1. What data structure is commonly used to implement a graph? a. Adjacency List b.Array c. Stack d. Heap
2. What is the worst-case scenario for quick sort? a. O(n^2) b. O(n log n) c. O(log n) d. O(2n)
3. What algorithm is used to find the shortest path from x node to y node in a weighted graph? a. Dijkstra b. BFS c. DFS
4. Which sorting algorithm always uses a pivot point to partition the data? a. Merge sort b. Quick sort c. Bubble Sort d.sort
5. What is the big O running time for merge sort?a. n log n b. n^2 c. log n d. n
6. Which of the following is true about Dijkstra's algorithm? c. Dijkstra's algorithm uses a priority queue to select the next node with the smallest known distance. 7. Which sorting algorithm is theoretically the fastest? a. bubble b. merge c. insertion
8. What does DFS stand for? a. Depth First Search b. Depth First Sort c. Deep First Sort d. Deep First Search
9. Which graph searching algorithm should NOT be implemented recursively? a. DFS b. BFS
10. Question: What does Dijkstra's algorithm do? b. find the shortest path between nodes in a graph
11. For disjoint sets, which of the following is NOT a way to construct a union? a. size b. height c. rank d. width
12. Which of these is a stable sorting algorithm? a. quick sort b. merge sort c. bubble sort d. insertion sort
13. Which sorting algorithm does not use divide and conquer. merge sort b. quick sort c. monkey/bogosort .
14. What container does BFS use? (n = 2) a. Stack b. Vector c. Array d. queue
15. What container does DFS use? (n = 2) a. stack b. vector c. array d. queue

---

1) weighted/ unweighted directed/ undirected? weighted = number directed = arrows
2) What is a real-life weighted graph? gps navigation b/c it takes into account distance which is numerical
3) What is a graph? Provide at least one example where it can be useful. A graph is a data structure consisting of nodes (contain data) and edges (connection between two nodes). they can be directed (edges have direction) and they can be weighted (edges have a weight). GPS navigator the locations is the nodes, the edges is the roads between the locations
4) Explain the difference between BFS and DFS. What data structures are used with each, and why? BFS = one level at a time visiting all nodes that are neighbors of the current nodes and then moves on to next node queue because nodes are visited level by level and need to be processed in the order visited FIFO DFS = go down one branch as deep as possible, until it reaches a dead end or a previously visited node then backtrack to last node that has unvisited adjacent neighbor stack to be able to pop elements off since we need to keep track of where to backtrack to when needed LIFO
5) Describe potential advantages of BFS over DFS? bfs = shortest path in unweighted graph and explore a graph in levels .
6) Compare and contrast the advantages/disadvantages of adjacency lists vs adjacency matrix? matrix = 1. allows you to check whether an edge exists between two nodes in constant time O(1), because it's simply a matter of indexing into a 2D array., 2. more memory but faster edge lookup , 3.if you are looking for neighbors, you have to scan an entire row of the list = 1.requires a linear search through the list of neighbors to check for the existence of an edge, which takes O(n) time, 2. uses less memory but slower edge lookup , 3. faster to iterate over the neighbors
connected = there is a path b/t every pair of nodes acyclie = does not contain any cycles simple = no self loop, no multiple edges
7) Which sorting algorithm involves selecting a random partition and sorting the elements on each side of the partition? quicksort
8) Explain the primary difference between Merge Sort and Quick Sort in terms of their approach to sorting and performance characteristics merge sort = the list is recursively split in half until each element is its own sublist and then each sublist is combined with another in order to make a new ordered list this is done recursively until there is only one ordered list/ stable/ time complexity always O(n log n) because always divides the array into halves/ space complexity O(n) quicksort = find a pivot and split the list into two sublists one with elements smaller than the pivot and one with elements greater or equal to the pivot, put pivot in the correct spot and recursively apply the same process to each sublist/ not stable/ time complexity Best: O(n log n) Worst Case: O(n^2) depending on the picking of the pivot/ space complexity O(log n)
9) Explain why a divide and conquer algorithm can be O(n log n). This is because the array is divided into log n levels, and each level requires O(n) operations for merging.
10) For quicksort, explain why having a good pivot is important and the consequences of choosing a bad pivot. b/c it makes sure that the list is equally divided which means recursive is calls less causing a more efficient time complexity of O(n log n)
11) quick sort real world? the best average case efficiency and it has low memory usage since it is an in-place sorting algorithm
12) What is your favorite sort algorithm and why? merge because it is simple and it is consistent no matter the size of the data
13) In your own words, what is a disjoint set and how union is used with them? A disjoint set is a group of separate sets where each item only belongs to one set. The union operation is used to combine two separate groups into one larger group
14) In a disjoint set: what is the difference between union by height, and union by rank with path compression? In union by height you find the height of each set and then the node with the smaller height points to the node with the bigger height. In rank with path compression the rank is determined by the theoretical high,when find is called it compresses the path to the root
15) Why use Dijkstra's algorithm over DFS/BFS? it gives the shortest path in weighted graphs from source node to every other node
16) What is stability in a sorting algorithm? stability is that if two / more elements have equal values their relative order in the sorted list is the same as the inputted list this is useful when sorting on multiple factors such as last name first name.
merge = stable insertion = stable quick = not stable What is one use case for a heap data structure? priority queue

17) linked list locate an element based on its value = o(n) Insert an element in the front = o(1) Insert an element in the back = o(n)
18) doubly linked list better time complexity then singly linked list? deleting a node single = go through all list to find previous node o(n) double = there is a pointer to previous node so easier o(1)
19) Reheapification restores the heap property after an element is added to the end of the heap. In a max heap, if the newly inserted element is larger than its parent, it needs to move upward in the tree. in a min heap, if the newly inserted element is smaller than its parent, it also moves upward. The time complexity of this process is O(logn).