UCS 2403 Design & Analysis of Algorithms

Assignment 10

Date of Exercise: 20.05.2024

Aim: To gain understanding and proficiency on solving problems using Branch and Bound and Iterative Improvement

Question 1:

Given a preference matrix for n research scholars and n advisors, use the Gale Shapely algorithm to find an allocation of research scholars to guides such that there is no blocking pair. Develop the Python code to implement the algorithm.

Algorithm:

Algorithm:

```
Function   advisorPrefersScholar(prefer, advisor, scholar,
                                 current_scholar)
               for i ← range(no_scholars):
                   if prefer[advisor][i]
                                 == current_
                                    scholar
                       return False
                   if prefer[advisor][i] == scholar
                       return True
               return False

Function   galeShapley(prefer):
               advisor_match ← [-1 for i in range
                                   (no_scholars)]
               scholar_free ← [False for _ in range
                                   (no_scholars)]
               free_scholars_count ← no_scholars
```

```
while free_scholars_count > 0:
    scholar ← 0
    while scholar < no_scholars:
        if not scholar_free[scholar]
            break
        scholar += 1

    for i in range(no_scholars):
        if scholar_free[scholar]:
            break
        advisor ← prefer[scholar][i]
        advisor_index ← advisor
                        - no_scholars
        if advisor_match[advisor_index] == -1:
            advisor_match[advisor_index]
                        ← scholar
            scholar_free[scholar] = True
            free_scholars_count -= 1
        else:
            current_scholar ← advisor_match
                        [advisor_index]
            if advisorPrefersScholar(prefer,
                        advisor, scholar,
                        current_scholar):
                advisor_match[advisor_index]
                        ← scholar
                scholar_free[scholar] ← True
                scholar_free[current_
                        scholar] ← False
```

Code:

```
def advisorPrefersScholar(prefer, advisor, scholar, current_scholar):
    for i in range(no_scholars):
        if prefer[advisor][i] == current_scholar:
            return False
        if prefer[advisor][i] == scholar:
            return True
    return False

def galeShapley(prefer):
    advisor_match = [-1 for _ in range(no_scholars)]
    scholar_free = [False for _ in range(no_scholars)]
    free_scholars_count = no_scholars
```

```
    while free_scholars_count > 0:
        scholar = 0
        while scholar < no_scholars:
            if not scholar_free[scholar]:
                break
            scholar += 1
        for i in range(no_scholars):
            if scholar_free[scholar]:
                break
            advisor = prefer[scholar][i]
            advisor_index = advisor - no_scholars
            if advisor_match[advisor_index] == -1:
                advisor_match[advisor_index] = scholar
                scholar_free[scholar] = True
                free_scholars_count -= 1
            else:
                current_scholar = advisor_match[advisor_index]
                if advisorPrefersScholar(prefer, advisor, scholar,
current_scholar):
                    advisor_match[advisor_index] = scholar
                    scholar_free[scholar] = True
                    scholar_free[current_scholar] = False

    print("\nResearch Scholars \t Advisors")
    for i in range(no_scholars):
        print(advisor_match[i], "\t\t\t", i + no_scholars)

no_scholars = int(input("Enter the number of scholars: "))

research_scholars = []
advisors = []

for i in range(no_scholars):
    research_scholars.append(i)
for k in range(no_scholars, 2 * no_scholars):
    advisors.append(k)

print("Research scholars are: ", research_scholars)
print("Advisors/Guides are: ", advisors)
print("\n")
prefer = []
for i in range(2 * no_scholars):
    prefs = list(map(int, input(f"Enter preferences for {'research scholar' if i <
no_scholars else 'advisor'} {i % no_scholars + 1}: ").split()))
    prefer.append(prefs)

galeShapley(prefer)
```

Output:

```
PS C:\Users\Mugilkrishna D U\OneDrive\Desktop\My Files\.vscode> & "C:/Users/Mugilkrishna D U/AppDat
a/Local/Programs/Python/Python311/python.exe" "c:/Users/Mugilkrishna D U/OneDrive/Desktop/My Files/
SSN/SEM4/DESIGN AND ANALYSIS OF ALGORITHMS/LAB/10.1.py"
Enter the number of scholars: 5
Research scholars are:  [0, 1, 2, 3, 4]
Advisors/Guides are:  [5, 6, 7, 8, 9]


Enter preferences for research scholar 1: 8 6 7 5 9
Enter preferences for research scholar 2: 9 7 6 5 8
Enter preferences for research scholar 3: 6 9 5 8 7
Enter preferences for research scholar 4: 9 6 8 7 5
Enter preferences for research scholar 5: 8 5 6 7 9
Enter preferences for advisor 1: 3 1 4 2 0
Enter preferences for advisor 2: 1 0 3 2 4
Enter preferences for advisor 3: 0 2 4 3 1
Enter preferences for advisor 4: 3 0 2 1 4
Enter preferences for advisor 5: 1 4 0 2 3

Research Scholars        Advisors
4                        5
3                        6
2                        7
0                        8
1                        9
PS C:\Users\Mugilkrishna D U\OneDrive\Desktop\My Files\.vscode>
```

```
PS C:\Users\Mugilkrishna D U\OneDrive\Desktop\My Files\.vscode> & "C:/Users/Mugilkrishna D U/AppDat
a/Local/Programs/Python/Python311/python.exe" "c:/Users/Mugilkrishna D U/OneDrive/Desktop/My Files/
SSN/SEM4/DESIGN AND ANALYSIS OF ALGORITHMS/LAB/10.1.py"
Enter the number of scholars: 3
Research scholars are:  [0, 1, 2]
Advisors/Guides are:  [3, 4, 5]


Enter preferences for research scholar 1: 5 3 4
Enter preferences for research scholar 2: 5 4 3
Enter preferences for research scholar 3: 3 4 5
Enter preferences for advisor 1: 0 1 2
Enter preferences for advisor 2: 1 2 0
Enter preferences for advisor 3: 2 1 0

Research Scholars        Advisors
0                        3
2                        4
1                        5
PS C:\Users\Mugilkrishna D U\OneDrive\Desktop\My Files\.vscode>
```

Time Complexity:

The Time Complexity of the Gale Shapley algorithm is O(n^2) where n is the number of scholars

Question 2:

There are n people who need to be assigned to execute n jobs, one person per job. (That is, each person is assigned to exactly one job and each job is assigned to exactly one person.) The cost that would accrue if the ith person is assigned to the jth job is a known quantity C[i, j] for each pair i, j = 1 . . . n. Using branch-and-bound, develop a Python code that assigns the people to the jobs to minimize the total cost of the assignment.

Algorithm:

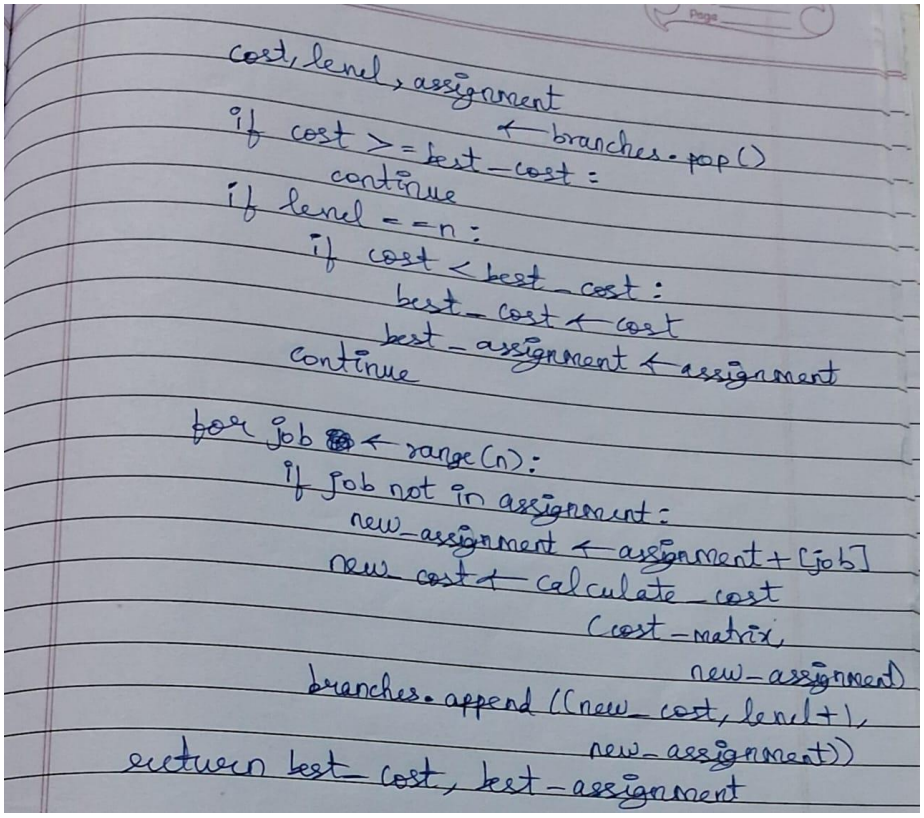
Algorithm:

