**Name : Ayush Patil**

**Roll no : TYCOC210**

**Batch : C-3**

**Prn no : 122B1B210**

**Assignment – 6**

**Title :**

Let there be N students and N clubs. Any student can be assigned to any club, incurring some cost that may vary depending on the student club assignment. It is required to allocate all clubs by assigning exactly one student to each club and exactly one club to each agent in such a way that the total cost of the assignment is minimized. Implement club assignment problem using Branch and bound

**Theory :**

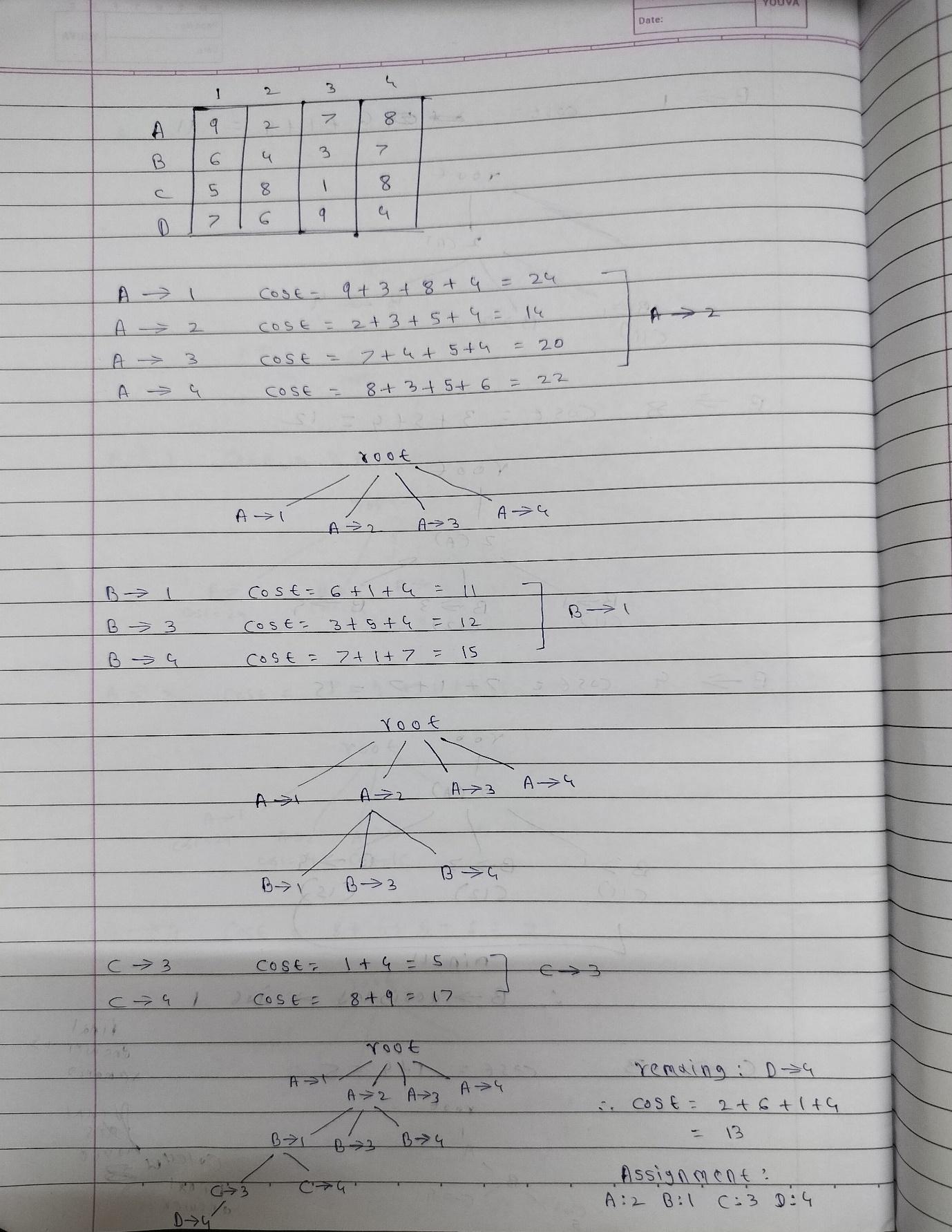
The club assignment problem, a variant of the assignment problem, requires allocating N students to N clubs while minimizing the total cost associated with these assignments. Each student can be assigned to any club, but the costs for each student-club pairing may vary, leading to the need for an optimal solution that balances these costs.

To efficiently tackle this challenge, algorithms like branch and bound are employed. This approach breaks down the problem into smaller subproblems (branching) while maintaining a record of the best solution found so far (bounding). By calculating lower bounds on potential costs in each branch, the algorithm can eliminate entire branches from consideration if they exceed the current best solution. This significantly reduces the number of assignments evaluated.

The branch and bound algorithm begins with a root node representing the initial state, where no students are assigned. It explores potential assignments iteratively using a priority queue to select the most promising nodes based on their costs. When a node reaches a state where all clubs have been assigned, the total cost is compared to the best solution found. If a better assignment is identified, the algorithm updates the best cost and assignment.

**Time Complexity**: The worst-case scenario involves exploring all possible assignments, which is O(N!) However, due to the bounding mechanism, many branches are pruned, and the effective number of explored nodes is significantly reduced. In practice, the time complexity can be closer to O(N^2) to O(N^3) depending on the implementation specifics and the nature of the cost matrix.

**Space Complexity**: The space complexity primarily arises from storing the nodes in the priority queue and maintaining the cost matrix. This can be approximated as O(N) for storing nodes and O(N^2) for the cost matrix, resulting in an overall space complexity of O(N^2).



**Code :**

import numpy as np

from queue import PriorityQueue

class Node:

def \_\_init\_\_(self, level, cost, assignment, visited):

self.level = level

self.cost = cost

self.assignment = assignment

self.visited = visited

def \_\_lt\_\_(self, other):

return self.cost < other.cost

def branch\_and\_bound(cost\_matrix):

N = len(cost\_matrix)

min\_cost = float('inf')

best\_assignment = None

root = Node(level=0, cost=0, assignment=[-1] \* N, visited=[False] \* N)

pq = PriorityQueue()

pq.put(root)

while not pq.empty():

current\_node = pq.get()

if current\_node.level == N:

if current\_node.cost < min\_cost:

min\_cost = current\_node.cost

best\_assignment = current\_node.assignment.copy()

continue

for club in range(N):

if not current\_node.visited[club]:

new\_assignment = current\_node.assignment.copy()

new\_assignment[current\_node.level] = club

new\_cost = current\_node.cost + cost\_matrix[current\_node.level][club]

new\_node = Node(

level=current\_node.level + 1,

cost=new\_cost,

assignment=new\_assignment,

visited=current\_node.visited.copy()

)

new\_node.visited[club] = True

if new\_cost < min\_cost:

pq.put(new\_node)

return best\_assignment, min\_cost

N = int(input("Enter the number of students/clubs: "))

cost\_matrix = []

print("Enter the cost matrix (row by row):")

for i in range(N):

row = list(map(int, input().split()))

cost\_matrix.append(row)

cost\_matrix = np.array(cost\_matrix)

assignment, total\_cost = branch\_and\_bound(cost\_matrix)

# Convert assignment to 1-based indexing

assignment = [job + 1 for job in assignment]

print("Optimal assignment:", assignment)

print("Minimum cost:", total\_cost)

**Output :**

