BUBBLE SORT - greatest element will be at the last position - multiple swaps in each iteration - time: O(n^2) -space: O(1) inplace

In [1]:
```
import time
import random

def bubble_sort(num):
    for i in range(len(num)):
        for j in range(0, len(num)-i-1):
            if num[j] > num[j+1]: # then swap
                temp = num[j]
                num[j]= num[j+1]
                num[j+1] = temp

while True:
    print()
    user_input = input('Enter the numbers or type random for random list or press q to quit:')
    if user_input.lower()== 'q':
        print('EXIT')
        print("THANKYOU")
        break

    elif user_input.lower()== 'random':
        num= [random.randint(1, 1000) for i in range(500)]
    else:
        try:
            num = list(map(int, user_input.strip().split()))
        except:
            print("INVALID ENTRY")
            print("ENTER INTEGER VALUE ONLY")
            continue

    start_time = time.perf_counter()
    bubble_sort(num) # call the function on the list
    end_time= time.perf_counter()
    print('Sorted list:', num[:5])
    print('Length of the input is:', len(num))
    print(f'Execution time: {end_time - start_time:.4f} seconds')

# inner loop will do the swapping
```

```
# outer loop for number of passes depending on the number of elements
```

```
Sorted list: [1, 3, 5, 9, 9]
Length of the input is: 500
Execution time: 0.0306 seconds

EXIT
THANKYOU
```

INSERTION SORT - time: O(n^2) - space: O(1)

In [3]:
```
import random
import time


def insertion_sort(arr):
    for i in range(1, len(arr)):
        key = arr[i]
        j= i-1
        while j>=0 and key<arr[j]: # then swap
            arr[j+1]= arr[j]
            j -= 1
        arr[j+1] = key


while True:
    print()
    user_input = input('Enter the numbers or type random for random list or press q to quit:')
    if user_input.lower()== 'q':
        print('EXIT')
        print("THANKYOU")
        break
    elif user_input.lower()== 'random':
        arr= [random.randint(1, 1000) for i in range(500)]
    else:

        try:
            arr = list(map(int, user_input.strip().split()))
        except:
            print("INVALID ENTRY")
            print("ENTER INTEGER VALUE ONLY")
```

```
            continue

    start_time = time.perf_counter()
    insertion_sort(arr) # call the function on the list
    end_time= time.perf_counter()
    print('Sorted list:', arr[:5])
    print('Length of the input is:', len(arr))
    print(f'Execution time: {end_time - start_time:.4f} seconds')
```

```
Sorted list: [2, 6, 11, 12, 15]
Length of the input is: 500
Execution time: 0.0080 seconds

EXIT
THANKYOU
```

insertion sort works faster than bubble sort

SELECTION SORT - gets minimum element on the first position after pass 1 - one swap each iteration (ie is the min value swap) - Time: O(n^2) - space:O(1)

In [5]:
```
import random
import time


def selection_sort(my_list):
    for i in range(len(my_list)):
        min_position = i
        for j in range(i+1, len(my_list)):
            if my_list[j] < my_list[min_position]: #then swap
                min_position = j
        tem_val = my_list[i] # after getting first min pass1
        my_list[i]= my_list[min_position]
        my_list[min_position] = tem_val


while True:
    print()
    user_input = input('Enter the numbers or type random for random list or press q to quit:')
    if user_input.lower()== 'q':
        print('EXIT')
```

```
        print("THANKYOU")
        break
    elif user_input.lower()== 'random':
        my_list= [random.randint(1, 1000) for i in range(500)]
    else:

        try:
            my_list = list(map(int, user_input.strip().split()))
        except:
            print("INVALID ENTRY")
            print("ENTER INTEGER VALUE ONLY")
            continue

    start_time = time.perf_counter()
    selection_sort(my_list) # call the function on the list
    end_time= time.perf_counter()
    print('Sorted list:', my_list[:10])
    print('Length of the input is:', len(my_list))
    print(f'Execution time: {end_time - start_time:.4f} seconds')
```

```
Sorted list: [1, 3, 3, 5, 10, 11, 16, 16, 17, 18]
Length of the input is: 500
Execution time: 0.0086 seconds

EXIT
THANKYOU
```

MERGE SORT - Time: O(nlogn) - Space: O(n) memory usage grows linearly with the size of the input array.

In [7]:
```
# merge sort
import random
import time

def merge_sort(arr):
    if len(arr) <=1:
        return arr  #base case when there is only one eelement in the list or its empty
    mid = len(arr) // 2
    left = arr[:mid]
    right =arr[mid:]

    #print(f"Splitting: {arr} : {left} || {right}")
```

```
    left = merge_sort(left)
    right = merge_sort(right)


    return merge_two_sorted_list(left, right)

def merge_two_sorted_list(a,b):
    sorted_list=[]
    len_a = len(a)
    len_b = len(b)
    i=j=0

    while i< len(a) and j< len_b:
        if a[i] <= b[j]:
            sorted_list.append(a[i])
            i +=1
        else:
            sorted_list.append(b[j])
            j+=1

    while i<len(a):
        sorted_list.append(a[i])
        i+=1
    while j<len(b):
        sorted_list.append(b[j])
        j+=1


    return sorted_list
if __name__ == '__main__':
    while True:
        print()
        user_input = input('Enter the numbers or type random for random list or press q to quit:')
        if user_input.lower()== 'q':
            print('EXIT')
            print("THANKYOU")
            break
        elif user_input.lower()== 'random':
            arr= [random.randint(1, 1000) for i in range(500)]
        else:

            try:
```

```
                arr = list(map(int, user_input.strip().split()))
            except:
                print("INVALID ENTRY")
                print("ENTER INTEGER VALUE ONLY")
                continue
        start_time = time.perf_counter()
        sorted_arr = merge_sort(arr)
        end_time = time.perf_counter()


        print("Sorted array:", sorted_arr[:10])
        print('Length of the input is:', len(arr))
        print(f'Execution time: {end_time - start_time:.4f} seconds')
```

```
Sorted array: [5, 5, 6, 9, 9, 13, 15, 18, 20, 23]
Length of the input is: 500
Execution time: 0.0013 seconds

EXIT
THANKYOU
```

