DISCO PROJECT

By-
MAHATVA GARG(2021B
MIHIKA SINGHAL(2022A7PS0345G)
PRISHA GUPTA(2022A7PS0164G)

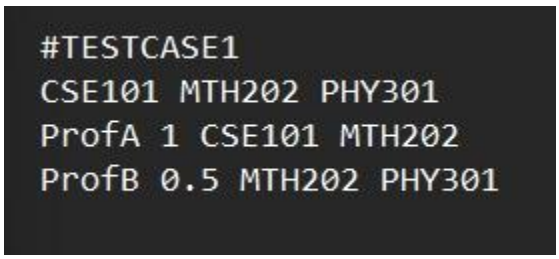
```
\documentclass{article}
\usepackage{amsmath}
\usepackage{algorithm}
\usepackage{algpseudocode}
\usepackage{graphicx}
\usepackage{adjustbox}

\begin{document}

\section*{Hungarian Algorithm Implementation}

\begin{algorithm}
\caption{Hungarian Algorithm Functions}
\begin{algorithmic}[1]
\Function{find\_min\_zero\_row}{zeromatrix, zeromarked}
    \State minrow $\gets$ [$\infty$, -1]
    \For{numrow in range(zeromatrix.shape[0])}
        \State countzeroes $\gets$ np.sum(zeromatrix[numrow] == 0)
        \If{countzeroes > 0 and minrow[0] > countzeroes}
            \State minrow $\gets$ [countzeroes, numrow]
        \EndIf
    \EndFor
    \State indexzero $\gets$ np.where(zeromatrix[minrow[1]] == 0)[0][0]
    \State zeromarked.append((minrow[1], indexzero))
    \State zeromatrix[minrow[1], :] $\gets$ $\infty$
    \State zeromatrix[:, indexzero] $\gets$ $\infty$
\EndFunction

\Function{markedmatrix}{matrix}
    \State culmatrix $\gets$ matrix
    \State zeroboolmat $\gets$ (culmatrix == 0)
    \State zeroboolmatcopy $\gets$ zeroboolmat.copy()
    \State zeromarked $\gets$ []

    \While{True in zeroboolmatcopy}
        \Call{find\_min\_zero\_row}{zeroboolmatcopy, zeromarked}
    \EndWhile

    \State zeromarkedrow $\gets$ [pos[0] for pos in zeromarked]
    \State zeromarkedcol $\gets$ [pos[1] for pos in zeromarked]
    \State rownonmarked $\gets$ list(set(range(culmatrix.shape[0])) - set(zeromarkedrow))
    \State columnmarked $\gets$ []
    \State checkit $\gets$ True
```

```
    \While{checkit}
        \State checkit $\gets$ False
        \For{i in range(len(rownonmarked))}
            \State row\_array $\gets$ zeroboolmat[rownonmarked[i], :]
            \For{j in range(row\_array.shape[0])}
                \If{row\_array[j] == True and j not in columnmarked}
                    \State columnmarked.append(j)
                    \State checkit $\gets$ True
                \EndIf
            \EndFor
        \EndFor

        \For{numrow, col\_num in zeromarked}
            \If{numrow not in rownonmarked and col\_num in columnmarked}
                \State rownonmarked.append(numrow)
                \State checkit $\gets$ True
            \EndIf
        \EndFor
    \EndWhile

    \State marked\_rows $\gets$ list(set(range(matrix.shape[0])) - set(rownonmarked))
    \State \Return zeromarked, marked\_rows, columnmarked
\EndFunction

\Function{adjmatrix}{matrix, cover\_rows, cover\_cols}
    \State culmatrix $\gets$ matrix
    \State numberofnonzeroelements $\gets$ []

    \For{row in range(len(culmatrix))}
        \If{row not in cover\_rows}
            \For{i in range(len(culmatrix[row]))}
                \If{i not in cover\_cols}
                    \If{culmatrix[row, i] != $\infty$}
                        \State numberofnonzeroelements.append(culmatrix[row, i])
                    \EndIf
                \EndIf
            \EndFor
        \EndIf
    \EndFor

    \State min\_num $\gets$ min(numberofnonzeroelements)

    \For{row in range(len(culmatrix))}
```

```
        \If{row not in cover\_rows}
            \For{i in range(len(culmatrix[row]))}
                \If{i not in cover\_cols}
                    \If{culmatrix[row, i] != $\infty$}
                        \State culmatrix[row, i] $\gets$ culmatrix[row, i] - min\_num
                    \EndIf
                \EndIf
            \EndFor
        \EndIf
    \EndFor

    \For{row in range(len(cover\_rows))}
        \For{col in range(len(cover\_cols))}
            \State culmatrix[cover\_rows[row], cover\_cols[col]] $\gets$ culmatrix[cover\_rows[row],
cover\_cols[col]] + min\_num
        \EndFor
    \EndFor

    \State \Return culmatrix
\EndFunction

\Function{hungalgo}{matrix}
    \State dim $\gets$ matrix.shape[0]
    \State culmatrix $\gets$ matrix

    \For{numrow in range(matrix.shape[0])}
        \State min\_val $\gets$ np.min(culmatrix[numrow])
        \If{min\_val > 0}
            \State culmatrix[numrow] $\gets$ culmatrix[numrow] - min\_val
        \EndIf
    \EndFor

    \For{col\_num in range(matrix.shape[1])}
        \State min\_val $\gets$ np.min(culmatrix[:, col\_num])
        \If{min\_val > 0}
            \State culmatrix[:, col\_num] $\gets$ culmatrix[:, col\_num] - min\_val
        \EndIf
    \EndFor

    \State zero\_count $\gets$ 0

    \While{zero\_count < dim}
        \State ans\_pos, marked\_rows, columnmarked $\gets$ \Call{markedmatrix}{culmatrix}
        \State zero\_count $\gets$ len(marked\_rows) + len(columnmarked)
```

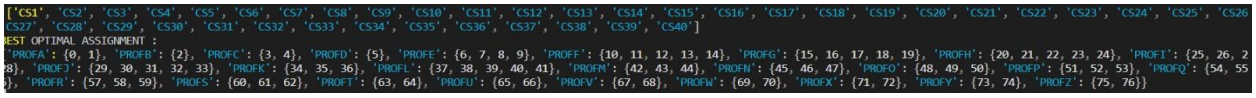
```
        \If{zero\_count < dim}
            \State culmatrix $\gets$ \Call{adjmatrix}{culmatrix, marked\_rows, columnmarked}
        \EndIf
    \EndWhile

    \State \Return ans\_pos
\EndFunction

\Function{calculate\_answer}{matrix, pos}
    \State total $\gets$ 0
    \State ans\_matrix $\gets$ np.zeros((matrix.shape[0], matrix.shape[1]))

    \For{i in range(len(pos))}
        \State total += matrix[pos[i][0], pos[i][1]]
        \State ans\_matrix[pos[i][0], pos[i][1]] $\gets$ matrix[pos[i][0], pos[i][1]]
    \EndFor

    \State \Return total, ans\_matrix
\EndFunction

\Function{solve}{matrix}
    \State cost\_matrix $\gets$ matrix
    \State ans\_pos $\gets$ \Call{hungalgo}{cost\_matrix.copy()}
    \State ans, ans\_
```

EXPECTED OUTPUT WITH REFERENCE TO THE TESTCASES

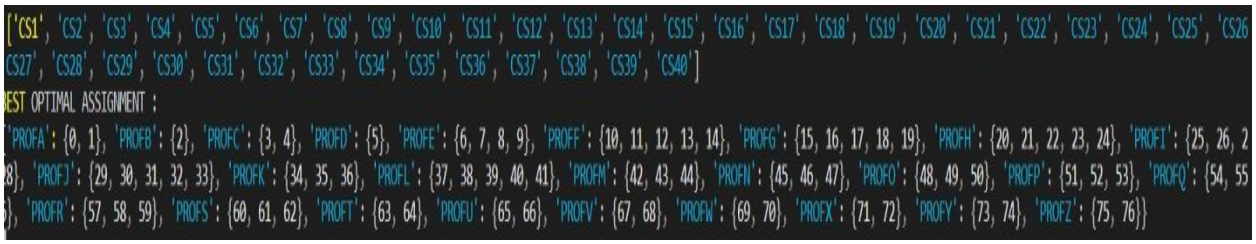


```
LIST OF COURSES OFFERED :
['CSE101', 'MTH202', 'PHY301']
BEST OPTIMAL ASSIGNMENT :
 {'ProfA': {0, 1}, 'ProfB': {2}}
```

```
#TESTCASE2
CS1 CS2 CS3 CS4 CS5 CS6
PROFA 1 CS1 CS2 CS3
PROFB 0.5 CS2 CS4 CS6
PROFC 1 CS3 CS5
PROFD 0.5 CS1 CS4
```




```
#TESTCASE3
CS1 CS2 CS3 CS4 CS5 CS6 CS7 CS8 CS9 CS10 CS11 CS12 CS13 CS14 CS15 CS16 CS17 CS18 CS19 CS20
CS21 CS22 CS23 CS24 CS25 CS26 CS27 CS28 CS29 CS30 CS31 CS32 CS33 CS34 CS35 CS36 CS37 CS38
CS39 CS40
PROFA 1 CS1 CS2 CS3 CS4
PROFB 0.5 CS2 CS4 CS6
PROFC 1 CS3 CS5
PROFD 0.5 CS1 CS4
PROFE 1 CS5 CS10 CS15 CS20
PROFF 0.5 CS6 CS12 CS18 CS24 CS30
PROFG 1 CS7 CS14 CS21 CS28 CS35
PROFH 0.5 CS8 CS16 CS24 CS32 CS40
PROFI 1 CS9 CS18 CS27 CS36
PROFJ 0.5 CS10 CS20 CS30 CS40
PROFK 1 CS11 CS22 CS33
PROFL 0.5 CS12 CS24 CS36
PROFM 1 CS13 CS26 CS39
PROFN 0.5 CS14 CS28 CS42
PROFO 1 CS15 CS30 CS45
PROFP 0.5 CS16 CS32 CS48
PROFQ 1 CS17 CS34 CS51
PROFR 0.5 CS18 CS36 CS54
PROFS 1 CS19 CS38 CS57
PROFT 0.5 CS20 CS40
PROFU 1 CS21 CS42
PROFV 0.5 CS22 CS44
PROFW 1 CS23 CS46
PROFX 0.5 CS24 CS48
PROFY 1 CS25 CS50
PROFZ 0.5 CS26 CS52
```




```
#TESTCASE 4
CS1 CS2 CS3 CS4 CS5 CS6 CS7
PROFA 1 CS1 CS2 CS3 CS4
PROFB 0.5 CS2 CS4 CS6
PROFC 1 CS3 CS5
PROFD 0.5 CS1 CS4
```

```
LIST OF COURSES OFFERED :
['CS1', 'CS2', 'CS3', 'CS4', 'CS5', 'CS6', 'CS7']
BEST OPTIMAL ASSIGNMENT :
Not Possible
```

Algorithm may not produce an optimal assignment due to the structure of the input matrix and constraints we considered.