Function
```
calculate_cost (cost_matrix, assignment):
    cost ← 0
    for i, job in enumerate (assignment):
        cost += cost_matrix [i] [job]
    return cost
```

function
```
branch_and_bound (cost_matrix):
    n ← len (cost_matrix)
    branches ← []
    branches. append ((0, 0, []))

    best_cost ← float ('inf')
    best_assignment ← None

    while branches:
        branches. sort (reverse = True)
```



```
cost, level, assignment
                ← branches.pop()
if cost >= best_cost :
    continue
if level == n :
    if cost < best_cost :
        best_cost ← cost
        best_assignment ← assignment
    continue

for job ← range(n):
    if job not in assignment :
        new_assignment ← assignment + [job]
        new_cost ← calculate_cost
                    (cost_matrix,
                     new_assignment)
        branches.append((new_cost, level+1,
                         new_assignment))
return best_cost, best_assignment
```

Code:

```python
def calculate_cost(cost_matrix, assignment):
    cost = 0
    for i, job in enumerate(assignment):
        cost += cost_matrix[i][job]
    return cost

def branch_and_bound(cost_matrix):
    n = len(cost_matrix)
    branches = []
    branches.append((0, 0, []))
    best_cost = float('inf')
    best_assignment = None

    while branches:
        branches.sort(reverse=True)
        cost, level, assignment = branches.pop()
        if cost >= best_cost:
            continue
        if level == n:
            if cost < best_cost:
                best_cost = cost
                best_assignment = assignment
            continue
```

Name: Mugilkrishna D U
Reg. No.: 3122225001073

```
        for job in range(n):
            if job not in assignment:
                new_assignment = assignment + [job]
                new_cost = calculate_cost(cost_matrix, new_assignment)
                branches.append((new_cost, level + 1, new_assignment))

    return best_cost, best_assignment

n = int(input("Enter the number of jobs: "))
print("\n")

cost_matrix = []
for i in range(n):
    row = []
    for j in range(n):
        cost = int(input(f"Enter the cost for person {i + 1} to job {j+ 1}: "))
        row.append(cost)
    print("\n")
    cost_matrix.append(row)

best_cost, best_assignment = branch_and_bound(cost_matrix)

print("Total cost:", best_cost)
print("Best assignment:", ["Job " + str(job + 1) for job in best_assignment])
```

Output:

```
PS C:\Users\Mugilkrishna D U\OneDrive\Desktop\My Files\.vscode> & "C:/Users/Mugilkrishna D U/AppDat
a/Local/Programs/Python/Python311/python.exe" "c:/Users/Mugilkrishna D U/OneDrive/Desktop/My Files/
SSN/SEM4/DESIGN AND ANALYSIS OF ALGORITHMS/LAB/10.2.py"
Enter the number of jobs: 4


Enter the cost for person 1 to job 1: 9
Enter the cost for person 1 to job 2: 2
Enter the cost for person 1 to job 3: 7
Enter the cost for person 1 to job 4: 8


Enter the cost for person 2 to job 1: 6
Enter the cost for person 2 to job 2: 4
Enter the cost for person 2 to job 3: 3
Enter the cost for person 2 to job 4: 7


Enter the cost for person 3 to job 1: 5
Enter the cost for person 3 to job 2: 8
Enter the cost for person 3 to job 3: 1
Enter the cost for person 3 to job 4: 8


Enter the cost for person 4 to job 1: 7
Enter the cost for person 4 to job 2: 6
```

```
Enter the cost for person 4 to job 3: 9
Enter the cost for person 4 to job 4: 4


Total cost: 13
Best assignment: ['Job 2', 'Job 1', 'Job 3', 'Job 4']
PS C:\Users\Mugilkrishna D U\OneDrive\Desktop\My Files\.vscode>
```

```
PS C:\Users\Mugilkrishna D U\OneDrive\Desktop\My Files\.vscode> & "C:/Users/Mugilkrishna D U/AppDat
a/Local/Programs/Python/Python311/python.exe" "c:/Users/Mugilkrishna D U/OneDrive/Desktop/My Files/
SSN/SEM4/DESIGN AND ANALYSIS OF ALGORITHMS/LAB/10.2.py"
Enter the number of jobs: 3


Enter the cost for person 1 to job 1: 3
Enter the cost for person 1 to job 2: 6
Enter the cost for person 1 to job 3: 7


Enter the cost for person 2 to job 1: 2
Enter the cost for person 2 to job 2: 3
Enter the cost for person 2 to job 3: 4


Enter the cost for person 3 to job 1: 9
Enter the cost for person 3 to job 2: 1
Enter the cost for person 3 to job 3: 4


Total cost: 8
Best assignment: ['Job 1', 'Job 3', 'Job 2']
PS C:\Users\Mugilkrishna D U\OneDrive\Desktop\My Files\.vscode>
```

Time Complexity:

The Time Complexity of the Job Assignment problem is O(n!) where n is the number of jobs

Learning Outcome:

Upon completing this exercise, I have understood the applications of Branch and Bound and it's various uses for solving problems in an effective manner. I have also learnt how to improve the iterations and to optimize it properly. I have now learnt to implement the Gale Shapely algorithm for matching research scholars with advisors/guides. I have also learnt how to solve the best job assignment problem.