1..n])

If n > 1:

C1 = CountAndSort (Q[1..[n/2]]);

C2 = CountAndSort (Q[[n/2]+1..n]);

C3 = CountAndMerge (Q[1..n], [n/2]);

Return (C1 + C2 + C3);

Else

Return 0;

CountAndMerge(Q[1..n], [n/2]):

i = 1; j = [n/2]+1;count = 0; total = 0;

For k = 1 to n;

If j > n

L[k] = Q[i]; i ++; total =+count;

Else if i > m:

L[k] = Q[j]; j++; count++;

Ekse if Q[i] < Q[j]

L[k] = Q[i]; i++; total =+ count;

Else

L[k] = Q[j]; j++; count ++;

For k = 1 to n:

Q[k] = L[k];

Return total;

Running time = O(n logn), calculated by drawing the recursion tree with T(n) = 2T(n/2) + O(n)

(b)