Time complexity analysis for problem 1:

Given an array of size nxn and array of size n, the time complexity is O(n^2). This is because we need to loop through each element in each row of A exactly once, and multiply that by each element in x. For an individual row in A, n things are multiplied together, giving O(n) time complexity. You do this n times, so you get n*O(n)=O(n^2) time complexity. Basically, you loop through two for loops each n times. n*n=n^2.

The derivation of the algorithm, and its time complexity analysis for problem 3:

The derivation of the algorithm for number 3 basically came from the fact that you can recursively define Hadmat multiplication of size n in terms of multiplication by a Hadmat matrix of size n/2 and the first half of x (also size n/2). This observation allows you to divide and conquer: Write T(n) in terms of T (n/2). We get, through this, that T(n)=2T(n/2)+O(n), which as we have seen from mergesort, gives T(n)=O (nlogn).

Your comments on the trends that you observe for problem 5:

As expected, matmult grows at a polynomial rate, backing up the claim that it is O(n^2). Likewise, hadmatmult grows at a slower rate, backing up the claim that it is O(nlogn).