QUICK SORT- Time: O(n^2) picking first or last element as pivot O(nlogn) median as pivot - Space: O(n) recursive nature

In [9]:
```
import time
import random


def quick_sort(apple):
    if len(apple) <=1:
        return apple
    p= apple[-1]

    L= [x for x in apple[:-1] if x <=p]
    R= [x for x in apple[:-1] if x >p]

    L= quick_sort(L)
    R = quick_sort(R)

    return L + [p] +R
```

```
while True:
    print()
    user_input = input('Enter the numbers or type random for random list or press q to quit:')
    if user_input.lower()== 'q':
        print('EXIT')
        print("THANKYOU")
        break

    elif user_input.lower()== 'random':
        apple= [random.randint(1, 1000) for i in range(500)]
    else:
        try:
            apple = list(map(int, user_input.strip().split()))
        except:
            print("INVALID ENTRY")
            print("ENTER INTEGER VALUE ONLY")
            continue

    start_time = time.perf_counter()
    sorted_list= (quick_sort(apple))
    #print(sorted_list)

   # quick_sort(apple) # call the function on the list
    end_time= time.perf_counter()
    print('Sorted list:', sorted_list[:15])
    print('Length of the input is:', len(apple))
    print(f'Execution time: {end_time - start_time:.4f} seconds')
```

```
Sorted list: [1, 2, 3, 10, 10, 11, 11, 14, 23, 24, 26, 28, 31, 31, 35]
Length of the input is: 500
Execution time: 0.0019 seconds

EXIT
THANKYOU
```

QUICK SORT USING 3 MEDIANS:

In [11]:
```
import time
import random
```

```
def three_median(mlist):
    first = mlist[0]
    middle = mlist[len(mlist)// 2]
    last = mlist[-1]

    return sorted([first, middle, last])[1]
def qs_3median(mlist):
    if len(mlist)<=1:
        return mlist

    pivot = three_median(mlist)

    L=[x for x in mlist if x<pivot]
    P=[x for x in mlist if x==pivot]
    R=[x for x in mlist if x>pivot]

    return qs_3median(L)+ P+ qs_3median(R)



while True:
    print()
    user_input = input('Enter the numbers or type random for random list or press q to quit:')
    if user_input.lower()== 'q':
        print('EXIT')
        print("THANKYOU")
        break
    elif user_input.lower()== 'random':
        mlist= [random.randint(1, 1000) for i in range(500)]
    else:

        try:
            mlist = list(map(int, user_input.strip().split()))
        except:
            print("INVALID ENTRY")
            print("ENTER INTEGER VALUE ONLY")
            continue

    start_time = time.perf_counter()
    sorted_lst = qs_3median(mlist)
```

```
    end_time= time.perf_counter()
    print('Sorted list:', sorted_lst[:5])
    print('Length of the input is:', len(mlist))
    print(f'Execution time: {end_time - start_time:.4f} seconds')
```

```
Sorted list: [2, 3, 3, 4, 4]
Length of the input is: 500
Execution time: 0.0017 seconds

EXIT
THANKYOU
```

HEAP SORT time: O(nlogn) space:O(1)

In [13]:
```
import time
import random


def swap(lst, i ,j):
    lst[i], lst[j] = lst[j], lst[i]
def shiftdown(lst,i, upper):
    while(True):
        l, r= i*2+1, i*2+2 # left and right children
        largest = i
        if l < upper and lst[l] > lst[largest]:
            largest = l
        if r < upper and lst[r] > lst[largest]:
            largest = r
        if largest != i:
            swap(lst, i, largest)
            i = largest
        else:
            break

def heapsort(lst):
    for j in range((len(lst)-2)//2, -1, -1):
        shiftdown(lst,j, len(lst)) # heapify

    for end in range(len(lst)-1,0,-1):
        swap(lst, 0, end)
        shiftdown(lst,0,end)
```

```
while True:
    print()
    user_input = input('Enter the numbers or type random for random list or press q to quit:')
    if user_input.lower()== 'q':
        print('EXIT')
        print("THANKYOU")
        break

    elif user_input.lower()== 'random':
        lst= [random.randint(1, 1000) for i in range(500)]
    else:
        try:
            lst = list(map(int, user_input.strip().split()))
        except:
            print("INVALID ENTRY")
            print("ENTER INTEGER VALUE ONLY")
            continue

    start_time = time.perf_counter()
    heapsort(lst)

    end_time= time.perf_counter()
    print('Sorted list:', lst[:15])
    print('Length of the input is:', len(lst))
    print(f'Execution time: {end_time - start_time:.4f} seconds')
```

```
Sorted list: [2, 5, 12, 12, 12, 13, 15, 17, 19, 20, 23, 30, 31, 31, 33]
Length of the input is: 500
Execution time: 0.0030 seconds

EXIT
THANKYOU
```

In [ ]: