1. Apply merge sort and the list of 8 elements. data d=(45, 67, -12, 5, 22, 30, 50) set up a recurrance relation for the numbers.

A. split into two halves

[45, 67, -12, 5] and [22, 30, 50, 20]

* Recursive split into each half.

[45, 67] and [-12, 5]

[22, 20] and [50, 20]

* continue spiliting

* 45, 67.
* -12, 5
* 22, 30.
* 50, 20.

conquer and combine:

* merge [45] and [67] to get (45, 67)
* merge (-12) and (5) to get (-12, 5).
* merge (22) and (30) to get (20, 50).

merge resulting sublists:

* merge (45, 67) and (-12, 5).

compare 45 and -12 -> take -12.

compare 45 and 5 -> take 5.

Remaining: 45, 67

Result: [-12, 5, 45, 67].

merge [22, 30] and [20, 30].

* compare 22 and 20 -> take 20.
* compare 22 and 50 -> take 22.

* compare 30 and 50 -> take 30.

* remaining: 50.

* Result: [20, 22, 30, 50].

merge the final two sublists:

compare -12 and 20 -> take -12.

* compare 5 and 20 -> take 5

* compare 45 and 20 -> take 20.

* compare 45 and 22 -> take 22.

* compare 45 and 30 -> take 30.

* compare 45 and 50 -> take 45.

* compare 67 and 50 -> take 50.

* Remaining: 67.

Result: [-12, 5, 20, 22, 30, 45, 50, 67].

sorted list: [-12, 5, 20, 22, 30, 45, 50, 67].

2. solving the recurrence relation:

for $n=1$, $T(1)=0$.

Applying the master theorem to solve the recurrence

$T(n) = 2T(n/2) + n-1$:

* $a=2$.
* $b=2$.
* $f(n) = n-1$ (which is $\theta(n)$).

According to the master theorem, when $f(n) = \theta(n^c)$

where $c = \log_b a$, here $c = \log_2 2 = 1$,

so $f(n) = \theta(n^1)$.

thus, $T(n) = \theta(n \log n)$.

Hence, the number of comparisons made by merge-sort $T(n) = \theta(n \log n)$.

3. Find the no. of times to perform swapping for selection sort. Also estimate the time complexity for the order of notation set (12, 7, 5, -2, 18, 6, 13, 4).

A.) finding the sort of [4, -2, 5].

* find the minimum element in the list [4, -2, 5], which is -2.
* swap -2 with the first element 4
* list after first pass [-2, 4, 5].

second pass:

* find the minimum element in the list [-2, 4, 5].
* No swap in the list.
* list after second pass: [-2, 4, 5].

Total number of swaps: 1

* the total number of comparison is:

$$(n-1) + (n-2) + \dots + 1 = \frac{n(n-1)}{2} = O(n^2).$$

Hence, the time complexity for selection sort is $O(n^2)$.

4. find the index of the target 10 using binary search from the following list of elements [2, 4, 6, 8, 10, 12].

A. Intialization:

* low = 0.
* high = 5.

first iteration:

* calculate 'mid': mid = $[\frac{0+5}{2}] = 2$.
* compare 'list [mid]' with the target:
  * list[2] = 6.
  * since 6 < 10, set 'low' to 'mid+1 = 3'.

second iteration:

* calculate 'mid' : mid = $\frac{3+5}{2}$ = 4.
* compare 'list [mid]' with the target :
  * list [4] = 10.
  * since 10 = = 10, the target is found at index

Find the time complexity of the below equation:

$$T(n) = \begin{cases} 2T(n-1) & \text{if } n>0 \\ 1 & \end{cases}$$

$$T(n-1) = 2T(n-2)$$

$$T(n-2) = 2T(n-3)$$

* substitute these back into the original equation

$$T(n) = 2(2T(n-2)) = 2^2 T(n-2).$$

$$T(n) = 2^2 (2T(n-3))$$

$$= 2^3 T(n-3).$$

* continue the pattern:

$$T(n) = 2^k T(n-k)$$

* Base case:
* when $k=n$, we reach the base case:

$$T(n) = 2^n T(0).$$

* substitute the base case $T(0) = 1$.

$$T(n) = 2^n \cdot 1$$

$$= 2^n.$$

∴ Time complexity:

$$T(n) = 2^n.$$