Algorithm

Sieve of Eratosthenes

MAIN IDEA | *Cross out all the multiple number from 2 → n*

EXAMPLE : Find all prime number in the following list

2, 3, 4, 5, 6, 7, 8, 9, 10

Step 1: 2, 3, ~~4~~, 5, ~~6~~, 7, ~~8~~, 9, ~~10~~

Step 2: 2, 3, ~~4~~, 5, ~~6~~, 7, ~~8~~, ~~9~~, ~~10~~

Prime : 2, 3, 5, 7

Merge Sort

MAIN IDEA | *Find the mid point and then split array into half*
Merge sort the left array
Merge sort the right array
Merge left and right array
Better apply to LinkedList
Rune time: O(nlogn)

EXAMPLE : Sort the following list

1. | 4 | 3 | 7 | 2 | 1 | 6 |

   ↙ ↘

2. | 4 | 3 | 7 | | 2 | 1 | 6 |

   ↙ ↘ ↙ ↘

3. | 4 | 3 | | 7 | | 2 | 1 | | 6 |

   ↙ ↘ ↙ ↘

4. | 4 | | 3 | | 7 | | 2 | | 1 | | 6 |

   ↘ ↙ ↘ ↙

5. | 3 | 4 | | 7 | | 1 | 2 | | 6 |

   ↘ ↙ ↘ ↙

6. | 3 | 4 | 7 | | 1 | 2 | 6 |

   ↘ ↙

7. | 1 | 2 | 3 | 4 | 6 | 7 |

Quick Sort

MAIN IDEA | *All about partition*
Pick one pivot random, first number, mid or last number
Any number less then pivot going to left array
Any number equal to pivot going to mid array
Any number greater then pivot going to right array
Recursively sort the left and right array
Better apply to array
Run time: worse case-O(n2), best case-O(nlogn)

EXAMPLE : Sort the following list

→ ←

1. | 4 | 3 | 7 | 2 | 1 | 6 | pivot: 7

→ ←

2. | 4 | 3 | 7 | 2 | 1 | 6 | pivot: 7

→ ←

3. | 4 | 3 | 6 | 2 | 1 | 7 | pivot: 7

→ ←

4. | 4 | 3 | 6 | 1 | 2 | 7 | pivot: 7

→ ← → ←

5. | 4 | 3 | 6 | 2 | pivot: 3 | 1 | 7 | pivot: 1

.....

6. | 1 | 2 | 3 | 5 | 6 | 7 |

Quick Select

MAIN IDEA | *Similar as quick sort*
The difference is quick sort recursively sort left, mid and right Quick Select only recursively sort left or right

Selection Sort

MAIN IDEA | *Iterate through the array then pick the smallest item and put it in the front*
Run Time: O(n2)

Insertion Sort

MAIN IDEA | *Iterate through the array then place the item in its correct position*
If current item is smaller then previous swap them and continues until it reach its correct position
Run Time: O(n2)

Bubble Sort

MAIN IDEA | *Repeatedly iterate through the array, if current number is greater then the next number swap them until it find its correct position*
Run Time: O(n2)

Binary Search

MAIN IDEA | *Find mid point then compare with the target*
Base case if the mid is equal to target then return it
If the mid is less then target recursive left half
If the mid if greater than target recursive right half
Run time: O(logn)

EXAMPLE : Find 7 in the following sorted array

1 3 5 7 8
L M R

1. | 1 | 3 | 5 | 7 | 8 | mid: 5, $5 < 7$
L M R

2. | 1 | 3 | 5 | 7 | 8 | mid: 7, $7 == 7$