In [ ]:
```
1.find min and max
```

In [2]:
```
arr = [3, 5, 1, 2, 4, 8, 7]
def find_min_max(arr, low, high):
    if low == high:
        return arr[low], arr[low]
    elif high == low + 1:
        if arr[low] < arr[high]:
            return arr[low], arr[high]
        else:
            return arr[high], arr[low]
    else:
        mid = (low + high) // 2
        min1, max1 = find_min_max(arr, low, mid)
        min2, max2 = find_min_max(arr, mid + 1, high)
        return min(min1, min2), max(max1, max2)
min_val, max_val = find_min_max(arr, 0, len(arr) - 1)
print(f"Minimum value:{min_val}, Maximum value: {max_val}")
```

```
Minimum value: 1, Maximum value: 8
```

2.merge sort

In [3]:
```
def merge_sort(arr):
    if len(arr) <= 1:
        return arr
    mid = len(arr) // 2
    left = merge_sort(arr[:mid])
    right = merge_sort(arr[mid:])
    return merge(left, right)
def merge(left, right):
    result = []
    while left and right:
        if left[0] < right[0]:
            result.append(left.pop(0))
        else:
            result.append(right.pop(0))
    result.extend(left or right)
    return result
arr = [12, 11, 13, 5, 6, 7]
print("Given array is:", arr)
sorted_arr = merge_sort(arr)
print("Sorted array is:", sorted_arr)
```

```
Given array is: [12, 11, 13, 5, 6, 7]
Sorted array is: [5, 6, 7, 11, 12, 13]
```

In [ ]:
```
3.quick sort
```

In [5]:
```
def quick_sort(arr):
    if len(arr)<= 1:
        return arr
    pivot=arr[len(arr)// 2]
    left=[x for x in arr if x<pivot]
    middle=[x for x in arr if x==pivot]
    right=[x for x in arr if x>pivot]
    return quick_sort(left)+ middle+quick_sort(right)
arr = [12, 11, 13, 5]
print("Given array is:", arr)
sorted_arr = quick_sort(arr)
print("Sorted array is:", sorted_arr)
```

```
Given array is: [12, 11, 13, 5]
Sorted array is: [5, 11, 12, 13]
```

In [ ]:
```
4.binary search
```

In [6]:
```
def binary_search(arr, target):
    low= 0
    high=len(arr)- 1
    while low<=high:
        mid=(low + high)// 2
        if arr[mid]<target:
            low=mid+1
        elif arr[mid]>target:
            high=mid-1
        else:
            return mid
    return -1
arr=[2, 5, 8, 12, 16, 23, 38, 56, 72, 91]
target= 2
result=binary_search(arr, target)
if result!=-1:
    print("Element is there at index", result)
else:
    print("Element is not there in the array")
```

```
Element is there at index 5
```

In [ ]:
```
5.strassen matrix multiplication
```

In [1]:
```
def add(A, B):
    return [[A[i][j] + B[i][j] for j in range(len(A))] for i in range(len(A
def sub(A, B):
    return [[A[i][j] - B[i][j] for j in range(len(A))] for i in range(len(A
def strassen(A, B):
    n = len(A)
    if n == 1:
        return [[A[0][0] * B[0][0]]]
    half = n // 2
    A11=[[A[i][j] for j in range(half)] for i in range(half)]
    A12=[[A[i][j] for j in range(half, n)] for i in range(half)]
    A21=[[A[i][j] for j in range(half)] for i in range(half, n)]
    A22=[[A[i][j] for j in range(half, n)] for i in range(half, n)]
    B11=[[B[i][j] for j in range(half)] for i in range(half)]
    B12=[[B[i][j] for j in range(half, n)] for i in range(half)]
    B21=[[B[i][j] for j in range(half)] for i in range(half, n)]
    B22=[[B[i][j] for j in range(half, n)] for i in range(half, n)]
    M1=strassen(add(A11, A22), add(B11, B22))
    M2=strassen(add(A21, A22), B11)
    M3=strassen(A11, sub(B12, B22))
    M4=strassen(A22, sub(B21, B11))
    M5=strassen(add(A11, A12), B22)
    M6=strassen(sub(A21, A11), add(B11, B12))
    M7=strassen(sub(A12, A22), add(B21, B22))
    C11=add(sub(add(M1, M4), M5), M7)
    C12=add(M3, M5)
    C21=add(M2, M4)
    C22=add(sub(add(M1, M3), M2), M6)
    C=[[0]*n for _ in range(n)]
    for i in range(half):
        for j in range(half):
            C[i][j] = C11[i][j]
            C[i][j + half]= C12[i][j]
            C[i+ half][j]= C21[i][j]
            C[i+ half][j+ half]= C22[i][j]
    return C
A = [
    [1, 2],
    [3, 4]
]
B = [
    [5, 6],
    [7, 8]
]
result = strassen(A, B)
for row in result:
    print(row)
```

```
[19, 22]
[43, 50]
```

In [3]:
```
def karatsuba(x, y):
    if x<10 or y<10:
        return x*y
    n = max(len(str(x)),len(str(y)))
    half = n//2
    x_high,x_low = divmod(x,10**half)
    y_high,y_low = divmod(y,10**half)
    z0=karatsuba(x_low, y_low)
    z1=karatsuba((x_low + x_high),(y_low+y_high))
    z2= karatsuba(x_high, y_high)
    return (z2*10**(2*half))+((z1-z2-z0)*10**half)+z0
x=12
y=56
result=karatsuba(x, y)
print(result)
```

```
672
```

In [ ]:
```
8.Median of medians
```

In [2]:
```
def subsetsum(subset):
    sums = set()
    n = len(subset)
    for i in range(1 << n):
        current_sum = 0
        for j in range(n):
            if i & (1 << j):
                current_sum += subset[j]
        sums.add(current_sum)
    return sums
def mim(arr, target):
    n = len(arr)
    left_part = arr[:n//2]
    right_part = arr[n//2:]
    left_sums = subsetsum(left_part)
    right_sums = subsetsum(right_part)
    for sum in left_sums:
        if (target - sum) in right_sums:
            return True
    return False
arr = [3, 34, 4, 12, 5, 2]
target = 9
result = mim(arr, target)
print(result)
```

```
True
```

In [ ]:
```
9.
```

In [3]:
```
def partition(arr, pivot):
    less = []
    equal = []
    greater = []
    for element in arr:
        if element < pivot:
            less.append(element)
        elif element == pivot:
            equal.append(element)
        else:
            greater.append(element)
    return less, equal, greater
def med(arr, k):
    if len(arr) <= 5:
        arr.sort()
        return arr[k-1]
    sublists = [arr[i:i + 5] for i in range(0, len(arr), 5)]
    medians = [sorted(sublist)[len(sublist) // 2] for sublist in sublists]
    medpivot = med(medians, len(medians) // 2)
    less, equal, greater = partition(arr, medpivot)
    if k <= len(less):
        return med(less, k)
    elif k <= len(less) + len(equal):
        return medpivot
    else:
        return med(greater, k - len(less) - len(equal))
arr = [12, 3, 5, 7, 4, 19, 26]
k = 3
result = med(arr, k)
print(result)
```

```
5
```

In [ ]: