1. **When Bubble Sort performs well?**The best case is when the data is already sorted.
2. **When Bubble Sort performs worst?**The worst situation for bubble sort is when the list's smallest element is in the last position. In this situation, the smallest element will move down one place on each pass through the list.
3. **When insertion sort performs well?**Insertion sort has a fast best-case running time and is a good sorting algorithm to use if the input list is already mostly sorted.
4. **When selection sort is performing worst?**The worst case for insertion sort will occur when the input list is in decreasing order.
5. **When quick sort is performing well?**n\*log(n)
6. **When quick sort performs worst?**The worst occurs in following cases.   
   1) Array is already sorted in same order.   
   2) Array is already sorted in reverse order.
7. **When merge sort performs best?**Merge sort's best case is when the largest element of one sorted sub-list is smaller than the first element of its opposing sub-list, for every merge step that occurs
8. **When merge sort performs worst?**The worst case of merge sort will be the one where merge sort will have to do maximum number of comparisons
9. **When linear search performs best?**  
   The best case is when the value is equal to the first element of the list, in which case only one comparison is needed.  
    ***Searching***
10. **When linear performs worst?**  
    The worst case is when the value is not in the list (or occurs only once at the end of the list), in which case n comparisons are needed.
11. **When binary search performs well?**Where the target value is the middle element of the array, its position is returned after one iteration. In terms of iterations
12. **When binary search performs worst?**The binary search finds the item at the end of the list/array when it's narrowed down to a single item. As we saw earlier with the list of size 32, to narrow it down to a single item we make at least 5 comparisons.

