

PL-1

QUESTIONS

برعاية :

HELWAN CODERS

Q1: What is the maximum number of swaps required in the worst case of Bubble Sort on an array of 5 elements?

- a. 10
- b. 5
- c. 4
- d. 0



Q2: Which of the following changes the minimum number of elements per pass in Bubble Sort?

- a. Decreasing the number of passes
- b. Skipping already sorted elements
- c. Reducing array size
- d. Increasing the starting index

Q3: (Tracing) What will be the state of the array {6, 2, 9, 1} after one pass of Bubble Sort?

- a. {2, 6, 1, 9}
- b. {2, 6, 9, 1}
- c. {2, 6, 1, 9}
- d. {6, 2, 9, 1}

Q4: In Linear Search, if the key is at the last index, how many comparisons will be made (array size = n)?

- a. n
- b. ~~n - 1~~
- c. ~~log n~~
- d. 1

Q5: Which of the following conditions must be satisfied before applying Binary Search?

- a. The array must have even number of elements

- b. The array must be sorted

- c. All elements must be unique

- d. The key must exist

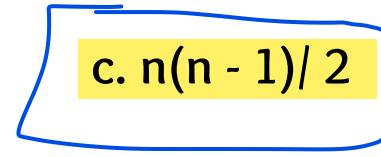
Q6: (Tracing) What is the final sorted array after applying Bubble Sort on {5, 1, 4, 2}?

- a. {1, 2, 4, 5}
- b. {2, 1, 4, 5}
- c. {5, 4, 2, 1}
- d. {1, 4, 2, 5}



Q7: Which of the following correctly represents the number of comparisons in Bubble Sort in the worst case for an array of size n ?

- a. n
- b. $n(n - 1)$
- c. $n(n - 1)/2$
- d. $\log_2(n)$



Q8: Which line in the following code is responsible for decreasing the number of comparisons in each pass of Bubble Sort?

- a. for (pass = 1; pass < SIZE; pass++)
- b. swap(&a[i], &a[i + 1])
- c. for (i = 0; i < SIZE; pass; i++)
- d. if (a[i] > a[i + 1])



Q9: (Tracing) How many passes are needed to fully sort {3, 1, 4, 2} using Bubble Sort?

- a. 3
- b. 2
- c. 4
- d. 1

Q10: In the recursive version of Binary Search, what happens if low > high?

- a. It goes to the next element
- b. It returns the last index
- c. It returns -1 (not found)
- d. It starts over from middle

Q11: What is the purpose of using pass in the Bubble Sort implementation?

- a. To track the number of elements sorted so far
- b. To track how many swaps occurred
- c. To count how many comparisons were made
- d. To reduce the range of unsorted elements



Q12: Which of the following is not a correct reason for Bubble Sort being inefficient?

- a. Requires nested loops $\rightarrow O(n^2)$
- b. Performs unnecessary ~~swaps~~^{comparisons} even when array is sorted
- c. Has worst-case time complexity of $O(\log n)$
- d. Has quadratic time complexity in average case

Q13: (Tracing) Given the array {7, 2, 4, 1}, how many comparisons and swaps happen in total during full Bubble Sort?

- a. 6 comparisons, 5 swaps
- b. 6 comparisons, 4 swaps
- c. 5 comparisons, 3 swaps
- d. 7 comparisons, 4 swaps

Q14: Which condition is used in Binary Search to determine the middle index?

- a. $(\text{low} + \text{high} + 1) / 2$
- b. $\text{low} + (\text{high} - \text{low}) / 2$
- c. $(\text{low} * \text{high}) / 2$
- d. $(\text{low} + \text{high}) / 2$

Q15: (Tracing) In recursive Binary Search, if key = 85 in array {10, 15, 30, 45, 50, 60, 75, 80, 85, 90, 95}, how many recursive calls are made?

- a. 2
- b. 3
- c. 4
- d. 5

Q16: In the iterative version of Binary Search, which condition ends the loop?

- a. $\text{low} == \text{high}$
- b. $\text{middle} == \text{key}$
- c. $\text{low} > \text{high}$
- d. $\text{low} < \text{high}$

Q17: If the Binary Search is applied on an unsorted array, what could be the result?



- a. It will always find the key
- b. It will find the key in logarithmic time
- c. It might give incorrect or undefined results
- d. It will sort the array before searching

Q18: Which of the following is true about the recursive Binary Search function?

- a. It does not use base cases
- b. It searches in halves by making new function calls
- c. It is faster than the iterative version in all cases
- d. It always takes fewer steps than iterative

Q19: Why is Binary Search considered more efficient than Linear Search on large sorted datasets?

- a. It checks every element
- b. It always finds the key in $O(n)$
- c. It eliminates half the array with each comparison
- d. It uses recursion

$$O(g) = \frac{1}{2} + \frac{1}{2} = 4$$

Q20: (Tracing) In array {3, 6, 11, 17, 24, 38, 41, 56, 69, 75}, searching for 24 using iterative Binary Search - which indices are checked in order?

- a. 4 → found
- b. 5 → 2 → 3 → 4
- c. 4 → 2
- d. 5 → 2 → 4

38 ? 24 → 7 → 5

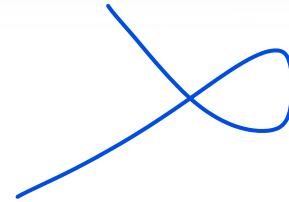


Q A

ANSWERS



| 1 | A



| 2 | B

| 3 | C

| 4 | A

| 5 | B

| 6 | A

| 7 | C

| 8 | C

| 9 | A

| 10 | C

Q A

| 11 | D

| 12 | C

| 13 | B

| 14 | D

| 15 | C

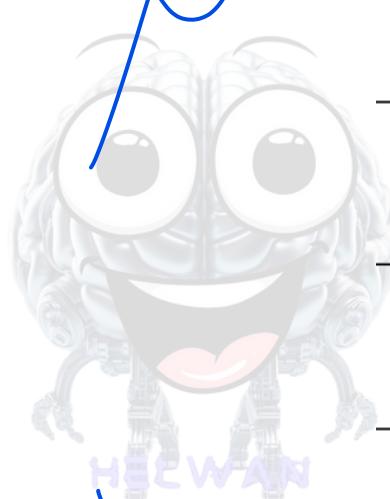
| 16 | C

| 17 | C

| 18 | B

| 19 | C

| 20 | D



{6}