UNIT 27

pop quiz 1

8 swaps were executed in the given code

qn 2

12 comparison operations were executed during the given process

In [1]:
```
# qn 1

def are_anagrams(word1, word2):
    return sorted(word1) == sorted(word2)

w1 = input("Enter first word: ")
w2 = input("Enter second word: ")

if are_anagrams(w1, w2):
    print("Anagrams")
else:
    print("Not Anagrams")
```

```
Anagrams
```

In [6]:
```
# qn 2

def selection_sort2(s):
    s = list(s)
    for i in range(len(s)):
        min_idx = i
        for j in range(i+1, len(s)):
            if s[j] < s[min_idx]:
                min_idx = j
        s[i], s[min_idx] = s[min_idx], s[i]
    return ''.join(s)

def are_anagrams_selection(word1, word2):
    return selection_sort2(word1) == selection_sort2(word2)


print(are_anagrams_selection("listen", "silent") );
print(are_anagrams_selection("anagram", "nagaram") );
print(are_anagrams_selection("listen", "silence") );
print(are_anagrams_selection("anagram", "anagrams") );
```

```
True
True
False
False
```

In [5]:
```python
def insertion_sort2(s):
    s = list(s)
    for i in range(1, len(s)):
        key = s[i]
        j = i - 1
        while j >= 0 and s[j] > key:
            s[j + 1] = s[j]
            j -= 1
        s[j + 1] = key
    return ''.join(s)

def are_anagrams_insertion(word1, word2):
    return insertion_sort2(word1) == insertion_sort2(word2)

print(are_anagrams_insertion("listen", "silent") );
print(are_anagrams_insertion("anagram", "nagaram") );
print(are_anagrams_insertion("listen", "silence") );
print(are_anagrams_insertion("anagram", "anagrams") );
```

```
True
True
False
False
```

In [ ]:
```python
```

unit 28

In [ ]:
```python
# pop quiz 1

array 1 - merge2() function was called 9 times
array 2- merge2() function was called 12 times
```

In [7]:
```python
# pop quiz 2
def multiway_merge(lists):
    result = []
    for lst in lists:
        result += lst
    result.sort()
    return result

N = int(input("Input the number of list: "))
list_of_nums = []

for i in range(N):
    nums = list(map(int, input("Input a list of numbers: ").split()))
```

```
    list_of_nums.append(nums)

sorted_list = multiway_merge(list_of_nums)
print("Merged into:", sorted_list)
```

```
Merged into: [1, 2, 3, 4, 5, 6, 7]
```

In [ ]:

unit 29

In [9]:
```
# pop quiz 1
[13, 10, 12, 15, 25, 20, 22]
```

Out[9]:
```
[13, 10, 12, 15, 25, 20, 22]
```

In [ ]:
```
# qn 1

Step 1: Start
Step 2: Read the array A containing N elements
Step 3: Read the value of K
Step 4: Sort the array A in ascending order
Step 5: Compute the index of the Kth largest element
   Index = N - K
Step 6: Retrieve the element at position A[N - K]
Step 7: Return A[N - K] as the Kth largest element
Step 8: Stop
```

Step 1: Start Step 2: Read array A of N elements Step 3: Read value of K Step 4: Compute target index Target = N − K Step 5: Call QuickSelect(A, 0, N − 1, Target) Step 6: Return the result Step 7: Stop

QUICK SELECT : Step 1: If low ≤ high, continue Step 2: Call partition(A, low, high) Let pivotIndex = returned index Step 3: If pivotIndex == Target Return A[pivotIndex] Step 4: Else if pivotIndex > Target Recursively call QuickSelect(A, low, pivotIndex − 1, Target) Step 5: Else (pivotIndex < Target) Recursively call QuickSelect(A, pivotIndex + 1, high, Target)

PARTITION PROCEDUE Step 1: Select the last element as pivot pivot = A[high] Step 2: Initialize i = low − 1 Step 3: For j from low to high − 1 do: If A[j] ≤ pivot then Increment i by 1 Swap A[i] and A[j] Step 4: After loop ends, swap A[i + 1] and A[high] Step 5: Return i + 1 (final pivot position)

In [ ]:

In [ ]:

In [ ]: