**SUBJECT**: DESIGN AND ANALYSIS OF ALGORITHMS

**CODE**: 503040

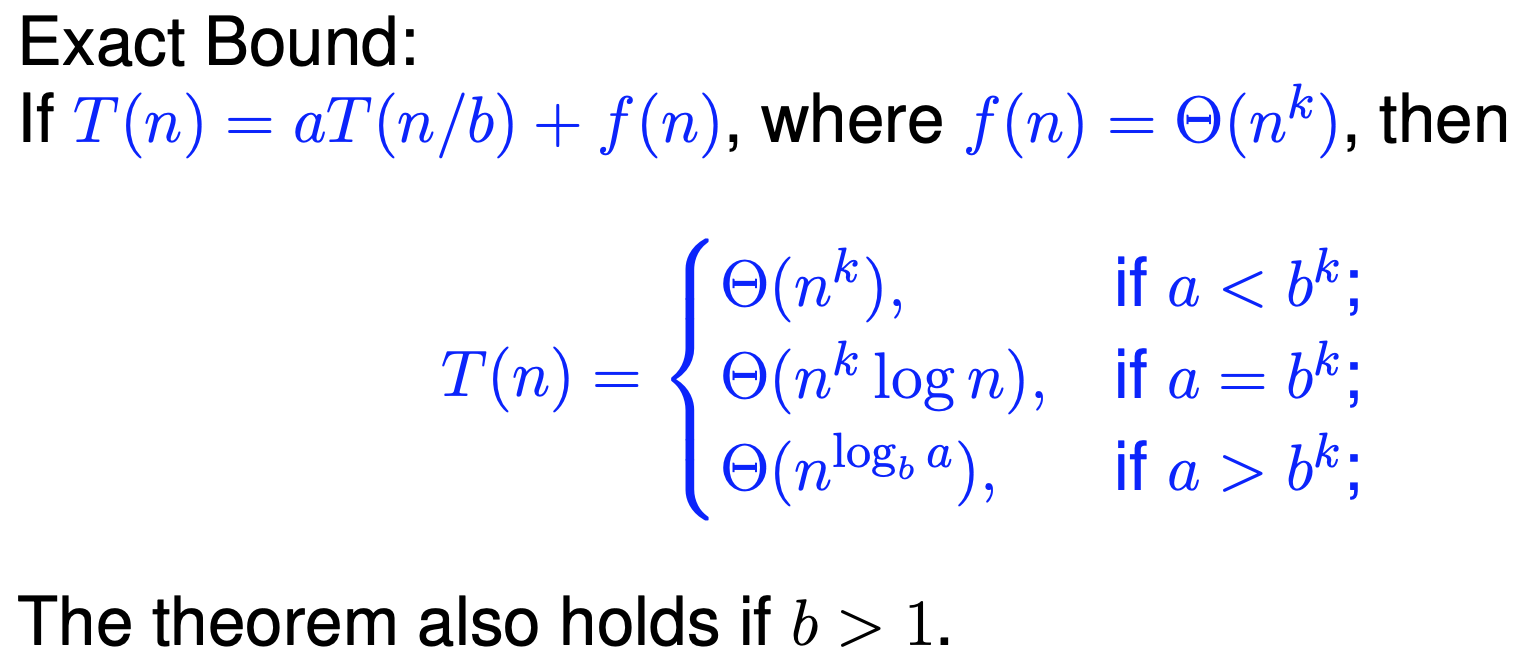
Duration: 135 minutes

Allowed to use materials.

**LAB 04: Divide-and-Conquer**

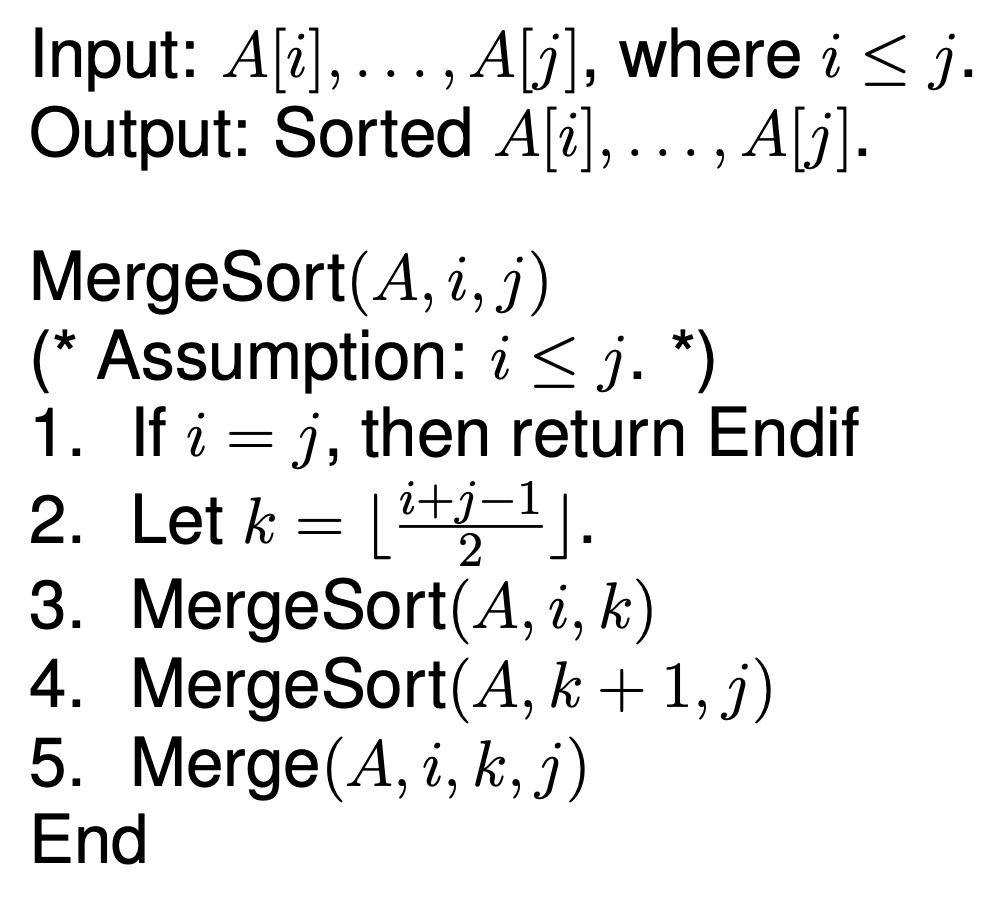
# Idea

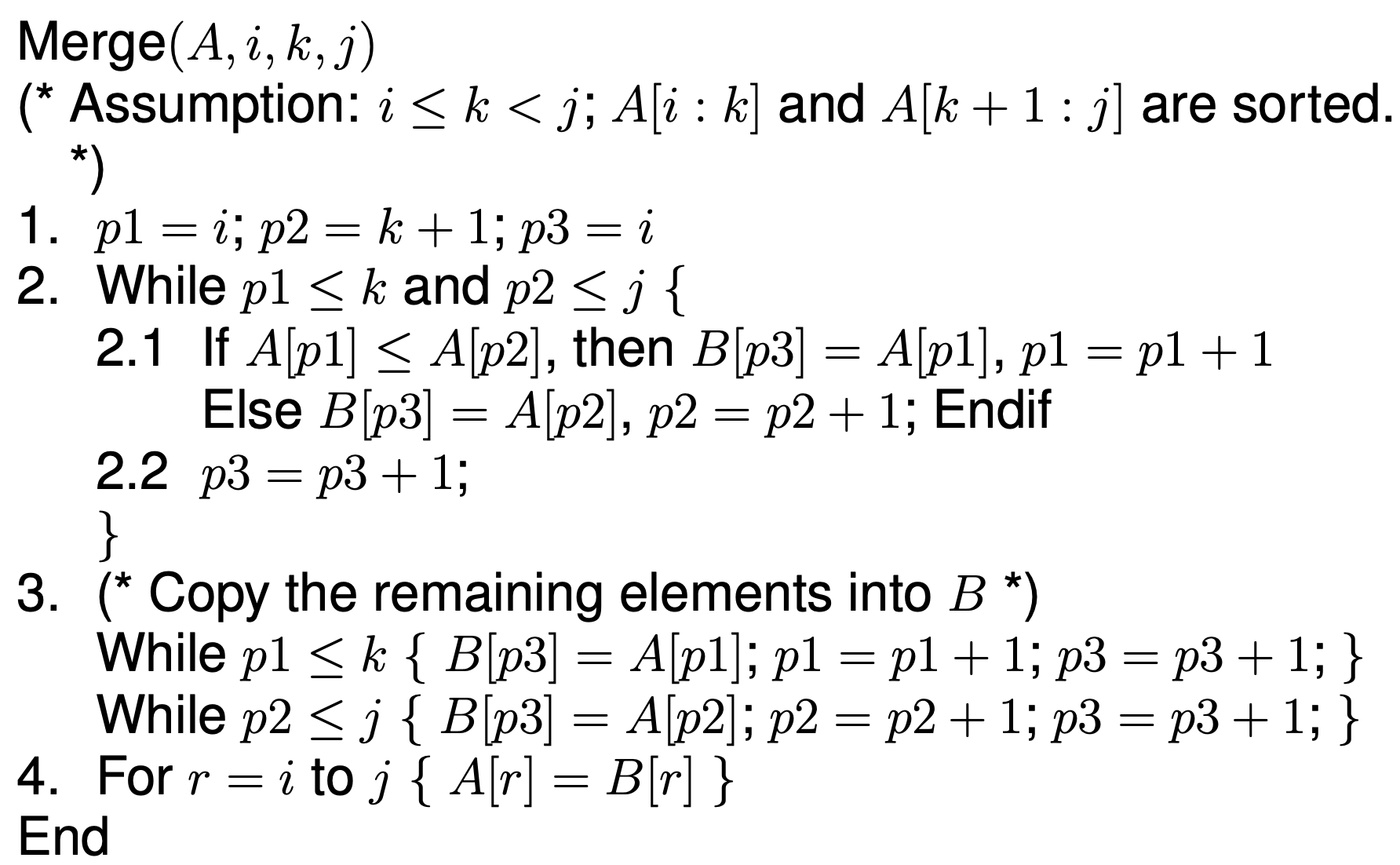
1. Divide the problem into parts.
2. Solve each part recursively.
3. Combine the Solutions.
4. Complexity is usually of form
5. Recurrence relation can be solved with Master Theorem.



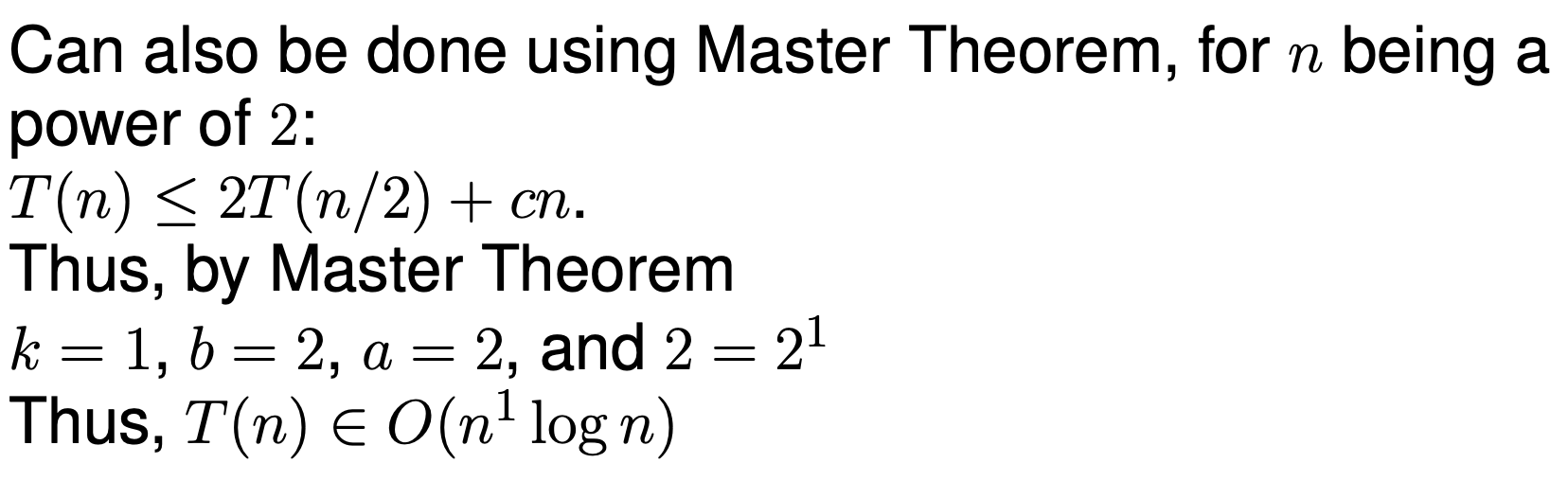
## An example of a divide-and-conquer algorithm

Pseudocode of MergeSort algorithm is given as follows:





The MergeSort algorithm analysis



Implementation of MergeSort in Python is presented as follows

import timeit

from random import shuffle

def mergesort(A, i, j):

if i == j: return None

k = (i + j)//2

mergesort(A,i,k)

mergesort(A,k+1,j)

merge(A, i, k, j)

def merge(A,i,k,j):

B = [0 for \_ in range(len(A))]

p1, p2, p3 = i, k+1, i

while p1 <= k and p2 <= j:

if A[p1] < A[p2]:

B[p3] = A[p1]

p1 += 1

else:

B[p3] = A[p2]

p2 += 1

p3 += 1

while p1 <= k:

B[p3] = A[p1]

p3 += 1

p1 += 1

while p2 <= j:

B[p3] = A[p2]

p3 += 1

p2 += 1

for r in range(i, j+1):

A[r] = B[r]

x = list(range(400))

shuffle(x)

print(x)

start = timeit.default\_timer()

mergesort(x,0, len(x) - 1)

stop = timeit.default\_timer()

print(x)

print('Time: ', stop - start)

# Exercises

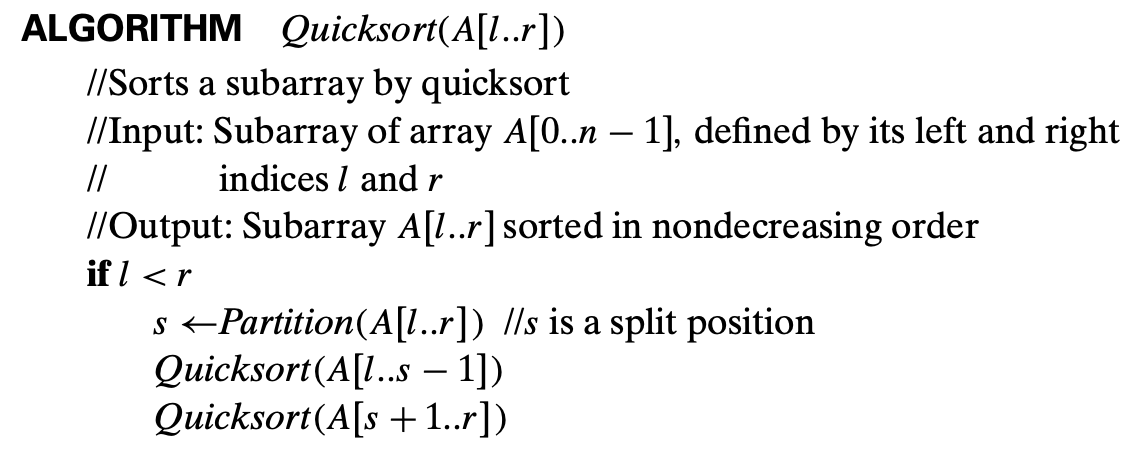
For each of the problems in this section, implement (in Python) and analyze a divide-and-conquer algorithm to solve the problem.

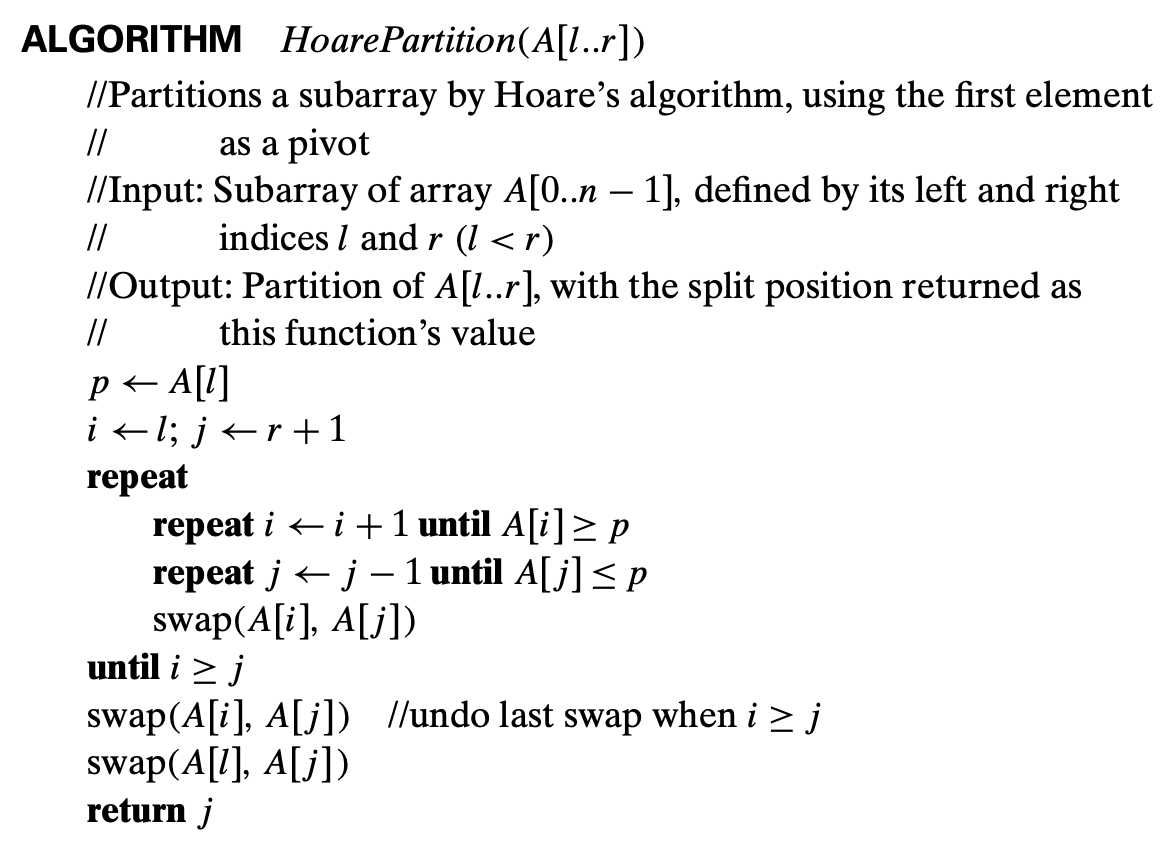
**Warm-up exercise**

1. Find an element in an ascending sorted array.

**Intermediate** **exercises**

1. Sort an array with Quicksort





1. Identify the height of a binary tree. Suppose that the height of a tree containing a single node is 0.
2. Traverse a binary tree in Pre-order, Post-order, In-order.

Node = (value, Left, Right)

Node = (1, None, None)

Node0 = (1, Node, None)