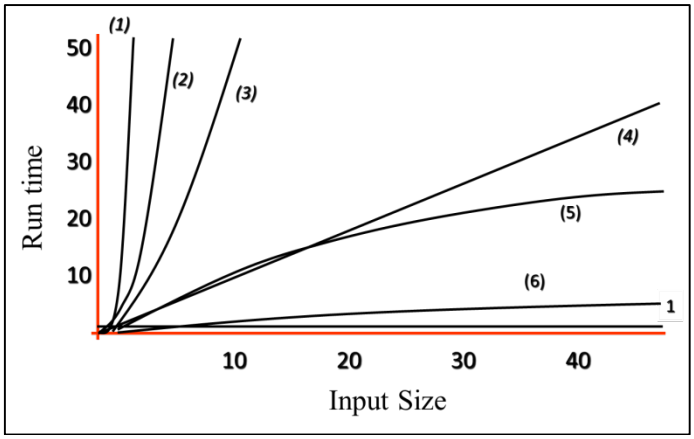
1. **Complexity**

Match each curve in the graph with one of the O() functions -- $N\log_2 N$, $N^3$, $(\log_2 N)^2$, N, $N^2$, $\log_2 N$.




| Curve | O() function |
|---|---|
| (1) | |
| (2) | |
| (3) | |
| (4) | |
| (5) | |
| (6) | |

2. **Recursive algorithm.** Write pseudo-code to solve the following two problems.

(1) Print elements of a linked list in reverse order by using same single linked list.

(2) Definition: to rotate an array by k times, we shift all items in the array to the right by k slots, and put the most right k items in the original array to the left of the rotated array. For instance:

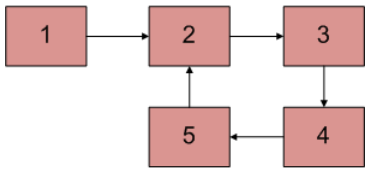
Original array: **1, 3, 6, 9, 12, 15**

Rotated by 2: **12, 15, 1, 3, 6, 9**

Find the minimum element in a rotated sorted array (we don't know how many time it has been rotated).

3. **Singly Linked List.** In text (bulleted list) or pseudo-code, describe the algorithms. Assume the length of the list is N, state the space and time complexity.
(1) Write an algorithm to retrieve the middle element in a singly linked list.

(2) A malformed singly linked list with a loop (see the mini example below) causes iteration over the list to fail because the iteration will never reach the end of the list. Write an algorithm to detect loop in a singly linked list.



4. **Sorting**. Sort an array containing the digits 71808294 in ascending order. Show how the order of the digits changes during each step of the following sorting algorithms. Clearly state what data structures are used in each algorithm (tree, array, list, etc).

(1) insertion sort

(2) selection sort

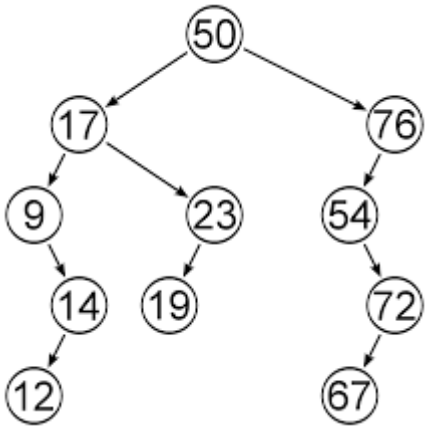
(3) mergesort

(4) quicksort (always choosing the first element of any subarray to be the pivot)

(5) quicksort (always choosing the last element of any subarray to be the pivot)

(6) heapsort

5. **Trees**. Traverse the following tree.



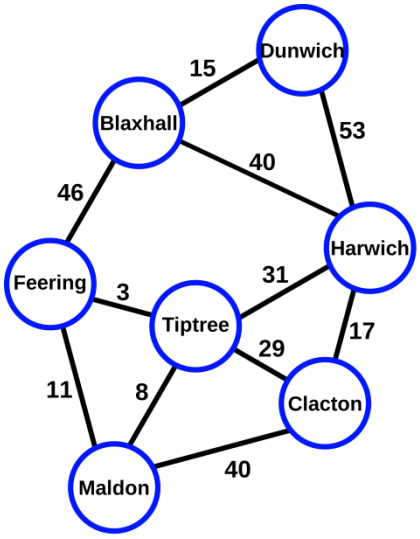

(1) In-order traversal: show the sequence of nodes being visited.

(2) In-order + iterative: Illustrate the stack content in each step.

(3) In-order + iterative: write down the pseudo-code that implements the traversal.

(4) Post-order traversal: write down the sequence of nodes being visited.

6. **Graphs.** Here is a map of the East Anglia towns. Edge weight indicates physical distance.




(a) Represent the graph in adjacency matrix and adjacency lists. You may list the nodes in alphabetic order.

(b) Run BFS starting from Tiptree. Assume that the Visit() procedure considers all neighbors of a given vertex in alphabetic order. In each step: mark the current vertex; mark vertices whose neighbors have all been considered; illustrate the queue Q.

(c) You are setting up high-speed fibers to connect all towns. Run Prim's to identify MST. In each step, highlight the current MST. Initial node: Harwich.

(d) You are traveling from Maldon to Dunwich. Run Dijkstra's to identify the shortest path.

7. **Hashtable.** Design algorithms in text (bulleted list) or pseudo-code. Describe your data structures and the time/space complexity.

(1) Given an array of integers, identify the most common integer(s).

(2) Given two arrays of integers, identify integer(s) that show up in both arrays.

8. **Graph.** You are given a weighted directed graph $G = (V, E, w)$ and the shortest path distances $\delta(s, u)$ from a source vertex $s$ to every other vertex in $G$. However, you are not given $\pi(u)$ (the predecessor pointers). With this information, give an algorithm to find a shortest path from $s$ to a given vertex $t$ in $O(V + E)$ time.

9. **Applications.** Design algorithms in text (bulleted list) or pseudo-code. Describe your data structures and the time/space complexity.

(1) Numbers are randomly generated and passed to a method. Write a program to find and maintain the median value as new values are generated.

(2) You have 10 giga integers, each of which is 8 bytes. All original numbers are stored in a hard drive; your computer has a physical memory of 4 giga bytes. Design an algorithm to find the largest 1 million numbers. Why you choose the algorithms over others? (1 giga = $10^9$; 1 million = $10^6$)