Problem statement  
Find the largest distance between any two even integers in an integer array.  
Algorithm 1.  
Create a new array consisting of even numbers only. Then use nested loops to solve the problem using the newly created array of even numbers only.  
Algorithm largestDistance(A, n)  
 Input array A of n integers  
 Output largest difference between two even integers  
  
 even <- new Array(n)  
 count <- 0  
 for i <- 0 to n-1 do  
 if A[i] % 2 = 0 then  
 even[count] <- A[i]  
 increment count  
  
 distance <- 0  
 dif <- 0  
 for j<-0 to count-1 do  
 for k<-0 to n-1 do  
 if A[k] %2=0 then  
 dif<-Math.abs(even[j]-A[k])  
 if distance<dif then  
 distance <- dif  
  
 return distance  
  
 The worst case time complexity is O(n²)+ O(n) which is O(n²)  
  
  
 Compare your empirical results from W1D1 with W1D2. Are they consistent? What did you learn from  
 W1D1 and W1D2?  
  
 I checked the results from W1D1 and W2D2. I am amazed to see they are consistent and exactly behave the way according  
 to their worst case time complexity. For smaller data Algorithm 1 with worst case O(n²) and Algorithm 2 with worst  
 case O(n²) produced similar kind of result. So I checked them with very big data like array of 40000 integers and then I found  
 out that Algorithm 2 is slightly better than Algorithm 1. From these three algorithms, Algorithm 3 had the linear time complexity  
 which can be clearly seen in the graph as well. From these two labs, I learnt that the performence of an algorithm depends on its  
 time complexity and its running time can never exceed the worst case time complexity.

For Algorithm 2 and 3, please check the attached image files.